

Министерство науки и высшего образования Российской Федерации  
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

---

Г.В. ТРОШИНА

# ЧИСЛЕННЫЕ РАСЧЕТЫ В СРЕДЕ MATLAB

Утверждено Редакционно-издательским советом университета  
в качестве учебного пособия

НОВОСИБИРСК  
2020

УДК 004.42(075.8)  
Т 766

Рецензенты:

канд. техн. наук, доцент *С.П. Ильиных*  
канд. физ.-мат. наук, доцент *Ю.А. Котов*

Работа подготовлена на кафедре вычислительной техники

**Трошина Г.В.**

Т 766 Численные расчеты в среде MatLab: учебное пособие /  
Г.В. Трошина. – Новосибирск: Изд-во НГТУ, 2020. – 72 с.

ISBN 978-5-7782-4092-6

В учебном пособии представлены основные приемы создания М-файлов и графического интерфейса в среде MatLab. Предназначено для студентов, обучающихся по направлениям 09.03.01 – «Информатика и вычислительная техника», 09.03.04 – «Программная инженерия».

УДК 004.42(075.8)

ISBN 978-5-7782-4092-6

© Трошина Г.В., 2020  
© Новосибирский государственный  
технический университет, 2020

## **Введение**

Цель написания данного учебного пособия – дать теоретическую базу, необходимую для выполнения лабораторных, контрольных и расчетно-графических работ по курсам «Средства инженерных и научных расчетов», «Вычислительная математика».

Каждый из разделов учебного пособия включает в себя примеры, графики, контрольные вопросы и тесты, необходимые для лучшего понимания излагаемого материала и получения практических навыков использования среды MatLab.

Большое внимание уделено рассмотрению установившегося режима в задаче оценивания параметров динамических объектов. Приведена методика формирования информационной матрицы Фишера, рассмотрены вопросы моделирования итерационной процедуры оценивания динамических параметров в задаче активной идентификации.

# 1. Создание М-файлов и операции с функциями

## 1.1. Создание М-файлов

При использовании командного окна среды MatLab в диалоговом режиме пользователь формирует одну или несколько команд в текущей строке. Несмотря на широкие возможности, которые предоставляет пользователю среда MatLab, такая последовательность действий подходит только для разовых расчетов. При изменениях в исходных данных или при выполнении большого количества однотипных действий этот режим не всегда является удобным.

Необходимую цепочку команд можно включить в текстовый файл. В результате неоднократное обращение к одним и тем же данным позволит сделать вычислительный процесс более удобным. Подобного рода программы принято называть сценариями. Сценарий является самым простым типом М-файлов – у него нет входных и выходных аргументов. Он используется для автоматизации многократно выполняемых вычислений.

Сценарий может быть подготовлен в любом текстовом редакторе и записан с расширением *.m*. Такие файлы используют единое рабочее пространство – *Workspace*. Переменные изменяются и сохраняются в течение всего сеанса работы. Это позволяет выполнить последовательно несколько файлов-сценариев в текущем сеансе работы с системой. Значения переменных передаются через *Workspace*, а промежуточные данные сохраняются в файлах.

Среда MatLab располагает собственным редактором М-файлов, который вызывается после активизации команды *File/New/M-File*. Приступить к редактированию существующего М-файла можно после выбора команды *File/Open*. При формировании последовательности команд в окне тестового редактора осуществляется автоматическая нумерация строк.

При оформлении текста программ на языке MatLab есть свои особенности. Каждый оператор должен быть описан в отдельной строке.

Нажатие клавиши *<Enter>* означает окончание оператора. Длинный оператор записывается в несколько строк, но в этом случае в конце предыдущей строки ставятся три точки. Символ *< % >* означает начало комментария в тексте программы. Все строчные и прописные буквы при определении имен переменных различаются. Длинный файл-сценарий можно разделить на отдельные программы, каждый из которых также можно представить в виде самостоятельного файла-сценария. Файл-сценарий позволяет управлять всем вычислительным процессом.

Пример. Рассчитать значение выражения:

$$(\sin(3 * \pi) + \cos(0) + \log(1))/\exp(1)$$

Решение: `>> (sin(3*pi) + cos(0) + log(1))/exp(1)`

`ans =`

`0.3679`

Пример. Написать М-файл с именем *primer.m*, который будет считать факториал:

`a = 1`

`for i = 1:5`

`a = a * i`

`end`

Этот файл является файлом-сценарием. Ввод команды *primer.m* в командной строке системы MatLab вызывает выполнение операторов этого сценария. Точка с запятой в конце оператора подавляет вывод результата на экран.

## 1.2. Создание файлов-функций

Файлы-функции являются разновидностью М-файлов. Эти файлы имеют расширение *.m*, а в качестве заголовка используют оператор *function*. В общем виде первая строка файла-функции может быть записана следующим образом:

*function <выходные параметры>=<имя функции>(входные параметры)*

Файлы-функции и файлы-сценарии нельзя отличить по расширению имени файла *.m*. Процедуры, предназначенные для многократного использования файлами-сценариями, оформляются в виде файлов-функций. Переменные, используемые в рамках файлов-функций, являются локальными переменными. Область, используемая локальными переменными, после выполнения файла-функции освобождается для других переменных. Переменные единого рабочего пространства с помощью оператора *global* могут быть объявлены в качестве глобальных переменных. В этом случае функция может обратиться к указанным переменным.

Необходимо, чтобы имя М-файла, в котором записывается программа вычислительного процесса, совпадало с именем функции. Имя функции не должно превышать 31 символ. Имя может быть и длиннее, но система MatLab принимает во внимание только первые 31 символ. Имя файла-функции должно начинаться с буквы, остальные символы могут быть любой комбинацией букв, цифр и подчеркиваний.

Пример. Написать функцию для вычисления факториала. Файл должен быть сохранен с именем *factorial.m*:

```
function y = factorial(n)
```

```
k = 1;
```

```
for i = 1:n
```

```
k = k * i;
```

```
end
```

```
y = k;
```

Для вычисления 5! достаточно набрать в текущей строке команду

```
>> k = factorial(5)
```

```
k =
```

```
120
```

*Структура файла-функции.* Файл-функция состоит из строки определения функции, первой строки комментария, собственно комментария, тела функции, строчных комментариев.

Строка определения функции сообщает системе MatLab, что процедура является файлом-функцией, а также определяет список входных аргументов.

Пример:

```
function y = average(x)
```

где *function* – ключевое слово, определяющее файл-функцию, *y* – выходной аргумент, *average* – имя функции, *x* – входной аргумент.

Если функция имеет более одного выходного аргумента, то список выходных аргументов помещается в квадратные скобки. Входные аргументы, если они присутствуют, помещаются в круглые скобки. Для отделения аргументов во входном и выходном списках применяются запятые.

Пример:

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

Первая строка комментария. Для функции *average* первая строка комментария выглядит так:

```
% AVERAGE Среднее значение элементов вектора
```

Это первая строка текста, которая появляется, когда пользователь набирает команду *help* <имя\_функции>.

*Комментарий.* Для М-файлов можно создать online-подсказку, вводя текст в одной или более строках. При вводе команды подсказки *help* <имя\_функции> система MatLab отображает строки комментария, которые размещаются между строкой определения функции и первой пустой строкой либо началом программы. Команда *help* <имя\_функции> игнорирует комментарии, размещенные вне этой области.

*Тело функции.* Тело функции содержит код языка MatLab, который выполняет вычисления и присваивает значения выходным аргументам. Операторы в теле функции могут состоять из вызовов функций, программных конструкций для управления потоком команд, интерактивного ввода/вывода, вычислений, присваиваний, комментариев и пустых строк.

*Вызов функции.* При вызове *m*-функции система MatLab транслирует функцию в псевдокод и загружает в память. Псевдокод остается в памяти до тех пор, пока не будет использована команда *clear* или завершен сеанс работы.

Существует несколько разновидностей *m*-функций. Функции, имена которых совпадают с именами М-файлов, называются головными функциями.

Подфункции представляют собой такой вид функций, описания которых находятся в М-файле после головной функции. Особенность использования подфункций М-файла заключается в том, что они не могут быть вызваны извне. Эти функции предусматривают внутреннее использование. Вызов подфункций осуществляет головная функция М-файла. К вложенной функции всегда может обращаться окружающая ее функция.

Для организации ветвлений внутри выполнения вычислительной процедуры применяются условные операторы.

Ниже приведены конструкции условных операторов:

1-й вариант: *if* <условие>

    <операторы>

*end*

Операторы (тело выражения) выполняются только в том случае, если условие истинно, если условие ложно, то тело выражения не выполняется.

2-й вариант: *if* <условие>

    <операторы 1>

*else*

    <операторы 2>

*end*

3-й вариант: *if* <условие1>

    <операторы 1>

*elseif* <условие2>

    <операторы2>

*elseif* <условие3>

    <операторы3>

    ...



*else*

*<операторы>*

*end*

В среде MatLab применяются следующие операторы сравнения: < – меньше; <= – меньше или равно; > – больше; >= – больше или равно; = – равно; ~= – не равно. В среде MatLab предусмотрено использование следующих логических операций: & – логическое «и» (and); | – логическое «или» (or); ~ – логическое отрицание (not). В результате выполнения логических операций получают значения 0 (false) и 1 (true).

Как правило, при разработке программ требуется использовать операторы цикла. Условный оператор цикла (или оператор цикла с предусловием) осуществляет повторение операторов нефиксированное число раз. Формат оператора имеет следующий вид:

*while <условие>*

*<операторы>*

*end*

Операторы выполняются, если переменная «условие» имеет ненулевые элементы.

Формат арифметического оператора цикла можно представить следующим образом:

*for <имя> = <начальное значение>: <шаг>: <конечное значение>*

*<операторы>*

*end*

где <имя> – имя переменной цикла, <начальное значение> – начальное значение переменной цикла, <конечное значение> – конечное значение управляющей переменной. Величина <шаг> указывает значение приращения переменной цикла в процессе ее изменения от начального значения до конечного значения. Если величина шага не задана, то по умолчанию значение шага принимается равным единице.

Для вывода информации в диалоговом режиме можно использовать команду *disp*. Эта команда имеет один аргумент. Для вывода нескольких переменных необходимо создать объект, содержащий все значения. Например, это можно сделать, если сформировать вектор соответствующих переменных. Аналогично можно объединять и текстовые

переменные. Некоторые трудности возникают при объединении текстовых и числовых переменных, так как они являются данными разных типов. В этом случае используют функцию *num2str*, которая дает возможность преобразовать числовое значение переменной в текстовую переменную. Другой способ, позволяющий объединить в одну строку текст и числовую переменную, состоит в использовании функции *sprintf*. После выполнения команды

*disp(sprintf('<текст1> %g <текст2>', X))*

получаем текстовую строку, в состав которой входит текст, заданный в части <текст1>, значения переменной *X* в соответствии с форматом %g, и текст, содержащийся в части <текст2>.

### 1.3. Графическое оформление результатов вычислений

В среде MatLab представлены широкие возможности графического оформления информации. Это дает возможность для построения двухмерных и трехмерных графиков функций, заданных как в аналитическом виде, так и в виде векторов и матриц, что позволяет построить множество функций на одном рисунке, а также иллюстрировать графики разными цветами, типами точек и линий в различных системах координат.

Пользователь среды MatLab способен строить диаграммы, гистограммы и графики специальных функций. В качестве основных функций двухмерной графики используются следующие функции:

*plot(x, y)*

*plot(x, y, s)*

*plot(x<sub>1</sub>, y<sub>1</sub>, s<sub>1</sub>, x<sub>2</sub>, y<sub>2</sub>, s<sub>2</sub>, ..., x<sub>n</sub>, y<sub>n</sub>, s<sub>n</sub>)*

где *x* – аргумент функции, задаваемой в виде вектора;

*y* – функция, представленная в аналитическом виде или в виде вектора или матрицы;

*s* – вектор стилей графика, может быть константа, определяющая цвет линий графика, тип точек и линий;

*x<sub>1</sub>, x<sub>2</sub>, ..., x<sub>n</sub>* – аргументы *n* функций, изображаемых на одном графике;

*y<sub>1</sub>, y<sub>2</sub>, ..., y<sub>n</sub>* – функции, изображаемые на одном графике.

Функция  $plot(x,y)$  позволяет строить график при задании функции  $y = f(x)$  в аналитическом виде, в виде вектора или матрицы.

Ниже перечислены функции, которые используются при построении графиков:

*loglog* – использование логарифмического масштаба при построении функции;

*semilogx* – использование полулогарифмического масштаба при построении функции (*log* по оси  $x$ );

*semilogy* – использование полулогарифмического масштаба при построении функции (*log* по оси  $y$ );

*polar* – использование полярной системы координат при построении функции;

*mesh* – изображение графика функции трехмерной поверхности;

*contour* – изображение графика с контурными линиями – уровнями равных высот;

*bar* – изображение графика функции столбцовой гистограммы;

*stairstep* – изображение графика функции в виде ступенчатой линии.

Нижеперечисленные операторы используются для оформления графиков:

*text* – размещение заданного текста на графике;

*title* – оформление титульного заголовка;

*xlabel* – размещение текста по оси  $x$ ;

*ylabel* – размещение текста по оси  $y$ ;

*grid* – изображение масштабной сетки.

При формировании графических иллюстраций принято использовать следующие операторы:

*axis(<масштаб>)* – указание масштаба при построении осей графика;

*hold* – предоставляет возможность сохранения предыдущих построений;

*subplot(m,n,p)* – данная операция осуществляет разбиение исходного окна на меньшие окна, где  $m$  – количество окон по вертикали,  $n$  – по горизонтали,  $p$  – номер подокна.

В табл. 1 приведены стили графиков системы MatLab.

Стили графиков

Тип точек		Цвет линии		Тип линии	
.	Точка	Y	Желтый	-	Сплошная
O	Окружность	M	Фиолетовый		
X	Крест	C	Голубой	:	Двойной пунктир
+	Плюс	R	Красный		
*	Звездочка	G	Зеленый	-.	Штрихпунктирная
S	Квадрат	B	Синий		
D	Ромб	W	Белый	--	Штриховая
P	Пятиугольник	K	Черный		

При задании стиля символ  $s$  представляется в виде вектора, элементами которого являются: тип точки, цвет линии и тип линии, разделенные запятыми и выделенные одиночными кавычками.

Пример: `plot(x, y, ['R ', ' +', ' - '])`

Это график красного цвета, точки графика в виде плюса, линия сплошная.

Функция `plot(x1, y1, s1, x2, y2, s2, ..., xi, yi, si, ..., xn, yn, sn)` дает возможность разместить несколько функций на одном графике. Здесь приняты следующие обозначения:

$x_i$  – массив аргументов для  $i$ -й функции;

$y_i$  – массив значений для  $i$ -й функции;

$s_i$  – стиль графика для  $i$ -й функции.

В случае, если стиль для графика не указан, то MatLab сам установит стиль. Используя команду `hold on`, можно вывести все графики на один рисунок. Отменить этот режим можно командой `hold off`.

К построенному графику можно подключить так называемую *легенду* – пояснения в виде линий с текстовой информацией. Команда `legend('x1', 'x2', 'x3')` позволяет сформировать отрезки линий графиков с поясняющей текстовой информацией 'x<sub>1</sub>', 'x<sub>2</sub>', 'x<sub>3</sub>', размещаемой внутри графика или около него. Если на рисунке изображено три функции, то легенда будет расположена в правом верхнем углу графика.

