



А.А. ОВИННИКОВ

# ОСНОВЫ РАБОТЫ В СРЕДАХ МАТЛАВ И SIMULINK

УЧЕБНОЕ ПОСОБИЕ

**А.А. ОВИННИКОВ**

# **ОСНОВЫ РАБОТЫ В СРЕДАХ MATLAB И SIMULINK**

**Учебное пособие**

*Рекомендовано  
Научно-методическим советом РГРТУ  
в качестве учебного пособия для студентов,  
изучающих дисциплину 2.11.03.02 «Инфокоммуникационные  
технологии и системы связи» (квалификация «бакалавр»)*

**Москва  
КУРС  
2023**

**УДК 004.43(075.8)**  
**ББК 32.973я73**  
**ОЗ1**

ФЗ № 436-ФЗ	Издание не подлежит маркировке в соответствии с п. 1 ч. 4 ст. 11
----------------	---

*Рецензенты:*

**Рыжов А.П.** — доктор технических наук, профессор кафедры сетей связи и систем коммутации Московского Технического Университета Связи и Информатики;

**Тимошенко Г.Н.** — доктор физико-математических наук, профессор;

**Аверкин А.Н.** — кандидат физико-математических наук, доцент.

**ОЗ1 Овинников А.А.,**  
**Основы работы в средах Matlab и Simulink : учебное пособие /**  
**А.А. Овинников. — Москва: КУРС, 2023. — 136 с.**

ISBN 978-5-907228-97-9

В учебном пособии рассматриваются основы работы в средах MATLAB и Simulink в части работы с командным окном, написание и отладка программ во встроенном отладчике, а также созданию простейших имитационных динамических моделей. Для каждой темы приводится практическое задание с набором индивидуальных вариантов, которые способствуют закреплению теоретического материала.

УДК 004.43(075.8)  
ББК 32.973я73



ISBN 978-5-907228-97-9

© Овинников А.А., 2020  
© КУРС, 2020

# ОГЛАВЛЕНИЕ

<b>Часть 1. Знакомство со средой автоматизации математических расчетов – MATLAB.....</b>	<b>5</b>
1. Назначение среды MATLAB, ее преимущества и недостатки.....	5
2. Интерфейс среды MATLAB, особенности работы с ним.....	6
3. Работа в командном окне. Реализация простейших математических операций.....	8
3.1. Основные типы данных и способы их форматирования.....	8
3.2. Операции с действительными и комплексными числами.....	10
3.3. Операции с векторами и матрицами.....	13
3.3.1. Ввод и модификация массивов.....	13
3.3.2. Математические операции с массивами и дополнительные функции.....	18
4. Работа в командном окне. Визуализация полученных результатов.....	20
4.1. Построение и редактирование двумерных графиков.....	20
4.2. Построение специальных графиков.....	24
5. Практическая работа с командным окном среды MATLAB.....	25
<b>Часть 2. Основы программирования в среде MATLAB.....</b>	<b>35</b>
1. Введение.....	35
2. Формы и типы данных.....	35
3. Работа с отладчиком в редакторе m-файлов.....	38
4. Программы (скрипты) и функции языка MATLAB.....	40
5. Логика работы программ. Условный оператор, ветвления, циклы в программах MATLAB.....	45
6. Практическая работа. Разбор готовой программы на элементарные составляющие. Составление блок-схем алгоритмов.....	48
<b>Часть 3. Программирование в MATLAB. Дополнительные функциональные возможности.....</b>	<b>87</b>
1. Работа с файлами в среде MATLAB.....	87

1.1.	Сохранение/загрузка файлов в формате .mat. Функции save и load .....	87
1.2.	Сохранение/загрузка двоичных файлов. Функции fwrite и fread .....	87
1.3.	Сохранение/загрузка текстовых файлов. Функции fscanf и fprintf.....	92
2.	Функции для преобразования и обработки разнородных типов данных.....	95
3.	Оптимизация работы программ.....	95
4.	Практическая работа. Разработка элементарной программы в среде MATLAB.....	97
<b>Часть 4. Создание простейших моделей в среде Simulink.....</b>		<b>102</b>
1.	Назначение и особенности среды Simulink.....	102
2.	Запуск среды Simulink. Знакомство с каталогом библиотеки.....	102
3.	Запуск среды моделирования. Знакомство со средой для создания динамических моделей – Simulink.....	105
4.	Этапы создания S-модели. Общие принципы.....	108
5.	Настройка параметров S-модели.....	109
6.	Разработка простейшей модели. Пошаговое описание необходимых действий, узлов и систем.....	112
6.1.	Разработка и редактирование параметров простейшей S-модели.....	113
6.2.	Разработка и редактирование подсистем.....	121
6.3.	Оформление S-модели и изменение типов используемых данных.....	123
7.	Практическая работа по созданию и моделированию простейшей S-модели.....	127
Библиография.....		132

# **ЧАСТЬ 1. ЗНАКОМСТВО СО СРЕДОЙ АВТОМАТИЗАЦИИ МАТЕМАТИЧЕСКИХ РАСЧЕТОВ – MATLAB**

## **1. Назначение среды MATLAB, ее преимущества и недостатки**

MATLAB — одна из мощнейших систем автоматизации математических расчетов. Название «MATLAB» представляет собой сокращение от Matrix Laboratory и отражает тот факт, что основные численные операции в этой среде производятся над матрицами. Система MATLAB содержит в своем составе огромное количество разработанных библиотек, помогающих решать широкий перечень прикладных задач по множеству направлений, начиная с решения дифференциальных уравнений и заканчивая моделированием систем связи 4-го поколения, расшифровкой генома человека и разработкой самых современных автомобильных двигателей. При столь впечатляющих возможностях нельзя не упомянуть об одном существенном недостатке MATLAB — чрезвычайно высокой стоимости этого программного продукта. Это заставляет разработчиков по всему миру создавать аналоги этой системы. В настоящее время существует несколько альтернатив среды MATLAB, а именно: GNU Octave, FreeMat и Scilab. Все эти пакеты являются альтернативами, но не могут дать полной функциональности (вычислительная эффективность работы с большими массивами данных, программная наполненность тематических библиотек), которую обеспечивает MATLAB. Единственным достаточно близким решением с точки зрения предлагаемых возможностей может быть GNU Octave при работе в ОС Linux.

### **Преимущества MATLAB**

1. Возможность оперативной проверки научно-технических идей благодаря огромной библиотеке встроенных нетривиальных функций и модулей.
2. Удобство отладки программного кода, что значительно уменьшает время получения рабочего проекта.
3. Акцент при разработке смещен с вопроса «Как это запрограммировать?» на «Что я программирую?», многие операции интуитивно понятны.

4. Среда позволяет удобно для пользователя отображать огромное количество различных графических форм для иллюстрации всевозможных функциональных связей.

5. Каждая функция в среде отлично документирована, и поиск ее описания в библиотеке не составляет никакого труда.

### **Недостатки MATLAB**

1. Медлительность системы в работе с алгоритмами, требующими использования большого числа вложенных циклов и ветвлений, а также существенные ограничения при работе с большими массивами данных.

2. Высокая стоимость лицензионного продукта, в особенности для научно-технических разработок.

На сегодняшний день области применения среды MATLAB чрезвычайно обширные, начиная от решения сугубо математических задач и заканчивая разработкой сложных динамических моделей, в частности системы LTE, физического уровня 802.11n, а также работой с целым парком подключаемого оборудования. Подробная информация о MATLAB доступна на сайтах [www.mathworks.com](http://www.mathworks.com), [www.softline.ru](http://www.softline.ru), [www.matlab.ru](http://www.matlab.ru) и [www.exponenta.ru](http://www.exponenta.ru).

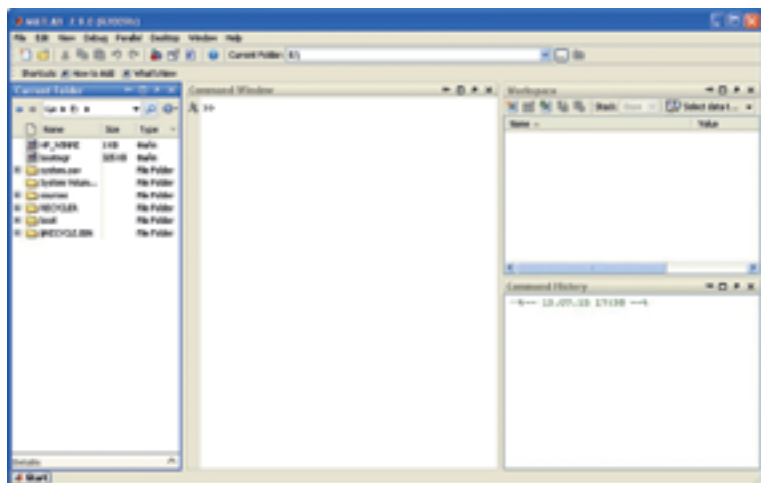
## **2. Интерфейс среды MATLAB, особенности работы с ним**

Программный комплекс MATLAB состоит из четырех основных компонентов: MATLAB, Simulink, Toolbox и Blockset. Компонент MATLAB является ядром системы, подсистема Simulink — расширение предназначенное для блочного моделирования динамических систем. Компоненты Toolbox (набор инструментов) и Blockset (набор блоков) — это пакеты расширения MATLAB и Simulink, сгруппированные по специализированным приложениям, назначение которых отображается в названиях.

Интерфейс среды разработки MATLAB состоит из следующих основных элементов (рис. 1.1):

1. Command Window (Командное окно) — основное окно интерактивной системы MATLAB с активизированной командной строкой. Окно Command Window является ключевым в общении пользователя со средой, фактически через него осуществляется взаимодействие со всеми без исключения пакетами и расширениями среды MATLAB.

2. Current Folder (Текущая папка) — в этом окне выводится содержимое папки, имя которой отображается в раскрывающемся списке Current Folder на панели инструментов окна MATLAB. По умолчанию текущей считается папка со стандартным расширением work,



**Рис. 1.1.** Главное окно MATLAB

предназначенная для хранения файлов и папок, создаваемых пользователем.

3. **Workspace (Рабочая область памяти)** – в этом окне выводится список текущих переменных, массивов, структур и других типов данных, сохраняемых в рабочей области памяти **Workspace** до момента выхода из **MATLAB**.

4. **Command History (История команд)** – в этом окне выводится построчный список объектов языка **MATLAB** в ходе текущей и предыдущих сессий.

Изменение состава и расположения окон интерфейса среды **MATLAB** можно изменить путем обращения к пункту меню **Desktop**, выбирая или удаляя интересующие пункты.

Отдельного внимания заслуживает система помощи **MATLAB**, электронная справочная система которой состоит из следующих компонентов.

1. Встроенная справочная система – формируется автоматически при установке системы **MATLAB**, исходя из ее состава, определяемого пользователем. Она является информативно наиболее краткой и содержит иллюстративные примеры, которые можно копировать и выполнять.

2. Справочная система в формате **HTML** – автономна по отношению к системе **MATLAB**, ее состав определяется пользователем при установке. Эта система по сравнению со встроенной дополнена иллю-



страционными примерами, которые можно копировать и выполнять в окне Command Window.

3. Справочная система в формате PDF – также автономна по отношению к системе MATLAB. Она представляет собой библиотеку «электронных книг», состоящую из целого ряда серий, каждая из которых посвящена какой-то выбранной теме, а входящие в ее состав книги – конкретным вопросам данной темы.

4. Демонстрационные примеры (Demos), которые включены во все компоненты Toolbox и Blockset. Эти примеры наглядно демонстрируют практическое использование различных функции в рамках определенных систем.

### **3. Работа в командном окне. Реализация простейших математических операций**

#### **3.1. Основные типы данных и способы их форматирования**

Главными объектами MATLAB являются двумерные массивы, однако все они включают в себя числа в различных форматах, большинство из которых представлено в табл. 1.1. По умолчанию используется формат double. Преобразование типов данных может быть выполнено с использованием функций вида single(число), int8(число) и т.д.

*Таблица 1.1.*

**Основные типы данных в MATLAB**

Название типа данных	Описание типа данных, размер
double	Вещественный, 64 бита
single	Вещественный, 32 бита
int8	Знаковый целочисленный, 8 битов
int16	Знаковый целочисленный, 16 битов
int32	Знаковый целочисленный, 32 бита
int64	Знаковый целочисленный, 64 бита
uint8	Беззнаковый целочисленный, 8 битов
uint16	Беззнаковый целочисленный, 16 битов
uint32	Беззнаковый целочисленный, 32 бита
uint64	Беззнаковый целочисленный, 64 бита

Отображение действительных чисел осуществляется по следующим правилам:

- 1). для отображения дробной части мантиссы используется десятичная точка либо запятая в зависимости от настроек среды MATLAB;
- 2). десятичный показатель числа записывается целым числом после символа *e*;
- 3). между записью мантиссы числа и символа *e* не должно быть никаких других символов.

Допустим, в рабочую область среды MATLAB введено число 0,000333, тогда после нажатия клавиши Enter в этом окне появится запись 3.330e-004. Несложно обнаружить, что число, отображенное на экране, не совпадает с введенным. Это обусловлено форматом представления чисел, который установлен по умолчанию. Его можно изменить, если обратиться к разделу меню File\Preferences. Далее необходимо обратиться к редактору переменных (variable editor), в котором можно изменить формат выбора данных по умолчанию (Default array format) (рис. 1.2). Альтернативным вариантом является использование команды **format**, информация о которой доступна в соответствующих разделах справочной системы MATLAB.

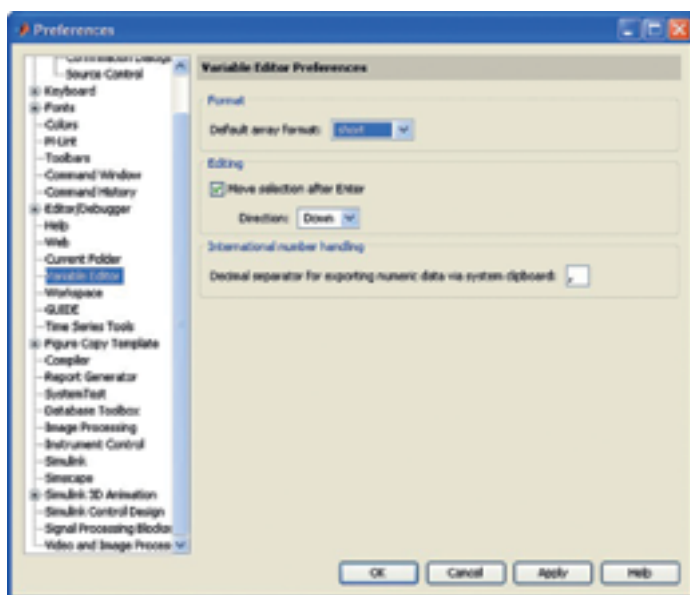


Рис. 1.2. Изменение формата представления чисел

### 3.2. Операции с действительными и комплексными числами

Рассмотрим принципы работы в командном окне среды MATLAB и некоторые наиболее распространенные команды. При нахождении значений простейших математических выражений с использованием операций (+, −, \*, /) по умолчанию результат вычисления записывается в переменную `ans`.

---

```
>>7+5  
ans =  
    12
```

---

Однако в большинстве случаев целесообразно использовать переменные с понятными не только автору именами, на которые, в свою очередь накладываются определенные ограничения:

- имя переменной может состоять из символов латинского алфавита, знака подчеркивания и цифр, но начинается обязательно с символа алфавита;
- прописные и строчные буквы различаются;
- пробел не входит в имя переменной.

Для записи в выбранную переменную результата вычислений необходимо использовать оператор присваивания «=», например:

---

```
>>a=7+5  
a =  
    12
```

---

В том случае, когда нет необходимости выводить результат вычисления в рабочую область, в конце выражения следует использовать оператор «;»:

---

```
>>a=7+5;
```

---

Результат вычисления будет доступен в окне *Workspace* среды MATLAB.

Помимо действительных чисел, используются также и комплексные, для разделения мнимой и реальной компонент которых применяется символ «i» или «j», что можно представить в рабочей области в виде

---

```
>>a=7+i*5;
```

---

Таким образом, будет сформирована комплексная переменная  $a$ , в которой будет храниться одно комплексное число. Помимо идентификатора мнимой единицы в среде MATLAB зарезервирован ряд констант и переменных, краткая характеристика по ним представлена в табл. 1.2.

Таблица 1.2.

**Зарезервированные константы и переменные среды MATLAB**

Имя	Значение
$i, j$	Мнимая единица
$\pi$	Число $\pi$
$\inf$	Обозначение машинной бесконечности
$\text{NaN}$	Обозначение неопределенного результата (0/0 или $\inf/\inf$ )
$\text{eps}$	Погрешность операций над числами с плавающей точкой
$\text{ans}$	Результат последней операции без знака присваивания
$\text{realmax}$ и $\text{realmin}$	Максимально и минимально возможные значения, которые могут использоваться в системе

Возможности среды выходят далеко за простые математические операции. В программе MATLAB имеется огромное множество разнообразных встроенных математических функций и ключевых слов. На практике наибольшее количество раз приходится использовать ключевое слово «help», которое позволяет вызывать в рабочую область справочную информацию по интересующей встроенной или пользовательской функции. В качестве примера рассмотрим справку по основным элементарным функциям, запустив следующую строку на выполнение в рабочей области среды MATLAB.

---

```
>> help elfun
```

---

В результате будет выдан список встроенных функций, включающий четыре категории:

- тригонометрические (Trigonometric);
- экспоненциальные (Exponential);
- комплексного переменного (Complex);
- округления и нахождения остатка (Rounding and remainder).

Путем нажатия на любую из функций в каждой категории пользователь получит более подробную информацию о правилах ее использования. В качестве примера использования встроенных функций рассмотрим вычисление тригонометрического тождества вида  $\sin^2(x) + \cos^2(x) = 1$ .

---

```
>>a=(sin(pi/2))^2+(cos(pi/2))^2;
```

---

Символ «^» означает возведение в степень находящейся слева от него функции или аргумента. Тригонометрические и любые другие функции могут работать не только с численными значениями входных аргументов, но и переменными, а также более сложными структурами данных. Также отдельно стоит отметить функции комплексного переменного, краткое описание которых приведено в табл. 1.3.

Таблица 1.3.

Основные функции комплексного аргумента среды MATLAB

Функция	Описание
real(Z)	Выделяет действительную часть комплексного аргумента
imag(Z)	Выделяет мнимую часть комплексного аргумента
angle(Z)	Вычисляет значение аргумента комплексного числа в диапазоне от $-\pi$ до $\pi$ в радианах
conj(Z)	Выдает число, комплексно сопряженное Z

Для того чтобы отобразить значение интересующей переменной в командном окне из рабочей области памяти, следует воспользоваться командой **disp(x)**.

В процессе решения задач могут возникать ситуации, когда происходит деление на ноль. Для их разрешения в среде MATLAB предусмотрен контроль с помощью специальных ключевых слов *Inf* (результат деления на ноль произвольного числа, кроме нуля) или *NaN* (0/0) (см. табл. 1.2).

Для подробного рассмотрения переменных, используемых в текущей рабочей сессии, используется команда **whos**. Стоит обратить внимание, что по умолчанию все переменные имеют размер 8 байтов и являются двумерными массивами (size 1×1). Подробная информация для переменных разных типов, полученная по команде **whos**, показана на рис. 1.3.

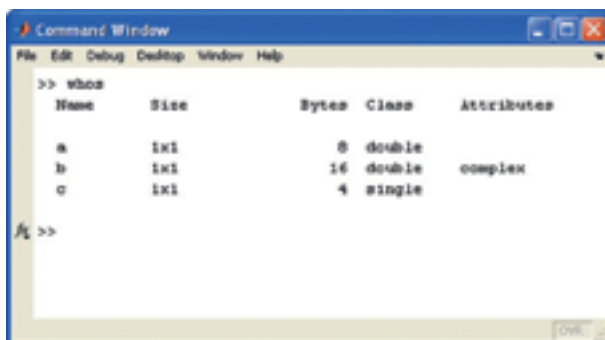


Рис. 1.3. Информация, по команде **whos**

### 3.3. Операции с векторами и матрицами

#### 3.3.1. Ввод и модификация массивов

Наиболее простым способом ввода массивов на начальном этапе знакомства со средой MATLAB является поэлементный ввод. Например, вектор-столбец представляется как

---

```
>>a=[0.2; -3.9; 4.6];
```

---

Отличительной особенностью вектора-строки является отсутствие знаков (:) для разделения чисел:

---

```
>>b=[7.6 0.1 -3.7 8.1];
```

---

Язык MATLAB предоставляет возможность сокращенного ввода вектора, значения элементов которого составляют арифметическую прогрессию. При обозначении начального элемента как *start*, шага – *step* и конечного значения – *end* можно ввести вектор в короткой форме:

---

```
>>c=0.5:0.05:1;
```

---

Если значение шага прогрессии не указывать, то он по умолчанию будет принят равным единице. Удобство работы с векторами объясняется также возможностью их объединения таким же образом, как отдельные числовые значения объединяются в вектор:

---

```
>> a=[0.2 -3.9 4.6]; b=[7.6 0.1 -3.7 8.1]; c=[a b];
```

---

Отличие в правилах ввода матриц от векторов заключается лишь в том, что необходимо задавать элементы как по строкам, так и по столбцам, например:

---

```
>> A=[0.25 -0.9; 5.6 7.9];
```

---

В результате такой операции получается матрица размером  $2 \times 2$ .

Ввод векторов и матриц может осуществляться с помощью ряда специальных функций среды MATLAB. Рассмотрим подробнее их практическое применение, которое представлено в табл.1.4.

Таблица 1.4.

**Функции оперативного формирования массивов**

Название	Описание	Примеры использования
zeros(M,N)	Создает матрицу размером $M \times N$ с нулевыми элементами	<pre>&gt;&gt; zeros(2,3)</pre> <pre>ans =</pre> <pre>0    0    0</pre> <pre>0    0    0</pre>
ones(M,N)	Создает матрицу размером $M \times N$ с единичными элементами	<pre>&gt;&gt; ones(2,3)</pre> <pre>ans =</pre> <pre>1    1    1</pre> <pre>1    1    1</pre>
eye(M,N)	Создает матрицу размером $M \times N$ с единичными элементами на главной диагонали	<pre>&gt;&gt; eye(2,3)</pre> <pre>ans =</pre> <pre>1    0    0</pre> <pre>0    1    0</pre>
rand(M,N)	Создает матрицу размером $M \times N$ , значения элементов которой распределены равномерно в диапазоне [0,1]	<pre>&gt;&gt; rand(2,3)</pre> <pre>ans =</pre> <pre>4.41e-002  9.35e-001  8.31e-001</pre> <pre>6.31e-003  3.15e-001  5.77e-002</pre>
randn(M,N)	Создает матрицу размером $M \times N$ , значения элементов которой распределены по нормальному закону. Математическое ожидание нулевое, СКО равно единице	<pre>&gt;&gt; randn(2,3)</pre> <pre>ans =</pre> <pre>1.79    1.44   -0.64</pre> <pre>0.26   -0.71    0.57</pre>

Помимо функций, осуществляющих оперативное формирование необходимой матрицы, важно отметить ряд функций, которые служат для преобразования уже имеющихся матрицы или вектора. Наиболее простым вариантом преобразования является транспонирование ('). Если элементы массива являются комплексными, то преобразование приводит к транспонированию с комплексным сопряжением. Более сложные функции с кратким описанием представлены в табл. 1.5. Пусть для преобразования используется матрица и вектор вида

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$v = [1 \ 2 \ 3];$$

Таблица 1.5.

**Функции, осуществляющие  
элементарные преобразования массивов**

Название	Описание	Примеры использования
diag(A) либо diag(v,k)	Формирует или извлекает диагональ матрицы. В случае когда $A$ существует и является матрицей, происходит процедура извлечения. Если $v$ – вектор, то процедура сформирует квадратную матрицу с элементами вектора на главной диагонали. Параметр $k$ указывает номер диагонали, которую необходимо заполнить элементами вектора	<pre>&gt;&gt;diag(A)' ans=     1     5     9 &gt;&gt;diag(v) ans=     1     0     0     0     2     0     0     0     3 &gt;&gt; diag(v, -1)     0     0     0     0     1     0     0     0     0     2     0     0     0     0     3     0</pre>
fliplr(A)	Формирует матрицу, зеркально отображенную относительно вертикальной оси, расположенной посередине	<pre>&gt;&gt; fliplr(A)     3     2     1     6     5     4     9     8     7</pre>



flipud(A)	Формирует матрицу, зеркально отображенную относительно горизонтальной оси, расположенной посередине	<pre>&gt;&gt; flipud(A) 7   8   9 4   5   6 1   2   3</pre>
rot90(A)	Поворот матрицы $A$ на $90^\circ$ против часовой стрелки	<pre>&gt;&gt; rot90(A) 3   6   9 2   5   8 1   4   7</pre>
reshape(A, M, N)	Выделяет матрицу размером $[M \times N]$ из $A$ , распределяя элементы по $N$ столбцам, каждый из которых содержит $M$ элементов. Общее количество элементов не должно изменяться в результате преобразования.	<pre>&gt;&gt; reshape(A, 1, 9) 1 2 3 4 5 6 7 8 9</pre>
tril(A)	Образует нижнюю треугольную матрицу на основе матрицы $A$ путем обнуления элементов выше главной диагонали	<pre>&gt;&gt; tril(A) 1   0   0 4   5   0 7   8   9</pre>
triu(A)	Образует верхнюю треугольную матрицу на основе матрицы $A$ , путем обнуления ее элементов ниже главной диагонали	<pre>&gt;&gt; triu(A) 1   2   3 0   5   6 0   0   9</pre>
cat(dim, A, B)	Объединяет векторы, матрицы в направлении выбранной размерности dim	<pre>&gt;&gt; cat(2,A,v') 1   2   3   1 4   5   6   2 7   8   9   3</pre>

Большое количество примеров нетрудно найти в справочной системе MATLAB, просто вызвав команду **help** и указав интересующую функцию, например:

---

```
>> help cat
```

---

Если перейти по ссылке `doc cat`, то мы автоматически окажемся в справочной системе, где в левой стороне окна можно обнаружить раздел справки, в которой описана требуемая функция. Для метода `cat` этот раздел соответствует `MATLAB\Functions\Mathematics\Arrays and Matrices\cat`.

Далее перейдем к рассмотрению способов обращения к отдельным элементам массивов и их частям. Следует учитывать, что в среде MATLAB индексация массивов производится с единицы, в отличие от многих языков программирования. Для дальнейшего изучения вопроса индексации будем использовать матрицу вида

$A =$

1	2	3	-1
4	5	6	-2
7	8	9	-3

Значение любого элемента матрицы  $A$  можно получить, обращаясь к нему, как  $A(k,l)$ , где  $k$  — номер строки, а  $l$  — номер столбца. Для изменения элемента необходимо использовать операцию присваивания, например:

---

```
>> A(1,1)=pi;
```

---

Такое преобразование приведет к тому, что элемент матрицы, расположенный в первой строке и первом столбце, примет значение, равное  $\pi$ .