Для того чтобы русский текст воспроизводился, необходимо задать код шрифта командой `set`. Команда `gca` позволяет выбрать шрифт для координатных осей.

В полярной системе координат выполнить построение графиков можно с помощью следующих функций:

$polar(Q, r);$

$polar(Q, r, s)$

где  $Q$  – угол функции  $r(Q)$ ,  $r$  – радиус,  $s$  – вектор стилей.

Для построения трехмерного графика функции  $z = f(x, y)$  используются матрицы значений переменных  $x$  и  $y$ . В этом случае применяют функции:

$[X, Y] = meshgrid(x, y)$

$[X, Y] = meshgrid(x)$

$[X, Y, Z] = meshgrid(x, y, z)$

Функция  $meshgrid(x, y)$  осуществляет преобразование областей векторов  $x$  и  $y$  в массивы значений  $X$  и  $Y$  соответственно. Эти массивы значений в дальнейшем используются при вычислении функции  $z = f(x, y)$ .

В результате вычисления функции  $[X, Y, Z] = meshgrid(x, y, z)$  получаем массив, который необходим для построения соответствующего графика.

Для построения графиков трехмерных поверхностей используются следующие функции:

$plot3(x, y, z)$

$plot3(X, Y, Z)$

$plot3(X, Y, Z, s)$

$plot3(x_1, y_1, z_1, s_1, x_2, y_2, z_2, s_2, \dots, x_n, y_n, z_n, s_n)$

где  $x, y, z$  представляют собой векторы аргументов функции,  $X, Y, Z$  – матрицы,  $s$  – указанные стили графика.

Все перечисленные функции строят точки на графике и соединяют их в соответствии с выбранным стилем.

**Задание 1.** Выполнить построение графиков функций:  $F1 = x$ ;  $F2 = \sin^2(x)$ ;  $F3 = \cos^2(x)$ ;  $F4 = \ln^2(x)$ .

Решение:

$>> f1=x;$

$>> f2=\sin(x).^2;$

```

>> f3=cos(x).^2;
>> f4=ln(x).^2;
>> Hold On
>> xlabel('x')
>> ylabel('y')
>> title('Функция')
>> plot(x,f1,['R','+','-'],x,f2,['G','+','-'],x,f3,['B','+','-'],x,f4,['C','+','-'])

```

На рис. 1 показано построение функций в одном окне.

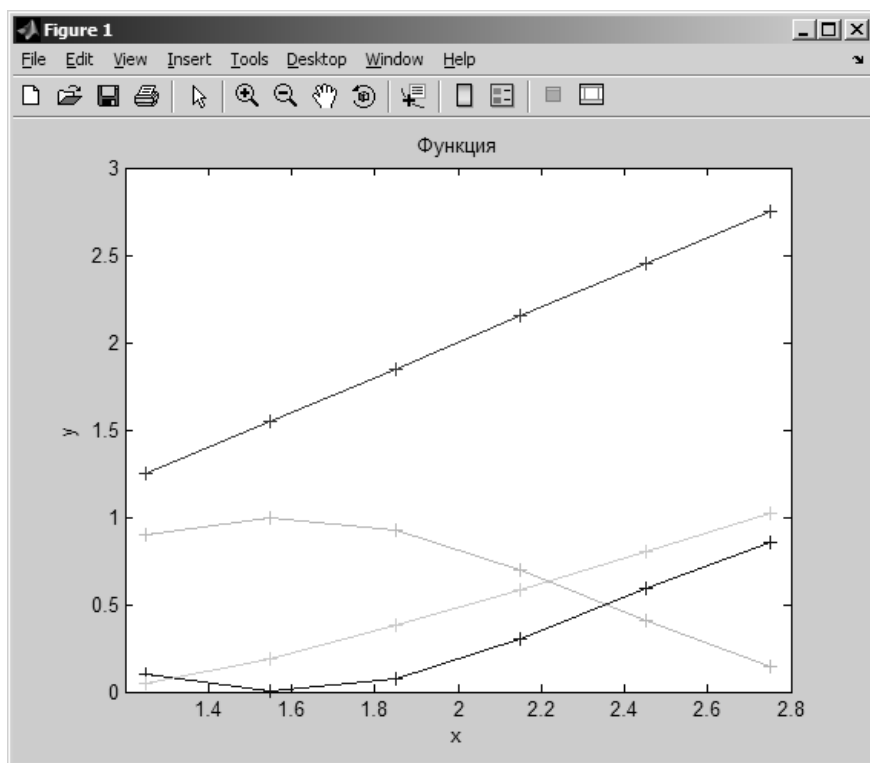


Рис. 1. Построение нескольких функций в одном окне

**Задание 2.** Выполнить построение графиков функций:  $F1 = x \cdot e^x$ ;  $F2 = \ln(x) + x^2$ ;  $F3 = \cos(x) + x^2$ ;  $F4 = \cos(x^2) \cdot |x|$ .

Решение:

```
>> x=0:0.1:10;  
>> f1=x.*exp(x);  
>> f2=log(x)+x.^2;  
>> f3=cos(x)+x.^2;  
>> f4=cos(x.^2).*abs(x);  
>> subplot(2,2,1),plot(x,f1),xlabel('x'),ylabel('y'),title('x*e^x');  
>> subplot(2,2,2),plot(x,f2),xlabel('x'),ylabel('y'),title('ln(x)+x^2');  
>> subplot(2,2,3),plot(x,f3),xlabel('x'),ylabel('y'),title('cos(x)+x^2');  
>> subplot(2,2,4),plot(x,f4),xlabel('x'),ylabel('y'),title('cos(x^2)*|x|');
```

На рис. 2 показано построение функций в нескольких окнах.

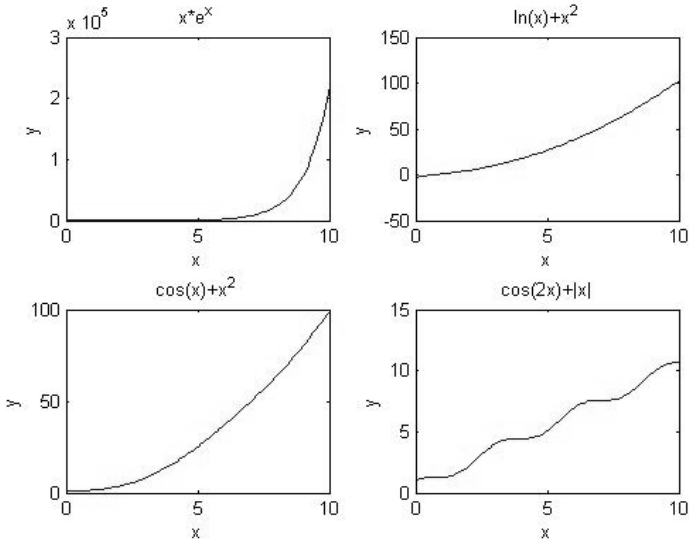


Рис. 2. Построение графиков функций в нескольких окнах

**Упражнение 1.** Построить графики функций.

Т а б л и ц а 2

**Функции**

№	Функции	№	Функции
1	$f1 = \ln(x) + e^x$ ; $f2 = \sin(x) + \cos(x)$ ; $f3 = x^2 + \ln(x)$ ; $f4 = \cos(x) + x^2$	7	$f1 = x * \sin(x)$ ; $f2 = x * \cos(x)$ ; $f3 = x^2$ ; $f4 = x * \ln(x)$
2	$f1 = x * e^x$ ; $f2 = \sin(x) + x$ ; $f3 = \sin(x) +  x $ ; $f4 = \sin(x) + x^2$	8	$f1 = \ln(x) + e^x$ ; $f2 = x * \cos(x)$ ; $f3 = x^2$ ; $f4 = \cos(x) + x^2$
3	$f1 = \sin(x)$ ; $f2 = \cos(x)$ ; $f3 = \ln(x)$ ; $f4 = x^2$	9	$f1 = \ln(x) + e^x$ ; $f2 = \sin(x) + x$ ; $f3 = \cos^2(x)$ ; $f4 = x^2$
4	$f1 = \cos(x) + e^x$ ; $f2 = \cos^2(x) + x$ ; $f3 = \cos(x^2) +  x $ ; $f4 = e^x + x^2$	10	$f1 = \sin(x)$ ; $f2 = \ln(x) + x^2$ ; $f3 = \cos^2(x)$ ; $f4 = e^x + x^2$
5	$f1 = \cos(x^2) * e^x$ ; $f2 = \cos(x^2) * x^2$ ; $f3 = \cos(x^2) * x$ ; $f4 = \cos(x^2) *  x $	11	$f1 = x$ ; $f2 = \cos(x)$ ; $f3 = \cos(x^2) * x$ ; $f4 =  x $
6	$f1 = \sin^2(x) + \cos^2(x)$ ; $f2 = \ln(x) * e^x$ ; $f3 = x^2 + \ln(x)$ ; $f4 = \cos(x) + x^2$	12	$f1 = x * e^x$ ; $f2 = \sin^2(x)$ ; $f3 = \cos(x) + x^2$ ; $f4 = \cos(x^2) *  x $

Выполнить построение графиков функций в нескольких окнах.

**Задание 3.** Выполнить построение сетчатой поверхности.

Решение:

$[x,y]=meshgrid(-15:0.2:15,-15:0.2:15);$

$R=sqrt(x.^2+y.^2)+0.001;$

$Z=5*sin(R)./R$

$meshc(Z), xlabel('X'), ylabel('Y'), zlabel('Z'), title('Z-поверхность')$



На рис. 3 показано построение сетчатой поверхности.

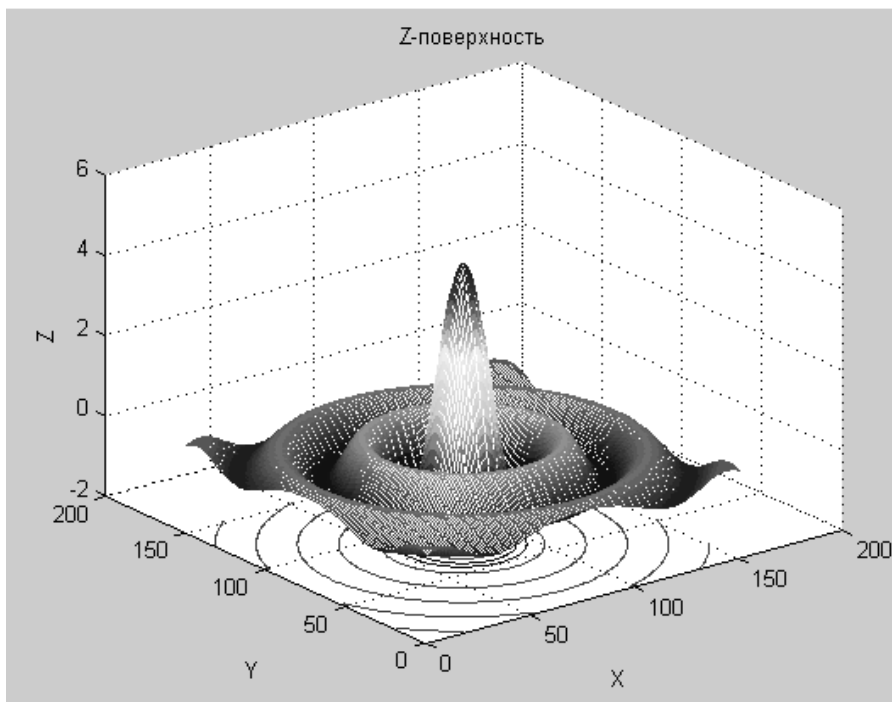


Рис. 3. Построение сетчатой поверхности

**Задание 4.** Формирование сферы.

Решение: `>> [x,y,z]=sphere(40);`

`>> x5=5*x;`

`>> y5=y*5;`

`>> z5=5*z;`

`>> plot3(x5,y5,z5),grid`

На рис. 4 показано построение сферы.

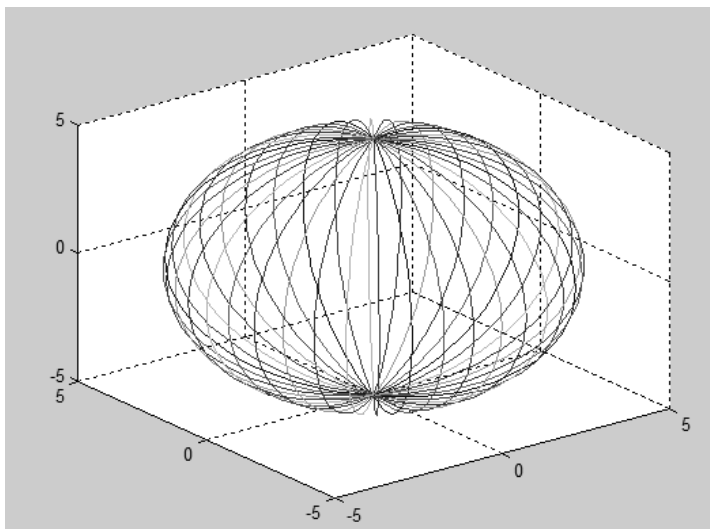


Рис. 4. Построение сферы

Для наглядного представления результатов работы необходимо их правильно оформлять.

#### 1.4. Решение систем обыкновенных дифференциальных уравнений

Для решения систем обыкновенных дифференциальных уравнений (ОДУ) в системе MatLab имеются функции *ode23*, *ode45*, *ode113*, *ode15s*, *ode23s*, *ode23t* и *ode23tb*.

Наиболее употребительной является функция *ode45*, реализующая алгоритм Рунге–Кутты 4–5-го порядка (разные порядки точности используются для контроля шага интегрирования).

Пусть необходимо решить систему  $n$  дифференциальных уравнений, разрешенных относительно первых производных функций  $y_1, y_2, \dots, y_n$ :

$$y_1' = F_1(t, y_1, y_2, \dots, y_n);$$

$$y_2' = F_2(t, y_1, y_2, \dots, y_n);$$

...

$$y_n' = F_n(t, y_1, y_2, \dots, y_n).$$

Введем вектор-столбцы  $Y$  и  $F$ , состоящие из  $y_1, y_2, \dots, y_n$  и  $F_1, F_2, \dots, F_n$ , соответственно. Тогда система дифференциальных уравнений примет следующий векторный вид:

$$Y' = F(t, Y).$$

Решатель обыкновенных дифференциальных уравнений обеспечивает возможность выбора метода, задания начальных условий и др.

$$[T, Y] = \text{solver}('F', [DT], Y0, \dots)$$

где  $DT$  – диапазон, содержащий начальное и конечное значение аргумента,  $Y0$  – вектор начальных значений переменных состояния,  $F$  – имя функции вычисления правых частей системы обыкновенных дифференциальных уравнений,  $\text{solver}$  – имя используемой функции (*ode45* – метод Рунге–Кутты 4 и 5-го порядков, *ode23* – тот же метод 2 и 3-го порядков, *ode113* – метод Адамса для нежестких систем, *ode23s* и *ode15s* – для жестких систем и др.). Версии решателя различаются используемыми методами и временем решения.