Иногда требуется выбрать или изменить строки и столбцы двумерного массива. В таком случае на помощь придет оператор `(:)`, который отмечает все элементы по строкам  $A(:, 1)$  и столбцам  $A(1, :)$  соответственно. Аналогично можно использовать оператор `(:)` для выбора/замещения группы элементов, например:

---

```
>> B=A(1:3,2:3)
```

---

$B =$

2	3
5	6
8	9

---

Для упрощения процедуры выделения/замещения части столбца/строки массива можно воспользоваться оператором `end`, который ука-

зывает на индекс последнего элемента. Результат предыдущего примера также может быть получен как  $B=A(1:end, 2:end)$ .

В некоторых случаях после многочисленных изменений размера используемого вектора или матрицы требуется узнать размерность этого объекта. Поэтому в среде MATLAB предусмотрены две специальные функции, которые позволяют выполнить такую операцию:

1. `length(v)` – выдает число, указывающее на количество элементов вектора  $v$ ;
2. `[M N] = size(A)` – выдает размерность матрицы  $A$ , где  $M$ ,  $N$  – число строк, столбцов соответственно.

### 3.1.2. Математические операции с массивами и дополнительные функции

Все операции, которые могут быть выполнены с векторами и матрицами в среде MATLAB, условно разделяются на разрешенные и запрещенные в математике. Примеры всевозможных операций с описанием представлены в табл. 1.6.

Таблица 1.6.

**Математические операции**

Наименование	Описание	Примеры использования
<+>, <->, <*> (строго математические)	Операции сложения, вычитания и умножения выполняются с векторами и матрицами одинаковой размерности в соответствии с правилами математики	>>x=[1 2 3]; >>y=[6 5 4]; >>v=x+y v= 7 7 7 >>v=2*v >>v= 14 14 14
cross(a,b) (строго математическая)	Векторное произведение двух трехкомпонентных векторов	>>a=[1 2 3]; >>b=[4 5 6]; >>cross(a,b) ans= -3 6 -3
<+>, <->, <.*>, <./>, <.\>, <.^> (поэлементные операции)	Операции «сумма» и «разность» используются при сложении массива с числом. Умножение и деление предполагают выполнение заданного действия с двумя массивами одинаковой размерности. Процедура обратного деления обозначает, что расположение делимого находится справа от операнда. Можно возводить массив в степень числа или степень другого массива	>>x=[0.1 0.2 0.3]; >>y=[3 7 9]; >>z=x+0.7 z= 0.8 0.9 1 >>x=x.*10 x= 1 2 3 >>z=x.\y z= 3 3.50 3 >>z=x.^y z= 1 128 19683

inv(A)	Вычисление обратной матрицы, которая должна быть квадратной	>>A=[0.1 2 3; 4 5 6; 7 8 9]; >>inv(A) ans= -1.11    2.22    -1.11 2.22    -7.44    4.22 -1.11    4.89    -2.78
--------	---	---

Рассмотренные ранее функции (тригонометрические, экспоненциальные, комплексного переменного и округления) применимы не только к отдельным числам, но и массивам. Кроме этого, существует ряд дополнительных функций, работающих исключительно с векторами и матрицами. Описание и примеры их использования даны в табл. 1.7.

Все функции, представленные в табл. 1.7, применимы к матрицам в той же степени, что и к векторам. Отличие заключается лишь в том, что для любой функции, рассмотренной выше, двумерный массив является набором векторов, для каждого из которых необходимо вычислить требуемую процедуру. Таким образом, конечный результат всегда будет иметь формат вектора, значения которого определяются выбранной функцией, примененной к N векторам столбцов.

Таблица 1.7.

### Функции обработки данных

Наименование	Описание	Примеры использования
min(v), max(v)	Нахождение минимального и максимального элементов в векторе. Возможно получение двух выходных параметров, причем второй из них является индексом минимального или максимального значения вектора v	>>y=[3 7 9]; >>z=min(y) z=3 >> z=max(y) z=9
mean(v), std(v)	Определение среднего значения или среднеквадратического отклонения для заданного вектора v	>>z=mean(y) z=6.3333 >>z= std(y) z=3.0551
sum(v), prod(v)	Вычисление суммы или произведения элементов вектора v	>>z=sum(y) z=19 >>z=prod(y) z=189

cumsum(v), cumprod(v)	Формирование векторов кумулятивной суммы или произведения. Каждый последующий элемент вектора вычисляется как сумма или произведение предыдущих элементов	>>z= cumsum(y) z=3 10 19 >>z= cumprod(y) z=3 21 189
sort(v)	Сортировка элементов вектора v	>>y=[7 1 4 2 9 8]; >>z=sort(y) z=1 2 4 7 8 9
diff(v)	Создается вектор, размер которого на единицу меньше размера исходного вектора v. Элементы рассчитываются как разность между соседними значениями вектора v	>>z=diff(y); z=[-6 3 -2 7 -1]

## 4. Работа в командном окне. Визуализация полученных результатов

### 4.1. Построение и редактирование двумерных графиков

Среда MATLAB предоставляет широчайшие возможности по визуализации экспериментальных и расчетных данных. Наиболее простой и удобной операцией среди прочих является функция `plot(x1, y1, s1, x2, y2, s2 ...)`, которая позволяет строить двумерные графики.

Рассмотрим параметры функции `plot`. Векторы `x1, y1` представляют отсчеты аргумента и соответствующей функции, отвечающие первой кривой графика. Вторая кривая, которая будет построена на графике, определяется значениями `x2` и `y2`. Предполагается, что значения аргумента откладываются вдоль горизонтальной оси, а значения функции – вдоль вертикальной. Переменные `s1, s2` отвечают за способ отображения графиков, являются символьными. Их использование не обязательно, но может быть крайне полезным при анализе графической информации. Рассматриваемые переменные могут содержать до трех специальных символов:

- тип линии, которая соединяет отдельные точки графика;
- тип точки графика;
- цвет линии.

По умолчанию для первого графика цвет линии является синим, отображение происходит по точкам, которые соединены обыкновенными отрезками прямой.

Графики всегда выводятся в отдельное графическое окно, если не используются специальные дополнительные команды. Вызов графиче-

ческого окна для открытия существующего графика осуществляется по команде **figure**. Рассмотрим пример построения сложного графика на примере:

$$f(x) = e^x \sin(\pi x) + x^2 \text{ на отрезке } [-2, 2].$$

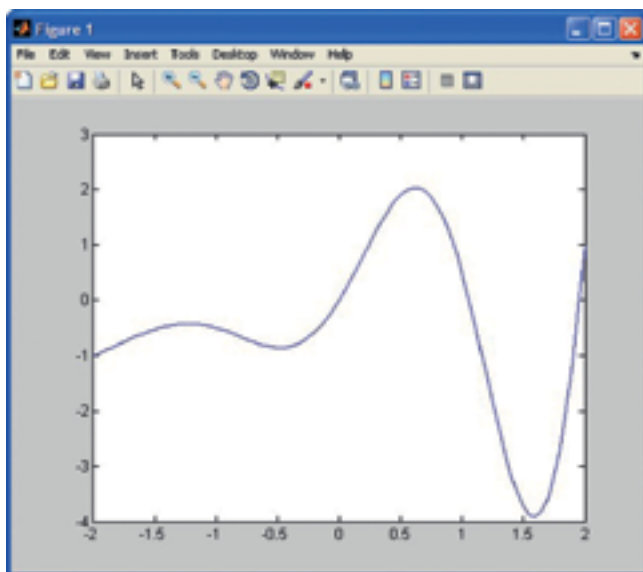
В данном случае шаг можно выбрать произвольно. В рабочем окне для построения графика потребуется ввести следующие строки:

---

```
>> x=[-2:0.05:2];  
>> f=exp(x).*sin(pi*x)+x.^2;  
>> plot(x,f);
```

---

Результат приведенных операций представлен на рис. 1.4.



**Рис. 1.4.** Графическое отображение функции

Добавим дополнительный график к уже построенному. Для этого используем команду **hold on/off**, которая позволяет отобразить множество графиков на одном рисунке:

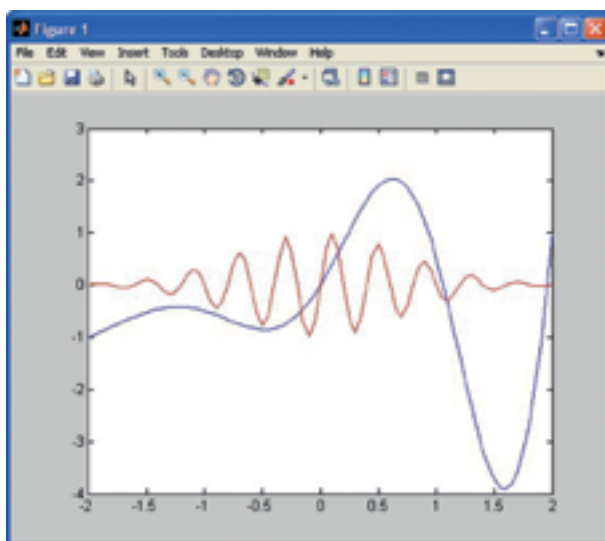
---

```
>> hold on;  
>> g=exp(-x.^2).*sin(5*pi*x);  
>> plot(x,g, '-r');  
>> hold off;
```

---

Результат приведенных операций показан на рис. 1.5, аналогичный эффект можно получить используя функцию

```
>> plot(x,f,'-b', x,g, '-r');
```



**Рис. 1.5.** Отображение нескольких функций на одном графике

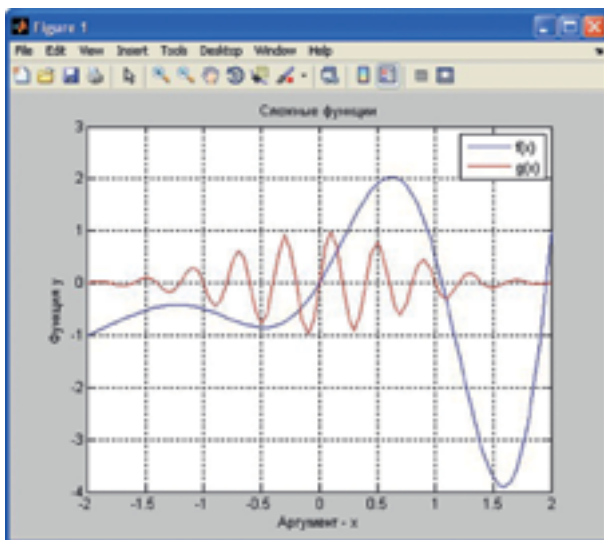
Существенным недостатком представленных графиков является сложность анализа представленной информации, так как отсутствует разметка осей, сетка, название графиков и какие-либо отметки о различии функциональных зависимостей. На текущем этапе редактирование графика может идти по двум совершенно разным путям – с помощью команд либо путем редактирования параметров в окне графика. Рассмотрим более сложный путь – использование команд. Наиболее простой элемент, который можно добавить на график – это сетка (grid). Ее особенностью является то, что она всегда соответствует целым шагам изменения, что делает графики более наглядными. Далее с помощью команды **title**(‘текст’) отображается заголовок. Аналогично можно вывести объяснения к графику, которые размещаются вдоль горизонтальной **xlabel**(‘текст’) и вертикальной **ylabel**(‘текст’) осей. И наконец, последний штрих – добавление так называемой легенды (**legend**(‘текст1’, ‘текст 2’, ...)) – смыслового пояснения каждой зависимости. Таким образом, результирующий список команд в рабочем окне будет выглядеть следующим образом:

---

```
>>grid;  
>>title('Сложные функции');  
>>xlabel('Аргумент - x');  
>>ylabel('Функция y');  
>>legend('f(x)', 'g(x)');
```

---

а график примет вид, представленный на рис. 1.6.



**Рис. 1.6.** Пример отредактированного графика

Если необходимо отображать графически сразу большое количество различных функциональных зависимостей, то следует присмотреться к функции `subplot(m, n, p)`. Ее параметры имеют следующее смысловое значение: `m` указывает, на сколько частей разбивается графическое окно по вертикали; `n` — аналогично, но по горизонтали, а последний параметр `p` является номером подокна, в котором будет строиться график. При этом подокна нумеруются слева направо построчно сверху вниз. Для того чтобы поместить предыдущий график на два горизонтально расположенных окна необходимо воспользоваться следующим набором команд:

---

```
>>subplot(1,2, 1); plot(x, f,'-k'); subplot(1,2, 2); plot(x, g,'-k');
```

---



## 4.2. Построение специальных графиков

При внимательном рассмотрении параметров графического окна можно обнаружить, что масштаб отображения построенных зависимостей можно менять на логарифмический по различным осям. В случае когда есть понимание, что такой масштаб будет удобным, можно вместо функции `plot` воспользоваться методами логарифмического и полулогарифмического построения `loglog(x, Y, s)`, `semilogx(x, Y, s)` или `semilogy(x, Y, s)`. Параметры этих функций идентичны `plot`. Функция `semilogx()` логарифмирует масштаб по горизонтальной оси, а `semilogy()` — по вертикальной.

К специальным графикам, имеющим практический интерес в инженерных разработках и научных исследованиях, также можно отнести различные столбцовые диаграммы, в частности функции `bar(Y)` и `hist(Y,x)`. Функция `bar()` строит столбцовый график элементов вектора  $Y$ , значения по горизонтали определяются вектором  $1:m$ , где  $m$  — количество точек в  $Y$ . Классическая гистограмма (`hist`) характеризует графически числа попаданий значений элемента вектора  $Y$  в  $m$  интервалов, с представлением этих чисел в виде столбцовой диаграммы. Набор команд, осуществляющих построение рассмотренных столбцовых диаграмм, представлен далее. В результате получены графики для диаграммы типа `bar` и `hist`, изображенные на рис. 1.7.

---

```
>>subplot(1,2,1); bar(rand(1,15), 'k');  
>>subplot(1,2,2); x=-6:0.1:6; y=randn(5000,1); hist(y,x,'k');
```

---

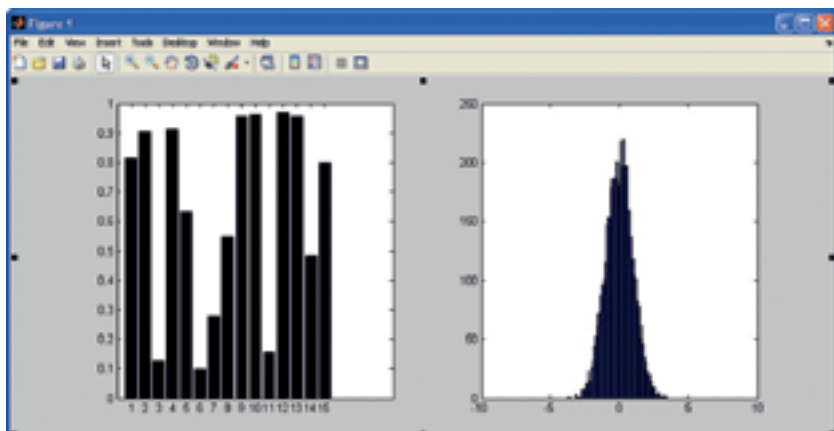


Рис. 1.7. Столбцовые диаграммы

## 5. Практическая работа с командным окном среды MATLAB

Ознакомление с принципами работы в командном окне MATLAB наиболее эффективно при выполнении ряда практических упражнений, в частности работе с переменными и простыми математическими операциями, знакомстве со встроенными функциями, применении массивов (одномерных и двумерных), а также построении и редактированию графических данных.

Для практической проработки теоретических аспектов работы в командном окне MATLAB предлагается выполнить ряд индивидуальных заданий, в результате чего необходимо заполнить отчет о выполнении задания. Отчет выполняется в произвольной форме по пунктам задания, требующим соответствующих записей.

Необходимые для выполнения задания шаги указаны далее.

1. Запустить MATLAB.
2. Создать папку рабочей директории по адресу D:\Группы\фамилия. Примером рабочей директории может быть D:\0110\Pupkin. Фамилия и номер группы не должны содержать букв русского алфавита. Диск может быть выбран произвольно, кроме системного.
3. В рабочем окне задать вектор **a** (см. вариант в табл. 1.8).
4. Вызвать помощь по команде **format**. Вывести значения элементов вектора **a** в форматах (*short*, *long*, *hex*). Записать результат наблюдения в отчет по заданию. Установить режим *short*.
5. Очистить Command Window командой **clc**. Вызвать помощь по команде **clc** и выяснить, какая команда позволяет выполнять очистку рабочей области (**Workspace**). Записать эту функцию в отчет по заданию.
6. Вычислить значение переменной  $sa=\sin(a)$ . Занести результаты в отчет.
7. Ввести значение переменной **b**. Найти  $ab=a+b$ ;  $bt=b^T$ ;  $atb=a^Tb$ ;  $abt=a*b^T$ ;  $atbt=a^Tb^T$ . Результаты занести в отчет.
8. Ввести матрицу **c**. Вычислить матрицу  $d=c^{-1}$ . Вычислить поэлементное умножение **d** и **c**. Вычислить произведение матриц  $dc=d*c$ . Результаты занести в отчет.
9. Сохранить переменные рабочей области памяти **Workspace**, используя команду **save** (посмотреть помощь по команде **help save**). Параметры команды посмотреть, вызвав помощь по этой команде (сохранять переменные в своей директории).
10. Выяснить какие переменные активны в данный момент, при помощи команды **whos**. Очистить память рабочей области при помощи

команды **clear**. Убедиться, что в памяти рабочей области нет никаких переменных.

11. Загрузить переменные из файла при помощи команды **load** (параметры команды узнать с помощью **help load**).

12. При помощи встроенных функций для заполнения стандартных матриц, индексации двоеточием и, возможно, поворота, транспонирования или вычеркивания получите матрицы в соответствии с вариантом задания (табл. 1.9, колонка 1). Полученную в результате работы правильную последовательность команд отразить в отчете.

13. Сконструировать блочные матрицы (используя функции для заполнения стандартных матриц) и применить функции обработки данных и поэлементных операций для нахождения заданных величин (см. табл. 1.9, колонка 2). Полученную в результате работы правильную последовательность команд отразить в отчете.

14. Ввести вектор  $x$  со значениями в диапазоне и с шагом согласно варианту задания (см. табл. 1.8). Вычислить значения заданной функции. Результат занести в переменную  $y$ . Построить график функции при помощи команды **plot(x,y)**. Сохранить и после этого закрыть полученный график.

15. Построить графики функций одной переменной на указанных интервалах (см. табл. 1.9). Вывести графики различными способами:

- в отдельные графические окна;
- в одно окно на одни графические оси;
- в одно окно на отдельные оси.

Графики необходимо группировать в соответствии с интервалами для переменной  $x$ , т.е. группируются графики функций  $f(x)$ ,  $g(x)$  и  $u(x)$ ,  $v(x)$ .

16. Отредактировать график, на котором представлены две функции одновременно ( $f(x)$  и  $g(x)$ ), добавив на него сетку, описание заголовка и отдельных осей, а также легенду. Сохранить график в свою рабочую директорию.

17. Продемонстрировать полученные результаты работ преподавателю (отчет, а также данные из рабочей области среды MATLAB).

Таблица 1.8.

## Варианты задания, часть 1

№	a	b	c	x	f(x)
1	(0.1, -3.4, 2.12)	(-0.3, 1.2, 6)	$\begin{bmatrix} 0.1 & -1 & 3 \\ 2 & -4 & 0.1 \\ 5 & 6 & -2 \end{bmatrix}$	От 0 до $\pi$ , Шаг 0.01	$\sin(2x)$
2	(-0.1, 1/3, 2)	(-5, -3, 21.2)	$\begin{bmatrix} 1 & -1 & 2 \\ 7 & -4 & 0.1 \\ -0.01 & 2 & -3 \end{bmatrix}$	От $-\pi$ до $\pi$ Шаг 0.03	$\cos(2x)$
3	(0.3, 1/7, 0.1)	(-3.4, 6, 8)	$\begin{bmatrix} 2 & -4 & 0.1 \\ 0.1 & -1 & 3 \\ 5 & 6 & -2 \end{bmatrix}$	От 0 до $\pi/2$ Шаг $\pi/10$	$\tan(x/2)$
4	(6, 7, $\pi/2$ )	(12, -4, 1)	$\begin{bmatrix} -4.1 & -1 & 5 \\ 0.2 & -2 & 0.6 \\ 0.3 & 1 & -1 \end{bmatrix}$	От $-\pi$ до $\pi$ Шаг 0.1	$\sin(x)$
5	(-7, 1.25, 3/7)	(4, 2, 1)	$\begin{bmatrix} 1/2 & 1/3 & 1/4 \\ 1/5 & 1/6 & 1/7 \\ 1/8 & 1/9 & 1/10 \end{bmatrix}$	От 0 до $6\pi$ Шаг $\pi/8$	$\cos(2x)$
6	(1.23, 4.5, 2)	(-1, -2, -6)	$\begin{bmatrix} 0.1 & -1 & 3 \\ 5 & 6 & -2 \\ 2 & -4 & 0.1 \end{bmatrix}$	От $-2\pi$ до $2\pi$ Шаг $\pi/50$	$\sin(3x)$
7	(0, -3, 0.0001)	(-12, 62, 100)	$\begin{bmatrix} 1 & -1 & 3 \\ 2 & -2 & 0.1 \\ 5 & 60 & -20 \end{bmatrix}$	От $-\pi/2$ до $\pi/2$ Шаг $\pi/12$	$\cos(3x)$
8	(1, 4, $1/\pi$ )	(-7, 6, -5.432)	$\begin{bmatrix} 0.1 & -0.1 & 0.2 \\ 1/27 & -4 & 27 \\ 55 & 67 & 23 \end{bmatrix}$	От 0 до $3\pi$ Шаг $1/14$	$\tan(x)$
9	(-1/3, 1/6, 1/7)	(0.1, -0.1, 0.01)	$\begin{bmatrix} 1 & -6 & 5 \\ 2 & -1.3 & 0.1 \\ -4 & 1 & -0.2 \end{bmatrix}$	От -1 до 1 Шаг 0.01	$\arccos(x)$

Продолжение табл. 1.8

10	(0.2, 1/12, 2.1)	(-0.2, 1.2, 6)	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	От 0 до 100 Шаг 1	cos(x/10)
11	(-4, 2, 5)	(5, -7, 8)	$\begin{bmatrix} 0.3 & 0.65 & -1 \\ 1 & -1.22 & -0.28 \\ 0.96 & -0.27 & 0.46 \end{bmatrix}$	От 1 до 10 Шаг 0.05	ln(x)
12	(2, 4, -13)	(-1, 14, 11)	$\begin{bmatrix} 0.44 & -0.05 & -2 \\ -0.24 & 0.9 & 1.1 \\ -0.78 & 0.6 & -1 \end{bmatrix}$	От 0 до 3*pi Шаг 0.01	sin(x/3)
13	(18, -6, -8)	(27, -1, 15)	$\begin{bmatrix} 1 & -1.15 & -1.7 \\ 0.25 & 0.55 & -0.4 \\ -1 & 1.5 & -0.08 \end{bmatrix}$	От 0 до 5 Шаг 0.1	cos(x/2)
14	(6, 30, 22)	(-16, 15, 2)	$\begin{bmatrix} -2 & 2 & -1 \\ 0.84 & -0.4 & -0.6 \\ -0.42 & 0.4 & -5 \end{bmatrix}$	От 5 до 15 Шаг 0.5	sqrt(x)
15	(21, 2, -3)	(25, 2, 12)	$\begin{bmatrix} 0.4 & 0.45 & 1.2 \\ 1.3 & -0.5 & 0.12 \\ -0.6 & 0.1 & -1 \end{bmatrix}$	От 0 до 5*pi Шаг 0.15	tan(x/5)
16	(17, -1, 0)	(33, 2, -14)	$\begin{bmatrix} 0.54 & 0.86 & -0.43 \\ 1.83 & 0.32 & 0.34 \\ -2.26 & -1.31 & 3.58 \end{bmatrix}$	От 0 до 5 Шаг 0.1	exp(x)
17	(-12, -45, 5)	(6, 82, 55)	$\begin{bmatrix} 2.78 & 0.75 & -0.07 \\ -1.17 & 1.77 & -0.5 \\ -1.85 & 1.22 & 0.23 \end{bmatrix}$	От 103 до 105 Шаг 103	log10(x)
18	(-2, 0, 91)	(9, -60, 51)	$\begin{bmatrix} 1.33 & -0.86 & -0.52 \\ -0.42 & -1.04 & 0.18 \\ -0.14 & -0.27 & 0.87 \end{bmatrix}$	От 5 до 14 Шаг 1	pow2(x)

Окончание табл. 1.8

19	(-48, -5, 13)	(8, -83, 73)	$\begin{bmatrix} 0.71 & -0.33 & -0.01 \\ 1.41 & 0.71 & -1.15 \\ -1.6 & 0.31 & -1 \end{bmatrix}$	От $2\pi$ до $3\pi$ Шаг 0.03	$\sec(x)$
20	(1, 63, 17)	(-67, -7, 0)	$\begin{bmatrix} -0.22 & 0.07 & 1.01 \\ 0.54 & 0.03 & -1.34 \\ 0.39 & 2.23 & -1.03 \end{bmatrix}$	От $-\pi$ до $\pi$ Шаг 1/17	$\csc(x)$

Таблица 1.9.

## Варианты задания, часть 2

№	Матрицы	Блочные матрицы	Функции
1	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 7 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 0 & 0 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 & 0 & 4 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \end{bmatrix}$ $m = \max_{j=1,2,\dots,6} \left\{ \sum_{i=1}^6 a_{ij}^2 \right\}$	$\begin{aligned} f(x) &= \sin x; \\ g(x) &= \sin^2 x; \\ x &\in [-2\pi, 3\pi]; \\ u(x) &= 0,01x^2; \\ v(x) &= e^{- x }; \\ x &\in [-0,2; 9,4] \end{aligned}$
2	$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 5 \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 7 & 0 \\ 5 & 0 & 0 & 0 & 0 & -1 & 8 \end{bmatrix}$	$A = \begin{bmatrix} 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \end{bmatrix}$ $s = \sum_{i=1}^6 \sum_{j=1}^6  a_{ij} $	$\begin{aligned} f(x) &= \sin x^2; \\ g(x) &= \cos x^2; \\ x &\in [-\pi, \pi]; \\ u(x) &= x / 20; \\ v(x) &= e^x; \\ x &\in [-2; 2] \end{aligned}$
3	$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ -1 & -2 & -3 & -4 & -5 & -6 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix}$ $m = \min_{i,j=1,\dots,6} a_{ij}^3$	$\begin{aligned} f(x) &= x^3 + x^2 + 1; \\ g(x) &= (x-1)^2; \\ x &\in [-1; 1]; \\ u(x) &= \sqrt{x}; \\ v(x) &= e^{-x^2}; \\ x &\in [0; 1] \end{aligned}$
4	$\begin{bmatrix} 4 & 3 & 3 & 3 & 3 & 3 & 9 \\ 3 & 4 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 4 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 4 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 4 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 4 & 3 \\ 9 & 3 & 3 & 3 & 3 & 3 & 4 \end{bmatrix}$	$A = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 3 & 0 & 0 \\ 2 & 2 & 2 & 0 & 3 & 0 \\ 2 & 2 & 2 & 0 & 0 & 3 \end{bmatrix}$ $s = \sum_{k=1}^6 a_{kk}^3$	$\begin{aligned} f(x) &= \ln x; \\ g(x) &= x \cdot \ln x; \\ x &\in [0,2; 10]; \\ u(x) &= x^{1/3}; \\ v(x) &= \sqrt{x}; \\ x &\in [0; 8] \end{aligned}$

5	$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 0 & 0 & 0 & 7 & 0 & 1 \\ 3 & 0 & 0 & 7 & 0 & 7 & 1 \\ 4 & 0 & 7 & 0 & 7 & 0 & 1 \\ 5 & 7 & 0 & 7 & 0 & 0 & 1 \\ 6 & 0 & 7 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 1 & -3 & -3 & -3 & -3 \\ 1 & 1 & -3 & -3 & -3 & -3 \\ -3 & -3 & 2 & 0 & 0 & 0 \\ -3 & -3 & 0 & 2 & 0 & 0 \\ -3 & -3 & 0 & 0 & 2 & 0 \\ -3 & -3 & 0 & 0 & 0 & 2 \end{bmatrix}$ $s = \sum_{i=1}^5 a_{i(i+1)}$	$f(x) =  2x ^3;$ $g(x) =  2x ^5;$ $x \in [-0,5; 0,5];$ $u(x) = \sqrt{ x };$ $v(x) = x^{1/5};$ $x \in [-0,6; 0,5]$
6	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$A = \begin{bmatrix} -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{bmatrix}$ $s = \sum_{i=1}^6 a_{ii} + \sum_{i=1}^5 a_{i(i+1)}$	$f(x) = x^2;$ $g(x) = x^3;$ $x \in [-1; 1];$ $u(x) = x^4;$ $v(x) = x^5;$ $x \in [-1; 1]$
7	$\begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 & 4 \\ 3 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{bmatrix}$ $s = \sum_{i=1}^6 \sum_{j=1}^6 \sin\left(\frac{\pi}{6} a_{ij}^2\right)$	$f(x) = \arcsin(x);$ $g(x) = \arccos(x);$ $x \in [-1; 1];$ $u(x) = \operatorname{arctg}(x);$ $v(x) = \operatorname{arctg}(3x);$ $x \in [-1; 1]$
8	$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 2 & 3 & -1 & 0 & 0 \\ 1 & 2 & 3 & 0 & -1 & 0 \\ 1 & 2 & 3 & 0 & 0 & -1 \\ 0 & 0 & 7 & 2 & 2 & 2 \\ 0 & 7 & 0 & 2 & 2 & 2 \\ 7 & 0 & 0 & 2 & 2 & 2 \end{bmatrix}$ $m = \max_{i=1, \dots, 6} \min_{j=1, \dots, 6} a_{ij}$	$f(x) = \operatorname{sh} x;$ $g(x) = \operatorname{ch} x;$ $x \in [-1; 1];$ $u(x) = e^x;$ $v(x) = e^{-x};$ $x \in [-6; 6]$
9	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 0 & 0 & 1 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 1 & 0 & 0 & 1 & -3 & -3 \\ -2 & -2 & -2 & -2 & 4 & 4 \\ -2 & -2 & -2 & -2 & 4 & 4 \end{bmatrix}$ $s = \sum_{i=1}^6 \max_{j=1, \dots, 6} (a_{ij} + a_{ji})$	$f(x) = \frac{\sin x}{x};$ $g(x) = e^x \cos x;$ $x \in [0, 0,1; 2\pi];$ $u(x) = \sin(\ln(x+1));$ $v(x) = \cos(\ln(x+1));$ $x \in [0; 2\pi]$



10	$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ -1 & -1 & -1 & 4 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & 0 \\ -1 & -1 & -1 & 0 & 0 & 4 \end{bmatrix}$ $p = \prod_{i=1}^6 \sum_{j=1}^6 (a_{ij})^{a_{ij}}$	$f(x) = x^x;$ $g(x) = x^{x^x};$ $x \in [0, 1];$ $u(x) = \frac{1}{1+x};$ $v(x) = \frac{1}{1 + \frac{1}{1+x}};$ $x \in [0, 1]$
11	$\begin{bmatrix} 1 & 0 & 0 & 8 & 0 & 0 & 7 \\ 0 & 2 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 3 & 0 & 5 & 0 & 0 \\ 9 & 0 & 0 & 4 & 0 & 0 & 10 \\ 0 & 0 & 3 & 0 & 5 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 6 & 0 \\ 1 & 0 & 0 & 11 & 0 & 0 & 7 \end{bmatrix}$	$A = \begin{bmatrix} 5 & 0 & 4 & 0 & 0 & 2 \\ 0 & 5 & 0 & 4 & 0 & 2 \\ 3 & 0 & 5 & 0 & 4 & 2 \\ 0 & 3 & 0 & 5 & 0 & 2 \\ 0 & 0 & 3 & 0 & 5 & 2 \\ 1 & 1 & 1 & 1 & 1 & 5 \end{bmatrix}$ $s = \sum_{i=1}^6 \min_{j=1, \dots, 6} (a_{ij} + a_{ji})$	$f(x) = 2x^3 + 1;$ $g(x) = (x+5)^2;$ $x \in [-2; 2];$ $u(x) = x^3;$ $v(x) = e^{-x};$ $x \in [0; 1]$
12	$\begin{bmatrix} 1 & 0 & 3 & 0 & 5 & 0 & 7 \\ 0 & 1 & 0 & 3 & 0 & 5 & 0 \\ 2 & 0 & 1 & 0 & 3 & 0 & 5 \\ 0 & 2 & 0 & 1 & 0 & 3 & 0 \\ 4 & 0 & 2 & 0 & 1 & 0 & 3 \\ 0 & 4 & 0 & 2 & 0 & 1 & 0 \\ 6 & 0 & 4 & 0 & 2 & 0 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 & 1 & 1 \\ 0 & 0 & 4 & 0 & 0 & 1 \\ -1 & 0 & 0 & 8 & 0 & 0 \\ -1 & -1 & 0 & 0 & 16 & 0 \\ -1 & -1 & -1 & 0 & 0 & 32 \end{bmatrix}$ $s = \sum_{i=1}^6 \sqrt{a_{ii}}$	$f(x) = \frac{\ln x}{x};$ $g(x) = \sqrt{x};$ $x \in [0, 2; 10];$ $u(x) = \frac{1}{x+1};$ $v(x) = x^{1/7};$ $x \in [0; 8]$
13	$\begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ -1 & 0 & 1 & 2 & 3 & 0 & 0 \\ 0 & -1 & 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & -1 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & -1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 15 \\ 0 & 3 & 0 & 0 & -7 & 0 \\ 0 & 0 & 5 & 8 & 0 & 0 \\ 0 & 0 & -3 & 7 & 0 & 0 \\ 0 & 5 & 0 & 0 & 9 & 0 \\ -1 & 0 & 0 & 0 & 0 & 11 \end{bmatrix}$ $s = \sum_{i=1}^6 \left( \sqrt{a_{ii}} + \frac{a_{(7-i)i}}{15} \right)$	$f(x) =  x ^3 + 1;$ $g(x) = \ln(2e^{x^2});$ $x \in [-0,7; 0,7];$ $u(x) = \sqrt{ x };$ $v(x) = \sin x;$ $x \in [-0,6; 0,5]$

14	$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 11 & 0 & 1 & 0 & 11 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 11 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 11 & 0 & 1 & 0 & 11 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 11 & 1 & 1 & 11 & 1 \end{bmatrix}$ $s = \sum_{i=1}^6 \sum_{j=1}^6 \tan(a_{ij})$	$f(x) = \sin(2x);$ $g(x) = \cos(x/2);$ $x \in [-\pi, \pi];$ $u(x) = \frac{x^2}{\sin x};$ $v(x) = e^x;$ $x \in [-2; 2]$
15	$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & -3 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 0 & -19 & 0 & 23 & 0 \\ 0 & -1 & 0 & -11 & 0 & 29 \\ 0 & 0 & 3 & 0 & 31 & 0 \\ 0 & 0 & 0 & -7 & 0 & 17 \\ 0 & 0 & 0 & 0 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 \end{bmatrix}$ $s = \sum_{i=1}^6 a_{ii} + \sum_{i=1}^4 a_{i(i+2)} + \sum_{i=1}^2 \sqrt{a_{i(i+4)}}$	$f(x) = x^{2x};$ $g(x) = x^{\sqrt{x}};$ $x \in [0, 1];$ $u(x) = \frac{1}{1 + \sqrt{x}};$ $v(x) = \frac{1}{1 + \frac{1}{1 + \sqrt{x}}};$ $x \in [0, 0.1; 1]$
16	$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & -7 \\ 0 & 2 & 0 & 3 & 0 & -6 & 0 \\ 0 & 0 & 3 & 0 & -5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 7 & 0 \\ 0 & 0 & -3 & 0 & 5 & 0 & 9 \\ 0 & -2 & 0 & 0 & 0 & 6 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 7 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 17 & 4 & 0 & 0 & 0 & 0 \\ 31 & 33 & 9 & 0 & 0 & 0 \\ 22 & 333 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 & 0 \\ 1 & 0 & 0 & 0 & 0 & 36 \end{bmatrix}$ $s = \sum_{i=1}^6 a_{i1} + \sum_{i=2}^4 a_{i2} / 3 + \sum_{i=1}^{26} \sqrt{a_{ii}}$	$f(x) = \log(2x);$ $g(x) = 3x + 4;$ $x \in [0; 10];$ $u(x) = \sin(2x);$ $v(x) = \cos(2x);$ $x \in [0; 2\pi]$
17	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 13 \\ 0 & 3 & 0 & 0 & 0 & 11 & 0 \\ 0 & 0 & 5 & 0 & 9 & 0 & 0 \\ 1 & 3 & 5 & 7 & 9 & 11 & 13 \\ 0 & 0 & 5 & 0 & 9 & 0 & 9 \\ 0 & 3 & 0 & 0 & 0 & 11 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 13 \end{bmatrix}$	$A = \begin{bmatrix} 0 & 0 & 3 & 0 & 0 & -1 \\ 0 & 2 & 0 & 4 & -2 & 0 \\ 1 & 0 & 11 & -3 & 5 & -5 \\ 0 & 10 & 13 & 12 & -4 & 6 \\ 0 & 0 & 9 & 0 & 7 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 \end{bmatrix}$ $s = \sum_{i=2}^4 \sum_{j=2}^4 a_{ij} - \sum_{i=2}^4 \sum_{j=4}^6 (a_{ij})^2$	$f(x) = x^3;$ $g(x) = 13\sqrt{x};$ $x \in [-0.5; 0.5]$ $u(x) = 3x + 5x^2 - x^3;$ $v(x) = 7\sqrt{x} - 4\sqrt[3]{x};$ $x \in [-0.5; 0.5]$

18	$\begin{bmatrix} 11 & 11 & 0 & 11 & 0 & 9 & 9 \\ 11 & 0 & 11 & 11 & 9 & 0 & 9 \\ 0 & -7 & 5 & 5 & 0 & 9 & 0 \\ 0 & -7 & 5 & 0 & 9 & 0 & 9 \\ 0 & -7 & 0 & 5 & 9 & 9 & 0 \\ 3 & 3 & -1 & 0 & 0 & 0 & -1 \\ 3 & 3 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$	$A = \begin{bmatrix} 11 & -9 & -7 & 6 & 5 & 4 \\ -5 & 13 & -1 & 3 & 2 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 3 & -3 & 2 & -2 & 1 & -1 \end{bmatrix}$ $s = \sum_{i=1}^6  a_{ii}  - \sum_{i=1}^6  a_{2i}  + \prod_{i=1}^6 a_{6i}$	$f(x) = \log(\sqrt{x});$ $g(x) = \ln(\sqrt{x});$ $x \in [1; 10];$ $u(x) = \exp(\sqrt{x});$ $v(x) = 2^{\sqrt{x}};$ $x \in [1; 10]$
19	$\begin{bmatrix} 11 & 11 & 11 & 11 & 9 & 9 & 9 \\ 11 & 11 & 11 & 11 & 9 & 9 & 9 \\ -7 & -7 & 5 & 5 & 9 & 9 & 9 \\ -7 & -7 & 5 & 5 & 9 & 9 & 9 \\ -7 & -7 & 5 & 5 & 9 & 9 & 9 \\ 3 & 3 & -1 & -1 & -1 & -1 & -1 \\ 3 & 3 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$	$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 20 & 21 & 22 & 22 & 24 & 7 \\ 19 & 32 & 33 & 34 & 25 & 8 \\ 18 & 31 & 36 & 35 & 26 & 9 \\ 17 & 30 & 29 & 28 & 27 & 10 \\ 16 & 15 & 14 & 13 & 12 & 11 \end{bmatrix}$ $s = \sum_{i=1}^6  a_{ii}  + \sum_{i=1}^6  a_{i(7-i)} $	$f(x) = x^3 + x - 1;$ $g(x) = \sqrt{x} - \sqrt[3]{x};$ $x \in [-0,5; 0,5];$ $u(x) = x - x^3;$ $v(x) = \sqrt{x} / (1 + x);$ $x \in [-0,5; 0,5]$
20	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & -19 & 0 & 0 & 0 & -11 & 0 \\ 0 & 0 & 17 & 5 & 9 & 0 & 0 \\ 0 & 0 & 0 & 13 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & -3 & 0 & 0 & -1 & -1 & -1 \\ -3 & 3 & 0 & 0 & -1 & -1 & -1 \end{bmatrix}$	$A = \begin{bmatrix} 35 & 1 & 6 & 26 & 19 & 24 \\ 3 & 32 & 7 & 21 & 23 & 25 \\ 31 & 9 & 2 & 22 & 27 & 20 \\ 8 & 28 & 33 & 17 & 10 & 15 \\ 30 & 5 & 34 & 12 & 14 & 16 \\ 4 & 36 & 29 & 13 & 18 & 11 \end{bmatrix}$ $s = \sum_{i=1}^6  a_{ii}  - \sum_{i=1}^6  a_{i(7-i)} $	$f(x) = x \log(x);$ $g(x) = \ln(x - 1);$ $x \in [-1; 1];$ $u(x) = \sin^2(x);$ $v(x) = -\cos^2(x);$ $x \in [0; 4\pi]$

# ЧАСТЬ 2. ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ MATLAB

## 1. Введение

Интерактивная работа с MATLAB в режиме командной строки очень удобна при освоении системы и при необходимости поэкспериментировать, чтобы испытать разные подходы к анализу и обработке имеющихся данных. Однако по мере приобретения опыта, а также при решении конкретной задачи вы заметите, что очень часто набираете одни и те же последовательности команд. Это означает, что настало время превратить такие последовательности в программы — сценарии (script) или функции (function). Эти программы и функции представляют из собой текстовые (ASCII) файлы с расширением .m, в которых записаны команды и операторы (MATLAB). Файлы и программы могут создаваться с помощью любого (ASCII) редактора, однако удобнее и комфортнее разрабатывать их с помощью встроенного редактора (начиная с пятой версии MATLAB). Описание редактора и его возможностей будет выполнено позднее, а сначала следует рассмотреть более сложные типы данных по сравнению с переменными и массивами, которые могут быть более востребованными при написании программ.

## 2. Формы и типы данных

Создание программы, как правило, начинается с определения переменных и способа представления данных. Следовательно, чтобы правильно организовать описание данных программы, необходимо знать, как задавать переменные в MATLAB и какие типы данных возможны. Простые типы данных были рассмотрены ранее.

Однако в среде MATLAB поддерживаются объекты с более сложной организацией. Первыми из них будут рассмотрены **многомерные массивы**. Для обращения к их элементам используется необходимое число индексов, однако заполнять многомерный массив числами и выражениями приходится по частям, так как синтаксис MATLAB позволяет записывать поэлементное содержимое только для одно- и двумерных массивов. Создадим трехмерный массив размером  $2 \times 2 \times 2$ :

---

```
>> x(:, :, 1) = [1 2; 3 4]; %первый слой
>> x(:, :, 2) = [5 6; 7 8]; %второй слой
>>x
x(:, :, 1) =
     1     2
     3     4
x(:, :, 2)
     5     6
     7     8
```

---

Вывод значений многомерного массива также осуществляется слоями. Для обработки многомерных данных обычно применяются функции среды MATLAB с отличительным идентификатором *n*, например **fft<sub>n</sub>** и **ifft<sub>n</sub>**.

Следующим типом данных являются **строки**. Для задания строковых констант используются апострофы:

---

```
>> s1 = 'Year';
>> s2 = 'Month';
>> s3 = 'Day';
```

---

С точки зрения среды MATLAB строки представляют собой массивы символов. Поэтому операция соединения строк выполняется не с помощью оператора сложения, а с использованием синтаксиса горизонтального соединения матриц:

---

```
>> s4 = [s3 ' ' s2 ' ' s1]
s4 =
Day Month Year
```

---

Основные функции для работы со строками можно найти по команде **help strfun** в командном окне MATLAB.

Одним из наиболее удобных типов данных, которые используются в программировании в среде MATLAB, является **структура** (structure). Этот тип данных позволяет в одной переменной объединять разнородные данные (поля — field) и осуществлять доступ к ним.

В MATLAB для создания структуры не требуется специальных объявлений — достаточно присвоить значение какому-либо полю. Имена полей указываются после имени переменной через точку. Например, для хранения информации, извлеченной из wav-файла, можно использовать следующую структуру:

---

```
>> w.name = 'c:\windows\media\tada.wav'; %имя файла
>> w.Fs = 22050; %частота дискретизации
>> w.bits = 16; %число битов на отсчет
>> w.channels = 2; %число каналов
>> w.samples = 42752; %число отсчетов
>> w.data = wavread(w.name); %двумерный массив отсчетов
```

---

В среде MATLAB отображаются значения переменных-структур следующим образом:

---

```
>> w
w=
    name: 'c:\windows\media\tada.wav'
      Fs: 22050
     bits: 16
 channels: 2
 samples: 42752
    data: [42752x2 double]
```

---

Значения полей, имеющих короткие представления, выводятся полностью, а для больших массивов указываются только размеры и типы данных. Можно также создавать массивы структур, для этого используются индексы. Например:

---

```
>> w(2).name = 'c:\windows\media\tada.wav'
w=
1x2 struct array with fields
    name
      Fs
     bits
 channels
 samples
    data
```

---

Для массива структур MATLAB показывает только размеры массива и список полей. При работе со структурами периодически используются функции:

- **isstruct(S)** — возвращает истину, если аргумент — структура;
- **isfield(S, 'name')** — возвращает истину, если имеется такое поле;
- **fieldnames(S)** — возвращает массив строк с именами всех полей.

Последним рассматриваемым типом данных является массив ячеек, который, так же как и структура, позволяет в одной переменной объединить разнородные данные. Однако обращение к этим данным производится не по имени полей, а по числовым индексам. Чтобы отличить массив ячеек от обычного массива, его индексы записываются в фигурных скобках. В качестве примера представлен аналогичный применяемой ранее структуре массив ячеек:

---

```
>> w{1} = 'c:\windows\media\tada.wav'; %имя файла
>> w{2} = 22050;                      %частота дискретизации
>> w{3} = 16;                         %число бит на отсчет
>> w{4} = 2;                          %число каналов
>> w{5} = 42752;                      %число отсчетов
>> w{6} = wavread(w.name);            %двумерный массив отсчетов
```

---

В рассмотренном примере применение массива ячеек менее удобно по сравнению со структурой, так как теряется физическая связь каждого элемента ячейки с содержащимся в ней объектом. Однако массивы ячеек удобны в тех случаях, когда нужно создать массив из векторов разной длины или матриц разного размера.

Содержимое ячеек отображается в среде MATLAB следующим образом:

---

```
>> w
w=
    Columns 1 through 5
    [1x25 char]    [22050] [16]    [2]    [42752]
    Columns 6 through 6
    [42752x2 double]
```

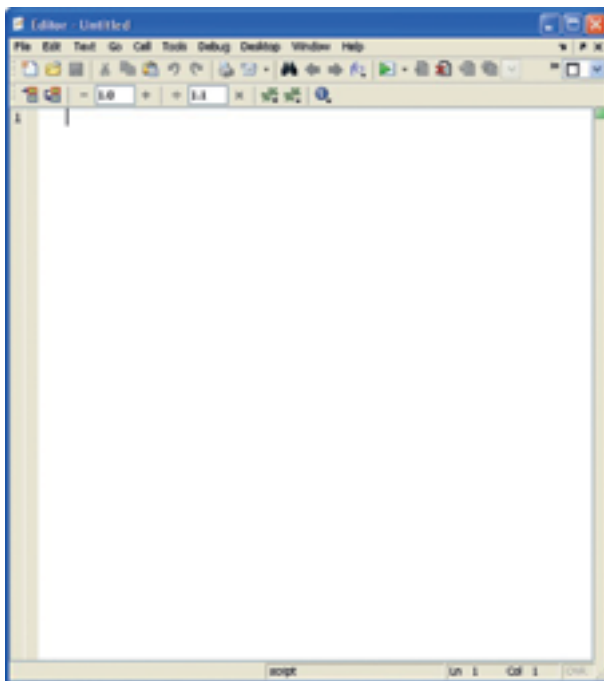
---

При этом переменные выводятся полностью, а для векторов и массивов указываются только размеры и формат данных.

### 3. Работа с отладчиком в редакторе m-файлов

Редактор/отладчик m-файлов позволяет выполнять синтаксический контроль файла, что значительно уменьшает число ошибок по сравнению с режимом работы через командное окно MATLAB. Запуск редактора/отладчика m-файлов производится путем нажатия в главном окне MATLAB кнопки «создать новый файл» (в меню New->m-file) либо через подпункт меню Open->File, после чего необходимо

выбрать файл с расширением .m. Окно редактора/отладчика показано на рис. 2.1.



**Рис. 2.1.** Редактор/отладчик m-файлов

Функции редактирования и работы с файлами являются типичными для любого текстового редактора, поэтому знакомство с ними не представляет особого труда для пользователя ПК. К специфическим возможностям редактора относятся следующие.

1. Цветовое выделение синтаксических элементов языка программирования MATLAB, которое при желании можно изменить через главное меню MATLAB (File->Preferences).

2. Команды меню Text, позволяющие закомментировать выделенный фрагмент текста (Comment), убрать знаки комментария (Uncomment), проверить соответствие правых и левых скобок (Balance Delimiters), а также отформатировать левый край выделенного фрагмента текста (Smart Indent).

Правая часть инструментов содержит кнопки управления отладчиком (также см. меню Debug и Breakpoints), которые представляют наибольший интерес.



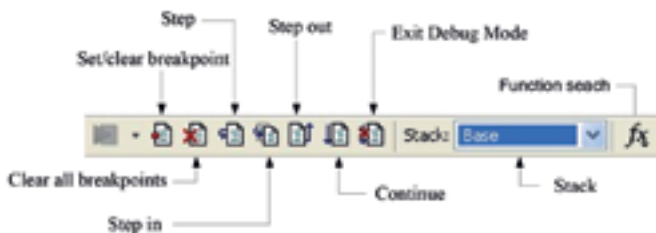
Кнопки панели инструментов имеют следующее назначение:

- Set/clear breakpoint – установка/сброс точки прерывания в выбранной строке;
- Clear all breakpoints – сброс всех точек прерывания;
- Step – выполнение одной строки программы, при этом вызов функций считается одним оператором и заход внутрь вызываемой функции не производится;
- Step in – выполнение одной строки программы с заходом внутрь вызываемой функции;
- Step out – выполнение оставшейся части функции, остановка программы производится после возврата из функции;
- Continue – продолжение выполнения программы;
- Exit Debug Mode – завершение отладки;
- Stack – этот раскрывающийся список позволяет переключаться между рабочим пространством MATLAB и вызываемыми функциями;
- Function seach – утилита поиска интересующей функции в Tootbox.

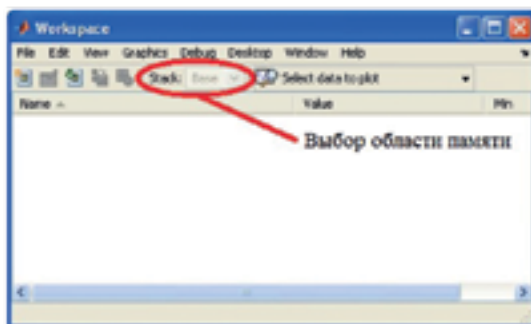
При остановке программы в отладочном режиме (в установленной точке прерывания или при пошаговом выполнении) в главном окне появляется отладочная подсказка K>>. При этом становятся доступными рабочие области памяти всех вызванных в данный момент функций. Переключаться между рабочими областями памяти всего стека функций можно с помощью раскрывающегося списка Stack на панели отладочных инструментов (рис. 2.2) либо на панели инструментов вкладки Workspace (рис. 2.3). Область памяти среды MATLAB фигурирует в списке Stack под именем Base. Для оценки наличия/отсутствия данных и также значения конкретных величин из окна редактора/отладчика необходимо навести курсор мыши на интересующую переменную (в установленной точке прерывания или при пошаговом выполнении). Появится всплывающая подсказка, отображающая значение переменной. Это удобно только для скалярных переменных и массивов небольшого размера, так как для больших массивов отображается только часть содержимого или только размер и тип данных.

#### **4. Программы (скрипты) и функции языка MATLAB**

Язык MATLAB ориентирован на так называемое структурное программирование и имеет много общего с языком Pascal в части синтаксиса. Однако в отличие от всех остальных языков программирования MATLAB характеризуется огромным наполнением в сочетании с от-



**Рис. 2.2.** Отладочные средства на панели инструментов окна редактора/отладчика



**Рис. 2.3.** Отображение списка Stack в рабочей области среды MATLAB

носительной простотой использования большого числа разнообразных встроенных функций, которые в одинаковой степени могут быть задействованы как в программах, так и в работе с командной строкой. Все программы, написанные на языке MATLAB, разделяются на две категории: сценарии или скрипты (Script) и функции (Function). Они отличаются друг от друга следующими положениями:

- 1). функции имеют заголовок function, а сценарии не имеют заголовка;
- 2). функции могут принимать входные параметры и возвращать результаты вычислений, а сценарии – нет;
- 3). программы используют рабочую область памяти среды MATLAB, а каждая функция при вызове создает свою собственную область памяти и общается с внешним миром только через параметры и глобальные переменные;
- 4). функции могут вызываться из программ и других функций в отличие от файлов сценариев, которые могут запускаться только из командной строки либо редактора/отладчика m-файлов.

Файлы обоих типов могут содержать множество команд, операторов и вызовов функций. В качестве разделителя команд используется

оператор «;» или «,». Их применение позволяет записывать последовательно идущие друг за другом команды в одну строку

---

```
a=2^64; b=sqrt(3); c=a+b;
```

---

или

---

```
a=2^64;  
b=sqrt(3);  
c=a+b;
```

---

Отличие между двумя разделителями заключается в том, что вывод результата в командное окно происходит лишь при использовании «;».