Под жесткостью принято понимать повышенное требование к точности – использование минимального шага во всей области интегрирования. При отсутствии информации о жесткости рекомендуется использовать решение с помощью *ode45* или *ode15s*.

Чтобы применить «решатель» *ode45*, нужно оформить в виде функции пользователя правую часть системы уравнений  $F(t, Y)$ .

Ниже приведены примеры использования М-файлов для решения систем обыкновенных дифференциальных уравнений.

**Задание 5.** Решить систему дифференциальных уравнений с заданными начальными условиями. Вывести графические результаты решения:

$$\dot{x}_1 = -x_1 + 6, \quad \dot{x}_2 = x_1 - 4x_2, \quad \dot{x}_3 = 2x_2 - 3x_3,$$

$$x_1(0) = 0, \quad x_2(0) = 10, \quad x_3(0) = 0.$$

Решение: определяем систему дифференциальных уравнений в файле *func1.m*:

$$\text{function } f = \text{func1}(t, a)$$

$$f = [-a(1) + 6;$$

$a(1)-4*a(2);$

$2*a(2)-3*a(3)];$

Далее определяем:

`>> x1=[0,10,0];`

`>> [T,Y]=ode45('func1',[0,2],x1);`

`>> plot(T,Y),title('Графики'),legend('x1','x2','x3'), xlabel('x'), ylabel('y'), grid on;`

На рис. 5 показано графическое решение системы дифференциальных уравнений.

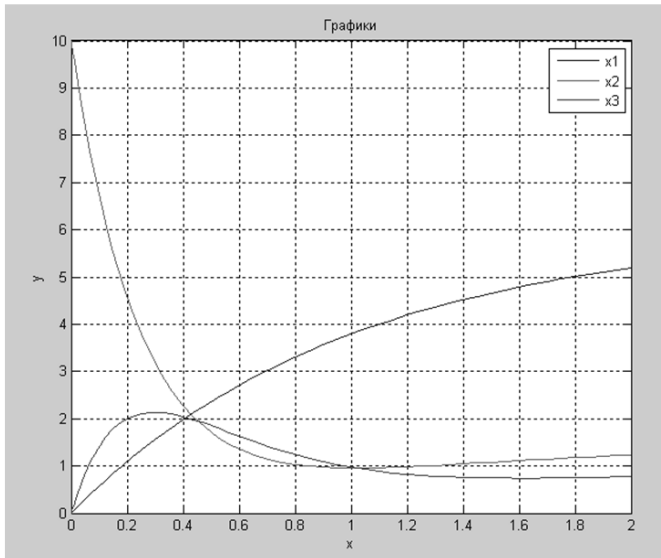


Рис. 5. Графическое решение системы дифференциальных уравнений

**Задание 6.** Решить систему дифференциальных уравнений с заданными начальными условиями:

$$y''-2y'+y=0 ; y(0)=1; y'(0)=-2$$

Вывести графические результаты решения.

Решение: *function dydt=vdp(t,y)*

*dydt=zeros(2,1);*

*dydt(1)=y(2);*

*dydt(2)=2\*y(2)-y(1);*

В основном файле имеем:

*[T,Y]=ode45(@vdp,[0 5],[0 -2]);*

*plot(T,Y);*

*hold on;*

*title('ode45');*

*legend('y1','y2');*

*xlabel('x');*

*ylabel('y');*

На рис. 6 показано построение графиков функций.

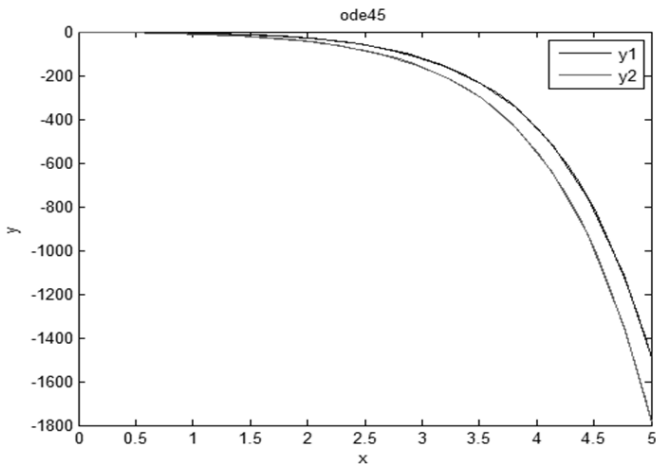


Рис. 6. Построение графиков функций

**Задание 7.** Решить систему дифференциальных уравнений, заданную в матричном виде. Вывести результаты решения в графическом виде.

$$x' = Ax; \quad A = \begin{pmatrix} -1 & -2 & 2 \\ -2 & -1 & 2 \\ -3 & -2 & 3 \end{pmatrix}, \text{ при начальных значениях } x_0 = [0, 2, 12].$$

```
Решение: function dy=vdp70000(t,y)
    A=[-1 -2 2; -2 -1 2;-3 -2 3]
    dy=zeros(2,1);
    for k=1:3;
        dy(k)=A(k,1)*y(1)+A(k,2)*y(2)+A(k,3)*y(3);
    end
```

В основном файле имеем:

```
[T,Y]=ode45(@vdp70000,[0 5],[0 2 12]);
plot(T,Y);
hold on;
title('ode15s');
legend('y1','y2','y3');
xlabel('x');
ylabel('y');
%gtext('y1'),gtext('y2');
```

На рис. 7 показано использование решателя *ode45*.

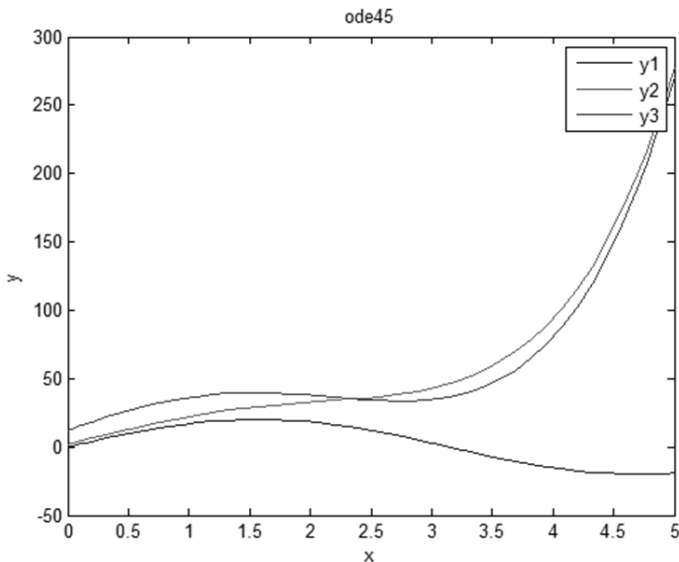


Рис. 7. Использование решателя *ode45*

**Упражнение 2.** Решить систему обыкновенных дифференциальных уравнений (табл. 3).

Таблица 3

### Система ОДУ

№	Система ОДУ	№	Система ОДУ
1	$\dot{x}_1 = -x_1 + 2, \quad \dot{x}_2 = 2x_1^2 - 0.5x_2,$ $x_1(0) = 10, \quad x_2(0) = 5$	6	$y'' - 4y' + 2y = 0,$ $y(0) = 4, \quad y'(0) = -3$
2	$\dot{x}_1 = -3x_1 + 10, \quad \dot{x}_2 = x_1 - 2x_2,$ $\dot{x}_3 = 4x_2 - x_3,$ $x_1(0) = 0, \quad x_2(0) = 0, \quad x_3(0) = 0$	7	$y'' - y' + 4y = 0,$ $y(0) = -1, \quad y'(0) = 0$
3	$\dot{x}_1 = -2x_1 + 5, \quad \dot{x}_2 = 2x_1 - 3x_2,$ $\dot{x}_3 = x_2 - 2x_3,$ $x_1(0) = 10, \quad x_2(0) = 5, \quad x_3(0) = 0$	8	$\dot{x}_1 = -3x_1 + 4, \quad \dot{x}_2 = 4x_1^2 - 3x_2,$ $x_1(0) = 2, \quad x_2(0) = 1$

№	Система ОДУ	№	Система ОДУ
4	$x' = Ax; A = \begin{pmatrix} 2 & -1 & -1 \\ 1 & 0 & -1 \\ 3 & -1 & -2 \end{pmatrix},$ $x_0 = [0, 1, 10]$	9	$x' = Ax; A = \begin{pmatrix} 2 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},$ $x_0 = [0, 10, 2]$
5	$x' = Ax; A = \begin{pmatrix} -2 & 1 & 2 \\ -1 & 0 & 2 \\ -2 & 0 & 3 \end{pmatrix},$ $x_0 = [0, 0.5, 1]$	10	$x' = Ax; A = \begin{pmatrix} 1 & -2 & 2 \\ 1 & 4 & -2 \\ 1 & 5 & -3 \end{pmatrix},$ $x_0 = [0, 1, 2]$

Реализовать вычисление системы обыкновенных дифференциальных уравнений в среде MatLab.

Наряду с решателями обыкновенных дифференциальных уравнений в среде MatLab можно оперировать и с дифференциальными алгебраическими уравнениями, которые так же, как и ОДУ, являются основой математического моделирования динамических линейных и нелинейных систем.

## 1.5. Вычисление интегралов в среде MatLab

Для вычисления интегралов методом трапеций в системе MatLab используется функция *trapz*:

$$integ = trapz(x, y);$$

В данном случае одномерный массив *x* содержит дискретные значения аргументов подынтегральной функции. В одномерном массиве *y* находятся значения подынтегральной функции в этих точках. Величина шага интегрирования определяет точность вычислений: чем меньше этот шаг, тем достигается большая точность вычислений интеграла.

В среде MatLab вычисление интегралов более высоких порядков точности реализуются функциями: *quad* (метод Симпсона) и *quad8* (метод Ньютона–Котеса 8-го порядка точности). У функции *quad8* более высокий порядок точности по сравнению с функцией *quad*, что очень хорошо для гладких функций, так как обеспечивается более высокая точность результата при большем шаге интегрирования (меньше



шем объеме вычислений). Функции *quad* и *quad8* могут принимать различное количество параметров. Минимальный формат вызова этих функций включает в себя три параметра:

*quad8(name, x1, x2)*

где *name* – имя подынтегральной функции, *x1* – нижний предел интегрирования, *x2* – верхний предел интегрирования. Четвертый параметр применяется в случае, если требуется относительная точность результатов вычислений.

Для вычисления двойных интегралов в среде MatLab применяется специальная функция *dblquad*:

*dblquad(name, x1, x2, x3, x4)*

где *x1* – нижний предел первого интеграла, *x2* – верхний предел первого интеграла, *x3* – нижний предел второго интеграла, *x4* – верхний предел второго интеграла.

**Упражнение 3.** Вычислить определенный интеграл (табл. 4).

Т а б л и ц а 4

### Определенный интеграл

№	Интеграл	№	Интеграл
1	$\int_1^e \frac{dx}{x\sqrt{1-(\ln x)^2}}$	6	$\int_0^1 \frac{\sqrt{x}dx}{4-x}$
2	$\int_{\frac{3}{4}}^0 \frac{3xdx}{\sqrt{(x+1)^3}}$	7	$\int_{-\pi}^{\pi} x \sin(x)dx$
3	$\int_0^3 \frac{\sqrt{x}dx}{1+x}$	8	$\int_0^4 \frac{dx}{\sqrt{x-3}}$
4	$\int_2^7 \frac{\sqrt{x+2}dx}{x}$	9	$\int_0^{\frac{\pi}{2}} \frac{dx}{3+2\cos x}$
5	$\int_{-8}^0 \frac{dx}{5-\sqrt[3]{x^2}}$	10	$\int_{-4}^1 \frac{xdx}{\sqrt{(5-x)^3}}$

Реализовать вычисление определенных интегралов в среде MatLab.

**Упражнение 4.** Вычислить двойной интеграл (табл. 5).

Т а б л и ц а 5

**Двойной интеграл**

№	Интеграл	№	Интеграл
1	$\int_0^1 \int_0^2 (y \sin(x) + \sin(x)) dx dy$	6	$\int_0^1 \int_0^2 \left( \frac{36x^2 + 9x}{\exp(y)} \right) dx dy$
2	$\int_0^1 \int_0^2 (\sin(y) - y \sin(x)) dx dy$	7	$\int_0^1 \int_0^2 x^2 \exp(x + \sin(y)) \cos(y) dx dy$
3	$\int_0^1 \int_0^2 \exp(\cos(y) - x) y^2 dx dy$	8	$\int_0^1 \int_0^2 (x^2 + \sqrt{(y + 3x)}) dx dy$
4	$\int_0^1 \int_0^2 (2x^2 + 7x + y^2) dx dy$	9	$\int_0^1 \int_0^2 (x^2 \sin(x) + 2 \cos(y)) dx dy$
5	$\int_0^1 \int_0^2 (x \sin(y) + y \sin(x)) dx dy$	10	$\int_0^1 \int_0^2 (\sin(x) + 2 \cos(y)) dx dy$

Реализовать вычисление двойных интегралов в среде MatLab.

## 1.6. Тест для самопроверки

1. Какая функция используется для построения двухмерных графиков?

1. *plot*

2. *ode45*

3. *trapz*

4. *plot3*

2. Какой символ определяет зеленый цвет линии?

1. Y
2. M
3. R
4. G

3. Какой символ указывает на штрихпунктирный тип линии?

1. --
2. :
3. -.
4. -

4. Какой оператор осуществляет сохранение предшествующих построений?

1. *hold*
2. *text*
3. *plot (x, y)*
4. *axis*

5. Какая команда позволяет осуществить разбивку окна на меньшие окна при построении графиков?

1. *plot (x, y, s)*
2. *subplot(m,n,p)*
3. *meshgrid(x, y, z)*
4. *plot3(x, y, z)*

6. Какая команда осуществляет построение графиков в полярной системе координат?

1. *stairs*
2. *contour*
3. *polar*
4. *polarstep*

7. Какая функция применяется при вычислении интегралов в среде MatLab?

1. *trapz(x, y)*
2. *plot (x, y)*
3. *plot3(x, y)*
4. *meshgrid(x, y)*

8. Какой оператор используется при построении графика функции в виде столбцовой гистограммы?

1. *bar*
2. *contour*
3. *stairs*
4. *plot*


9. Какую команду можно использовать для вывода информации в диалогом режиме?

1. *text*
2. *hold*
3. *plot*
4. *disp*

10. Какая функция осуществляет преобразование числового значения переменной в текстовую переменную?