Для улучшения читаемости кода программы длинные строки можно делить на части, используя оператор «...». Пример выглядит следующим образом:

---

```
M=[a b c d; ...  
    b f d a; ...  
    a c e f];
```

---

Важным атрибутом любой программы являются комментарии, поясняющие выполнение той или иной операции. Для обозначения комментариев в MATLAB используется знак «%». Блочные комментарии в среде отсутствуют, поэтому для комментирования нескольких строк необходимо в начале каждой из них ставить символ «%». Для блочного выделения определенного куса программы используется символ «%%». Он выполняет функцию комментария для первой строки, остальные просто подсвечиваются вплоть до повторного применения оператора «%%».

Как было сказано ранее, существует ряд различий в задании сценариев и функций в среде MATLAB. Однако большим количеством нюансов обладают файлы типа *function*, поэтому рассмотрим их более подробно.

Текст *m*-файла функции должен начинаться с заголовка *function*, имеющего следующий вид:

---

```
function [y1, y2, ...] = fname(x1, x2, ...)  
<тело функции>
```

---

Здесь **fname** — имя функции, по которой можно выполнять поиск в среде, также получить справку, *x1*, *x2* и т.д. — входные параметры, *y1*,

y2 и т.д. — выходные параметры, которые могут присутствовать или нет. На самом деле имя функции определяется не строкой **fname**, а именем, под которым сохранен m-файл. По этой причине имена функций, в отличие от имен переменных, не чувствительны к регистру символов. Также стоит отметить что для уменьшения путаницы стоит называть функцию и сохранять соответствующий ей m-файл одним именем. Рассмотрим пример реализации функции для вычисления евклидова расстояния:

---

```
function len = euclid(x1, y1, x2, y2)
len = sqrt((x1-x2)^2+(y1-y2)^2);
```

---

Продemonстрируем возможность возвращения нескольких параметров на примере вычисления ширины и высоты прямоугольника, заданного координатами левого верхнего угла (x1,y1) и правого нижнего (x2,y2):

---

```
function [width, height] = RectangleHW(x1,y1,x2,y2)
width = abs(x1-x2);
height = abs(y1-y2);
```

---

Данную функцию можно записать еще и с таким набором параметров:

---

```
function [width, height] = RectangleHW(P1, P2)
%Вычисление ширины (width) и высоты (height) прямоугольника
a = abs(P1(1)-P1(2));
b = abs(P2(1)-P2(2));
width=a;
height=b;
```

---

где P1 и P2 — векторы (массивы) размером в два элемента, описывают точку в двумерном пространстве. В последнем случае строка, введенная как комментарий, — это встроенная справка об использовании нашей функции. По команде **help RectangleHW** в командное окно выводятся строки комментария, расположенные в файле **fname.m** непосредственно после заголовка функции, а именно:

---

```
>> help RectangleHW
Вычисление ширины (width) и высоты (height) прямоугольника
```

---

Таким образом, можно написать сколь угодно подробную справку для создаваемой функции. Переменные *a* и *b*, создаваемые внутри

функции, являются локальными. Они существуют только во время выполнения команд текущей функции и уничтожаются при завершении ее работы. Если переменные с такими же именами существовали в рабочей области памяти MATLAB до вызова функции, их значения не изменятся.

Отдельно стоит отметить возможность использования переменных внутри функций и сценариев, а именно:

- переменные не требуют объявления; прежде чем переменной присвоить значение, необходимо убедиться, что всем переменным в правой части значения были присвоены ранее;
- любая операция присваивания создает переменную, если это необходимо, или изменяет значение существующей переменной;
- имена переменных начинаются с буквы, за которой следует любое количество букв, цифр и подчеркиваний; система MATLAB различает символы верхнего и нижнего регистров;
- имя переменной не должно превышать 31 символ, так как MATLAB воспринимает только первый 31 символ и игнорирует последующие.

Функция общается с внешним миром только через свои параметры и глобальные переменные. Для объявления глобальной переменной  $x$  необходимо использовать специальную директиву, которая также предварительно прописывается и в командном окне:

---

**global**  $x$

---

Для работы с глобальными переменными необходимо:

- объявить переменную как глобальную во все функции, где ее использование целесообразно;
- в каждой функции, желательно в начале, использовать команду **global** перед первым объявлением переменной.

Имена глобальных переменных обычно более длинные и более содержательные, чем имена локальных переменных, и часто используют заглавные буквы, что упрощает их визуальный поиск и идентификацию среди прочих переменных в рабочей области среды MATLAB.

Для обеспечения большей гибкости при работе с программой часто в теле функции используется оператор досрочного выхода — **return**. В любом случае результатами работы функции являются значения, которые выходные параметры имели на момент возврата.

При вызове функции можно использовать переменное количество входных и выходных параметров. Для этого вместо привычных пере-

менных в заголовке функции используются ключевые слова **varargin** и **varargout**. Система MATLAB упаковывает все заданные входные и выходные аргументы в массив ячеек. Каждая ячейка может содержать любой тип и любое количество данных. Типичный заголовок функции в таком случае приобретает вид

---

```
function [varargout] = myFunction(varargin)
```

---

Чтобы определить при каждом запуске текущее количество входных и выходных аргументов при их переменном числе, следует пользоваться ключевыми словами **nargin** и **nargout**. Обычно использование рассмотренных ключевых слов уместно для серьезного управления логикой работы программы, в частности для изменения используемого алгоритма в зависимости от числа входных и выходных параметров.

## **5. Логика работы программ. Условный оператор, ветвления, циклы в программах MATLAB**

Для формирования логических условий в среде MATLAB чаще всего используются операторы сравнения **==** (равно), **~=** (не равно), **<** (меньше), **>** (больше), **<=** (меньше или равно), **>=** (больше или равно). Следует иметь в виду, что для массивов они производят поэлементное сравнение, возвращая массив того же размера, как сравниваемые аргументы. Результирующий массив содержит единицы там, где сравнение элементов дало выполнение условия, а нули там, где условие сравнения выполнено не было.

Для формирования логических условий полезны также следующие функции:

- **all(x)** — возвращает единицу, если все элементы **x** отличны от нуля;
- **any(x)** — возвращает единицу, если хотя бы один элемент **x** отличен от нуля;
- **isequal(x,y)**. В отличие от конструкции **all(x==y)**, использование функции **isequal(x,y)** не приведет к ошибке, если размеры **x** и **y** не совпадают;
- **isempty(x)** — возвращает единицу, если **x** является пустой матрицей (т.е. ее размер равен  $0 \times 0$ ).

Имеется еще довольно много функций проверки разнообразных условий, имена которых начинаются с латинских букв **is...** . Подробнее узнать о том, что это за функции и каково их назначение, можно, введя **is** в командное окно MATLAB и нажав кнопку **Tab**, либо через справочную систему.

Условный оператор в MATLAB реализуется с помощью ключевых слов **if**, **else** и **end**:

---

```
if cond
    % ветвь, выполняемая, если cond истинно
    <операторы>
else
    % ветвь, выполняемая, если cond ложно
    <операторы>
end
```

---

Здесь переменная **cond** выступает в качестве проверяемого значения, например **cond**=(**x**>=0.5). При необходимости можно использовать оператор **if** с большим числом ветвлений:

---

```
if cond1
    % ветвь, выполняемая, если cond1 истинно
    <операторы>
elseif cond2
    % ветвь, выполняемая, если cond2 истинно, а cond1 ложно
    <операторы>
else
    % ветвь, выполняемая, если cond1 и cond2 ложны
    <операторы>
end
```

---

Аналогичным образом можно использовать **elseif** несколько раз.

Для компактной реализации процедуры выбора среди множества альтернатив в среде MATLAB используется оператор **switch**. Запись оператора выбора производится с помощью ключевых слов **switch**, **case**, **otherwise** и **end**:

---

```
switch x
    case x1
        % ветвь, выполняемая, если x равно x1
        <операторы>
    case x2
        % ветвь, выполняемая, если x равно x1
        <операторы>
    otherwise
        % ветвь, выполняемая, если все условия case
        не соблюдаются
```

## <операторы>

**end**

---

При выполнении оператора **switch** значение  $x$  поочередно сравнивается с  $x_1$ ,  $x_2$  и т.д. При обнаружении первого совпадения выполняются операторы соответствующей ветви, после чего производится выход из оператора выбора. Если такие совпадения отсутствуют, выполняется последний оператор **otherwise**.

Для реализации циклов в MATLAB имеется два оператора — **for** и **while**. В операторе **for** переменная-счетчик цикла поочередно принимает значения элементов некоторого вектора:

---

```
for k=x
    % тело цикла
    <операторы>
end
```

---

Операторы, входящие в тело цикла, будут выполняться при значении переменной  $k$ , равном  $x(1)$ , затем  $x(2)$  и т.д. до  $x(\text{end})$ . После завершения цикла значение  $k$  остается равным  $x(\text{end})$ . Возможны различные варианты использования счетчика:

---

```
for k=1:N
for k=-1:0.01:1
for k=[1 10 pi 9 ln(3)]
```

---

Второй тип циклов реализуется с помощью оператора **while**. Тогда тело цикла выполняется, пока условие **cond** остается истинным (т.е. значение **cond** отлично от нуля):

---

```
while cond
    % тело цикла
    <операторы>
end
```

---

Если **cond** изначально имеет нулевое значение, тело цикла не будет выполнено ни разу. Чтобы реализовать досрочное завершение цикла, используется оператор **break**. Разумеется, он должен применяться в сочетании с условным оператором **if**:

---

```
for k=1:N
```



```

% первая половина тела цикла
<операторы>
% проверка дополнительного условия завершения
if cond
    break;
end
% вторая половина тела цикла
<операторы>
end

```

---

Оператор **break** можно использовать для реализации циклов с проверкой условия завершения не в начале (как делает оператор **while**), а в произвольном месте цикла. Для этого с помощью операции **while** создается «вечный» цикл, а его завершение производится оператором **break**. Кроме оператора **break** для управления выполнением программы внутри цикла имеется оператор **continue**. Размещенный внутри цикла оператор **for** или **while** передает управление на проверку условия завершения цикла, пропуская при этом оставшийся фрагмент тела цикла. Это позволяет сделать код более наглядным, избежав использования длинного оператора **if**.

Одна из возможных реализаций бесконечного цикла с условным выходом:

```

while 1    % «вечный» цикл
% первая половина тела цикла
<операторы>
% проверка условия завершения
if cond
    break;
end
% вторая половина тела цикла
<операторы>
end

```

---

## 6. Практическая работа. Разбор готовой программы на элементарные составляющие. Составление блок-схем алгоритмов

Для того чтобы облегчить самостоятельное написание программного обеспечения на языке MATLAB, следует начать с анализа готовых программ, составления алгоритмов их работы и разработки блок-схем.

Это позволит освоить синтаксис языка и лучше понимать возможности, предоставляемые в MATLAB. В соответствии с этими умозаключениями в рамках практической части работы предлагается выполнить задание, состоящее из двух частей.

### 1. Анализ готовой программы. Составление таблицы вида.

		Наименование объекта
Тип данных	Переменные	Указать название и номера строк(и), в которых используется рассматриваемая переменная, вектор, массив, строка, структура, ключевое слово или функция (встроенная либо пользовательская)
	Векторы	
	Массивы	
	Строки	
	Структуры	
Функции в программе	Встроенные	
	Пользовательские	
Ключевые слова, используемые в программе		
Если в программе встречаются символы «?», определить исходя из типа данных, что должно быть поставлено вместо этих знаков		Указать номер строки (строк) и наименование переменной(ых), которая может подойти на место (места), где расположены символы «?»


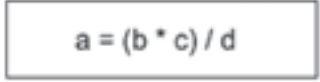




Для выполнения первого задания потребуется:

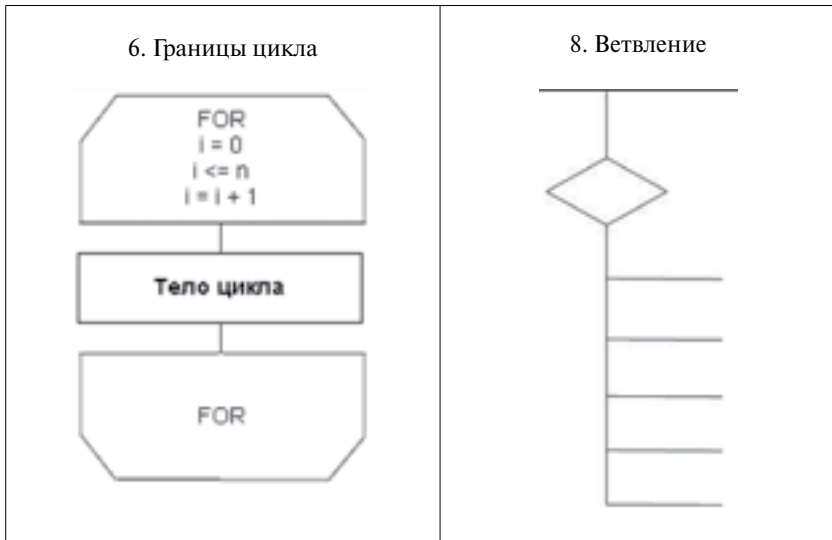
- выбрать вариант задания в соответствии со своим номером в списке группы, все варианты представлены в заключение параграфа 6;
- ввести код программы в редакторе кода (Editor) среды MATLAB;
- отладить работу программы в пошаговом режиме исполнения; программа, вероятнее всего, не запустится с первого раза в связи с внесением в нее ошибок при вводе (будьте предельно внимательны) и с наличием в ней пользовательских функций, которые не находятся в вашей рабочей папке (в этом случае нужно обратиться к преподавателю и получить необходимые программные коды);
- после безошибочного выполнения программы заполнить таблицу в части «Наименования объекта» обнаруженными переменными, векторами, массивами и т.д., соблюдая заданную классификацию;

- представить преподавателю **заполненную таблицу по вашей программе.**
2. Подготовка блок-схемы алгоритма работы программы.  
Блок-схема алгоритма составляется в соответствии с логикой работы программы. К основным блокам, составляющим схему, относятся (табл. 2.1):

Таблица 2.1.

**Основные элементы для реализации блок-схем алгоритмов**

<p>1. Терминатор (начало – конец программы)</p> 	<p>3. Процесс</p> 
<p>2. Ввод данных</p> 	<p>4. Предопределенный процесс (функция)</p> 
<p>5. Решение (условный оператор)</p> 	<p>7. Соединитель (при переходе соединительных стрелок)</p> 



В качестве примера рассмотрим код программы в среде MATLAB и соответствующую ей блок-схему алгоритма. Программа выполняет модуляцию входного двоичного потока с данными тремя способами – изменением амплитуды (ASK), частоты (FSK) и фазы (PSK) гармонического колебания по закону, соответствующему входной двоичной последовательности.

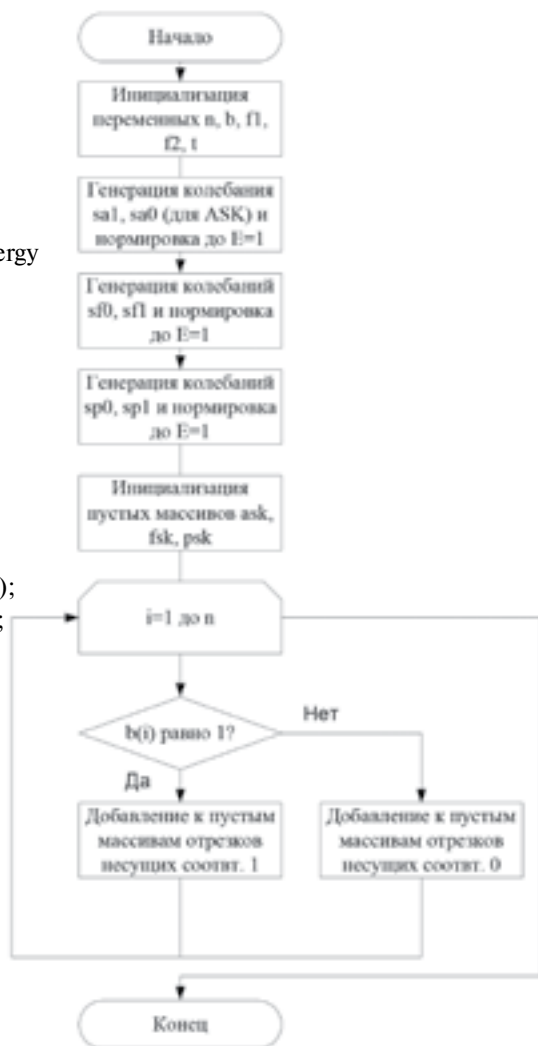
Каждому студенту в соответствии с вариантом задания необходимо выбрать код программы, ввести его в редакторе (если это еще не было сделано), добиться безошибочного выполнения программы и изобразить для нее блок-схему алгоритма по аналогии с примером, изображенным на рис. 2.4. Настоятельно рекомендуется использовать элементарные блоки, изображенные в табл. 2.1 для отображения различных действий и логических операций, выполняемых в коде программы.

В результате выполнения второго пункта практического задания преподавателю необходимо представить **блок-схему алгоритма** и **продемонстрировать работоспособность введенного кода программы**. Важно отметить, что для некоторых вариантов программы для составления таблицы по п. 1 и разработки блок-схемы по второму пункту могут отличаться. Однако работоспособность программ в любом случае должна быть достигнута.

```

n=msglen;
b=randint(1,n);
f1=1;f2=2;
t=0:1/30:1-1/30;
%ASK
sa1=sin(2*pi*f1*t);
E1=sum(sa1.^2);
sa1=sa1/sqrt(E1); %unit energy
sa0=0*sin(2*pi*f1*t);
%FSK
sf0=sin(2*pi*f1*t);
E=sum(sf0.^2);
sf0=sf0/sqrt(E);
sf1=sin(2*pi*f2*t);
E=sum(sf1.^2);
sf1=sf1/sqrt(E);
%PSK
sp0=-sin(2*pi*f1*t)/sqrt(E1);
sp1=sin(2*pi*f1*t)/sqrt(E1);
%MODULATION
ask=[];psk=[];fsk=[];
for i=1:n
    if b(i)==1
        ask=[ask sa1];
        psk=[psk sp1];
        fsk=[fsk sf1];
    else
        ask=[ask sa0];
        psk=[psk sp0];
        fsk=[fsk sf0];
    end
end
end

```



**Рис. 2.4.** Программа и соответствующая ей блок-схема алгоритма

### Вариант 1.

**Моделирование системы передачи данных с амплитудной модуляцией и когерентной демодуляцией:**

```
clear, clf
b=1;
M=2^b;
Tb=1;
Ts=b*Tb;
Ns=40;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
A=sqrt(Es);
wc=10*pi/Ts;
wcT=wc*T;
t=[0:Ns-1]*T;
tt=[0:LB-1]*T;
ss=[0; 1];
su(1,:)=sqrt(2/Ts)*cos(wc*t);
suT=su*T;
sw=[A*su(1,:); zeros(1,Ns)];
SNRdBs=[1:12];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRdB= SNRdBs(iter);
    SNR=10^(SNRdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    st=zeros(1,LB);
    y=zeros(1,LB);
    r=zeros(1,LB);
    yr=zeros(1,LB);
    nobe= 0;
    thrshld= sqrt(Es)/2;
    for k=1:MaxIter
        i= ceil(rand*M);
        s=ss(i);
        for n=1:Ns
            st=[st(2:LB) sw(i,n)];
```

```

    yn= suT*st(LBN1:LB)';
    y=[y(:,2:LB) yn];
    wct=wcT*(n-1);
    bp_noise= randn*cos(wct)-randn*sin(wct);
    r=[r(2:LB) sw(i,n)+sgmsT*bp_noise];
    yrn= suT*r(LBN1:LB)';
    yr=[yr(:,2:LB) yrn];
end
if iter==10
    subplot(223);
    hold on;
    if s==0
        plot(k,yrn(1),'o');
    else
        plot(k,yrn(1),'*');
    end
end
d= (yrn(1)<thrshld);
nobe = nobe+(s~=d);
if nobe>100
    break;
end
end
pobe(iter)= nobe/(k*b);
end
SNRdBt=0:0.1:12;
SNRt=10.^(SNRdBt/10);
poe_on_theory= Q(sqrt(SNRt/4));

```

## **Вариант 2.**

**Моделирование системы передачи данных с амплитудной модуляцией и некогерентной демодуляцией:**

```

clear, clf
b=1;
M=2^b;
Tb=1;
Ts=b*Tb;
Ns=40;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;

```

```

Es=2;
A=sqrt(Es);
wc=10*pi/Ts;
wcT=wc*T;
t=[0:Ns-1]*T;
wct=wc*t;
tt=[0:LB-1]*T;
ss=[0;1];
su=sqrt(2/Ts)*[cos(wct); -sin(wct)];
suT=su*T;
sw(1,:)= A*su(1,:);
sw(2,:)= zeros(1,Ns);
SNRdBs=[1:12];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRdB= SNRdBs(iter);
    SNR=10^(SNRdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    st=zeros(1,LB);
    y=zeros(1,LB);
    r=zeros(1,LB);
    yr=zeros(1,LB);
    nobe= 0;
    thrshld= Es/2;
    for k=1:MaxIter
        i=ceil(rand*M);
        s=ss(i);
        for n=1:Ns
            st=[st(2:LB) sw(i,n)];
            yn=su*st(LBN1:LB)'*T;
            y=[y(:,2:LB) yn(1)^2+yn(2)^2];
            wct= wcT*(n-1);
            bp_noise= randn*cos(wct)-randn*sin(wct);
            r=[r(2:LB) sw(i,n)+sgmsT*bp_noise];
            yrn=suT*r(LBN1:LB)';
            yrn(1)^2+yrn(2)^2];
        end
    if iter==10
        subplot(223)

```



```

    if s==0
        plot(k,yr(LB),'o');
    else
        plot(k,yr(LB),'*');
    end
    hold on
end
d= (yr(LB)<thrshld);
nobe = nobe+(s~=d);
if nobe>100
    break;
end
end
pobe(iter)= nobe/(k*b);
end

```

### Вариант 3.

**Моделирование системы передачи данных с дифференциальной фазовой модуляцией:**

```

clear, clf
b=2;
M=2^b;
SNRbdBt=0:0.1:10;
SNRbt=10.^(SNRbdBt/10);
pobet= (1+(b>1))/b*Q(sqrt(b*SNRbt/2)*sin(pi/M));
Tb=1;
Ts=b*Tb;
Nb=16;
Ns=b*Nb;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
sqEs=sqrt(Es);
ss=[0 0; 0 1; 1 1; 1 0];
phases=[0:M-1]*2*pi/M;
gs='>^<vs*x+d';
wc=8*pi/Ts;
t=[0:Ns-1]*T;
wcT=wc*T;
nd=1;

```

```

for m=1:M
    sw(m,:)= sqrt(2*Es/Ts)*cos(wc*t+phases(m));
end
su= sqrt(2/Ts)*[cos(wc*t); -sin(wc*t)];
suT= su*T;
SNRdBs=[1:10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRdB= SNRdBs(iter);
    SNRb=10^(SNRdB/10);
    N0=2*(Es/b)/SNRb;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    sws= zeros(1,LB);
    yr= zeros(2,LB);
    nobe=0;
    is0=1;
    th0=1;
    for k=1:MaxIter
        i= floor(rand*M);
        s=ss(i+1,:);
        is= mod(is0+i,M);
        is1=is+1;
        for n=1:Ns
            sws=[sws(2:LB) sw(is1,n)];
            wct= wcT*(n-1);
            bp_noise= randn*cos(wct)-randn*sin(wct);
            rn= sws(end-nd)+sgmsT*bp_noise;
            yr= [yr(:,2:LB) suT(:,n)*rn];
        end
        ycsk= sum(yr(:,LBN1:LB)');
        if iter==8&k<300
            subplot(221);
            hold on;
            plot(ycsk(1),ycsk(2),gs(i+1));
        end
        th=atan2(ycsk(2),ycsk(1));
        dth=th-th0;
        if dth<-pi/M
            dth=dth+2*pi;
        end
    end
end

```

```

[themmin,mmin]=min(abs(dth-phases));
d=ss(mmin,:);
nobe=nobe+sum(s~=d);
if nobe>100
    break;
end
is0=is;
th0=th;
end
pobe(iter)= nobe/(k*b);
end

```

#### **Вариант 4.**

**Моделирование системы передачи данных с частотной модуляцией и когерентной демодуляцией:**

```

clear, clf
b=1;
M=2^b;
SNRbdBt=0:0.1:10;
SNRbt=10.^(SNRbdBt/10);
ss=[0; 1];
pobet= Q(sqrt(SNRbt/2));
Tb=1;
Ts=b*Tb;
Ns=40;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
wc=8*pi;
dw=2*pi;
w=wc+[0:M-1]*dw;
wcT=wc*T;
t=[0:Ns-1]*T;
tt=[0:LBN1-1]*T;
nd=0;
td=0;
for m=1:M
    su(m,:)=sqrt(2/Ts)*cos(w(m)*(t-td));
    sw(m,:)=sqrt(2*Es/Ts)*cos(w(m)*t);
end
suT= su*T;

```

```

SNRdBs=[1:10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRdB= SNRdBs(iter);
    SNR=10^(SNRdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    st=zeros(1,LB);
    y=zeros(M,LB);
    r=zeros(1,LB);
    yr=zeros(M,LB);
    nobe=0;
    for k=1:MaxIter
        i= ceil(rand*M);
        s=ss(i,:);
        for n=1:Ns
            st=[st(2:LB) sw(i,n)];
            yn=suT*st(LBN1:LB)';
            y=[y(:,2:LB) yn];
            wct=wcT*(n-1);
            bp_noise= randn*cos(wct)-randn*sin(wct);
            r=[r(2:LB) sw(i,n)+sgmsT*bp_noise];
            yrn= suT*r(LBN1:LB)';
            yr=[yr(:,2:LB) yrn];
        end
        [yremax,lmax]= max(yrn);
        d=ss(lmax,:);
        nobe=nobe+sum(s~=d);
        if nobe>100
            break;
        end
    end
    pobe(iter)= nobe/(k*b);
end

```

### Вариант 5.

**Моделирование системы передачи данных с частотной модуляцией и некогерентной демодуляцией:**

```

clear, clf
b=1;
M=2^b;

```

```

SNRbdBt=0:0.1:10;
SNRbt=10.^(SNRbdBt/10);
pobet=exp(-SNRbt/4)/2;
Tb=1;
Ts=b*Tb;
Ns=32;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
wc=8*pi/Ts;
dw=2*pi/Ts;
w=wc+[0:M-1]*dw;
wcT=wc*T;
t=[0:Ns-1]*T;
tt=[0:LB-1]*T;
nd=0;
for m=1:M
    sw(m,:)=sqrt(2*Es/Ts)*cos(w(m)*t);
    su(2*m-1:2*m,:)=sqrt(2/Ts)*[cos(w(m)*t); sin(w(m)*t)];
end
suT=su*T;
ss=[0; 1];
SNRdBs=[1:10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRbdB= SNRdBs(iter);
    SNR=10^(SNRbdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    st=zeros(1,LB);
    sws=zeros(1,LB);
    y=zeros(M,LB);
    yn=zeros(M,1);
    r=zeros(1,LB);
    yr=zeros(M,LB);
    yrn=zeros(M,1);
    nobe= 0;
    for k=1:MaxIter
        i=ceil(rand*M);

```

```

s=ss(i,:);
sws=[sws(2:LB) sw(i,n)];
st=[st(2:LB) sws(end-nd)];
ycss=suT*st(LBN1:LB)';
for m=1:M
    yn(m)=ycss(2*m-1)^2+ycss(2*m)^2;
end
y=[y(:,2:LB) yn];
wct=wcT*(n-1);
bp_noise= randn*cos(wct)-randn*sin(wct);
r=[r(2:LB) sws(end-nd)+sgmsT*bp_noise];
ycss=suT*r(LBN1:LB)';
for m=1:M
    yrn(m)=ycss(2*m-1)^2+ycss(2*m)^2;
end
yr=[yr(:,2:LB) yrn];
end
[yremax,mmax]=max(yrn);
d=ss(mmax,:);
nobe=nobe+sum(s~=d);
if nobe>100
    break;
end
end
pobe(iter)= nobe/(k*b);
end

```

## Вариант 6.

### Моделирование системы передачи данных с MSK-модуляцией:

```

b=1;
M=2^b;
SNRbdBt=0:0.1:12;
SNRbt=10.^(SNRbdBt/10);
pbe_PSK= prob_error(SNRbdBt,'PSK',b,'bit');
pbe_FSK= prob_error(SNRbdBt,'FSK',b,'bit');
Tb=1;
Ts=b*Tb;
Nb=16;
Ns=Nb*b;
T=Ts/Ns;
LB=4*Ns;

```

```

LBN2=LB-2*Ns+1;
Es=2;
ds=[0 1];
wc=2*pi;
wcT=wc*T;
t=[0:2*Nb-1]*T;
pihTbt=pi/2/Tb*t;
su=sqrt(2/Tb)*[cos(pihTbt-pi/2).*cos(wc*(t-Tb)); ?];
su=[fftshift(su(1,:)); su(2,:)];
suT=su*T;
ik=[1 2 1 2 1 1 2 2 2 1];
sq2EbTb=sqrt(2*Es/b/Tb);
pihTbT=pi/2/Tb*T;
SNRdBs=[1 3 10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRdB=SNRdBs(iter);
    SNR=10^(SNRdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    yr=zeros(2,LB);
    yc=0;
    ys=0;
    th0=0;
    t=0;
    wct=0;
    tht=th0;
    mn=1;
    nobe= 0;
    for k=1:MaxIter
        i=ceil(rand*M);
        s(k)=ds(i);
        sgn=s(k)*2-1;
        for m=1:Nb
            bp_noise= randn*cos(wct)-randn*sin(wct);
            rn= sq2EbTb*cos(wct+tht) + sgmsT*bp_noise;
            yr=[yr(:,2:LB) suT(:,mn)*rn];
            t=t+T;
            wct=wct+wcT;
            tht=tht+sgn*pihTbT;
        end
    end
end

```

```

        mn=mn+1;
    end
    if tht<-pi2
        tht=tht+pi2;
    elseif tht>pi2
        tht=tht-pi2;
    end
    thtk(k+1)=tht;
    if mn>size(su,2)
        mn=1;
    end
    if mod(k,2)==1
        yc=sum(yr(? ,LBN2:LB));
        th_hat(k)=??*(yc<0);
    else
        ys=sum(yr(? ,LBN2:LB));
        th_hat(k)=???*(2*(ys<0)-1);
    end
    if k>1
        d=(dth_hat>0);
        if abs(dth_hat)>=pi
            d=~d;
        end
        nobe = nobe+(d~=s(???));
        if nobe>100;
            break;
        end
    end
end
pobe(iter)= nobe/((k-1)*b);
end

```

### **Вариант 7.**

#### **Моделирование системы передачи данных с OQPSK- модуляцией:**

```

clear, clf
b=2;
M=2^b;
SNRbdBt=0:0.1:10;
SNRbt=10.^(SNRbdBt/10);
pobet=prob_error(SNRbdBt,'PSK',b,'bit');
Tb=1;

```



```

Ts=b*Tb;
Nb=16;
Ns=Nb*b;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
sqEb=sqrt(Es/b);
ss=[0 0; 0 1; 1 1; 1 0];
wc=2*pi/Ts;
wcT=wc*T;
t=[0:Ns-1]*T;
su=sqrt(2/Ts)*?;
suT=su*T;
sw=sqEb*su;
SNRdBs=[1:3:10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRdB=SNRdBs(iter);
    SNRb=10^(SNRdB/10);
    N0=2*(Es/b)/SNRb;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    yr= zeros(2,LB);
    yc=zeros(1,2);
    ys=zeros(1,2);
    iq=0;
    nobe=0;
    for k=1:MaxIter
        i=ceil(rand*M);
        s=ss(i,:);
        wct=-wcT;
        for n=1:b
            if n==1
                ii=2*ss(i,1)-1;
            else
                iq=2*ss(i,2)-1;
            end
            mn=0;
            for m=1:Nb
                wct= wct+wcT;

```

```

mn=mn+1;
bp_noise= randn*cos(wct)-randn*sin(wct);
rn=ii*sw(1,mn)+iq*sw(2,mn)+sgmsT*bp_noise;
yr=[yr(:,2:LB) suT(:,mn)*rn];
end
if n==2
yc=[yc(2) sum(yr(:,LBN1:LB))];
else
ys=[ys(2) sum(yr(:,LBN1:LB))];
end
end
d=(|yc(?) ys(?)|>0);
if k>1
nobe=nobe+sum(s0~=d);
end
if nobe>100
break;
end
s0= s;
end
pobe(iter)= nobe/(k*b);
end

```

### **Вариант 8.**

**Моделирование системы передачи данных с ортогональными сигналами:**

```

clear, clf
b=1;
M=2^b;
Tb=1;
Ts=b*Tb;
Nb=20;
Ns=b*Nb;
T=Tb/Nb;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
su=[1 1; 1 -1];
su=ones(Ns/M,1)*reshape(su',1,M*M);
su=reshape(su(:),Ns,M)'/sqrt(Ts);
suT=su*T;
sw=sqrt(Es)*su;

```

```

for m=1:M
    ss(m,:)=deci2bin(m-1,b);
end
SNRdBs= [1:10];
N_Iter=6000;
for iter=1:length(SNRdBs)
    SNRdB= SNRdBs(iter);
    SNR= 10^(SNRdB/10);
    sigma2=(Es/b)/SNR;
    sgmsT=sqrt(sigma2/T);
    st= zeros(1,LB);
    y= zeros(M,LB);
    r= zeros(1,LB);
    yr= zeros(M,LB);
    nobe= 0;
    for k=1:N_Iter
        i= ceil(rand*M);
        s=ss(i,:);
        for m=1:Ns
            st=[st(2:LB) sw(i,m)];
            y=[y(:,2:LB) suT*st(LBN1:LB)'];
            r=[r(2:LB) sw(i,m)+sgmsT*randn];
            yre= suT*r(LBN1:LB).';
            yr=[yr(:,2:LB) yre];
        end
        [ymax,imax]=max(yre);
        d = ss(imax,:);
        nobe = nobe+sum(d~=s);
        if nobe>100
            break;
        end
    end
    pobe(iter)= nobe/(k*b);
end

```

### **Вариант 9.**

**Моделирование системы передачи данных с ФМ-модуляцией (для составления таблицы):**

```

clear, clf
Ac=1;
fc=50;
wc=2*pi*fc;

```

```

Tb=0.1;
T=1/fc/8;
Fs=1/T;
Nb=Tb/T;
lt=2^(nextpow2(3*Nb));
t=[1:lt]*T;
m=ones(Nb,1)*[4 -8 -4];
m=m(:).';
m=[m, zeros(1,lt-length(m))];
int_m(1)=0;
T2=T/2;
for i=1:lt-1
    int_m(i+1)=int_m(i)+(m(i)+m(i+1))*T2;
end
kF=30;
m_FM=Ac*cos(wc*t+kF*int_m);
th=unwrap(angle(hilbert(m_FM)))-wc*t;
y_FM=[0 diff(th)/T/kF];

```

**Моделирование системы передачи данных с РМ-модуляцией (для написания блок-схемы):**

```

clear, clf
Ac=1;
fc=50;
wc=2*pi*fc;
Tb=0.1;
ts=1/fc/8;
fs=1/ts;
Nb=Tb/ts;
lt=2^(nextpow2(3*Tb/ts));
t=[1:lt]*ts;
m=4*[ones(1,Nb), -2*ones(1,Nb), -ones(1,Nb)];
m=[m, zeros(1,lt-length(m))];
kP=0.01;
kP=30;
m_PM=Ac*cos(wc*t+kP*m);
th=unwrap(angle(hilbert(m_PM)))-wc*t;
y_PM=th/kP;
plot_MOD(ts,lt,m,m_PM,y_PM,'PM')
t=[1:lt]*ts;
m_PM2=modulate(m,fc,fs,'pm',kP);
y_PM2=demod(m_PM2,fc,fs,'pm',kP);

```

### Вариант 10.

#### Моделирование системы передачи данных с QPSK-модуляцией:

```
clear, clf
b=2;
M=2^b;
SNRbdBt=0:0.1:10;
SNRbt=10.^(SNRbdBt/10);
pobet= (1+(b>1))/b*Q(sqrt(b*SNRbt)*sin(pi/M));
Tb=1;
Ts=b*Tb;
Nb=32;
Ns=b*Nb;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
sqEs=sqrt(Es);
ss=[0 0; 0 1; 1 1; 1 0];
g_code=gray_code(b);
for m=1:M
    ss(m,:)=deci2bin(g_code(m),b);
end
phases=2*pi/M*[0:M-1];
gs='>^<vs*x+d';
wc=8*pi/Ts;
t=[0:Ns-1]*T;
wcT=wc*T;
nd=0;
for m=1:M
    sw(m,:)=sqrt(2*Es/Ts)*cos(wc*t+phases(m));
end
su=sqrt(2/Ts)*[cos(wc*t); -sin(wc*t)];
suT= su*T;
SNRdBs=[1:10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRbdB= SNRdBs(iter);
    SNR=10^(SNRbdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
```

```

sws= zeros(1,LB);
yr= zeros(2,LB);
nobe= 0;
for k=1:MaxIter
    i=ceil(rand*M);
    s=ss(i,:);
    for n=1:Ns
        sws=[sws(2:LB) sw(i,n)];
        wct= wcT*(n-1);
        bp_noise= randn*cos(wct)-randn*sin(wct);
        rn= sws(end-nd)+sgmsT*bp_noise;
        yr= [yr(:,2:LB) suT(:,n)*rn];
    end
    ycsk= sum(yr(:,LBN1:LB))';
    if iter==9&k<300
        subplot(221);
        hold on;
        plot(ycsk(1),ycsk(2),gs(i));
    end
    th= atan2(ycsk(2),ycsk(1));
    if th<-pi/M
        th=th+2*pi;
    end
    [themin,lmin]= min(abs(th-phases));
    d= ss(lmin,:);
    nobe=nobe+sum(s~=d);
    if nobe>100
        break;
    end
end
pobe(iter)= nobe/(k*b);
end

```

### **Вариант 11.**

#### **Моделирование системы передачи данных с QAM-модуляцией:**

```

clear, clf
b=4;
M=2^b;
L=2^(b/2);
Esum= 0;
SNRbdBt=0:0.1:15;

```

```

SNRbt=10.^(SNRbdBt/10);
Pm=2*(1-1/L)*Q(sqrt(3/2*b*SNRbt/(M-1)));
pobet= (1-(1-Pm).^2)/b;
Tb=1;
Ts=b*Tb;
Nb=16;
Ns=b*Nb;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
ssc=[0 0; 0 1; 1 1; 1 0];
sss=ssc;
wc=8*pi/Ts;
wcT=wc*T;
t=[0:Ns-1]*T;
su=sqrt(2/Ts)*[cos(wc*t); -sin(wc*t)];
suT=su*T;
for i=1:L
    for l=1:L
        s(i,l,1)=2*i-L-1;
        s(i,l,2)=2*l-L-1;
        Esum= Esum +s(i,l,1)^2 +s(i,l,2)^2;
        ss(L*(l-1)+i,:)= [ssc(i,:) sss(l,:)];
        sw(L*(l-1)+i,:)=s(i,l,1)*su(1,:)+s(i,l,2)*su(2,:);
    end
end
Eav=Esum/M;
Eav_s=2*(M-1)/3;
Es=2;
A=sqrt(Es/Eav);
sw=A*sw;
levels=A*[-(L-1):2:L-1];
SNRdBs=[1:15];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRbdB= SNRdBs(iter);
    SNR=10^(SNRbdB/10);
    N0=2*(Es/b)/SNR;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    yr= zeros(2,LB);

```

```

nobe= 0;
for k=1:MaxIter
    im= ceil(rand*L);
    in= ceil(rand*L);
    imn= (in-1)*L+im;
    s=ss(imn,:);
    for n=1:Ns
        wct= wT*(n-1);
        bp_noise= randn*cos(wct)-randn*sin(wct);
        rn= sw(imn,n) + sgmsT*bp_noise;
        yr= [yr(:,2:LB) suT(:,n)*rn];
    end
    ycsk=sum(yr(:,LBN1:LB)');
    [dmin_i,mi]= min(abs(ycsk(1)-levels));
    [dmin_l,ml]= min(abs(ycsk(2)-levels));
    d= ss((ml-1)*L+mi,:);
    nobe = nobe+sum(s~=d);
    if nobe>100
        break;
    end
end
pobe(iter)= nobe/(k*b);
end

```

## Вариант 12.

**Моделирование системы передачи данных с однополосной АМ-модуляцией:**

```

function sim_SSB_AM(U_or_L)
if nargin<1
    U_or_L=0;
end
clear, clf
Ac=1;
fc=50;
wc=2*pi*fc;
Tb=0.1;
T=1/fc/8; Fs=1/T;
Nb=Tb/T;
lt=2^(nextpow2(3*Nb));
t=[1:lt]*T;
m= ones(Nb,1)*[4 -8 -4];

```



```

m=m(:).';
m=[m, zeros(1,lt-length(m))];
ma=hilbert(m);
tmpc=real(ma).*cos(wc*t);
tmps=imag(ma).*sin(wc*t);
if U_or_L<=0
    m_ssb=Ac*(tmpc-tmps);
    str='USSB-AM';
else
    m_ssb=Ac*(tmpc+tmps);
    str='LSSB-AM';
end
y_ssb=m_ssb*2/Ac.*cos(wc*t);
Bd= fir1(20,fc*T); Ad=1;
y_dtr=filter(Bd,Ad,y_ssb);
m_ssb=modulate(m,fc,Fs,'amssb');
y_ssb=demod(m_ssb,fc,Fs,'amssb');

```

### **Вариант 13.**

**Моделирование системы передачи данных с  $\pi/4$  DQPSK-модуляцией:**

```

b=2;
M=2^b;
M2=M^2;
SNRbdBt=0:0.1:10;
SNRbt=10.^(SNRbdBt/10);
pobet=prob_error(SNRbdBt,'DPSK',b,'bit');
Tb=1;
Ts=b*Tb;
Nb=16;
Ns=b*Nb;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
Es=2;
ss=[0 0; 0 1; 1 1; 1 0];
wc=8*pi/Ts;
wcT=wc*T;
t=[0:Ns-1]*T;
nd=1;
for m=1:M2
    sw(m,:)=sqrt(2*Es/Ts)*cos(wc*t+(m-1)*pi/M);

```

```

end
su= sqrt(2/Ts)*[cos(wc*t); -sin(wc*t)];
suT=su*T;
SNRdBs =[1:3:10];
MaxIter=10000;
for iter=1:length(SNRdBs)
    SNRbdB=SNRdBs(iter);
    SNRb=10^(SNRbdB/10);
    N0=2*(Es/b)/SNRb;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    sws=zeros(1,LB);
    yr=zeros(2,LB);
    nobe=0;
    is0=1;
    th0=1;
    for k=1:MaxIter
        i= ceil(rand*M);
        s=ss(i,:);
        is= mod(is0+????,M2)+1;
        for n=1:Ns
            sws=[sws(2:LB) sw(is,n)];
            wct=wcT*(n-1);
            bp_noise= randn*cos(wct)-randn*sin(wct);
            rn= sws(end-nd) + sgmsT*bp_noise;
            yr=[yr(:,2:LB) suT(:,n)*rn];
        end
        ycsk=sum(yr(:,LBN1:LB))';
        th=atan2(ycsk(2),ycsk(1));
        dth=th-th0;
        if dth<0
            dth=dth+2*pi;
        end
        [themin,lmin]=min(abs(dth-[????]*2*pi/M));
        d= ss(lmin,:);
        nobe= nobe+sum(s~=d);
        if nobe>100
            break;
        end
        is0=is;
        th0=th;
    end
end

```

```

end
pobe(iter)= nobe/(k*b);
end

```

#### **Вариант 14.**

**Модулятор QAM** (для построения таблицы):

```

function qamseq=QAM(bitseq,b)
bpsym = nextpow2(max(bitseq));
if bpsym>0
    bitseq = deci2bin(bitseq,bpsym);
end
if b==1
    qamseq=bitseq*2-1;
    return;
end
N0=length(bitseq);
N=ceil(N0/b);
bitseq=bitseq(:).';
bitseq=[bitseq zeros(1,N*b-N0)];
b1=ceil(b/2);
b2=b-b1;
b21=b^2;
b12=2^b1;
b22=2^b2;
g_code1=2*gray_code(b1)-b12+1;
g_code2=2*gray_code(b2)-b22+1;
tmp1=sum([1:2:b1-1].^2)*b21;
tmp2=sum([1:2:b2-1].^2)*b12;
M=2^b;
Kmod=sqrt(2*(M-1)/3);
qamseq=[];
for i=0:N-1
    bi=b*i;
    i_real=bin2deci(bitseq(bi+[1:b1]))+1;
    i_imag=bin2deci(bitseq(bi+[b1+1:b]))+1;
    qamseq=[qamseq (g_code1(i_real)+j*g_code2(i_imag))/Kmod];
end

```

**Демодулятор QAM** (для реализации блок схемы алгоритма):

```

function bitseq=QAM_dem(qamseq,b,bpsym)
if b==1
    bitseq=(qamseq>=0);

```

```

    return;
end
N=length(qamseq);
b1=ceil(b/2);
b2=b-b1;
g_code1=2*gray_code(b1)-2^b1+1;
g_code2=2*gray_code(b2)-2^b2+1;
tmp1=sum([1:2:2^b1-1].^2)*2^b2;
tmp2=sum([1:2:2^b2-1].^2)*2^b1;
Kmod=sqrt((tmp1+tmp2)/2/(2^b/4));
g_code1=g_code1/Kmod;
g_code2=g_code2/Kmod;
bitseq=[];
for i=1:N
    [emin1,i1]=min(abs(real(qamseq(i))-g_code1));
    [emin2,i2]=min(abs(imag(qamseq(i))-g_code2));
    bitseq=[bitseq deci2bin1(i1-1,b1) deci2bin1(i2-1,b2)];
end
if (nargin>2)
    N = length(bitseq)/bpsym;
    bitmatrix = reshape(bitseq,bpsym,N).';
    for i=1:N
        intseq(i)=bin2deci(bitmatrix(i,:));
    end
    bitseq = intseq;
end

```

### **Вариант 15.**

#### **Реализации кодера источника – Хаффмана:**

```

function [h,L,H]=Huffman_code(p,opt)
zero_one=['0';'1'];
if nargin>1&&opt>0
    zero_one=['1';'0'];
end
if abs(sum(p)-1)>1e-6
    fprintf('\n The probabilities in huffman_code() does not add up to 1!');
    p=p/sum(p);
end
M=length(p);
N=M-1;
p=p(:);

```

```

h{1}=zero_one(1);
h{2}=zero_one(2);
h={zero_one(1),zero_one(2)};
if M>2
    pp(:,1)=p;
    for n=1:N
        [pp(1:M-n+1,n),o(1:M-n+1,n)]=sort(pp(1:M-n+1,n),1,'descend');
        if n==1
            ord0=o;
        end
        if M-n>1
            pp(1:M-n,n+1)=[pp(1:M-1-n,n); sum(pp(M-n:M-n+1,n))];
        end
    end
    for n=N:-1:2
        tmp=N-n+2;
        oi=o(1:tmp,n);
        for i=1:tmp
            h1{oi(i)}=h{i};
        end
        h=h1;
        h{tmp+1}=h{tmp};
        h{tmp}=[h{tmp} zero_one(1)];
        h{tmp+1}=[h{tmp+1} zero_one(2)];
    end
    for i=1:length(ord0)
        h1{ord0(i)}=h{i};
    end
    h=h1;
end
L=0;
for n=1:M
    L=L+p(n)*length(h{n});
end
H=-sum(p.*log2(p));

```

### **Вариант 16.**

#### **Моделирование ФАПЧ:**

```
clear, clf
```

```
Ts=1e-5;
```

```
Ns=40;
```

```

T=Ts/Ns;
MaxIter=400;
tt=[0:MaxIter]*T;
wc=8*pi/Ts; pi2=pi*2;
th= pi/4 + pi/50*sin(0.01*pi/T*tt);
zeta=0.707;
wn=pi/2/Ts;
T2=max(2*zeta/wn,T/pi);
T1=10*T2;
K=2*wn^2*T1;
[bLF,aLF]=bilinear([T2 1],[T1 0],1/T,1/T2/2/pi);
[bVCO,aVCO]=bilinear(K,[1 0],1/T);
thh(1)=0;
vLF(1)=0;
for n=1:MaxIter
    t=tt(n);
    vi(n)=sin(wc*t+th(n));
    vVCO(n)=cos(wc*t+thh(n));
    ve(n)=vi(n)*vVCO(n);
    if n<=1
        vLF(n+1)=-aLF(2)*vLF(n)+bLF(1)*ve(n);
    else
        vLF(n+1)=-aLF(2)*vLF(n)+bLF*ve(n:-1:n-length(bLF)+1)';
    end
    thh(n+1)= -aVCO(2)*thh(n)+bVCO*vLF(n+1:-1:n-length(bVCO)+2)';
    if (thh(n+1)>pi )
        thh(n+1)= thh(n+1)-pi2;
    elseif (thh(n+1)<-pi)
        thh(n+1)= thh(n+1)+pi2;
    end
end
subplot(211),
plot(tt,thh,'r', tt,th,'k')
lb_ve=10;
b_ve=zeros(1,lb_ve);
thh(1)=0;
vLF(1)=0;
KT=1/Ts;
for n=1:MaxIter
    t=tt(n);

```

```

vi(n)=sin(wc*t+th(n));
vVCO(n)=cos(wc*t+thh(n));
ve(n)= vi(n)*vVCO(n);
vLF(n+1)=vLF(n)+(ve(n)-b_ve(1))*T;
b_ve=[b_ve(2:lb_ve) ve(n)];
thh(n+1)= thh(n)+KT*vLF(n+1);
if (thh(n+1)>pi)
    thh(n+1)= thh(n+1)-2*pi;
elseif (thh(n+1)<-pi)
    thh(n+1)= thh(n+1)+2*pi;
end
end
subplot(212);
plot(tt,th,'k', tt,thh,'r');

```

### **Вариант 17.**

#### **Моделирование квадратичной петли:**

```

clear, clf
Ts=1e-5;
Ns=64;
T=Ts/Ns;
wc=8*pi/Ts;
ds=randint(1,20)*2-1;
N_ds=length(ds);
NT= N_ds*Ns;
th=pi/5 +pi/20*sin([1:NT]*pi/200);
fp=0.05;
fs=0.1;
Rp=3;
As=40;
[Norder,fc0]= ellipord(fp,fs,Rp,As);
[Bde,Ade]= ellip(Norder,Rp,As,fc0);
NB=length(Bde);
NA=length(Ade);
lb_ve=10;
b_ve=zeros(1,lb_ve);
thh_s2(1)=0;
vLF(1)=0;
KvT=10/Ts*T;
for KC=0:1
    n=1;

```

```

zi=zeros(1,max(NA,NB)-1);
for k=1:N_ds
    d(k)=ds(k);
    for m=1:Ns
        t(n)=(n-1)*T;
        wct=wc*t(n);
        wct2=wct*2;
        dt(n)=d(k);
        r(n)= d(k)*cos(wct+th(n));
        if KC==0
            r_dem(n)=r(n)*2*cos(wct);
        else
            vVCO(n)= cos(wct2+thh_s2(n));
            ve(n)= r(n)^2*vVCO(n);
            vLF(n+1)= vLF(n)+(ve(n)-b_ve(1))*KvT;
            b_ve= [b_ve(2:lb_ve) ve(n)];
            thh_s2(n+1)= thh_s2(n)+KvT*vLF(n+1);
            thh(n)= (thh_s2(n)-pi/2)/2;
            r_dem(n)=r(n)*2*cos(wct+thh(n));
        end
        [dh(n),zi]=filter(Bde,Ade,r_dem(n),zi);
        n=n+1;
    end
end
if (KC>0)
    subplot(311);
    plot(t,th,'k', t,thh,'r');
end
subplot(312+KC);
plot(t,dt,'k', t,dh,'r')
end

```

### **Вариант 18.**

#### **Моделирование петли Костаса для QPSK:**

```

clear, clf
KC=1;
b=2;
M=2^b;
Tb=1e-2;
Ts=b*Tb;
Nb=40;

```



```

Ns=b*Nb;
T=Ts/Ns;
LB=4*Ns;
LBN1=LB-Ns+1;
SNRbdBt=0:0.1:10;
pobet=prob_error(SNRbdBt,'PSK',b,'bit');
ss=[0 0; 0 1; 1 1; 1 0];
gs='>^<v';
wc=2*pi/Ts;
wcT=wc*T;
t=[0:Ns-1]*T;
wct=wc*t;
pi2=pi^2;
phases=[0:M-1]*(pi2/M);
gam=0.01;
su=sqrt(2/Ts)*[cos(wct); -sin(wct)]; suT=su*T;
thd=-pi/8;
Es=2;
for m=1:M
    sw(m,:)=sqrt(2*Es/Ts)*cos(wct+phases(m)+thd);
end
SNRbdBs=[1 5 9];
N_SNRbdBs=length(SNRbdBs);
MaxIter=40000;
Target_no_of_error=100;
for iter=1:N_SNRbdBs
    SNRbdB=SNRbdBs(iter);
    SNRb=10^(SNRbdB/10);
    N0=2*(Es/b)/SNRb;
    sigma2=N0/2;
    sgmsT=sqrt(sigma2/T);
    yr=zeros(2,LB);
    thh(1)=0;
    nobe= 0;
    rand('twister',5489);
    randn('state',0);
    for k=1:MaxIter
        i= ceil(rand*M);
        s=ss(i,:);
        for n=1:Ns
            wct=(n-1)*wcT;