1. *subplot*
2. *quad*
3. *stairs*
4. *num2str*

11. Установите соответствие между командой и ее действием (с помощью стрелки):

Команда		Действие
loglog		<i>Использование логарифмического масштаба при построении функции</i>
polar		Изображение графика функции столбцовой гистограммы
mesh		Изображение графика функции в виде ступенчатой линии
contour		Использование полярной системы координат при построении функции
bar		Изображение графика функции трехмерной поверхности
stairs		Изображение графика с контурными линиями – уровнями равных высот

## Контрольные вопросы и задания

1. Перечислите возможности графического представления информации в среде MatLab.
2. Каким образом можно задать стиль графика?
3. Как можно построить несколько функций на одном графике?
4. С какой целью используется команда *hold on*?
5. С помощью какой команды можно нанести координатную сетку?
6. Каким образом можно построить график в логарифмическом масштабе?
7. Как осуществляется построение графиков в полярной системе координат?
8. Какие функции используются для решения систем обыкновенных дифференциальных уравнений?
9. Какие типы М-файлов существуют?
10. Как определить тип точки, цвет линии и тип линии на графике?
11. Как используются локальные и глобальные переменные в среде MatLab?
12. Какие конструкции условных операторов используются в среде MatLab?
13. Поясните назначение аргументов функции *subplot(m,n,p)*.
14. Охарактеризуйте структуру файла-функции.
15. Каким образом можно задать надписи по осям координатной сетки?
16. Какие разновидности операторов цикла используются в среде MatLab?
17. С помощью каких функций можно осуществлять вычисление интегралов в среде MatLab?
18. От чего зависит точность вычисления интегралов?
19. С какого ключевого слова начинается описание файла-функции?
20. Как построить оси на графике с заданным масштабом?
21. Какие функции используются для создания трехмерных графиков?
22. С какой целью используется функция *disp*?
23. Поясните, что означает запись *plot(x, y, ['G ', ' \*', '-- '])*.
24. Перечислите особенности оформления текста программ на языке MatLab.
25. В каких случаях применяется функция *meshgrid*?

26. Какая команда дает возможность сформировать отрезки линий графиков с поясняющей текстовой информацией?
27. Какие логические операции используются в среде MatLab?
28. Перечислите отличия файла-функции от файла-сценария.
29. В каких случаях применяется функция *sprintf*?

## 2. Проектирование интерфейса пользователя

### 2.1. Компоненты графического интерфейса

Среда MatLab позволяет строить оконные Windows-приложения, где широко представлены средства управления отображением двухмерных и трехмерных объектов. Все компоненты, ориентированные на проектирование графического интерфейса, оформлены в соответствии со стандартами Windows. Набор свойств данных элементов соответствует основным функциональным возможностям, аналогичным компонентам, которые предлагаются современными системами визуального программирования. Перечень интерфейсных компонентов MatLab приведен в табл. 6.

Т а б л и ц а 6

**Перечень интерфейсных компонентов MatLab**

Наименование компоненты	Назначение
Push Button	Кнопка
Toggle Button	Кнопка, фиксирующаяся в утопленном состоянии
Radio Button	Переключатель – индикатор альтернативных вариантов
Checkbox	Окошко – индикатор неальтернативных вариантов
Edit Text	Поле для вывода, ввода и редактирования текста
Static Text	Область для вывода текста
Slider	Полоса прокрутки
Panel	Рамка, контейнер для интерфейсных компонентов
Button Group	Рамка для кнопок типа Radio Button и Toggle Button
Listbox	Список, окно для отображения массива строк
Popur Menu	Всплывающее меню

Каждый интерфейсный элемент обладает определенным набором свойств, который задает внешний вид и поведение компонента на каждом этапе выполнения программы. В среде MatLab любому интерфейсному элементу приписано 41 свойство, но некоторые из них имеют смысл только для определенных элементов. В качестве примера можно привести свойство *ListBoxTop*, которое определяет индекс отображаемого элемента списка. Это свойство имеет значение для объекта типа *Listbox*, а для других интерфейсных элементов данное свойство интереса не представляет. Аналогично можно отметить свойство *sliderstep*, которое используется объектами типа *slider*. Свойство *sliderstep* определяет минимальный и максимальный шаг перемещения ползунка.

При динамическом способе организации интерфейса с пользователем создаются те или иные графические объекты на каждой стадии выполнения программы, и в этом случае их свойствам присваиваются подходящие значения. Для создания любого интерфейсного элемента с определенными свойствами применяется функция *uicontrol*, которая возвращает указатель на формируемый компонент:

```
hUIC = uicontrol([hFig, ] 'Style', 'тип_компонента', ...
                'Свойство_1', Значение_1, ...
                'Свойство_2', Значение_2, ...
                ...
                'Свойство_k', Значение_k);
```

Следует отметить, что первый аргумент функции *uicontrol* не является обязательным. Если данный аргумент отсутствует, то владельцем создаваемого компонента будет текущий графический объект.

У уже существующего интерфейсного элемента можно изменить свойства с помощью функции *set*:

```
Set(UIC, 'Свойство_1', Значение_1, ...
      'Свойство_2', Значение_2, ...
      ...
      'Свойство_k', Значение_k)
```

С помощью функции *uicontrol*, в которой параметру *'style'* приписано значение *'pushbutton'*, можно выполнить создание кнопки *Push*



*Button*. Ниже представлена последовательность операторов для создания кнопки *Push Button*.

```
hfig=figure('Position', [200,200,300,100]);
```

```
hbtn=uicontrol('Style', 'pushbutton');
```

По умолчанию данная кнопка имеет серый цвет, находится в левом нижнем углу и не содержит никаких надписей. Всем свойствам данной кнопки присвоены значения по умолчанию.

Среда MatLab располагает возможностями для окрашивания поверхности кнопки в любой цвет и позволяет использовать любую цветовую гамму для символов, размещаемых на кнопке. В этом случае необходимо задать нужные значения свойств *BackgroundColor* и *ForegroundColor*.

Основное назначение командной кнопки состоит в вызове процедуры, реагирующей на соответствующее событие *callback*.

Событие, ассоциируемое со свойством *ButtonDownFcn*, может возникнуть в следующих ситуациях:

- при нажатии левой кнопки мыши в тот момент, когда курсор находится снаружи контура в пределах 5-пиксельной зоны кнопки;
- при нажатии левой кнопки мыши, когда курсор находится внутри контура кнопки.

При использовании события *ButtonDownFcn* не происходит изменений во внешнем виде кнопки. Если установить свойство *Enable* в состояние *off*, то есть запретить доступ, или в состояние *inactive*, то есть сделать кнопку неактивной, то таким образом можно заблокировать обработчик события *callback*. Для изменения размеров или положения кнопки можно воспользоваться обработчиком событий *ButtonDownFcn*.

Отличие кнопки типа *Toggle Button* от командной кнопки заключается в том, что она после первого нажатия фиксируется в утопленном состоянии и после следующего нажатия возвращается в начальное состояние. Такие кнопки могут использоваться на панелях инструментов. Как правило, на таких кнопках размещают небольшие изображения, которые изменяются в зависимости от состояния кнопки.

Для того чтобы разобраться в обработчиках событий, необходимо проанализировать значение свойства *value*. В случае, если значение данного свойства совпадает со значением *Max*, кнопка находится в утопленном состоянии. Совпадение свойства *value* со значением *Min* означает, что кнопка отжата.

В качестве рамок-контейнеров используются элементы типа *panel* и *Button Group*. Во внутренней области данных элементов группируются однотипные индикаторы альтернативных (*Radio Button*) или безальтернативных (*checkbox*) компонентов. Удобство таких элементов заключается в том, что они могут перемещаться по графическому окну. Все рамки-контейнеры могут быть снабжены пояснительными надписями.

Отличие кнопки типа *checkbox* (флажки) от переключателей заключается в том, что при выделении кнопки появляется галочка. Автоматическое выделение кнопки и связанное с ним событие *callback* происходит при щелчке на квадратике. По умолчанию значение свойства *Max* равно единице. Если кнопка типа *checkbox* выделена, то значение *value* совпадает со значением *Max*. При повторном нажатии на кнопку *checkbox* ее состояние автоматически изменяется на противоположное.

Объект типа *Static Text* дает возможность использовать поле отображения текстовой информации. Этот объект предназначен для вывода символьной строки (или нескольких строк) в выделенной области. Значение свойства *string* содержит отображаемый текст.

Объект типа *Edit Text* применяется как для вывода символьной информации, так и для ввода строк во время работы программы. Текст в поле ввода можно изменять. Буфер обмена используется для вставки и удаления редактируемого текста. После нажатия клавиши *<Enter>* процедура ввода генерирует событие *callback*.

Элемент типа *Listbox* представляет собой область, в которой могут находиться несколько строк. Если длина списка строк оказывается больше, чем высота окна, то для перемещения используется вертикальная полоса прокрутки.

Список, предоставляющий возможность выбрать единственную строку, характеризуется тем, что разность между значениями его свойств *Max* и *Min* равна единице. По умолчанию это условие выполнено, так как  $Max = 1$ , а  $Min = 0$ . Выбор строки осуществляется при нажатии левой кнопки мыши тогда, когда курсор указывает на выбираемую строку. Одновременно с подсветкой строки ее номер заносится в свойство *value* и генерируется событие *callback*. Строки в списке нумеруются, начиная с единицы.

Если разность между значениями *Max* и *Min* превышает единицу, то предоставляется возможность множественного выбора. Для того чтобы осуществить выбор нескольких разрозненных строк, необходимо при нажатой клавише *<Ctrl>* указать требуемое количество строк.

Каждая строка будет подсвечиваться, а ее номер будет сохраняться в векторе *value*.

Для того чтобы осуществить выбор нескольких строк, идущих друг за другом, необходимо при нажатой клавише *<Shift>* указать первую и последнюю строку в группе. Все строки, находящиеся между первой выделенной строкой и последней выделенной строкой, будут также выделены, а их номера будут сохранены в векторе *value*. При одновременном выделении нескольких подряд идущих строк событие *callback* возникает в момент нажатия клавиши мыши. Обработка выделенных строк может быть выполнена с помощью соответствующей кнопки. Событие *callback* позволит определить компоненты вектора *value*.

Элемент типа *Slider* определяет полосу прокрутки, которая используется в качестве средства ввода значений из диапазона  $[Min, Max]$ .

В зависимости от соотношения длины  $w$  и высоты  $h$ , задаваемых в качестве значения свойства  $Position = [x \ y \ w \ h]$ , полоса прокрутки принимает либо горизонтальное положение (для случая, если значение  $w$  больше значения  $h$ ), либо вертикальное положение (для случая, если значение  $w$  меньше значения  $h$ ). Точка с координатами  $(x, y)$  определяет левый нижний угол полосы.

Смещение ползунка изменяет значение свойства *value* в диапазоне от минимальной величины до максимального значения. По умолчанию  $Min = 0$ , а  $Max = 1$ .

Для изменения положения ползунка программным путем достаточно изменить значение свойства *value*, например, с помощью функции *set*.

## 2.2. Проектирование интерфейса в среде GUIDE

Если провести сравнение между возможностями, которые предоставляют различные среды визуального программирования, и возможностями, которые предоставляет динамический способ проектирования интерфейса среды MatLab, то можно выделить ряд недостатков.

Один из них заключается в том, что программа приложения значительно увеличивается за счет операторов, которые создают элементы интерфейса. В визуальных средах этого не происходит. Все элементы интерфейса встраиваются в приложение с помощью палитры компонентов. Настройка свойств осуществляется с помощью редактора.

Следующее отличие состоит в том, что в визуальной среде достаточно просто задавать положение и размеры элементов интерфейса на форме.

Следует отметить, что форматы вызова обработчиков различных событий в визуальных средах унифицированы. Визуальной средой автоматически формируется оболочка для каждого обработчика событий. Тело процедуры заполняется уже пользователем. Перечень сервисных средств, упрощающих проектирование интерфейса приложения, приведен в табл. 7.

Таблица 7

**Перечень сервисных средств MatLab**

Наименование	Назначение
GUIDE	Конструктор графического интерфейса
Property Inspector	Инспектор свойств
Object Browser	Просмотр объектов
M-file Editor	Редактор М-файлов
Component Callbacks	Создание функций обработки событий для компонентов
Figure Callbacks	Создание функций обработки событий для окон
Align Objects	Выравнивание объектов
Grid and Rulers	Управление сеткой и линейками
Menu Editor	Создание и редактирование меню
Tab Order Editor	Изменение порядка активизации компонентов при нажатии клавиши <Tab>

Как и любой процесс проектирования, процесс построения графического интерфейса пользователя можно разбить на следующие этапы: постановка задачи, создание формы интерфейса и создание на ней элементов управления, написание кода программы и кода обработки событий.

На первом этапе проводится анализ поставленной задачи и определяется количество и состав элементов управления.

На втором этапе создается форма графического интерфейса, на ней создаются и размещаются элементы управления. Здесь же описываются их свойства.

На третьем этапе создания графического интерфейса пользователя пишется код основной программы вычисления и код для обработки событий. Код основной программы вычисления пишется на языке программирования операционной среды MatLab в виде М-файлов. Созданные М-файлы закрепляются за событием какого-нибудь элемента управления или формы.

### *Вызов редактора GUIDE*

Редактор GUIDE запускается из командного окна с помощью команды *guide*. После запуска редактора появляется окно, содержащее две вкладки (рис. 8). Вкладка *<create new GUI>* дает возможность начать проектирование нового интерфейса. Вкладкой *<open existing GUI>* следует пользоваться в том случае, если интерфейс ранее был уже создан.

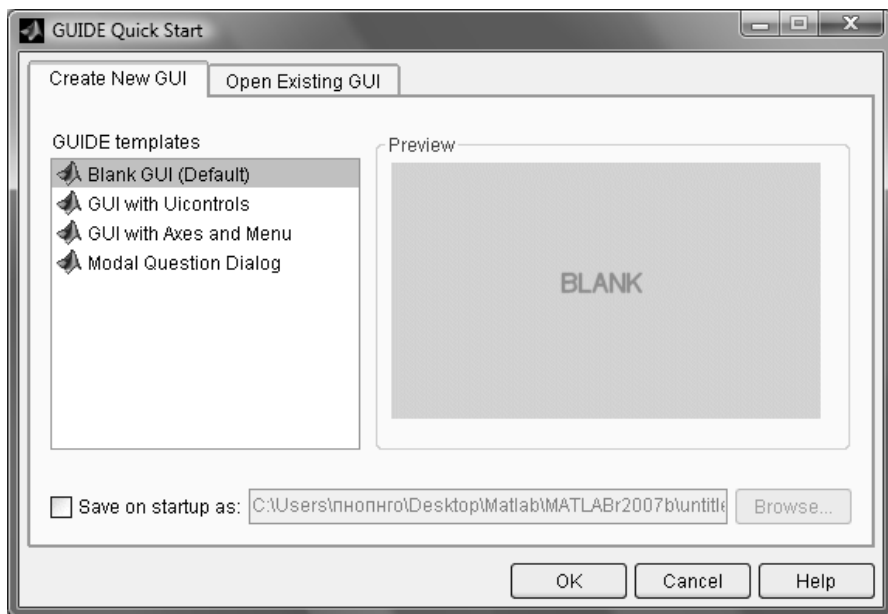


Рис. 8. Стартовое окно

Список шаблонов (*GUIDE template*) предусматривает следующие варианты компоновки, которые могут находиться в основе разрабатываемого приложения:

1) *Blank GUI* (шаблон интерфейса) – используется в случае, когда приложение разрабатывается с нуля;

2) *GUI with Uicontrols* (типовой интерфейс) – этот выбор предусматривает использование готового приложения в качестве прототипа;

3) *GUI with Axes and Menu* (интерфейс с координатными осями и меню) – этот вариант предполагает использование приложения, которое воспроизводит график одной из шести функций. Эти функции могут быть выбраны из раскрывающегося списка;

4) *Modal Question Dialog* (модальное диалоговое окно) – в этом случае в качестве прототипа можно использовать приложение, содержащее диалоговое окно с двумя вариантами ответа.