```

```

        rn=sw(i,n)+sgmsT*(randn*cos(wct)-randn*sin(wct));
        yr= [yr(:,2:LB) suT(:,n)*rn];
    end
    ycsk= sum(yr(:,LBN1:LB)');
    if (iter==N_SNRbdBs&k<min(200,MaxIter))
        subplot(221);
        plot(ycsk(1),ycsk(2),gs(i));
        hold on
    end
    ynejth=(ycsk(1)+j*ycsk(2));
    if (KC>0)
        ynejth=ynejth*exp(-j*thh(k));
    end
    th=angle(ynejth);
    if (th<-pi/M)
        th=th+pi2;
    end
    [thmin,lmin]=min(abs(th-phases));
    D=ss(lmin,:);
    thh(k+1)=thh(k)+gam*imag(exp(-j*phases(lmin))*ynejth);
    nobe=nobe+sum(s~=D);
    if (nobe>Target_no_of_error)
        break;
    end
end
end
pobe(iter)= nobe/(k*b);
end
subplot(222);
semilogy(SNRbdBt,pobet,'k-', SNRbdBs,pobe,'b*');

```

### **Вариант 19.**

**Моделирование схемы тактовой синхронизации с запаздыванием/  
опережением:**

```

clear, clf
KC=0;
Ns=80;
Ts=1;
T=Ts/Ns;
Delay=3;
rof=0.5;
Fd=1/Ts;

```

```

Fs=Fd*Ns;
SRRC_filter = rcosine(Fd,Fs,'fir/sqrt',rof,Delay);
[g,tg]=rcosflt([1],Fd,Fs,'filter',SRRC_filter);
g=g.';
Ng=length(g);
h=fliplr(g);
Es=1;
b=1;
SNRbdB=8;
b=1;
pobet= prob_error(SNRbdB,'PSK',b,'bit');
SNRb=10^(SNRbdB/10);
N0=2*(Es/b)/SNRb;
sigma2=N0/2;
sgmsT=sqrt(sigma2/T);
dT=Ts/2;
dTn=ceil(dT/T);
delta=0.01;
MaxIter=2000;
for ts=[3.1 2.9 2.1]*Ts
    tsN=ceil(ts/T);
    dd=round(Ng/Ns)*2-floor(ts/Ts)+2;
    lch=Ng+Ns-1;
    ch_input=zeros(1,lch);
    lrs=Ng+Ns-1;
    rs=zeros(1,lrs);
    lys=Ns*10;
    ys=zeros(1,lys);
    lds=100+dd;
    ds=zeros(1,lds);
    lDs=100;
    Ds=zeros(1,lDs);
    nobe=0;
    for k=1:MaxIter
        d(k)=(rand>0.5)*2-1;
        ds=[ds(2:end) d(k)];
        ch_input= [ch_input(Ns+1:end) d(k)*ones(1,Ns)];
        for n=1:Ns
            rn=ch_input(end-Ns+n-Ng+1:end-Ns+n)*g' +sgmsT*randn;
            rs=[rs(2:end) rn/Ns];
            yn=rs(end-Ng+1:end)*h';

```

```

        ys=[ys(2:end) yn];
    end
    dif=abs(ys(tsN-dTN))-abs(ys(tsN+dTN));
    if (KC>0)
        if (dif>delta)
            ts=ts-T;
        elseif (dif<-delta)
            ts=ts+T;
        end
    end
    ts_history(k)=ts;
    tsN=ceil(ts/T);
    D=(ys(tsN)>0)*2-1;
    Ds= [Ds(2:1Ds) D];
    if (nobe>100)
        break;
    end
end
pobe=nobe/k;
plot(ts_history) ;
end

```

## Вариант 20.

**Моделирование схемы восстановления несущей для QAM-демодуляции:**

```

clear, clf
KC=2;
b=4;
M=2^b;
L=2^(b/2);
SNRbdBt=0:0.1:15;
pobet=prob_error(SNRbdBt,'QAM',b,'bit');
Tb=0.025;
Ts=b*Tb;
Nb=20;
Ns=b*Nb;
T=Ts/Ns;
LB=Ns*4;
LBNs=LB-Ns;
ssc=[0 0; 0 1; 1 1; 1 0];
sss=ssc;

```

```

wc=12*pi/Ts;
t=[0:Ns-1]*T;
s2sT=sqrt(2/Ts);
su= s2sT*[cos(wc*t); -sin(wc*t)];
for m=1:L
    for n=1:L
        s(m,n,1)=2*m-L-1;
        s(m,n,2)=2*n-L-1;
        ss(L*(n-1)+m,:)= [ssc(m,:) sss(n,:)];
        sw(L*(n-1)+m,:)=s(m,n,1)*su(1,:)+s(m,n,2)*su(2,:);
    end
end
Es=2;
Eav=2*(M-1)/3;
A=sqrt(Es/Eav);
sw=A*sw; levels=A*[-(L-1):2:L-1].';
zeta=0.707;
wn=pi/2000/Ts;
T2= max(2*zeta/wn,T/pi);
T1=10*T2;
KvT= 4*wn^2*T1*T;
[bLF,aLF]=bilinear([T2 1],[T1 0],1/T,1/T2/2/pi);
SNRbdBs=[3 8 13];
MaxIter=4e4;
Target_no_of_error=100;
for iter=1:length(SNRbdBs)
    SNRbdB=SNRbdBs(iter);
    SNRb=10^(SNRbdB/10);
    N0=2*(Es/b)/SNRb;
    sigma2=N0/2;
    sigmaT=sqrt(sigma2/T);
    r= zeros(1,LB);
    yr= zeros(2,LB);
    ycs= zeros(2,LB);
    ve=zeros(1,LB);
    vLF=zeros(1,LB);
    thk=0;
    Ak=1;
    nobe= 0;
    rand('twister',5489);
    randn('state',0);

```

```

for k=1:MaxIter
    im= ceil(rand*L);
    in= ceil(rand*L);
    imn= (in-1)*L+im;
    s=ss(imn,:);
    if (k<5)
        Nd=0;
        thh=0;
    elseif (KC>0)
        Nd=1;
    end
    thd=-wc*Nd*T;
    for n=1:Ns
        wct= wc*(n-1)*T;
        bp_noise= randn*cos(wct)-randn*sin(wct);
        r=[r(2:LB) sw(imn,n)+sigmaT*bp_noise];
        yr=[yr(:,2:LB) s2sT*[cos(wct+thh); -sin(wct+thh)]*r(LB-Nd)];
        ycs=[ycs(:,2:LB) ycs(:,LB)+yr(:,LB)-yr(:,LBNs)];
        ve=[ve(2:LB) [-sin(thk) cos(thk)]/Ak*yr(:,LBNs)];
        vLF=[vLF(2:LB) vLF(LB)+bLF*ve(LB:-1:LB-1)'];
        thh= thh + KvT*vLF(LB);
        if (KC<2)
            thh=0;
        end
    end
    ycsk=ycs(:,LB)*T;
    yck=yck(1);
    ysk=ysk(2);
    if (iter==length(SNRbds)&k<200)
        subplot(341+KC);
        hold on;
        plot(yck(1),ysk(2),'.');
    end
    [dmin,lmn]=min(abs([yck(1)-levels ysk(2)-levels]));
    lm=lmn(1);
    ln=lmn(2);
    thk=atan2(levels(ln),levels(lm));
    D= ss((ln-1)*L+lm,:);
    nobe=nobe+sum(s~=D);
    if (nobe>Target_no_of_error)
        break;
    end
end

```

```
        end
    end
    pobe(iter)= nobe/(k*b);
end
subplot(345+KC);
semilogy(SNRbdBt,pobet,'k-', SNRbdBs,pobe,'b*');
title('BER for (16-ary) QAM Signaling');
```

# ЧАСТЬ 3. ПРОГРАММИРОВАНИЕ В СРЕДЕ MATLAB. ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

## 1. Работа с файлами в среде MATLAB

Создание программ часто предполагает сохранение результатов расчетов в файлы для их дальнейшего анализа, обработки, хранения и т.п. В связи с этим в MATLAB реализованы различные функции по работе с файлами, содержащие данные в самых разных форматах. В этой главе рассмотрим наиболее полезные функции для сохранения и загрузки результатов работы алгоритмов из файлов.

### 1.1. Сохранение/загрузка файлов в формате .mat. Функции *save* и *load*

В самом простом случае для сохранения и последующей загрузки каких-либо данных в MATLAB предусмотрены две функции:

**save** <имя файла> <имена переменных> % сохранение данных;  
**load** <имя файла> <имена переменных> % загрузка данных.

Функция **save** позволяет сохранять произвольные переменные программы в файл, который будет (по умолчанию) располагаться в рабочем каталоге (обычно поддиректория *work*) и иметь расширение *.mat*. Соответственно функция **load** позволяет загрузить из указанного *mat*-файла ранее сохраненные переменные. Ниже представлен пример использования данных функций:

---

```
x=ones(5); y=5; s='hello';  
save params x y s;  
x = zeros(5); y = 0; s = '';  
load params x y s;  
disp(x);  
disp(y);  
disp(s);
```

---

### 1.2. Сохранение/загрузка двоичных файлов. Функции *fwrite* и *fread*

Недостатком рассмотренных функций **save** и **load** является то, что они работают с определенными форматами файлов (обычно *mat*-



файлы) и не позволяют загружать или сохранять данные в других форматах. Между тем бывает необходимость загружать информацию, например, из бинарных файлов, созданных другими программными продуктами для дальнейшей обработки результатов в MATLAB. С этой целью были разработаны функции

```
fwrite(<идентификатор файла>, <переменная>, <тип данных>);  
и  
<переменная>=fread(<идентификатор файла>);  
<переменная>=fread(<идентификатор файла>, <размер>);  
<переменная>=fread(<идентификатор файла>, <размер>, <точность>).
```

Здесь <идентификатор файла> – это указатель на файл, с которым предполагается работать. Чтобы получить идентификатор, используется функция

<идентификатор файла> = **fopen**(<имя файла>, <режим работы>), где параметр <режим работы> может принимать значения, приведенные в табл. 3.1. Для закрытия файла используется функция **fclose**(<идентификатор файла>).

Таблица 3.1.

Режимы работы с файлами в MATLAB

Параметр <режим работы>	описание
'r'	Чтение
'w'	Запись (стирает предыдущее содержимое файла)
'a'	Добавление (создает файл, если его нет)
'r+'	Чтение и запись (не создает файл, если его нет)
'w+'	Чтение и запись (очищает прежнее содержимое или создает файл, если его нет)
'a+'	Чтение и добавление (создает файл, если его нет)
'b'	Дополнительный параметр, означающий работу с бинарными файлами, например 'wb', 'rb', 'rb+', 'ab' и т.п.

Если функция **fopen()** по каким-либо причинам не может корректно открыть файл, то она возвращает значение –1. Ниже представлен фрагмент программы записи и считывания данных из бинарного файла.

---

```
A = [1 2 3 4 5];
```

```
fid = fopen('my_file.dat', 'wb');    % открытие файла на запись
if fid == -1                          % проверка корректности открытия
    error('File is not opened');
end
```

```
fwrite(fid, A, 'double');           % запись матрицы в файл (40 байтов)
fclose(fid);                       % закрытие файла
```

```
fid = fopen('my_file.dat', 'rb');    % открытие файла на чтение
if fid == -1                          % проверка корректности открытия
    error('File is not opened');
end
```

```
B = fread(fid, 5, 'double');          % чтение 5 значений double
disp(B);                             % отображение на экране
fclose(fid);                         % закрытие файла
```

---

В результате работы данной программы в рабочем каталоге будет создан файл `my_file.dat` размером 40 байтов, в котором будут содержаться пять значений типа `double`, записанных в виде последовательности байтов (по 8 байтов на каждое значение). Функция **fread()** считывает последовательно сохраненные байты и автоматически преобразовывает их к типу `double`, т.е. каждые 8 байтов интерпретируются как одно значение типа `double`.

В приведенном примере явно указывалось число элементов (пять) для считывания из файла. Однако часто общее количество элементов бывает наперед неизвестным либо оно меняется в процессе работы программы. В этом случае было бы лучше считывать данные из файла до тех пор, пока не будет достигнут его конец. В MATLAB существует функция для проверки достижения конца файла:

**feof**(<идентификатор файла>),  
которая возвращает единицу при достижении конца файла и ноль — в других случаях. Перепишем программу для считывания произвольного числа элементов типа `double` из входного файла:

---

```
fid = fopen('my_file.dat', 'rb');    % открытие файла на чтение
if fid == -1
    error('File is not opened');
```

end	
B=0;	% инициализация переменной
cnt=1;	% инициализация счетчика
while ~feof(fid)	% цикл, пока не достигнут конец
файла	
[V,N] = fread(fid, 1, 'double');	%считывание одного значения
	double (V содержит значение
	элемента, N – число считанных
	элементов)
if N > 0	% если элемент был прочитан
	успешно
B(cnt)=V;	% формируем вектор-строку
	из значений V
cnt=cnt+1;	% увеличиваем счетчик на 1
end	
end	
disp(B); fclose(fid);	% отображение результата
	на экран и закрытие файла

---

В данной программе динамически формируется вектор-строка по мере считывания элементов из входного файла. Язык MATLAB автоматически увеличивает размерность векторов, если индекс следующего элемента на единицу больше максимального. Однако на такую процедуру тратится много машинного времени и программа начинает работать заметно медленнее, чем если бы размерность вектора B с самого начала была определена равной пяти элементам, например, так:

---

```
B = zeros(5,1);
```

---

Следует также отметить, что функция **fread()** записана с двумя выходными параметрами V и N. Первый параметр содержит значение считанного элемента, а второй – число считанных элементов. В данном случае значение N будет равно единице каждый раз при корректном считывании информации из файла, и нулю – при считывании служебного символа EOF, означающего конец файла. Приведенная ниже проверка позволяет корректно сформировать вектор значений B. С помощью функций **fwrite()** и **fread()** можно сохранять и строковые данные. Например, пусть дана строка:

---

```
str = 'Hello Matlab';
```

---

которую требуется сохранить в файл. В этом случае функция **fwrite()** будет иметь следующую запись:

---

```
fwrite(fid, str, 'int16');
```

---

Здесь используется тип `int16`, так как при работе с русскими буквами система MATLAB использует двухбайтовое представление каждого символа. Ниже представлена программа записи и чтения строковых данных, используя функции **fwrite()** и **fread()**. Код программы реализует последовательно процедуры записи текстовой информации побайтно, после чего выполняется побайтное чтение из файла с преобразованием чисел в символы. Результат выполнения программы будет иметь вид

---

```
Привет MATLAB
```

---

---

```
fid = fopen('my_file.dat', 'wb');
if fid == -1
    error('File is not opened');
end

str='Привет MATLAB';           % строка для записи
fwrite(fid, str, 'int16');      % запись в файл
fclose(fid);

fid = fopen('my_file.dat', 'rb');
if fid == -1
    error('File is not opened');
end

B='';                           % инициализация строки
cnt=1;
while ~feof(fid)
    [V,N] = fread(fid, 1, 'int16=>char'); % чтение текущего
                                           % символа и преобразование
                                           % его в тип char

    if N > 0
        B(cnt)=V;
        cnt=cnt+1;
    end
```

```
end  
disp(B); % отображение строки на экране  
fclose(fid);
```

---

### 1.3. Сохранение/загрузка текстовых файлов. Функции *fscanf* и *fprintf*

Описанные выше функции работы с файлами позволяют записывать и считывать информацию по байтам, которые затем требуется правильно интерпретировать для преобразования их в числа или строки. В то же время выходными результатами многих программ являются текстовые файлы, в которых явным образом записаны числа или текст. Например, при экспорте данных из MS Excel можно получить файл формата

```
174500,1.63820,1.63840,1.63660,1.63750,288  
180000,1.63740,1.63950,1.63660,1.63820,361  
181500,1.63830,1.63850,1.63680,1.63740,223  
183000,1.63720,1.64030,1.63720,1.64020,220
```

где числа записаны в столбик и разделены запятой.

Прочитать такой файл побайтно, а затем интерпретировать полученные данные — довольно трудоемкая задача, поэтому для этих целей были специально разработаны функции чтения и записи

```
[value, count] = fscanf(fid, format, size) % чтение текстовых файлов  
count = fprintf(fid, format, a,b,...) % запись текстовых файлов
```

таких данных в файл. Здесь value — результат считывания данных из файла; count — число прочитанных (записанных) данных; fid — указатель на файл; format — формат чтения (записи) данных; size — максимальное число считываемых данных; a,b,... — переменные для записи в файл.

Приведем пример чтения данных из файла, приведенного выше, с помощью функции `fscanf()`:

---

```
function fscanf_ex  
fid = fopen('my_excel.dat', 'r');  
if fid == -1  
    error('File is not opened');  
end  
S = fscanf(fid, '%d,%f,%f,%f,%f,%d');  
fclose(fid);
```

---

Здесь форматная строка состоит из спецификаторов  
 %d – работа с целочисленными значениями;  
 %f – работа с вещественными значениями  
 и записана в виде ‘%d,%f,%f,%f,%f,%d’. Это означает, что сначала должно быть прочитано целочисленное значение из файла, затем через запятую должно читаться второе вещественное значение, затем третье и т.д. до последнего целочисленного значения. Полный список возможных спецификаторов приведен в табл. 3.2.

Таблица 3.2.

**Список основных спецификаторов для функций *fscanf* и *fprintf***

Спецификатор	Описание
%d	Целочисленные значения
%f	Вещественные значения
%s	Строковые данные
%c	Символьные данные
%u	Беззнаковые целые значения

В результате работы программы переменная *S* будет представлять собой вектор-столбец, состоящий из 24 элементов:

---

$S = [174500 \ 1,6382 \ 1,6384 \ 1,6366 \ 1,6375 \ 288 \ 180000 \ 1,6374 \ 1,6395 \ 1,6366 \ 1,6382 \ 361 \ 181500 \ 1,6383 \ 1,6385 \ 1,6368 \ 1,6374 \ 223 \ 183000 \ 1,6372 \ 1,6403 \ 1,6372 \ 1,6402 \ 220]$ ;

---

Несмотря на то что данные были корректно считаны из файла, они из таблицы были преобразованы в вектор-столбец, что не соответствует исходному формату представления данных. Чтобы сохранить верный формат данных, функцию **fscanf()** в приведенном примере следует записать так:

---

$S = \text{fscanf}(\text{fid}, \text{'\%d,\%f,\%f,\%f,\%f,\%d'}, [6 \ 4]);$

---

Тогда на выходе получится матрица *S* размером в шесть столбцов и четыре строки с соответствующими числовыми значениями.

Для записи данных в текстовый файл в заданном формате используется функция **fprintf()**. Ниже представлен пример записи матрицы чисел

---

180000	1.28210	1.28240	1.28100	1.28120	490
190000	1.28100	1.28150	1.27980	1.28070	444
200000	1.28050	1.28080	1.27980	1.28000	399
210000	1.27990	1.28020	1.27880	1.27970	408
220000	1.27980	1.28060	1.27880	1.28030	437
230000	1.28040	1.28170	1.28020	1.28130	419
0	1.28140	1.28140	1.28010	1.28100	294
10000	1.28080	1.28190	1.28030	1.28180	458
20000	1.28190	1.28210	1.28080	1.28140	384
30000	1.28130	1.28170	1.28080	1.28140	313

---

в файл, в котором числовые значения должны разделяться точкой с запятой. Будем также предполагать, что данная матрица хранится в переменной *Y*.

---

```
function fprintf_ex
    fid = fopen('my_excel.txt', 'w');
    if fid == -1
        error('File is not opened');
    end
    fprintf(fid, '%6d;%4f;%4f;%4f;%4f;%d\r\n', Y); fclose(fid);
```

---

Следует отметить, что в функции **fprintf()** переменная *Y* имеет знак транспонирования, так как данные в файл записываются по столбцам матрицы. Кроме того, перед спецификаторами стоят числа, которые указывают, сколько значащих цифр числа должно быть записано в файл. Например, спецификатор **%6d** говорит о том, что целые числа должны иметь шесть значащих цифр, а спецификатор **%4f** означает, что после запятой будет отображено только четыре цифры. Наконец, в форматной строке были использованы управляющие символы (**'\r'** — возврат каретки, **'\n'** — переход на новую строку), которые необходимы для формирования строк в выходном файле. В итоге содержимое файла будет иметь вид:

---

```
180000;1.2821;1.2824;1.2810;1.2812;490
190000;1.2810;1.2815;1.2798;1.2807;444
200000;1.2805;1.2808;1.2798;1.2800;399
210000;1.2799;1.2802;1.2788;1.2797;408
220000;1.2798;1.2806;1.2788;1.2803;437
230000;1.2804;1.2817;1.2802;1.2813;419
```

---

```
0;1.2814;1.2814;1.2801;1.2810;294
10000;1.2808;1.2819;1.2803;1.2818;458
20000;1.2819;1.2821;1.2808;1.2814;384
30000;1.2813;1.2817;1.2808;1.2814;313
```

---

С помощью функции **fprintf()** можно записать значения двух и более переменных разного формата. Например, для записи числа и строки можно воспользоваться следующей записью:

---

```
str = 'Hello';
y = 10;
count = fprintf(fid, '%d\r\n%s\r\n', y, str);
```

---

Содержимое файла будет иметь вид

---

```
10
Hello
```

---

Таким образом, можно осуществлять запись разнородных данных в файл в требуемом формате.

## 2. Функции для преобразования и обработки разнородных типов данных

Работа с различными файлами операционной системы часто сопровождается процедурами модификации и конвертации данных, которые в них содержатся. Поэтому среди инструментов среды MATLAB стоит выделить ряд функций, которые отвечают за подобные преобразования. При работе с текстовыми файлами особый интерес представляют функции, связанные с обработкой текстовых данных, описание которых представлено в табл. 3.3.

## 3. Оптимизация работы программ

Оптимизация заключается в повышении скорости работы программы. В MATLAB применимы все или почти все общие приемы оптимизации программ (например, вынесение из цикла наружу всех вычислений, не зависящих от счетчика цикла). Однако для рассматриваемой среды необходимо учитывать следующее:

1. Среда MATLAB – интерпретирующая, а не компилирующая.
2. Функции векторно-матричных операций встроены в ядро системы и выполняются предельно быстро.



Таблица 3.3.

**Специальные функции, упрощающие анализ и преобразование данных**

Наименование	Описание	Примеры использования
<code>flag=strcmp(s1, s2)</code>	Функция сравнения строк. Возвращает единицу в случае совпадения строк <code>s1</code> и <code>s2</code> и ноль – в противном случае	<code>&gt;&gt;flag= strcmp('may', 'june')</code> Результат: <code>flag=0</code>
<code>k=findstr(s1, s2)</code>	Поиск подстроки в строке. Выходной аргумент – вектор <code>k</code> содержит позиции, с которых подстрока начинается в строке. Входными аргументами являются строки и строковые переменные. Подстрокой считается входной аргумент меньшей длины	<code>&gt;&gt;s1='abcdefghigk0123'; s2='a'; k=findstr(s1, s2)</code> Результат: <code>k=1</code>
<code>k1=lower(s), k2=upper(s)</code>	Преобразование всех строчных букв в прописные ( <code>lower</code> ) и наоборот ( <code>upper</code> )	<code>&gt;&gt;s='a'; k1=upper(s); k2=lower(k1);</code> Результат: <code>k1='A'; k2='a';</code>
<code>k=num2str(a) a=str2num(k)</code>	Преобразование числового значения, записанного в переменной <code>a</code> в строку <code>k</code> – функция <code>num2str</code> и обратно – <code>str2num</code>	<code>&gt;&gt;a=[1 2 3]; k=num2str(a);</code> Результат: <code>k='1 2 3'</code>
<code>k=bin2dec(a) a=dec2bin(k)</code>	Преобразовывает строку двоичных чисел в целое число – <code>bin2dec</code> и обратно – <code>dec2bin</code> . При обратном преобразовании можно указать количество битов на выходе в качестве второго параметра	<code>&gt;&gt;bin2dec('010111')</code> <code>ans =</code> <code>23</code> <code>&gt;&gt;bin2dec('010 111')</code> <code>ans =</code> <code>23</code> (пробелы игнорируются) <code>&gt;&gt;dec2bin(23)</code> <code>ans =</code> <code>10111</code>

Общие рекомендации заключаются:

1). в минимизации использования циклов в программном коде, заменять их векторно-матричными операциями (данное правило следует использовать с умом, так как некоторые алгоритмы плохо поддаются векторизации, что может привести к огромным затратам памяти);

2). выделении памяти под переменные, массивы и векторы до использования их в цикле, что значительно уменьшит время на операции выделения и освобождения памяти, которые бы могли выполняться по мере необходимости в цикле. Если число проходов в цикле заранее неизвестно и, как следствие, не известен итоговый размер массива, то можно использовать компромиссный подход, который предполагает увеличение размера массива внутри цикла, но редко и большими кусками.

Для оценки времени выполнения программ в среде MATLAB можно использовать функции **tic** и **toc**, которые измеряют время дискретными порциями с шагом 50–60 мс:

---

```
>>tic  
>>for k=1:10000, x(k)=sin(k); end  
>>toc
```

---

Более серьезным инструментом хронометража является утилита профилирования, вызываемая командой

---

```
>>profile on
```

---

Для вывода результатов профилирования используется следующая команда:

---

```
>>profile report
```

---

При этом генерируется отчет в HTML-формате, в котором показано общее время профилирования, число вызываемых функций с момента включения утилиты и время, затраченное на их выполнение. Крайне полезным атрибутом команды **profile report** является возможность анализа времени выполнения каждой команды, которые содержатся в запускаемых функциях.

#### 4. Практическая работа. Разработка элементарной программы в среде MATLAB

После знакомства с общими и более продвинутыми средствами разработки программ в среде MATLAB можно переходить к написа-

нию программ, которые выполняют обработку некоторой внешней информации. В настоящей практической работе каждый студент в соответствии с вариантом задания по журналу выполняет разработку простейшей программы на языке MATLAB. Любая из программ, создаваемая в рамках этого задания, включает в себя три этапа:

1. Чтение файла (двоичного, текстового или файла среды MATLAB `.mat`).

2. Выполнение одной или нескольких операций с полученными данными.

3. Запись измененного файла.

Общим этапом работы для всех вариантов является определение времени выполнения программы с помощью процедур `tic` и `toc`, а также предложение, если это указано в задании, направления оптимизации программы для ее ускорения.

Для выполнения задания необходимо:

- 1). написать и отладить код программы;
- 2). получить выходной файл (результат расчета — при необходимости) — результат выполнения программы;
- 3). зафиксировать время выполнения программы;
- 4). оптимизировать код программы, увеличив скорость ее выполнения минимум в 10 раз;
- 5). зафиксировать изменения в коде программы по сравнению с вариантом без оптимизации.

***ВАЖНО:** Для создания итогового сигнала в заданиях 9, 11, 12 воспользуйтесь командой `cat(.)`.*

### **Варианты заданий**

1. Прочитать текстовый файл «Дубровский.txt», посчитать общее количество символов `'.'` в тексте произведения. Записать текст произведения в выходной файл «Дубровский1.txt», удалив все символы `'.'` из текста. Сравнить размеры входного и выходного файлов. Оценить общее время выполнения программы с помощью процедур `tic` и `toc`.

2. Прочитать текстовый файл «Кавказский пленник.txt». Посчитать количество букв `'а'` и `'я'` в тексте программы. Найти соотношение  $n('а')/n('я')$ , где  $n(.)$  — посчитанное количество выбранных букв в тексте. Записать выходной файл «Кавказский пленник1.txt», преобразовав все строчные буквы `'а'` в заглавные `'А'`. Оценить общее время выполнения программы с помощью процедур `tic` и `toc`.

3. Прочитать текстовый файл «Капитанская дочка.txt», посчитать общее количество символов `'.'` в тексте произведения. Преобразовать первые 100 символов текста в соответствующее десятичное

представление с помощью команды **double(.)** и записать эти числа в бинарный файл «dataKD.bin». Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

4. Прочитать текстовый файл «Жених.txt», определить количество знаков ‘?’ и ‘!’ в тексте произведения. Соотнести это количество с общим числом знаков в файле. Записать символы из файла «Жених.txt» в файл «Жених1.txt» в обратном порядке. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

5. Прочитать текстовый файл «Медный всадник.txt», определить максимальное количество символов, разделяющих соседние буквы ‘ю’ в тексте произведения. Записать полученное значение в текстовый файл «максЮ.txt» файл. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

6. Прочитать двоичный файл «data01.bin» в формате ‘double’, записать имеющиеся в нем данные  $a_0, a_1, \dots, a_{9999}$  в квадратную матрицу размером  $100 \times 100$  построчно, т.е. строка 1:  $a_0, a_1, \dots, a_{99}$ , строка 2:  $a_{100}, a_{101}, \dots, a_{199}$  и.т.д. Записать данные из имеющейся матрицы по столбцам в текстовый файл «data11.txt». Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

7. Прочитать двоичный файл «data02.bin» в формате ‘double’, записать имеющиеся в нем данные  $a_0, a_1, \dots$  в квадратную матрицу  $A$  размером  $128 \times 128$  построчно. Преобразовать каждую строку матрицы с помощью функции **fft(.)** в строку матрицы  $F$ . Записать данные из  $F$ -матрицы по строкам в текстовый файл «data12.txt». Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

8. Прочитать текстовый файл «data03.txt». Преобразовать имеющиеся в нем данные в двоичный код, используя команду **dec2bin(.)**, выбрать из полученного кода только нечетные отсчеты и выполнить обратное преобразование в целые числа (**bin2dec(.)**). Записать полученные данные в текстовый файл «data13.txt». Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

9. Открыть файл «data04.mat» в среде MATLAB. Представить каждый ноль одним периодом гармонического колебания с частотой  $f_0=1$  кГц с периодом дискретизации  $T_d=0,02$  мс, а каждую единицу, аналогично, но с частотой  $f_1=2$  кГц. Записать полученный сигнал в двоичный файл «data14.bin» в формате ‘double’. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

10. Открыть текстовый файл «Жених.txt», прочитать и преобразовать данные из текста в ASCII-формат с помощью команды **unicode2native(.)**, получить двоичное представление данных, используя

команду **dec2bin(.)**. Определить общее количество битов, записанных в массив после преобразования. Записать двоичный эквивалент текстового файла «Жених.txt» в текстовом формате с разделителем «пробел» между отдельными битами данных. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

11. Открыть файл «data05.mat» в среде MATLAB. Представить каждый ноль одним периодом гармонического колебания с частотой  $f=1$  кГц с периодом дискретизации  $T_d=0.02$  мс и амплитудой 1 В, а каждую единицу, аналогично, но с амплитудой 2 В. Насколько увеличился объем передаваемых данных в результате амплитудной модуляции? Записать полученный сигнал в двоичный файл «data15.bin» в формате 'double'. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

12. Открыть файл «data06.mat» в среде MATLAB. Представить каждый ноль одним периодом гармонического колебания с частотой  $f=1$  кГц с периодом дискретизации  $T_d=0,02$  мс и нулевой начальной фазой, а каждую единицу — аналогично, но с начальной фазой, равной  $\pi$ . Записать полученный сигнал в двоичный файл «data16.bin» в формате 'double'. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

13. Прочитать текстовый файл «Дубровский.txt», посчитать количество каждой из букв 'а', 'б', 'в', 'г', 'д' в тексте произведения. Записать количество найденных букв в .mat файл. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

14. Прочитать тестовый файл «Кавказский пленник.txt». Посчитать количество появлений в тексте любых (на ваш выбор) 10 букв русского алфавита. Построить гистограмму — зависимость вероятности появления объекта от вида объекта. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

15. Прочитать текстовый файл «Капитанская дочка.txt», посчитать вероятность появления букв 'о' и 'е' в тексте произведения. Преобразовать первые 50 символов текста в соответствующее 16-ричное .mat файл. Команды, необходимые для такого преобразования, найти в справочной системе MATLAB. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

16. Прочитать текстовый файл «Медный всадник.txt», определить максимальное количество знаков препинания каждого наименования (';', ':', ';', '!', '?', '!'), используемых в тексте произведения. Записать вероятности появления каждого из этих знаков в текстовый файл «symb.

txt». Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

17. Прочитать текстовый файл «data03.txt». Преобразовать имеющиеся в нем данные в двоичный код, используя команду **dec2bin(.)**, выбрать из полученного кода только четные отсчеты и выполнить обратное преобразование в целые числа (**bin2dec(.)**). Записать полученные данные в текстовый файл «data13.txt». Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.

18. Открыть файл «data04.mat» в среде MATLAB. Представить каждый ноль одним периодом гармонического колебания с частотой  $f_0=1$  кГц с периодом дискретизации  $T_d=0,01$  мс и амплитудой 0,1, а каждую единицу — аналогично, но с частотой  $f_1=3$  кГц и амплитудой 0,2. Отобразить полученный сигнал на графике. Оценить общее время выполнения программы с помощью процедур **tic** и **toc**.


## ЧАСТЬ 4. СОЗДАНИЕ ПРОСТЕЙШИХ МОДЕЛЕЙ В СРЕДЕ SIMULINK

### 1. Назначение и особенности среды Simulink

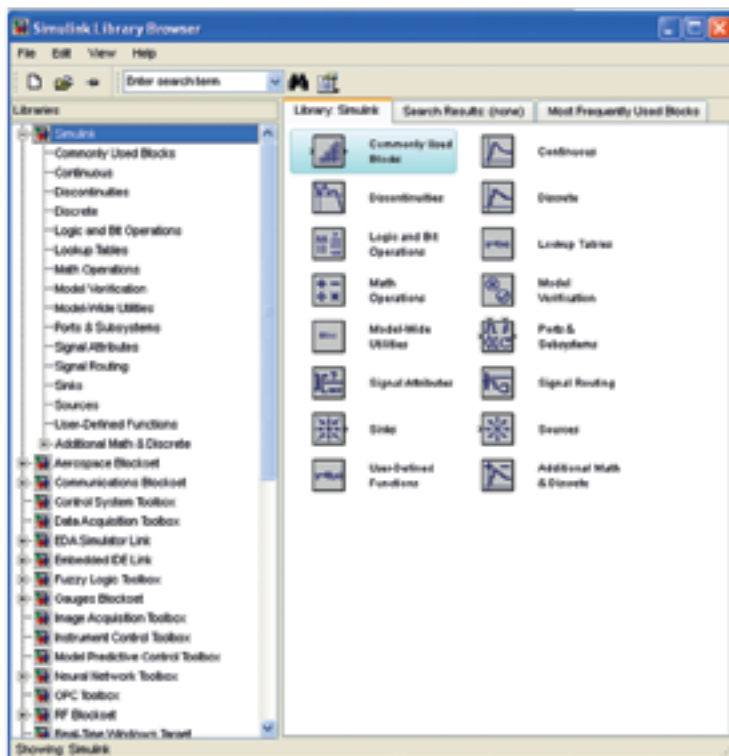
Simulink — главный пакет расширения системы MATLAB, реализующий имитационное блочное визуально-ориентированное моделирование систем и устройств как самого общего, так и конкретного назначения. Слово «simulink» образовано из комбинации первых четырех букв слова «simulation» (моделирование) и «link» (соединение). Основой визуально-ориентированного моделирования в среде Simulink служат S-модели или блок-схемы, которые могут работать во временной и частотной областях, с событийным управлением и по методу Монте-Карло.

S-модель фактически является программой, которую можно посмотреть с помощью текстового редактора или редактора файлов среды MATLAB. Файлы модели имеют расширение **.mdl**. Однако следует отметить, что эти файлы могут содержать тысячи строк кода, что отражает одно из основных свойств систем визуально-ориентированного программирования. Поэтому для простоты и удобства редактирования моделей используется уровень диаграмм и блоков без рассмотрения программных кодов. Simulink автоматизирует наиболее трудоемкий этап моделирования: он составляет и решает сложные системы алгебраических и дифференциальных уравнений, описывающих заданную функциональную схему (модель), обеспечивая удобный и наглядный визуальный контроль за поведением созданного пользователем виртуального устройства.

### 2. Запуск среды Simulink. Знакомство с каталогом библиотеки

Предварительным этапом перед ознакомлением с библиотеками Simulink является запуск самой среды моделирования. Для этого можно использовать пиктограмму  в панели управления главного рабочего окна MATLAB либо набрать команду **simulink** в рабочей области и запустить ее. Существуют и другие способы запуска среды Simulink, однако они подразумевают создание новой пустой модели, что будет рассмотрено далее. Работа в среде Simulink начинается с

окна Simulink Library Browser – SLB (каталог библиотек Simulink), графический интерфейс которого представлен на рис. 4.1.



**Рис. 4.1.** Графический интерфейс окна с каталогом библиотек среды Simulink

Интерфейс данного окна образуют следующие элементы.

1. Главное меню с пунктами (File, Edit, View и Help). Пункт по работе с файлами (File) содержит стандартные элементы – создания новой модели или библиотеки (New->Model/New->Library), открытия уже существующей модели/библиотеки (Open), закрытия текущей модели/библиотеки (Close), а также пункт настройки параметров среды моделирования (Preferences...). Для быстрой работы с блоками библиотеки служит пункт Edit, позволяющий оперативно находить интересующие пользователя системы и блоки с помощью функции поиска (Find), а также выбирать (Select ...) и добавлять (Add ...) блоки в открытую модель среды Simulink. Редактирование размеров пикто-



грамм блоков библиотек Simulink а также способов их отображения осуществляется в пункте меню (View). Оперативно получить справочную информацию по модели, блокам библиотеки или самой среде Simulink возможно через пункт меню help.

2. Панель инструментов с кнопками, дублирующими основные команды главного меню, а также содержащее поле для ввода компоненты поиска.

3. Панель Libraries – каталог библиотеки Simulink, которая имеет иерархическую структуру. Библиотека содержит обширнейший набор блоков общего и профильного назначения, упорядоченных путем многоуровневой системы вложений. Рассмотрим состав наиболее общей библиотеки среды Simulink, имеющей аналогичное название. Каждый из блоков этой библиотеки входит в состав подгруппы, объединенной по функциональному назначению. К таким подгруппам относятся:

a. **Commonly Used Blocks** – наиболее часто используемые пользователем блоки;

b. **Continuous** – блоки, необходимые для обработки аналоговых сигналов, компоненты с непрерывными характеристиками;

c. **Discontinious** – компоненты с разрывными характеристиками;

d. **Discrete** – компоненты для обработки дискретных сигналов;

e. **LookUp Table** – табличное задание зависимостей;

f. **Math Operations** – математические компоненты;

g. **Model Verifications** – верификация моделей;

h. **Model\_Wide Utilities** – дополнительные утилиты;

i. **Port & Subsystem** – порты и подсистемы;

j. **Signal Attributes** – блоки атрибутов сигналов;

k. **Signal Routing** – блоки маршрутизации сигналов;

l. **Functions & Tables** – функции и таблицы;

m. **Nonlinear** – нелинейные компоненты;

n. **Connections** – соединительные компоненты;

o. **Signals & Systems** – сигналы и системы;

p. **Sinks** – регистрирующие устройства;

q. **Sources** – источники сигналов и воздействий.

Подробное описание каждого из блоков библиотеки можно найти в разделах справочной информации либо в литературе [1–3]. Правее панели Libraries расположены панель отображения библиотек и блоков, расположенных в них.

4. Поле вывода информации (Block Description), расположенное внизу главного окна библиотеки Simulink, отображает сведения о текущем выбранном блоке или библиотеке.

Подробное описание справочной информации по пакету Simulink можно получить, введя в командное окно среды MATLAB следующую команду:

---

```
>>help simulink
```

---

В курсах, связанных с цифровой обработкой сигналов и телекоммуникациями, чаще всего задействованы блоки из трех библиотек среды Simulink.

1. Simulink – содержит каталоги блоков генерации, обработки, а также отображения простейших аналоговых и цифровых сигналов.
2. Communications Blockset – содержит каталоги кодирования, модуляции, синхронизации и эквалайзинга, моделирования каналов связи и визуализации происходящих в системах связи процессов.
3. Signal Processing Blockset – содержит каталоги фильтрации, в том числе многоскоростной и адаптивной, специализированные источники и приемники данных, спектрального анализа и потоковой обработки информации.

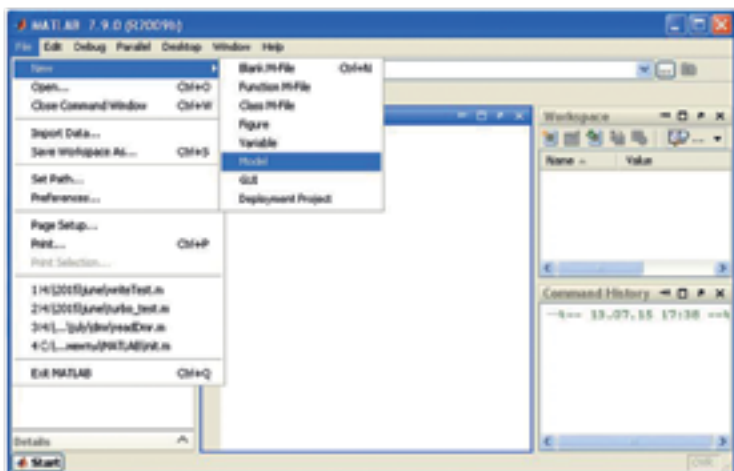
Подробное описание каждого из блоков в интересующей библиотеке можно получить в справочной литературе на русском языке [1–5], сайтах [6–8], а также непосредственно в справочной системе среды MATLAB.

### **3. Запуск среды моделирования. Знакомство со средой для создания динамических моделей – Simulink**

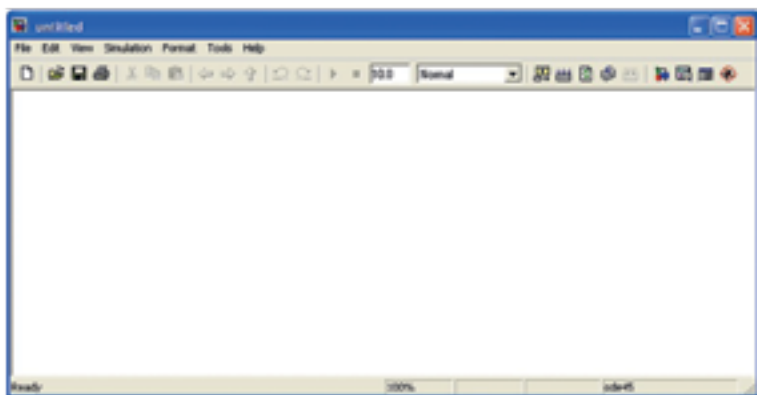
Как было отмечено ранее, при рассмотрении особенностей работы с библиотеками пакета расширения Simulink создание новой модели/библиотеки либо открытие уже существующей осуществляется через главное окно SLB. Альтернативой может служить создание/открытие модели или библиотеки через меню основного окна MATLAB. Графически процедура представлена на рис. 4.2.

После открытия окна для создания/редактирования моделей пользователь может приступить к процедуре добавления блоков, однако первоначально рекомендуется ознакомиться с элементами меню и панелью управления, которые изображены на рис. 4.3. Основное окно модели состоит из следующих элементов.

1. Заголовок с названием окна. Вновь созданному окну присваивается имя Untitled с соответствующим номером.
2. Панель меню, в составе элементов (File, Edit, View, Simulation, Format, Tools, Help).



**Рис. 4.2.** Элементы графического интерфейса главного окна системы MATLAB



**Рис. 4.3.** Окно модели Simulink 7.4

3. Панель инструментов.
4. Окно для создания схемы модели.
5. Строка информации, содержащая информацию о текущем состоянии модели.

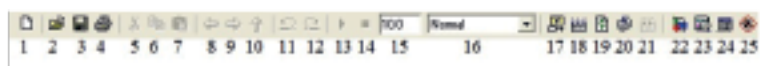
Меню окна содержит опции для редактирования модели, ее настройки и управления процессами расчета, работы с файлами и др., в частности:

- 1). File (Файл) – работа с файлами моделей и библиотек, а также редактирование параметров моделирования;

- 2). Edit (Редактирование) — изменение модели, создание подсистем и поиск блоков;
- 3). View (Вид) — управление показом элементов модели и интерфейса;
- 4). Simulation (Моделирование) — задание настроек для моделирования и управление процессом расчетов;
- 5). Format (Форматирование) — изменение внешнего вида блоков и модели в целом;
- 6). Tools (Инструментальные средства) — применение специальных средств для работы с моделью (отладчик, инструмент линейного анализа, преобразователь к фиксированной точке и т.д.);
- 7). Help (Справка) — открытие окна справочной системы в соответствии с различными разделами Simulink.

С полным списком подпунктов меню можно ознакомиться в специальной литературе [3].

Для работы с моделью удобно использовать кнопки на панели инструментов, цифровая идентификация которых представлена на рис. 4.4.



**Рис. 4.4.** Панель инструментов окна модели Simulink

Кнопки с 1 по 12 представляют собой инструменты для создания, открытия, сохранения, печати и других действий, связанных с редактированием модели. Их смысловое значение интуитивно понятно по иконкам или после прочтения названия соответствующего элемента. Кнопки 13 и 14 отвечают за процедуру запуска и завершения моделирования. Следует отметить, что повторное нажатие кнопки 13 приводит к остановке работы модели. Текстовое поле 15 отвечает за время моделирования, а меню выбора 16 предлагает возможность выбора режима работы S-модели (Normal, Accelerator, Rapid Accelerator и др.). Каждый режим работы модели характеризуется скоростью моделирования и необходимыми затратами вычислительных ресурсов и памяти. Следующие кнопки отвечают: за создание исполняемого кода модели (Incremental build — 18); обновление блоков модели (Refresh Model blocks — 19); обновление диаграмм (Update diagram — 20) и создание исполняемого кода для подсистем (Build subsystem — 21). Последние четыре элемента панели инструментов являются наиболее комплексными, в частности кнопка 22 позволяет запустить SLB из основного окна модели, по нажатию на 23 открывается окно настройки модели (Launch Model Explorer), отдельные элементы которого будут рассмо-

трены далее. Предпоследняя кнопка (24) отвечает за запуск окна обозревателя модели (Toggle Model Browser). При использовании данной опции в левой части окна модели будет открыто дополнительное окно, содержащее изображение иерархической структуры модели. И наконец, последняя кнопка отвечает за запуск отладчика.

В нижней части модели находится строка состояния, в которой отображаются краткие комментарии к кнопкам панели инструментов, а также к пунктам меню, когда указатель мыши находится над соответствующим элементом интерфейса. Это же текстовое поле используется и для индикации состояния Simulink: Ready (Готов) или Running (Выполнение). В строке состояния отображаются также:

- масштаб отображения блоков модели (в %, исходное значение равно 100%);
- индикатор продолжительности процесса моделирования;
- текущее значение модельного времени;
- используемый метод моделирования.

#### **4. Этапы создания S-модели. Общие принципы**

В общем случае технология создания S-модели состоит из следующих этапов:

1. Создания окна S-моделей. Подробное рассмотрение этого этапа представлено в предыдущем параграфе.

2. Поиск и добавление необходимых блоков в модель из каталога библиотеки среды Simulink. Создание единой системы в открытом окне, представленном на рисунке 4.3 возможно с помощью технологии “перетащить и отпустить” (drag-and-drop). Она заключается в последовательном перетаскивании блоков в окно S-модели при нажатой левой кнопки мыши из панели SLB.

3. Задание параметров блоков, которые определяют различные свойства сигнала на выходе или осуществляют настройку средств анализа. Для того, чтобы выполнить рассматриваемую процедуру необходимо дважды нажать на интересующий блок и установить требуемые для его работы параметры. Часто этапы 2 и 3 повторяются итеративно, когда по мере добавления новых блоков происходит установка необходимых значений внутри каждого из них.

4. Соединение блоков. Может выполняться в ручном и полуавтоматическом режиме. В первом случае соединение блоков происходит путем продолжительного нажатия левой кнопки мыши на выходе первого блока (символ стрелки) и перетаскивания появившейся соединительной линии к входу второго блока.

5. Сохранение модели осуществляется путем нажатия соответствующе кнопки в панели управления либо через меню файл. Имя файла может состоять из любой последовательности латинских букв, цифр и символа подчеркивания, начинающейся с буквы.

После сборки и сохранения модели с ней могут выполняться различные действия, такие как оформление, настройка, исследование. В зависимости от поставленных целей и сложности разработанной модели эти действия могут комбинироваться либо частично игнорироваться.

## **5. Настройка параметров S-модели в среде Simulink**

Рассмотрим более подробно вопросы, связанные с установкой параметров моделирования и базовыми принципами взаимодействия модели, разработанной в среде Simulink, с различными типами данных (переменными и массивами) MATLAB.

Для корректной работы модели необходимо, чтобы задаваемые параметры моделирования соответствовали тем требованиям, которые диктуются отдельными функциональными блоками, используемыми в ней. Поэтому перед началом симуляции (часто после сборки модели) задаются глобальные параметры моделирования. Задание параметров выполняется в панели меню Simulation/Configuration Parameters. Наибольшее значение для начинающего пользователя представляет вкладка решателя (Solver), элементы которой представлены на рис. 4.5.

Элементы, представленные во вкладке решателя, разделены на четыре группы.

1. Интервал моделирования (Simulation time), величина которого задается с помощью указателя начального (Start time) и конечного (Stop time) значений времени. Начальное время, как правило, задается равным нулю. Величина конечного времени задается пользователем, исходя из условий решаемой задачи.

2. Параметры решателя (Solver Options), при выборе которых необходимо указать метод интегрирования (Type) — либо с фиксированным (Fixed-step), либо с переменным (Variable-step) шагом. Подробное описание методов решения (Solver), а также дополнительные настройки можно найти в специализированной литературе [1].

3. Параметры управления задачами и интервалом времени (Tasking and sample time options). С их помощью контролируется интервал времени и процесс моделирования. В зависимости от выбора решателя (Solver) задаваемые в этой группе параметры могут сильно

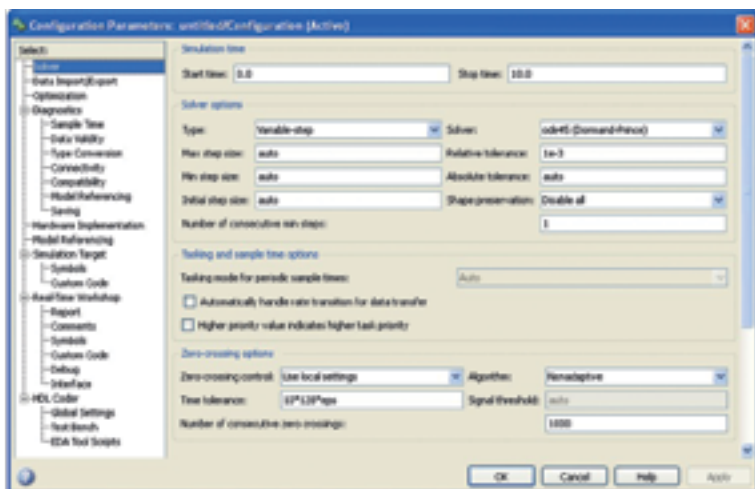


Рис. 4.5. Элементы вкладки решателя (Solver)

различаться. Подробнее об этом можно узнать в специализированной литературе [1].

4. Параметры, связанные с пересечением нуля в S-модели (Zero-crossing options). При выборе решателя с переменным шагом моделирования для аналоговых систем может использоваться метод «пересечение нуля», обеспечивающий контроль монотонности сигнала на каждом шаге моделирования и позволяющий максимально точно определять моменты резкого изменения значений сигнала.

При необходимости более подробную информацию по панели решателя можно получить в справочной системе MATLAB по Simulink, введя в командной строке следующую надпись:

---

```
>>help simulink
```

---

В результате появятся ссылки на различные разделы справочной информации по среде Simulink.

Следующей по значимости для пользователя вкладкой в панели меню Simulation/Configuration Parameters окна Select является вкладка настройки обмена данными S-модели системы с рабочим пространством памяти Workspace (Data Import/Export), которая представлена на рис. 4.6.

Параметры взаимодействия задаются в правой панели, разделенной на три группы.

1. Загрузка из рабочей области (Load from workspace) – выбираются параметры, контролирующие импорт из Workspace перед моделированием. Рассмотрим имеющиеся пункты подробнее.

- Флаг входные данные (Input) управляет импортом данных. При его установке указываются имена импортируемых данных, которые могут быть представлены в виде:
  - ✓ матрицы  $[t, u]$  (по умолчанию), где  $t$  – вектор-столбец возрастающих значений времени, а  $u$  – матрица, векторы-столбцы которой соответствуют значениям сигналов:  $i$ -й столбец соответствует значениям  $i$ -го сигнала в момент времени, заданный вектором  $t$ ;
  - ✓ структуры, заданной своим именем;
  - ✓ вектора времени  $t$ ;
  - ✓ выражением MATLAB в апострофах, представляющим собой функцию времени, например 'log(t)-0.1', где аргументу  $t$  в процессе моделирования будут автоматически присваиваться значения времени.

При использовании настроек по флагу Input передача данных из рабочего окна в модель будет осуществляться через блок In, расположенный в библиотеке по адресу Simulink/Sources/In).

- Флаг установки начального состояния (Initial state), по установке которого некоторое начальное состояние импортируется из Workspace в модель с помощью блока In.