#### *Управление конструктором графического интерфейса*




На рабочем поле конструктора нанесена вспомогательная сетка, позволяющая осуществлять выравнивание элементов графического интерфейса по узлам и линиям.







В разделе *Tools* (сервис) находятся команды, предусматривающие вызов инструментов построения графического интерфейса, например: *Align Objects*, *Grid and Rules*, *Menu Editor*, *Tab Order Options*.

В табл. 8 перечислены некоторые элементы управления, используемые при построении графического интерфейса приложения.

Таблица 8

#### Элементы управления

Элемент	Назначение
	Режим выделения объекта
	Режим создания кнопки
	Режим создания переключателя

Элемент	Назначение
	Режим создания флажка
	Режим создания поля для ввода текстовой информации
	Режим создания поля для ввода текстовой информации без возможности редактирования
	Режим создания списка
	Режим создания раскрывающегося списка
	Режим создания поля для вывода информации в виде графиков и диаграмм

### *Инспектор свойств (Property Inspector)*

Инспектор свойств вызывается по команде *View/Property Inspector*. Список свойств выделенного объекта представлен в левой половине окна, а в правой половине окна находятся их значения.

Инспектор свойств на стадии проектирования приложения в среде GUIDE выполняет те же действия, которые осуществляют функции *get* и *set* при выполнении работы приложения. При необходимости текущие значения свойств объектов заменяются новыми значениями.

### *Просмотр объектов (Object Browser)*

Вызов браузера объектов, расположенных на форме, осуществляется командой *View/Object Browser*.

Все объекты отображаются в порядке их размещения на форме. Ветви дерева объектов подчиняются взаимоотношениям между объектами-родителями и объектами-потомками.

### Создание меню (Menu Editor)

В нижней части редактора меню находятся две вкладки, которые позволяют создавать либо главное меню, либо всплывающие меню.

Для создания нового раздела меню используется кнопка *New Menu*. А для создания новой команды меню – кнопка *New Menu Item*.

В качестве примера построим графический интерфейс для вывода графика функции в заданном интервале.

Для выбора и размещения объектов будем использовать панель инструментов, изображенную на рис. 9. На рис. 10 показан инспектор свойств, позволяющий осуществлять редактирование значения свойств выделенного объекта.



Рис. 9. Панель элементов управления

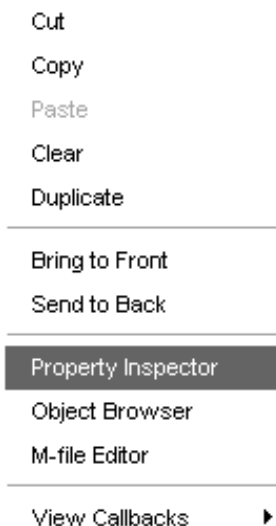


Рис. 10. Инспектор свойств


Инспектор свойств может быть вызван также с помощью кнопки . Создадим две области. Одна область предназначена для ввода выражения, а другая область – для ввода границ интервала вычисления выражения.





Рис. 11. Свойства Tag и String

Для первой области (области ввода функции) свойству *Tag* присвоим значение *ede*, а для второй области (области ввода заданного интервала) – значение *edi*. Для осей графика свойству *Tag* присвоим значение *ay*. Значения свойства *String* необходимо очистить (рис. 11).


Создаем кнопку с помощью элемента управления – кнопка . Список свойств этого элемента появляется при двойном нажатии кнопки, созданной в выбранной области графического интерфейса. Свойству *Tag* присвоим значение *Bp*, а свойству *String* – значение *Plot*. Для построения графика необходимо выполнение ряда команд. При нажатии правой кнопки мыши в контекстном меню выбираем команду *Callback* (рис. 12).



Рис. 12. Свойство Callback

Открывается содержимое М-файла, где надо найти свойство *Callback*, возникающее при нажатии кнопки *Bp*.

Укажем следующую последовательность команд:

```
cla
```

```
interval=str2num(get(handles.edi,'String'));
```

```
f=inline(get(handles.ede,'String'));
```


```
[x,y]=fplot(f,interval);
```

```
handles.line=plot(x,y,'b-');
```

```
handles.line=plot(x,y,'Color', My_color);
```

```
guidata(gcbo,handles);
```

```
hold on
```

В данном случае функция *get()* использует конкретное свойство заданного объекта. Функция *str2num()* выполняет преобразование строкового значения в числовое значение. Полученное значение используется переменной *interval*. Функция *inline()* преобразует строку, содержащую выражение, в функцию. Функция *fplot(f,interval)* осуществляет построение графика функции *f* на заданном интервале. Запуск программы осуществляется с помощью кнопки . На рис. 13 приведен график функции  $20 \cdot \sin(x) / x$  на интервале от  $-30$  до  $30$ .

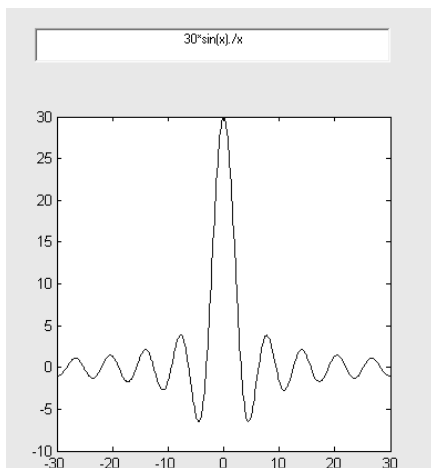


Рис. 13. График функции

**Задание 8.** Написать программу вычисления стоимости покупки с учетом скидки. Скидка 3 % предоставляется в том случае, если сумма покупки больше 2000 руб., скидка 5 % – если сумма покупки больше 3000 руб., 10 % – если сумма покупки больше 5000 руб., студентам скидка 7 % при любой покупке, пенсионерам скидка 15 % при любой покупке. Если покупка осуществляется в воскресенье с 8 до 9 часов утра, то предоставляется скидка 20 % для всех покупателей. Построить графический интерфейс.

Решение. Для разработки программы можно использовать встроенные в Matlab средства создания GUI.

Пользователю предложено ввести стоимость покупки без скидки. В данном примере эта возможность реализована с помощью элемента *Edit Text* (*tag=startCost*, *string=0*).

Выбор дня покупки реализован с помощью элемента *PopUp Menu* (*tag=day*), так как названия и количество дней фиксировано.

Ввод значений времени реализован с помощью двух полей типа *Edit Text* (*tag1=hours*, *string1=0*; *tag2=minutes*, *string2=0*). Элемент типа *Edit Text* более удобен в данном случае, так как избавляет от больших раскрывающихся списков.

Состояние, к какой категории относится покупатель, реализовано с помощью элемента типа *Panel* (*tag=uipanel1*) и трех элементов *Radio Button*. В этом случае установлены следующие свойства: *tag1=statusStudent*, *string1=Студент*; *tag2=statusPensioner*, *string2=Пенсионер*; *tag3=statusNone*, *string3=Не является студентом или пенсионером*. По умолчанию устанавливается параметр «Не является студентом или пенсионером» как наиболее часто используемый.

После нажатия на элемент *Push Button* (свойство *tag=calculate*, *string=Рассчитать стоимость с учетом скидки*) с надписью «Рассчитать стоимость с учетом скидки» происходит расчет и вывод итоговой стоимости с учетом выбранных пользователем условий в поле *Edit Text* (свойство *tag=result*, *string=Итоговая стоимость*), которое использовано для возможности скопировать результат в буфер обмена.

Также происходит вывод процента скидки и количества сэкономленных рублей в поля типа *Static Text* (*tag1=percent*, *string=-*; *tag2=rub*, *string=-*), так как данная информация является вспомогательной.

Выбор нужного процента скидки реализован с помощью вложенных операторов условия. Сначала происходит проверка на соответствие времени «воскресенье с 8 до 9 часов», так как в это время действует скидка, исключаяющая все остальные. Затем происходит провер-

ка следующего условия: является ли покупатель студентом или пенсионером. И если не выполнилось ни одно из предыдущих условий, то происходит расчет скидки на основе величины суммы покупки.

*if(hours==8 && day==7) % проверка, соответствует ли время воскресенью с 8 до 9 часов*

*set(handles.result,'String', num\*0.8); % сумма с учетом скидки*

*set(handles.percent,'String', 20); % значение скидки в процентах*

*set(handles.rub,'String', num-num\*0.8); % сумма сэкономленных средств*

*else*

*if(get(handles.statusStudent,'Value')==1) % проверка, является ли покупатель студентом*

*set(handles.result,'String', num\*0.93); % вычисление суммы с учетом скидки*

*set(handles.percent,'String', 7); % значение скидки в процентах*

*set(handles.rub,'String', num-num\*0.93); % скидка*

*end*

*if(get(handles.statusPensioner,'Value')==1) % проверка, является ли покупатель пенсионером*

*set(handles.result,'String', num\*0.85); % сумма с учетом скидки*

*set(handles.percent,'String', 15); % значение скидки в процентах*

*set(handles.rub,'String', num-num\*0.85); % скидка*

*end*

*if(get(handles.statusNone,'Value')==1) % скидка относительно стоимости покупки*

```

if(num>2000) % если сумма покупки больше 2000, но меньше 3000
    set(handles.result,'String', num*0.97); % сумма с учетом скидки
    set(handles.percent,'String', 3); % значение скидки в процентах
    set(handles.rub,'String', num-num*0.97); % скидка
if(num>3000) %, если сумма покупки больше 3000, но меньше 5000
    set(handles.result,'String', num*0.95); % сумма с учетом скидки
    set(handles.percent,'String', 5); % значение скидки в процентах
    set(handles.rub,'String', num-num*0.95);% скидка
if(num>5000) %, если сумма покупки больше 5000
    set(handles.result,'String', num*0.9) % сумма с учетом скидки
    set(handles.percent,'String', 10); % значение скидки в процентах
    set(handles.rub,'String', num-num*0.9); % скидка
end
end
else % если не выполнилось ни одно из предыдущих условий и сумма
меньше либо равна 2000, то скидка отсутствует
    set(handles.result,'String', num); % исходная сумма при отсутствии
скидки
    set(handles.percent,'String', 0); % нулевой процент скидки при ее
отсутствии
    set(handles.rub,'String', 0); % нулевая скидка
end
end
end
end

```

На рис. 14 приведен интерфейс разработанной программы.

The image shows a MATLAB GUI window with a light gray background. At the top, there is a label 'Стоимость покупки без скидки' followed by a text box containing '3750' and the word 'рублей'. Below this is a label 'День покупки' followed by a dropdown menu showing 'Четверг'. Underneath is a label 'Время покупки' followed by two text boxes: the first contains '17' and is followed by the word 'часов', and the second contains '35' and is followed by the word 'минут'. Below these is a group box containing three radio buttons: 'Студент', 'Пенсионер' (which is selected), and 'Не является студентом или пенсионером'. Below the group box is a button labeled 'Рассчитать стоимость с учётом скидки'. Below the button is a large text box displaying the result '3187.5'. At the bottom, there is a label 'Скидка составляет' followed by '15', a '%' symbol, the word 'или', '562.5', and 'рублей'.

Рис. 14. Расчет стоимости покупки

Аналогично можно осуществить построение интерфейса пользователя с использованием других, встроенных в Matlab средств создания GUI.

**Упражнение 5.** Пусть заданы окружности, координаты центров которых содержатся в массивах  $x$  и  $y$ . Значения радиусов находятся в массиве  $r$ . Известны координаты некоторой точки. Требуется вывести график, на котором маркером отмечено положение заданной точки. Окружности, внутри которых лежит точка, изображаются синим цветом, а все остальные окружности – красным цветом. Написать программу и построить графический интерфейс.

**Упражнение 6.** Построить графический интерфейс пользователя для нахождения корней квадратного уравнения вида  $ax^2 + bx + c$ . На форме должны быть поля для ввода значений  $a$ ,  $b$  и  $c$ .

**Упражнение 7.** Написать программу, которая рисует образец мишени с окружностью красного цвета в центре. Далее чередуются коль-

ца белого и черного цвета (по шесть колец). Построить графический интерфейс.

### 2.3. Тест для самопроверки

1. Может ли файл-функция иметь входные аргументы?
  1. Да
  2. Нет
  
2. Для чего используется символ «;» в конце строки?
  1. Конец команды
  2. Не выводить результат набранной команды
  3. Набранная команда имеет продолжение на следующей строке
  4. Это комментарий
  
3. Какая функция используется для создания интерфейсного компонента?
  1. *Component()*
  2. *Uicontrol()*
  3. *Set()*
  4. *Interface()*
  
4. С какого символа должны начинаться имена переменных?
  1. С цифры
  2. С символа подчеркивания
  3. С буквы
  4. С символа %
  
5. Как называется текущий документ, в котором пользователь вводит команды, а система выводит результат их исполнения?
  1. Дневник
  2. Сессия
  3. Журнал
  4. Файл
  
6. Как называется функция для формирования многомерного массива?
  1. *Mas*
  2. *Set*

3. *Cat*

4. *Len*

7. Если функция имеет более одного аргумента, то список выходных аргументов размещается в...

1. круглые скобки

2. квадратные скобки

8. Переменные, появляющиеся в теле функции, являются...

1. глобальными переменными

2. локальными переменными

9. Для чего используется функция `plot(x, y)`?

1. Для вывода значений матрицы

2. Для построения графика

3. Для создания кнопки размером  $x$  на  $y$

4. Для создания интерфейса

10. Какая функция используется для нахождения среднего значения элементов в массиве?

1. *Median*

2. *Std*

3. *Mean*

4. *Avg*

11. Имеет ли файл-сценарий входные аргументы?

1. Да

2. Нет

12. Как называется переменная, хранящая результат последней операции?

1. *otvet*

2. *ans*

3. *result*

4. *end*

13. Что произойдет, если запись оператора не будет заканчиваться символом «;»?

1. Появится сообщение об ошибке

2. Результат будет выведен в файл



3. Результат будет выведен в командном окне
4. В командном окне ничего не появится

14. Для изменения свойств в существующем интерфейсном компоненте используется команда...

1. *new*
2. *set*
3. *exchange*
4. *modify*

15. В чем заключается основное назначение *Property Inspector* на стадии проектирования приложения в среде GUIDE?

1. Добавлять новые компоненты интерфейса
2. Создавать меню приложения
3. Просматривать и изменять текущие значения свойств
4. Удалять компоненты интерфейса

16. Какое поле отображает имя указателя на объект?

1. *Callback*
2. *Tag*
3. *Label*
4. *Name*

17. В каком поле содержится название раздела меню?

1. *Name*
2. *Label*
3. *Title*
4. *Callback*

18. С помощью какой команды можно определить элемент *Инспектор свойств объекта*?

1. *Property Inspector*
2. *Figure Callbacks*
3. *Component Callbacks*
4. *Object Browser*

19. С помощью какого интерфейсного компонента можно определить полосу прокрутки?

1. *Listbox*
2. *Slider*

3. *Edit Text*

4. *Checkbox*

20. Какие свойства используются для окрашивания поверхности кнопки?

1. *Style*

2. *Toggle Button*

3. *Backgroundcolor*

4. *ButtonDownFcn*

21. Какое сервисное средство предоставляет возможность управлять сеткой и линейками?







1. *Object Browser*

2. *Grid and Rulers*





3. *Menu Editor*

4. *M-file Editor*


22. Установите соответствие между изображением кнопки и ее назначением (с помощью стрелки):

Кнопка		Назначение
		<i>Режим выделения объекта</i>
		Режим создания раскрывающегося списка
		Режим создания поля для вывода информации в виде графиков и диаграмм
		Режим создания кнопки
		Режим создания переключателя

Окончание таблицы

Кнопка		Назначение
		Режим создания флажка
		Режим создания поля для ввода текстовой информации
		Режим создания поля для ввода текстовой информации без возможности редактирования
		Режим создания списка

23. Установите соответствие между названием команды и ее результатом (с помощью стрелки):

Команда		Действие
GUIDE		<i>Конструктор графического интерфейса</i>
Property Inspector		Управление сеткой и линейками
Object Browser		Создание и редактирование меню
M-file Editor		Инспектор свойств
Component Callbacks		Просмотр объектов
Figure Callbacks		Редактор М-файлов
Align Objects		Создание функций обработки событий для компонентов
Grid and Rulers		Создание функций обработки событий для окон
Menu Editor		Выравнивание объектов

## Контрольные вопросы и задания

1. Перечислите компоненты, используемые при построении графического интерфейса.
2. С какой целью используется функция *uicontrol* при построении графического интерфейса?
3. Каким образом можно изменить свойства у существующего интерфейсного объекта?
4. Каким образом можно создать командную кнопку типа *Push Button*?
5. Что позволяют задать свойства *BackgroundColor* и *ForegroundColor*?
6. Чем отличается кнопка типа *Toggle Button* от командной кнопки?
7. В каком качестве используются компоненты типа *panel* и *Button Group*?
8. Каким образом используются кнопки типа *checkbox*?
9. Какой объект используется для вывода символьной строки (или нескольких строк) в выделенном прямоугольнике?
10. В каких случаях может использоваться интерфейсный элемент типа *Edit Text*?
11. Что собой представляет интерфейсный компонент типа *Listbox*?
12. Каким образом можно использовать интерфейсный компонент типа *Slider*?
13. Как установить полосу прокрутки в горизонтальном или вертикальном положении?
14. В каких случаях происходит вызов события *callback*?
15. Перечислите сервисные средства среды MatLab, упрощающие проектирование графического интерфейса приложения.
16. Какие варианты компоновки, которые можно положить в основу проектируемого приложения, предлагает список шаблонов *GUIDE template*?
17. Каким образом осуществляется управление конструктором графического интерфейса?
18. В чем заключается основное назначение *Property Inspector*?
19. Каким образом можно осуществить выравнивание объектов на форме?
20. Какое преобразование выполняет функция *str2num()*?
21. Какая функция выполняет преобразование строкового значения в числовое значение?

### **3. Установившийся режим в задаче оценивания параметров динамических объектов**

#### **3.1. Формирование матрицы Фишера в установившемся режиме**

Рассматривается динамический объект в следующей форме:

$$x_{k+1} = \Phi x_k + \Psi u_k + \Gamma w_k,$$

$$y_{k+1} = H x_{k+1} + v_{k+1},$$

где  $x_{k+1}$  – вектор состояния;  $u_k$  – вектор управления;  $w_k$  – вектор возмущения;  $y_{k+1}$  – вектор наблюдения,  $v_{k+1}$  – вектор ошибки измерения,  $\Phi$  – матрица состояния;  $\Gamma$  – матрица возмущения;  $\Psi$  – матрица управления;  $H$  – матрица наблюдения. При этом обозначим через  $Q$  неотрицательно определенную ковариационную матрицу вектора возмущения; через  $R$  – положительно определенную ковариационную матрицу вектора ошибки измерения. Предполагается, что все переходные процессы закончились, то есть рассматривается установившийся режим.

В задаче активной идентификации при оценивании параметров линейных динамических дискретных объектов во многих работах используется фильтр Калмана с обновленной последовательностью и информационная матрица Фишера [25–28]. В данном разделе рассматривается установившийся режим для вычисления фильтра Калмана и матрицы Фишера в среде Simulink.

Приведем процедуру вычислений оценок параметров динамических дискретных объектов для случая, когда неизвестные параметры находятся в матрице состояния. Вывод формул, используемых в данном разделе, подробно изложен в работе [29]. В среде Simulink построена модель в виде блоков для вычисления оценок параметров динами-

ческих объектов на основе метода наименьших квадратов. Каждый блок соответствует определенной группе формул. Ниже дан набор формул, необходимый для проведения расчетов.

Первая группа:

$$P_{k+1,k} = \Phi P_{k,k} \Phi^T + \Gamma Q \Gamma^T, \quad (1)$$

$$\Sigma_{k+1} = \left( H P_{k+1,k} H^T + R \right)^{1/2}, \quad (2)$$

$$K1_{k+1} = P_{k+1,k} H^T \Sigma_{k+1}^{-1}, \quad (3)$$

$$P_{k+1,k+1} = \left( I - K1_{k+1} \Sigma_{k+1}^{-1} H \right) P_{k+1,k}. \quad (4)$$

Первая группа формул соответствует вычислению фильтра Калмана с обновленной последовательностью в установившемся режиме [31]. На рис. 15 приведены результаты построения уравнения (1). Матрицы  $\Phi$ ,  $\Gamma$ ,  $Q$  являются входными данными для уравнения (1).

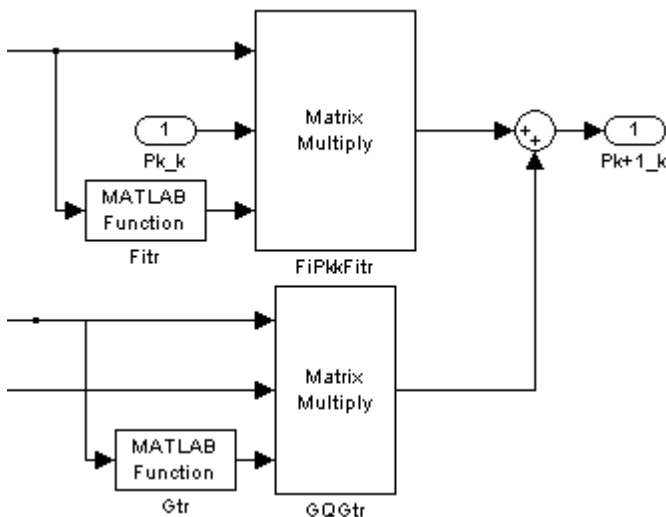


Рис. 15. Вычисление по уравнению (1)

На рис. 16 приведены результаты формирования уравнения (2). Матрицы  $H$  и  $R$  являются входными данными для уравнения (2).

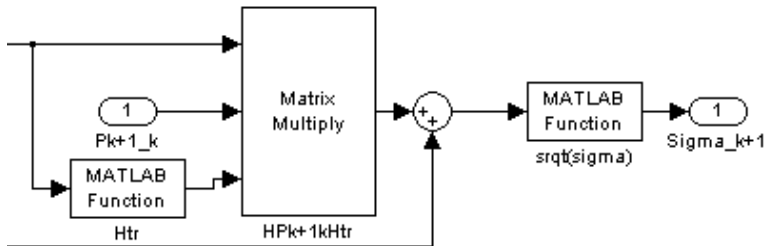


Рис. 16. Вычисление по уравнению (2)

На рис. 17 даны результаты моделирования уравнения (3).

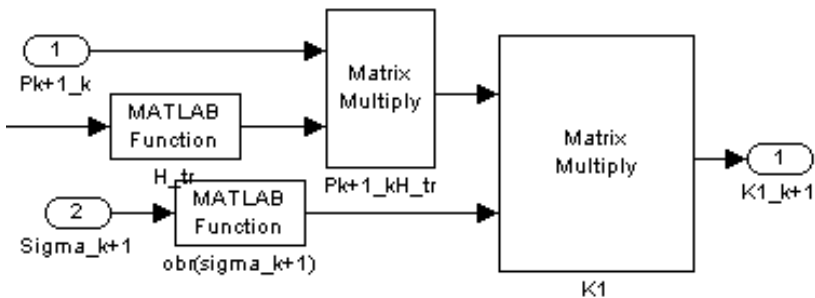


Рис. 17. Вычисление по уравнению (3)

Схема вычисления уравнения (4) показана на рис. 18.  
Уравнения второй группы формул:

$$\frac{\partial P_1}{\partial \theta_i} = \frac{\partial \Phi}{\partial \theta_i} P_0 \Phi^T + \Phi \left( \frac{\partial P_0}{\partial \theta_i} \Phi^T + P_0 \frac{\partial \Phi^T}{\partial \theta_i} \right), \quad (5)$$

$$\frac{\partial \Sigma_\infty^{-1}}{\partial \theta_i} = \left( -\frac{1}{2} \right) \Sigma_\infty^{-3} H \frac{\partial P_1}{\partial \theta_i} H^T, \quad (6)$$

$$\frac{\partial K1_\infty}{\partial \theta_i} = \frac{\partial P_1}{\partial \theta_i} H^T \Sigma_\infty^{-1} + P_1 H^T \frac{\partial \Sigma_\infty^{-1}}{\partial \theta_i}, \quad (7)$$

$$\frac{\partial P_0}{\partial \theta_i} = \left( -\frac{\partial K1_\infty}{\partial \theta_i} \Sigma_\infty^{-1} H - K1_\infty \frac{\partial \Sigma_\infty^{-1}}{\partial \theta_i} H \right) P_1 + \left( I - K1_\infty \Sigma_\infty^{-1} H \right) \frac{\partial P_1}{\partial \theta_i}, \quad (8)$$

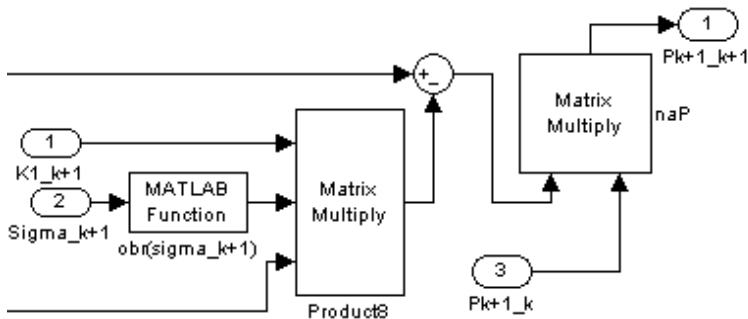


Рис. 18. Вычисление по уравнению (4)

На рис. 19 приведена схема моделирования уравнения (5).

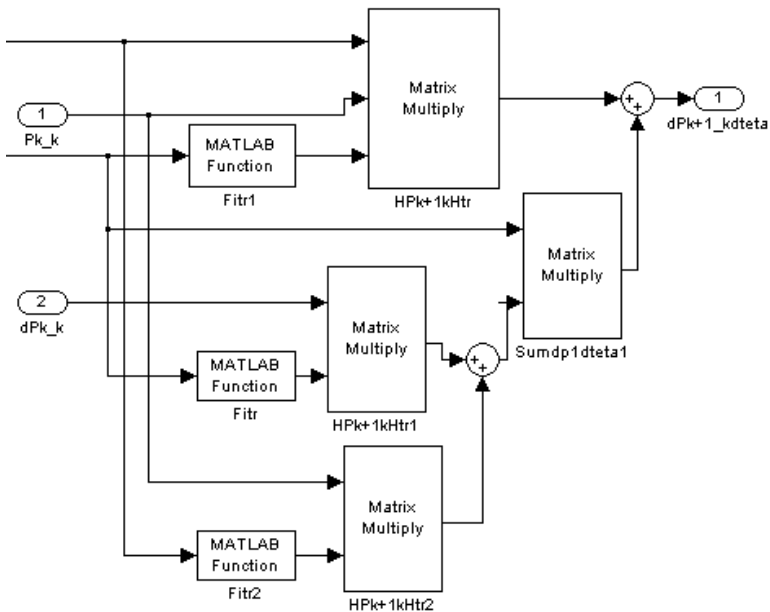


Рис. 19. Вычисление по уравнению (5)



На рис. 20 показан порядок формирования уравнения (6).

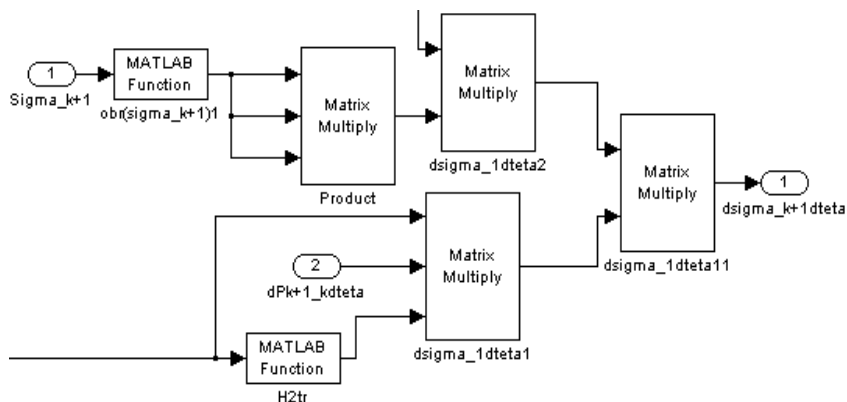


Рис. 20. Вычисление по уравнению (6)

На рис. 21 приводится схема вычислений уравнения (7).

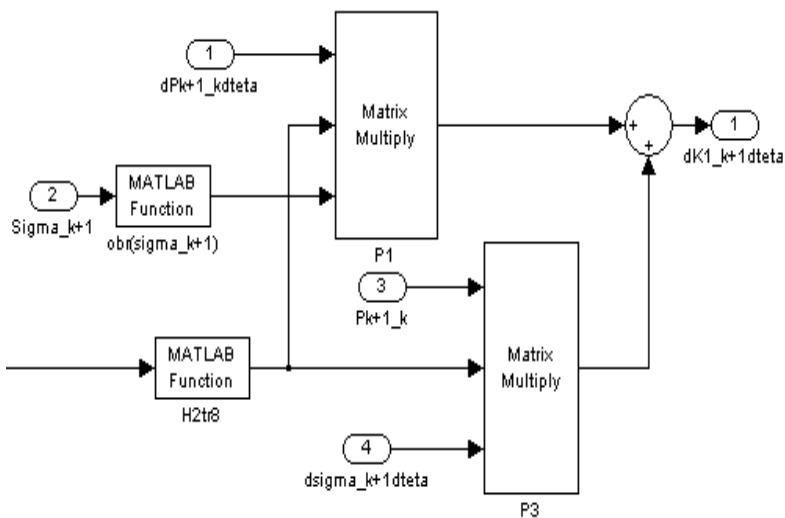


Рис. 21. Вычисление по уравнению (7)

Третья группа формул:

$$K2_{\infty} = \Phi K1_{\infty}, \quad \frac{\partial K2_{\infty}}{\partial \theta_i} = \frac{\partial \Phi}{\partial \theta_i} K1_{\infty} + \Phi \frac{\partial K1_{\infty}}{\partial \theta_i},$$

$$\Phi_{A,1} = \begin{bmatrix} \Phi & 0 & 0 \\ \frac{\partial \Phi}{\partial \theta_1} & \Phi - K2_{\infty} \Sigma_{\infty}^{-1} H & 0 \\ \frac{\partial \Phi}{\partial \theta_2} & 0 & \Phi - K2_{\infty} \Sigma_{\infty}^{-1} H \end{bmatrix},$$

$$K_{A,1} = \begin{bmatrix} K2_{\infty} \\ \frac{\partial K2_{\infty}}{\partial \theta_1} - K2_{\infty} \Sigma_{\infty}^{-1} \frac{\partial \Sigma_{\infty}}{\partial \theta_1} \\ \frac{\partial K2_{\infty}}{\partial \theta_2} - K2_{\infty} \Sigma_{\infty}^{-1} \frac{\partial \Sigma_{\infty}}{\partial \theta_2} \end{bmatrix}, \quad \Psi_{A,1} = \begin{bmatrix} \Psi \\ 0 \\ 0 \end{bmatrix},$$

$$\bar{x}_{A,1} = \Phi_{A,1} \bar{x}_{A,1} + \Psi_{A,1} u, \quad \Sigma_{A,1} = \Phi_{A,1} \Sigma_{A,1} \Phi_{A,1}^T + K_{A,1} K_{A,1}^T.$$

$$E \left\{ \left[ \frac{\partial \hat{x}_{k+1,k}}{\partial \theta_i} \right] \left[ \frac{\partial \hat{x}_{k+1,k}}{\partial \theta_j} \right]^T \right\} = c_i \left[ \Sigma_{A,k+1,k} + \bar{x}_{A,k+1,k} \bar{x}_{A,k+1,k}^T \right] c_j^T,$$

$$c_i = [0 : \dots : I : \dots : 0],$$

$$M_{ij} = \sum_{k=0}^{N-1} \left\{ Sp \left[ \frac{\partial \Sigma_{k+1}}{\partial \theta_i} \Sigma_{k+1}^{-2} \frac{\partial \Sigma_{k+1}}{\partial \theta_j} \right] + \right.$$

$$+Sp \left[ H^T \Sigma_{k+1}^{-2} H E \left\{ \left( \frac{\partial \hat{x}_{k+1,k}}{\partial \theta_i} \right) \left( \frac{\partial \hat{x}_{k+1,k}}{\partial \theta_j} \right)^T \right\} \right] +$$

$$Sp \left[ \Sigma_{k+1}^{-1} \frac{\partial \Sigma_{k+1}}{\partial \theta_i} \Sigma_{k+1}^{-1} \frac{\partial \Sigma_{k+1}}{\partial \theta_j} \right] \Bigg\}.$$

Третья группа формул касается вычисления информационной матрицы Фишера и здесь подробно не рассматривается ввиду громоздкости вычислений.

### 3.2. Моделирование итерационной процедуры оценивания динамических параметров в задаче активной идентификации

Для оценивания параметров динамических объектов используется рекуррентная схема метода наименьших квадратов [33]. Как известно, метод наименьших квадратов заключается в минимизации следующей функции квадратичного отклонения [34]:

$$s(\theta) = \sum_{i=1}^N (y_i - x_i^T \theta)^2.$$

Тогда оценку неизвестных параметров на основе метода наименьших квадратов можно получить исходя из формулы

$$\hat{\theta}_N = (X_N^T X_N)^{-1} X_N^T Y_N,$$

где приняты обозначения:

$$X_N = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix} = \begin{pmatrix} x_1^1 & x_1^2 \\ x_2^1 & x_2^2 \\ \vdots & \vdots \\ x_N^1 & x_N^2 \end{pmatrix}, \quad Y_N = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}.$$

Для получения рекуррентной оценки неизвестных параметров  $\hat{\theta}$  на основе  $N + 1$  измерений необходимо воспользоваться нижеприведенной группой формул [34]:

$$\hat{\theta}_{N+1} = \hat{\theta}_N + K_{N+1} (y_{N+1} - x_{N+1}^T \hat{\theta}_N),$$

$$K_{N+1} = P_N x_{N+1} / (1 + x_{N+1}^T P_N x_{N+1}),$$

$$P_{N+1} = \left( I - P_N \frac{x_{N+1} x_{N+1}^T}{1 + x_{N+1}^T P_N x_{N+1}} \right) P_N,$$

где  $K_{N+1}$ ,  $P_{N+1}$  вычисляются по результатам  $N + 1$  измерений.

Результаты вычислений в значительной мере зависят как от задания начального значения  $\theta_0$ , так и от начальных значений  $K_0$  и  $P_0$ . Заметим, что при нулевом значении  $P_0 = 0$  рекуррентный процесс не сходится и получить удовлетворительную оценку  $\hat{\theta}_N$  не удастся.

Моделирование итерационной процедуры вычисления оценки параметров объекта выполнено в среде Simulink. Схема вычисления параметров объекта представлена на рис. 22, где введены следующие блоки: Input\_Blok – генератор входного сигнала; Simulation – блок моделирования системы; Estimation – блок вычисления неизвестных параметров.

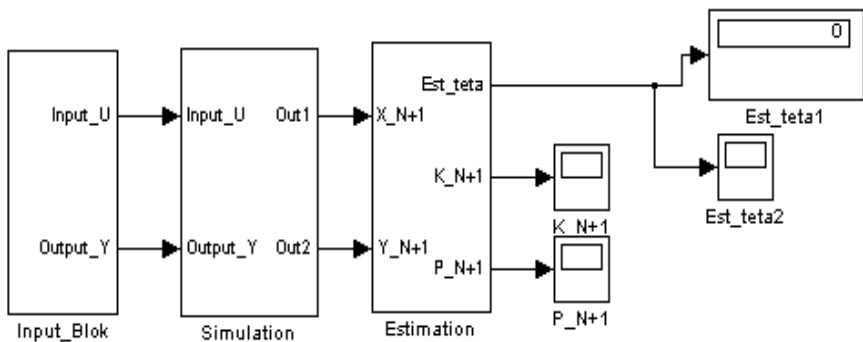


Рис. 22. Итерационная процедура вычисления оцениваемых параметров

Выходной сигнал объекта, итерационные значения коэффициента усиления  $K$  и матрицы  $P$  выведены на индикаторы. Блок Estimation приведен на рис. 23.

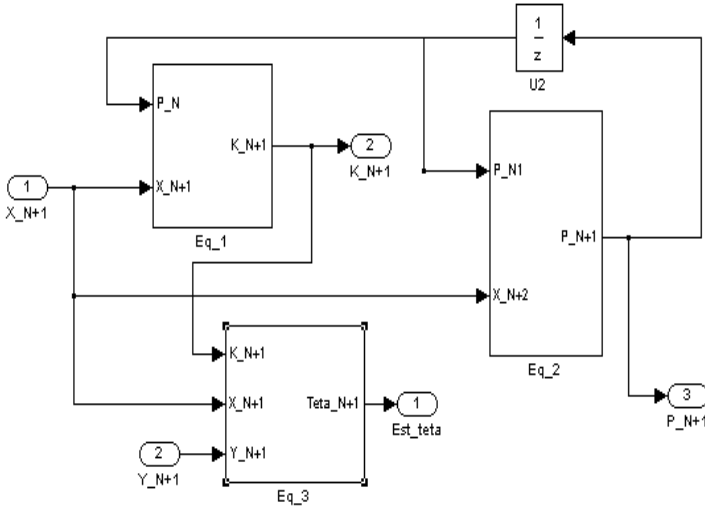


Рис. 23. Блок Estimation

Таким образом, процедура рекуррентного оценивания представлена несколькими уровнями. Верхний уровень отражает результаты моделирования динамического объекта, входных сигналов и оценивания параметров (рис. 22). Блоки следующего уровня формируют рекуррентный метод оценки параметров (рис. 23).

Пусть динамический объект рассматривается в следующей форме:

$$\begin{pmatrix} x_{k+1}^1 \\ x_{k+1}^2 \end{pmatrix} = \begin{pmatrix} 1 & -0.5 \\ \theta_1 & -\theta_1 + \theta_2 \end{pmatrix} \begin{pmatrix} x_k^1 \\ x_k^2 \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0 \end{pmatrix} u_k + \begin{pmatrix} w_{k+1}^1 \\ w_{k+1}^2 \end{pmatrix},$$

$$y_{k+1} = (0 \quad 1) x_{k+1} + v_{k+1},$$

где  $x_{k+1}$  – вектор состояния;  $u_k$  – вектор управления;  $w_{k+1}$  – вектор возмущения;  $y_{k+1}$  – вектор измерения;  $v_{k+1}$  – вектор ошибки измерения с нулевым математическим ожиданием [27–31].

Вычисления производятся шаг за шагом. После некоторых преобразований получаем следующие выражения для сигналов  $x_i$  и  $y_i$ :

$$\bar{x}_{k+1} = \begin{pmatrix} 0.5u_k + 0.5y_k - y_{k+1} \\ y_{k+1} - y_k \end{pmatrix}, \quad \bar{y}_{k+1} = y_{k+2} - y_{k+1}.$$

Эти выражения используются в дальнейшем в блоке моделирования (рис. 24).

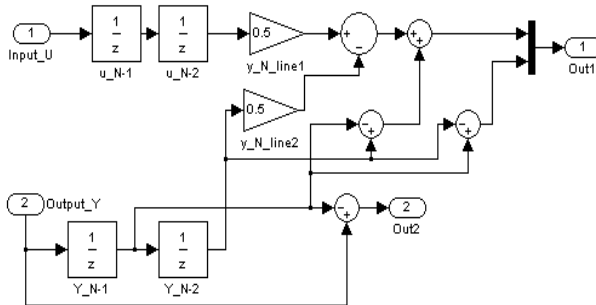


Рис. 24. Блок Simulation

Моделирование выполнено при базовых значениях объекта:  $\theta^1 = 0.5$ ,  $\theta^2 = 0.8$ . Моделирование дискретного объекта показано на рис. 25.

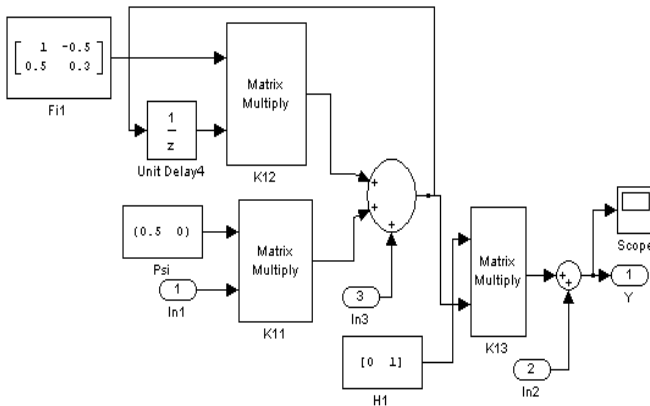


Рис. 25. Моделирование дискретного объекта

В качестве начального значения матрицы  $P$  принято следующее значение:  $P_0 = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}$ . В этой работе рассматривается активная идентификация. Это означает, что можно выбирать такой воздействующий сигнал, который в наибольшей степени «раскачивает» объект. В данном случае выбран входной сигнал типа меандра с периодом  $T = 20$  и с амплитудой, равной единице (рис. 26).

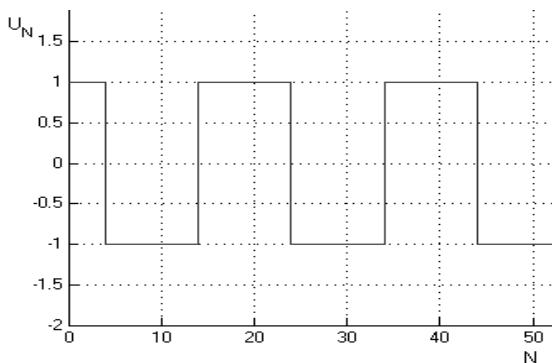


Рис. 26. Входной сигнал

График выходного сигнала объекта без шума измерений показан на рис. 27.

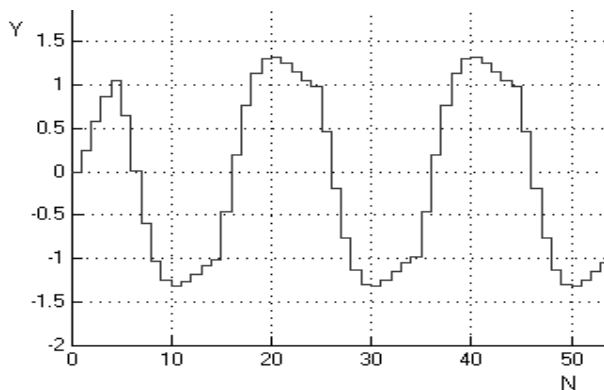


Рис. 27. Выходной сигнал объекта (без шума)

Эксперимент проводился для 120 измерений [33]. Для случая, когда шумы измерений отсутствуют, результат оценивания довольно точный, а именно  $\theta^1 = 0.4934$ ,  $\theta^2 = 0.8212$  (рис. 28).

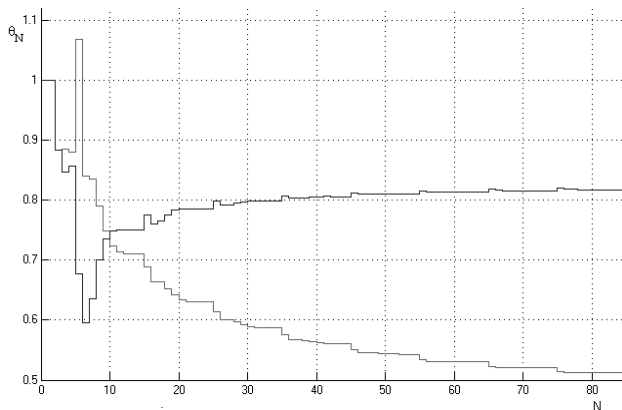


Рис. 28. Оценивание параметров (без шума)

Рекуррентная оценка сходится к истинному значению параметра примерно за 50–60 итераций. То же самое можно сказать и о сходимости значений  $P_N$  (рис. 29).

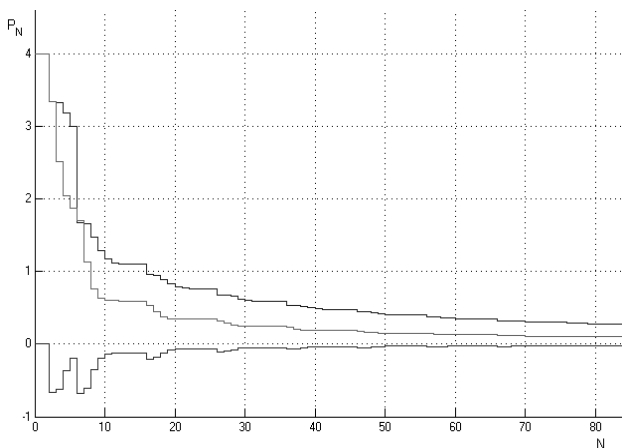


Рис. 29. Параметр  $P_N$  (без шума)



График сходимости параметра  $K_N$  представлен на рис. 30.

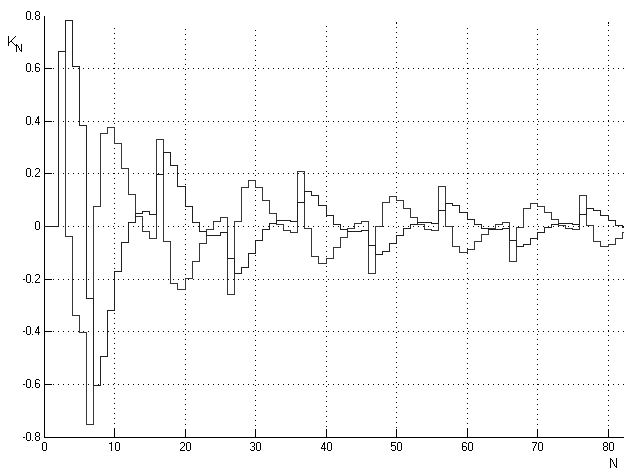


Рис. 30. Параметр  $K_N$  (без шума)

Результаты рекуррентного оценивания для случая, когда присутствуют шумы измерений с  $m = 0$  и  $\sigma^2 = 0.025$ , даны на рис. 31.

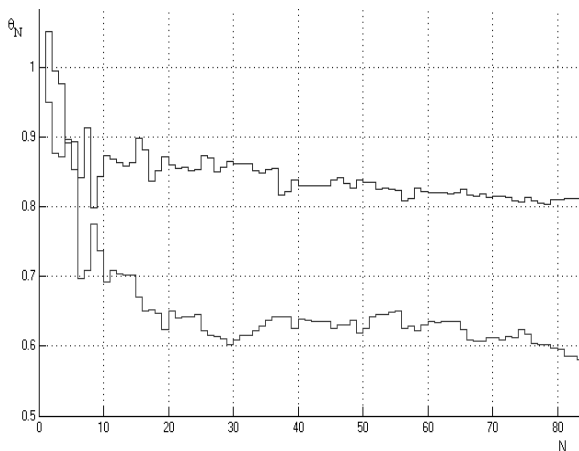


Рис. 31. Оценивание параметров (при наличии шума)

Оценки параметров получены после 120 измерений:  $\theta^1 = 0.5774$ ,  $\theta^2 = 0.8061$ . График сходимости параметров  $P_N$  для случая, когда присутствуют шумы измерений и помехи динамики, приводится на рис. 32.

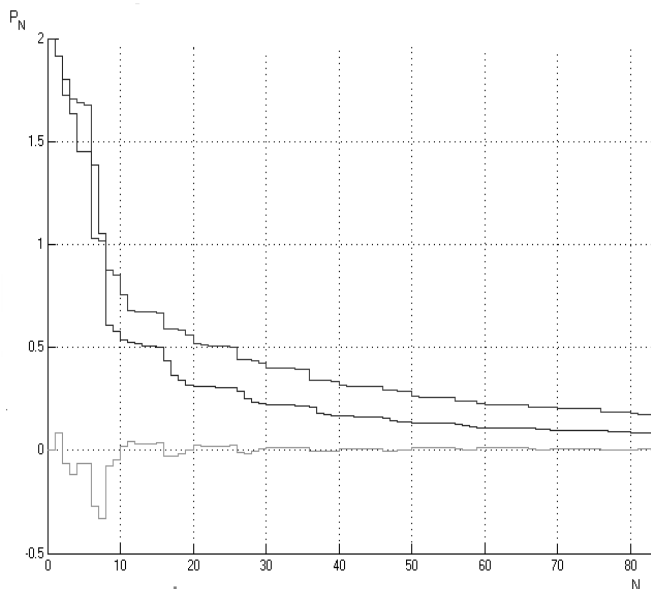


Рис. 32. Параметр  $P_N$  (с шумом)

Заметим, что погрешность измерений находится в пределах 5...7 %. Алгоритм успешно работает даже при наличии значительных шумов измерений.

**Упражнение 8.** Построить графический пользовательский интерфейс для вычисления оценок динамических параметров в задаче активной идентификации. Использовать поле *Edit Text* для ввода параметров модели динамической системы, кнопку *Button* – для старта различных вычислений, поле *Static Text* – для вывода результатов.

## **Контрольные вопросы и задания**

1. Получить оценки параметров системы для случая, когда на вход системы подается синусоидальный сигнал.
2. Получить оценки параметров системы для случая, когда на вход системы подается единичная ступенька.
3. Получить оценки параметров системы для случая, когда на вход системы подаются два меандра.
4. Что показывает коэффициент усиления в процедуре рекуррентного оценивания методом наименьших квадратов?
5. В каких случаях рекуррентный процесс оценивания методом наименьших квадратов не сходится?
6. Выполнить построение в среде Simulink уравнений третьей группы формул, используемой при формировании информационной матрицы Фишера в установившемся режиме.
7. Опишите этапы формирования матрицы Фишера в установившемся режиме.
8. Какие параметры являются входными для вычисления коэффициента усиления в процедуре рекуррентного оценивания методом наименьших квадратов?
9. Каким образом можно выполнить генерирование входных сигналов?
10. Как можно охарактеризовать оценки параметров, если шумы измерений отсутствуют?
11. Какое примерное количество измерений необходимо выполнить, чтобы рекуррентная оценка параметров удовлетворительно сошлась к истинному значению?
12. Какое влияние оказывают шумы измерений и помехи динамики на оценки параметров динамических систем?

## Библиографический список

1. *Кетков Ю. Л.* МАТЛАБ 7: Программирование, численные методы / Ю. Л. Кетков, А. Ю. Кетков, М. Н. Шульц. – Санкт-Петербург : БХВ-Петербург, 2005. – 752 с.
2. *Половко А. М.* MatLab для студента / А. М. Половко, П. Н. Бутусов. – Санкт-Петербург : БХВ-Петербург, 2005. – 320 с.: ил.
3. *Лазарев Ю.* Моделирование процессов и систем в МАТЛАБ. Учебный курс / Ю. Лазарев. – Санкт-Петербург : Питер ; Киев : Изд. группа ВНУ, 2005. – 512 с.
4. *Ануфриев И.* MatLab 7 в подлиннике / И. Ануфриев, А. Смирнов, Е. Смирнова. – Санкт-Петербург : БХВ-Петербург, 2005. – 1104 с.
5. *Гуляев А. К.* Визуальное моделирование в среде MatLab / А. К. Гуляев. – Санкт-Петербург : Питер, 2000. – 430 с.: ил.
6. *Дьяконов В. П.* MatLab 6: учебный курс / В. П. Дьяконов. – Санкт-Петербург : Питер, 2001. – 592 с.
7. *Дьяконов В. П.* Математические пакеты расширения MatLab. Специальный справочник / В. П. Дьяконов, В. В. Круглов. – Санкт-Петербург : Питер, 2001. – 480 с.
8. *Иглин С.* Математические расчеты на базе MatLab / С. Иглин. – Санкт-Петербург : БХВ-Петербург, 2005. – 640 с.: ил.
9. *Кетков Ю. Л.* MatLab 6х Программирование численных методов / Ю. Л. Кетков. – Санкт-Петербург : БХВ-Петербург, 2004. – 672 с.: ил.
10. *Кондрашов В. Е.* MatLab как система программирования научно-технических расчетов / В. Е. Кондрашов, С. Б. Королев. – Москва : Мир, 2002. – 350 с.
11. *Мэтьюз Дж. Г.* Численные методы. Использование MatLab : пер. с англ. / Дж. Г. Мэтьюз, К. Д. Финк. – Москва : Изд. дом «Вильямс», 2001. – 713 с.
12. *Потемкин В. Г.* Вычисления в среде MatLab / В. Г. Потемкин. – Москва : Диалог-МИФИ, 2004. – 720 с.
13. *Остром К.* Введение в стохастическую теорию управления / К. Остром. – Москва : Мир, 1973. – 320 с.
14. *Льонг Л.* Идентификация систем. Теория для пользователя / Л. Льонг ; под ред. Я. З. Цыпкина. – Москва : Наука, 1991. – 432 с.

15. *Эйкхофф П.* Основы идентификации систем управления / П. Эйкхофф. – Москва : Мир, 1975. – 683 с.
16. *Воевода А. А.* Вычисление информационной матрицы Фишера для линейных стационарных дискретных систем с неизвестными параметрами в моделях динамики и наблюдения / А. А. Воевода, Г. В. Трошина // Сборник научных трудов НГТУ. – Новосибирск, 2006. – Вып. 2 (44). – С. 29–34.
17. *Трошина Г. В.* О методах оценивания вектора состояния в задачах идентификации / Г. В. Трошина // Сборник научных трудов НГТУ. – 2012. – № 1 (67). – С. 69–78.
18. *Воевода А. А.* О некоторых методах фильтрации в задаче идентификации / А. А. Воевода, Г. В. Трошина // Сборник научных трудов НГТУ. – Новосибирск, 2014. – Вып. 2 (76). – С. 16–25.
19. *Трошина Г. В.* Об использовании фильтра Калмана при идентификации динамических систем / Г. В. Трошина // Сборник научных трудов НГТУ. – 2014. – № 3 (77). – С. 37–52.
20. *Трошина Г. В.* Об активной идентификации динамических объектов / Г. В. Трошина // Сборник научных трудов НГТУ. – 2014. – № 4 (78). – С. 41–52. – doi: 10.17212/2307-6879-2014-4-41-52.
21. *Troshina G. V.* Parameters estimation with fischer information matrix on the example of the control system of the inverted pendulum [Electronic resource] / G. V. Troshina, A. A. Voevoda // International Siberian Conference on Control and Communications (SIBCON–2015) : proceedings., Omsk, Russia, 21–23 May, 2015, pp. 1–4. doi: 10.1109/SIBCON.2015.7147243.
22. *Воевода А. А.* Моделирование фильтра Калмана с обновленной последовательностью в среде Simulink / А. А. Воевода, Г. В. Трошина // Сборник научных трудов НГТУ. – 2015. – Вып. 2 (80). – С. 7–17. – doi: 10.17212/2307-6879-2015-2-7-17.
23. *Воевода А. А.* Моделирование динамического объекта в среде Simulink / А. А. Воевода, Г. В. Трошина // Сборник научных трудов НГТУ. – 2015. – Вып. 3 (81). – С. 16–25.
24. *Воевода А. А.* Оценивание параметров моделей динамики и наблюдения для линейных стационарных дискретных систем с использованием информационной матрицы Фишера / А. А. Воевода, Г. В. Трошина // Научный вестник НГТУ. – 2006. – № 3(24). – С. 199–200.
25. *Mehra R. K.* Optimal input signal for parameter estimation in dynamic system – survey and new results / R. K. Mehra // IEEE Trans. Aut. Contr. – 1974. – Vol. AC – 19. – № 6. – P. 753–768. – doi: 10.1109/TAC.1974.1100701.
26. *Mehra R. K.* On the Identification of Variences and Adaptive Kalman Filtering / R. K. Mehra // IEEE Trans. Autom. Control. – 1970. – Vol. AC–15. – No. 2. – P. 175–184. – doi: 10.1109/TAC.1970.1099422.
27. *Voevoda A. A.* Active identification of linear stationary dynamic object on base of the Fisher information matrix: the ateady state / А. А. Воевода, G. V. Troshina // Proc. of the XII Intern. Conf. “Actual problems of electronic

instrument engineering (APEIE-2014)". – Novosibirsk : Novosibirsk State Technical University, 2014. – P. 745–749. – doi: 10.1109/APEIE.2014.7040785.

28. *Voevoda A. A.* Active identification of the inverted pendulum control system," / A. A. Voevoda, G. V. Troshina // Proc. of the 18th Intern. Conf. on Soft Computing and Measurements (SCM'2015). – Sankt-Peterburg : LETI Publ., 2015. – Vol. 1. – Pp. 153–156.

29. *Трошина Г. В.* Активная идентификация линейных динамических дискретных стационарных объектов во временной области : дис. ... канд. техн. наук : 05.13.01 / Г. В. Трошина ; Новосиб. гос. техн. ун-т. – Новосибирск, 2007. – 171 с.

30. *Воевода А. А.* Об оценке вектора состояния и вектора параметров в задаче идентификации / А. А. Воевода, Г. В. Трошина // Сборник научных трудов НГТУ. – 2014. – Вып. 4 (78). – С. 53–68. – doi: 10.17212/2307-6879-2014-4-53-68.

31. *Воевода А. А.* Рекуррентный метод оценивания параметров в динамическом объекте / А. А. Воевода, Г. В. Трошина // Научный вестник НГТУ. – 2016. – № 4 (65). – С. 7–18.

32. *Troshina G. V.* The steady-state in the parameters estimation problem for the dynamic objects / G. V. Troshina, A. A. Voevoda // International Multi-Conference on Engineering, Computer and Information Sciences, (SIBIRCON 2017). Novosibirsk, Akademgorodok, 18–22 Sept. 2017 / Novosibirsk: IEEE, 2017. – pp. 351–355.

33. *Troshina G. V.* The iterative procedure modeling for the dynamic parameters estimation at the active identification task / G. V. Troshina, A. A. Voevoda // Siberian symposium on data science and engineering (2017 SSDSE) : proc., Novosibirsk, Akademgorodok, 12–13 Apr. 2017 ; Novosibirsk : IEEE, 2017. – Pp. 80–83.

34. *Goodwin G. C.* Dynamic System Identification: Experiment Design and Data Analysis / G. C. Goodwin, R. L. Payne. – New York : Academic Press, 1977. – 291 p.

35. *Воевода А. А.* Моделирование матричных уравнений в задачах управления на базе MatLab/Simulink / А. А. Воевода, Г. В. Трошина. – Новосибирск : Изд-во НГТУ, 2015. – 48 с.

## Оглавление

Введение .....	3
1. Создание М-файлов и операции с функциями .....	4
1.1. Создание М-файлов .....	4
1.2. Создание файлов-функций .....	5
1.3. Графическое оформление результатов вычислений .....	10
1.4. Решение систем обыкновенных дифференциальных уравнений .....	18
1.5. Вычисление интегралов в среде MatLab .....	24
1.6. Тест для самопроверки .....	26
Контрольные вопросы и задания .....	29
2. Проектирование интерфейса пользователя .....	31
2.1. Компоненты графического интерфейса .....	31
2.2. Проектирование интерфейса в среде GUIDE .....	35
2.3. Тест для самопроверки .....	47
Контрольные вопросы и задания .....	52
3. Установившийся режим в задаче оценивания параметров динамических объектов .....	53
3.1. Формирование матрицы Фишера в установившемся режиме .....	53
3.2. Моделирование итерационной процедуры оценивания динамических параметров в задаче активной идентификации .....	59
Контрольные вопросы и задания .....	67
Библиографический список .....	68

**Трошина Галина Васильевна**

**ЧИСЛЕННЫЕ РАСЧЕТЫ В СРЕДЕ MATLAB**

**Учебное пособие**

Выпускающий редактор *И.П. Брованова*

Корректор *Л.Н. Кинит*

Дизайн обложки *А.В. Ладыжская*

Компьютерная верстка *С.И. Ткачева*

Налоговая льгота – Общероссийский классификатор продукции

Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

---

Подписано в печать 23.01.2020. Формат 60 × 84 1/16. Бумага офсетная. Тираж 50 экз.

Уч.-изд. л. 4,18. Печ. л. 4,5. Изд. № 295/19. Заказ № 321. Цена договорная

---

Отпечатано в типографии

Новосибирского государственного технического университета

630073, г. Новосибирск, пр. К. Маркса, 20