2. Сохранение в рабочей области (Save to workspace) – отвечает за экспорт данных в Workspace после завершения моделирования. К параметрам этой группы относятся следующие флаги:

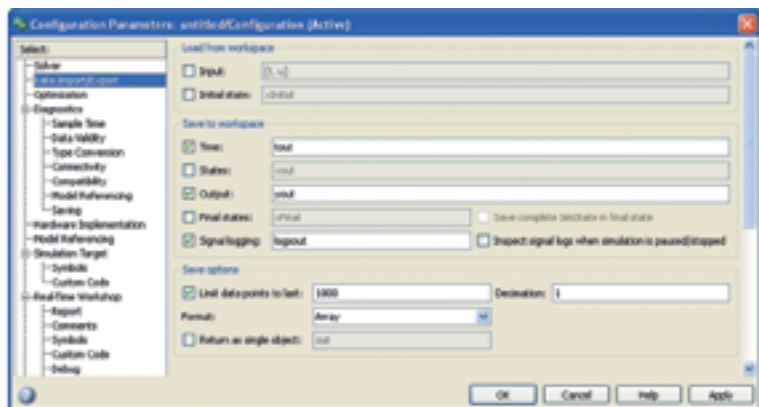
- времени (Time), отвечающий за экспорт значение указанного вектора в рабочую область;
- экспорта переменных состояний (States), при выставлении которого записанная в поле ввода переменная экспортируется в рабочую область переменных состояний системы. Вид экспортируемых переменных указывается в поле Format, которое будет рассмотрено далее;
- экспорта выходных данных (Output), при выставлении которого записанная в поле ввода переменная экспортируется в рабочую область с помощью блока Out (Simulink/Sinks/Out);
- экспорта конечных состояний (Final states) модели (по умолчанию – xFinal) в рабочую область переменных состояний на последнем шаге моделирования. При его установке активирует флаг сохранения результатов моделирования в последнем положении (Save complete SimState in final state);



- Регистрации сигнала (Signal logging), при установке которого в поле ввода указывается имя регистрируемого сигнала (по умолчанию logsout) и активируется флаг проверки регистрации сигнала при завершении моделирования или во время паузы (Inspect signal logs when simulation is paused/stopped).
3. Сохранение параметров (Save options). В состав этой группы входят следующие элементы.
- Флаг, разрешающий установку граничного числа данных для сохранения (Limit data points to last), с числовыми параметрами количества отсчетов и прореживающего значения (Decimation), указанных в соответствующих полях ввода.
  - Поле выбора формата (Format) экспортируемых данных, которые могут быть представлены в одном из трех видов:
    - ✓ массив (Array);
    - ✓ структура (Structure) — массив записей с пустым полем time;
    - ✓ структура с временем (Structure with time) — массив записей с полем time.
  - Поле параметров вывода (Output options), который используется только для решателей с переменным шагом моделирования и может принимать значения:
    - ✓ скорректированного вывода (Refine output), при выборе которого в поле ввода коэффициента коррекции (Refine factor) можно задать дополнительные точки для вывода экспортируемых данных, причем значение по умолчанию означает отсутствие дополнительных точек вывода;
    - ✓ дополнительного вывода (Produce additional output), позволяющего задать с помощью коэффициента коррекции дополнительные точки вывода в виде вектора в квадратных скобках;
    - ✓ определенного вывода (Produce specified output only), позволяющего задать только те точки, в которых будут выводиться выходные данные в виде вектора в квадратных скобках.
  - Флаг установки единого выхода (Return as single object), смысл которого заключается в объединении всех выходных данных модели в единый объект с именем, указанным в поле ввода.

## 6. Разработка простейшей модели. Пошаговое описание необходимых действий, узлов и систем

Рассмотрим пошагово все описанные выше этапы на примере создания и моделирования простейшей S-модели в среде Simulink. Модель представляет собой систему обработки гармонического колебания.



**Рис. 4.6.** Элементы вкладки обмена данными S-модели с рабочим пространством памяти при включении фиксированного шага моделирования в решателе




Создание модели в рассматриваемом примере содержит следующие этапы.

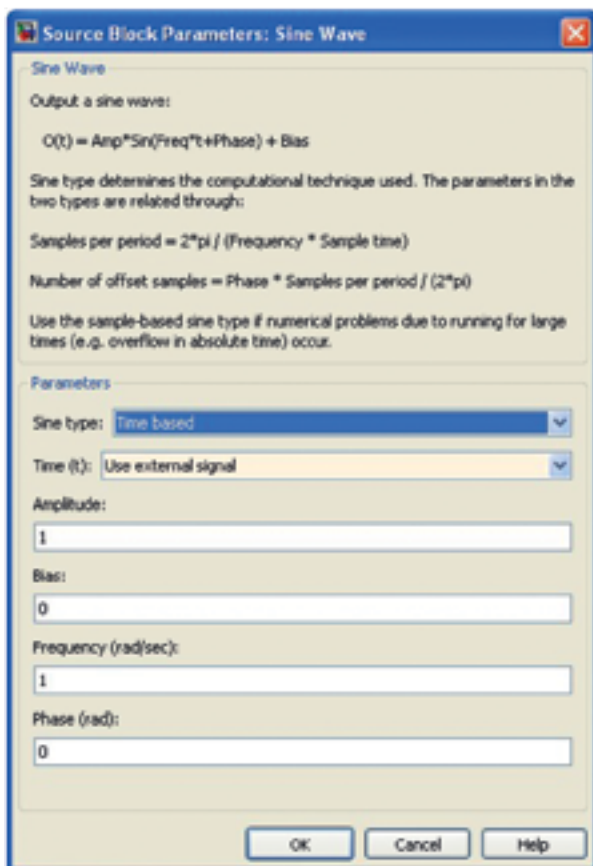
1. Создание и настройка простейшей S-модели из блоков библиотеки Simulink.
2. Добавление в модель уровней иерархии путем объединения простых блоков в подсистемы и их редактирование.
3. Изменение текстового оформления модели и типов данных, используемых в модели.

Рассмотрим эти этапы подробнее.

## 6.1. Разработка и редактирование параметров простейшей S-модели

Процедура создания S-модели из блоков библиотеки состоит из следующих действий.

1. Открываем SLB с помощью команды `simulink`, введенной в рабочую область главного окна MATLAB.
2. Создаем новую S-модель путем нажатия соответствующей кнопки (вставить кнопку) в библиотеке среды Simulink.
3. Добавляем в окно S-модели блоки из библиотеки:
  - генератора гармонических колебаний (Simulink\Sources\Sine Wave) ;
  - интегратора (Simulink\Continuous\Integrator) ;
  - визуализации данных (Simulink\Sinks\Scope) .

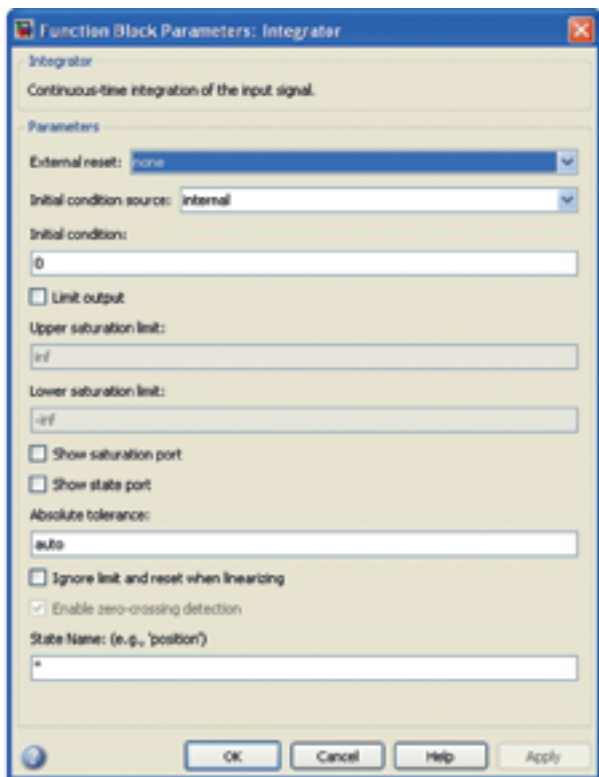


**Рис. 4.7.** Параметры генератора гармонических колебаний

4. Редактируем параметры блоков и знакомимся с их содержанием.

Генератор гармонических колебаний (рис. 4.7) формирует сигнал с заданной амплитудой, частотой, фазой и смещением.

Для формирования сигнала блоком могут использоваться два алгоритма, тип которых определяется параметром Sine Type (способ формирования сигнала) — Time based (по текущему времени) или Sample-based (по величине шага модельного времени). При выборе способа Time based формирование сигнала происходит по первой формуле, представленной в описании блока, где значения амплитуды (Amplitude), постоянной составляющей (Bias), частоты (Frequency)



**Рис. 4.8.** Параметры интегратора

(rad/sec)) и фазы (Phase) могут быть записаны в соответствующие текстовые поля.

Отдельно стоит остановиться на шаге модельного времени (Sample time), который может принимать три принципиально разных значения. При Sample time = -1 период дискретизации наследуется от предыдущего блока, при нулевом значении генератор выдает аналоговый сигнал. Во всех остальных случаях значение Sample time соответствует дискретному шагу генерации гармонического колебания.

Интегратор выполняет функцию накопления поступающих отсчетов в непрерывном масштабе времени и обладает рядом специфических параметров, указанных на рис. 4.8.

Ключевые параметры и их обозначение представлены далее.

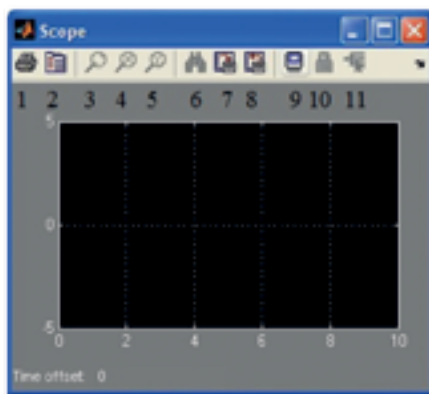
- Внешний сброс интегратора (External reset). Сброс производится по переднему (rising), заднему (falling) или обоим (either) фронтам управляющего импульса. Также можно регулировать

функцию сброса по уровню (level) и порогу (level hold). При отсутствии сброса (none) интегрирование происходит непрерывно. Для систем передачи данных наиболее актуально интегрирование с возможностью сброса по управляющему сигналу.

- Источник начального состояния интегратора (Initial condition source) задается внутренним (internal), что приводит к появлению текстового поля для его задания (Initial condition), либо внешним. В этом случае появляется дополнительный вход в блоке для подключения необходимого сигнала управления.

Остальные параметры используются реже, и их назначение можно посмотреть в разделе справочной информации нажатием кнопки **Help** или в специальной литературе [5].

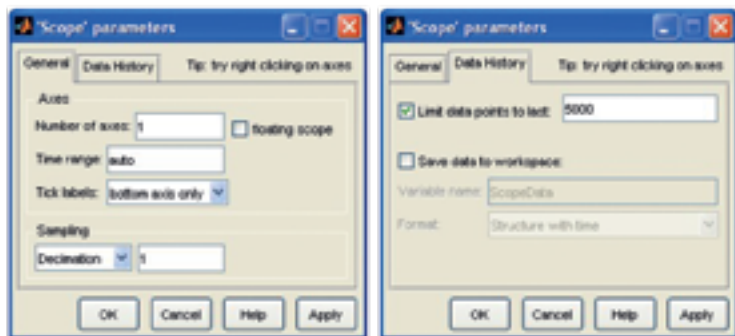
Блок визуализации данных выполняет функцию отображения поступающих на него сигналов в двумерном виде. Его графическое представление дано на рис. 4.9.



**Рис. 4.9.** Графическое представление блока визуализации данных

В отличие от интегратора и генератора блок визуализации настраивается до начала моделирования, а отображение информации и дополнительные функции могут быть изменены уже по завершении работы модели. Рассмотрим элементы панели инструментов:

- 1). печать содержимого;
- 2). вызов окна настройки параметров блока Scope;
- 3). изменение масштаба одновременно по обеим осям графика;
- 4). изменение масштаба по горизонтальной оси;
- 5). изменение масштаба по вертикальной оси;
- 6). автоматическая установка масштаба осей по умолчанию;
- 7). сохранение установок параметров осей;



**Рис. 4.10.** Окно настройки блока Scope

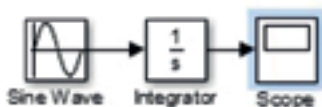
- 8). включение холостого подсоединения блока;
- 9). шлюз селектора сигналов;
- 10). селектор сигналов.

Кнопки 3–5 альтернативные, так как использование одной из них запрещает одновременное включение остальных. Они недоступны до момента появления графика, поэтому применяются только после завершения моделирования.



Главная форма настройки параметров блока Scope вызывается по нажатию второй кнопки (рис. 4.10). В диалоговом окне Scope parameters имеются две вкладки: «Общие» (General), позволяющая установить параметры осей, и «Представление данных» (Data history), предназначенная для определения параметров представления данных блока Scope. В первой вкладке окна General устанавливается количество отображаемых осей (Number of axes), которое соответствует количеству входных портов блока Scope, также задается верхняя граница модельного времени, откладываемого по оси абсцисс. С помощью раскрывающегося списка меток осей (Tick Labels) задается вид оформления осей координат на графиках окна Scope. Установка флажка floating scope отключает все входы блока Scope; если блок был подключен, все связи автоматически обрываются. В области дискретизации вкладки General находится список, в котором по умолчанию выбран элемент «Прореживание» (Decimation) с коэффициентом, равным единице. Этот факт символизирует о том, что все данные, которые поступают в блок Scope, будут отображены, если имеется достаточный объем памяти. Увеличение коэффициента прореживания либо установка параметра Sample Time приведет к изменению количества отображаемых точек графика. Вкладка Data History позволяет задать максимальное количество (начиная с конца) элементов массивов данных,

которые используются для построения графиков в окне Scope. Для этого используется поле, определяющее максимальное количество точек (Limit data points to last). Если установить флажок записи данных в рабочую область (Save data to workspace), то все отображаемые блоком Scope данные будут продублированы в массиве или структуре с соответствующим именем (Variable name) и форматом (массив – Array, структура – Structure, структура со временем – Structure with time).

После знакомства с параметрами отдельных блоков и их установкой в соответствии с имеющимся заданием или логикой работы модели необходимо объединить блоки в единую систему вида:



Соединение блоков можно выполнять тремя различными способами:

- 1). зажатием левой кнопки мыши (ЛКМ) на значке  выходного порта блока источника и доведением полученной линии (указатель меняется на перекрестие) до входа блока приемника , не отпуская левую кнопку мыши до момента появления символа соединения блоков (двойное перекрестие);
- 2). нажатием на блок источника ЛКМ, зажатием клавиши Ctrl и последующим нажатием ЛКМ на блоке приемника;
- 3). перетаскиванием ЛКМ блока в любое место линии соединения имеющихся блоков, причем момент появления специфических символов разрыва будет свидетельствовать о возможности вставки блока.

При первом запуске модели обратимся к окну задания параметров моделирования с помощью решателя (Simulink/Parameters/Solver). Зададим время расчета указанием начального и конечного времени (Start time, Stop time) и выберем тип моделирования с переменным шагом (Variable step), решатель оставим без изменения. Подробнее настройки модели рассмотрены в п.5, справочной системе Simulink и специализированной литературе [1].



После запуска модели результаты ее работы можно наблюдать в окне блока Scope, открыв его двойным нажатием ЛКМ. В данном случае нам станут доступны инструменты, в частности масштабирование, предназначенные для работы с моделью после завершения моделирования.

Усложним модель, добавив блоки усилителя (Simulink/Math Operations\Gain) и ограничителя амплитуды (Simulink\Discontinuities\

Saturation). Пиктограммы блоков, их настройка и описание представлены в табл. 4.1.

Таблица 4.1.

**Описание, назначения и параметров блоков**

Пиктограмма	Описание назначения и настроек
	<p>Блок Gain предназначен для изменения амплитуды входного сигнал(ов) на указанное в нем значение коэффициента усиления.</p> <p>Основная вкладка настроек (Main) усилителя (рис. 4.11) состоит из трех полей:</p> <ul style="list-style-type: none"> <li>• коэффициента усиления (Gain), который может быть переменной, вектором или массивом;</li> <li>• типа умножения (Multiplication) в зависимости от типа данных и логики работы модуля (поэлементное либо матричное);</li> <li>• интервала дискретизации (Sample time), по умолчанию принимающего значение <math>-1</math>, т.е. наследуется от предыдущего блока.</li> </ul>
	<p>Блок Saturation реализует линейную зависимость с насыщением. Выходная величина этого блока совпадает с входной, если последняя находится в указанном диапазоне. Если же входная величина выходит за границы диапазона, то выходной сигнал принимает значение ближайшей из границ.</p> <p>Основная вкладка (Main) настроек ограничителя амплитуды (рис. 4.12) состоит из трех полей и двух дополнительных установок:</p> <ul style="list-style-type: none"> <li>• поля верхнего и нижнего уровней ограничения (Upper limit и Lower limit);</li> <li>• интервала дискретизации (Sample time), по умолчанию принимающего значение <math>-1</math>, т.е. наследуется от предыдущего блока.</li> </ul> <p>С наименованием значений дополнительных установок можно познакомиться в справочной системе среды Simulink</p>

В результате редактирования блоков коэффициенту усиления блока Gain присваивается векторное значение 3:10, а интервал амплитудного ограничения задается верхней —  $\text{rand}(8,1)*10$  и нижней —  $-\text{rand}(8,1)*10$  границами. Таким образом, получается многоканальная система обработки информации:





Количество каналов обработки определяется размерностью вектора, указывающего на коэффициенты усиления соответствующего блока.

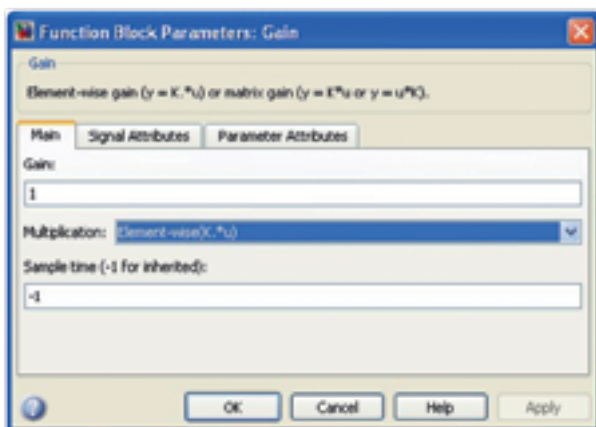


Рис. 4.11. Панель настроек блока усиления сигнала

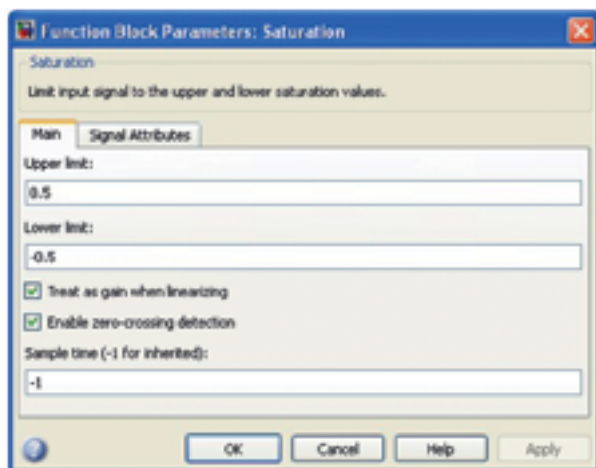


Рис. 4.12. Панель настроек блока амплитудного ограничения сигнала


После запуска процесса моделирования в окне блока визуализации будут отражены отрезки гармонических колебаний с различной амплитудой и уровнем ограничения.

Полученная модель работает исключительно с аналоговыми сигналами, для преобразования ее к дискретному виду потребуется добавить в нее ряд блоков, выполняющих процедуру дискретизации (пе-

редискретизации), представленных в табл. 4.2 (Simulink\Discrete\Zero-order Hold).

Таблица 4.2.

**Описание, назначение и параметры блока дискретизации**

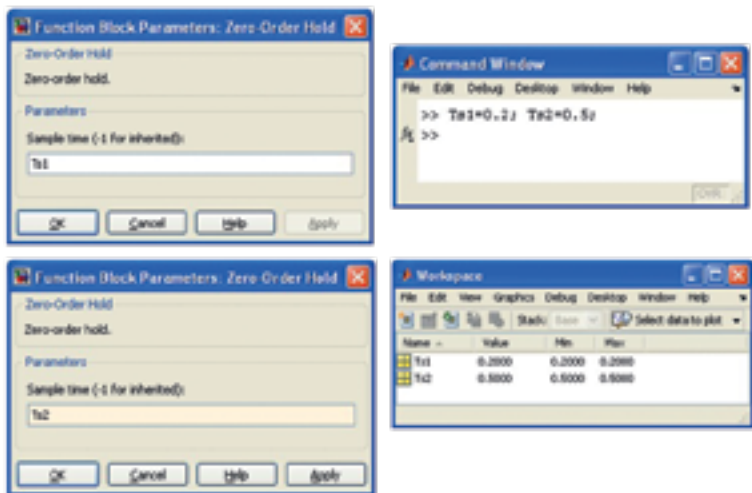
Пиктограмма	Описание назначения и настроек
	<p>Блок Zero-order Hold предназначен для преобразования непрерывного сигнала в дискретный либо повторной дискретизации сигнала.</p> <p>Рассматриваемый преобразователь имеет единственный параметр – интервал дискретизации (Sample time), по умолчанию принимающий значение –1, т.е. наследуется от предыдущего блока</p>

Вводим элементы передискретизации в S-модель в количестве двух штук, преобразуя аналоговую систему в дискретную. Редактируем значение периода дискретизации в каждом из блоков через окно Workspace ( $Ts1=0.2$ ,  $Ts2=0.5$ ). Результат установки параметров для выбранных блоков представлен на рис. 4.13. После добавления двух входов в блоке Scope S-модель примет вид:




## 6.2. Разработка и редактирование подсистем

Для упрощения визуального понимания модели, а также знакомства с понятием подсистем создадим подсистему обработки, которая состоит из блоков усилителя, амплитудного ограничителя и двух дискретизаторов. Для создания подсистемы необходимо выделить интересующие блоки S-модели, нажать на одном из них правой кнопкой мыши (ПКМ) и в открывшемся окне выбрать пункт Create Subsystem. После этого на месте выбранных блоков появится всего один – новая подсистема. Нажав на нее ПКМ, в открывшемся окне выберем пункт маскирования (Mask Subsystem...) для редактирования параметров вновь созданной подсистемы (рис. 4.14). В открывшемся окне редактора напишем команду в пункте Icon Drawing commands надпись



**Рис. 4.13.** Установка параметров блоков S-модели через командное окно среды MATLAB

«plot(peaks)», которая изменит внешний вид подсистемы. Далее введем два управляемых параметра в подсистему во вкладке «Параметры» (Parameters) редактора маски (Mask Editor). Для этого необходимо нажать кнопку «Добавить новый параметр» . После этого появится новая строка в таблице с соответствующими текстовыми полями:

- текстовое описание переменной (Prompt), в которой раскрывается смысловое значение переменной;
- название переменной (Variable), используемое в блоках внутри подсистемы вместо цифровых констант;
- способ задания переменной (Type->edit/checkbox/popup/DataTypeStr/Minimum/Maximum).

Введем два управляемых параметра в подсистему: «Prompt: Sample Time 1, Variable: Ts1» и «Prompt: Sample Time 2, Variable: Ts2». И наконец, оформим документацию к блоку. Для этого в редакторе маски необходимо открыть вкладку Documentation и заполнить соответствующие текстовые поля:

- тип маски (Mask type), в котором указывается краткое название подсистемы, например «This is my custom subsystem»;
- описание маски (Mask description), например «This subsystem samples the signals at a specified rate»;
- справочная информация о подсистеме (Mask help). Заполнение этого текстового поля приводит к созданию специального .html-файла, который будет входить в состав основной библи-

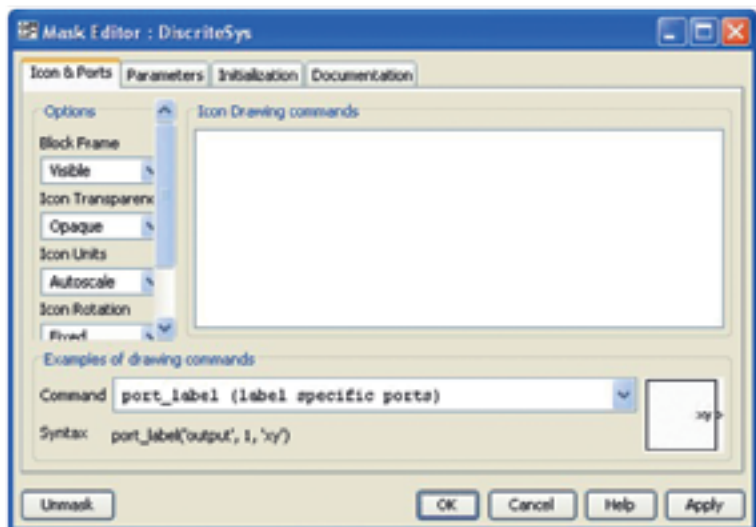



Рис. 4.14. Редактор маски подсистемы (Mask Editor)

отеки и вызываться по нажатию клавиши help в открытом окне редактора подсистемы.

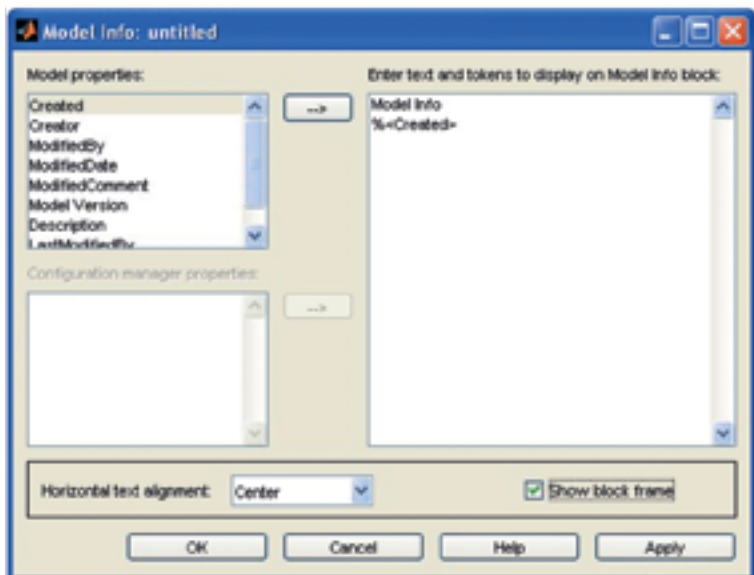
С учетом внесенных изменений S-модель имеет вид



### 6.3. Оформление S-модели и изменение типов используемых данных

Добавим текстовое описание S-модели. Для этого щелкаем ЛКМ на любое свободное место модели, где нет блоков и соединительных линий, и вводим в открывшемся текстовом поле необходимую информацию, например «Дискретизация модифицированного гармонического сигнала». После этого добавим блок информации (Simulink\ Model-Wide Utilities\Model Info) о текущей версии модели, дате создания, авторских правах и некоторых других особенностях. При открытии данного блока появляется меню редактора, который представлен на рис. 4.15. Редактирование отображаемой на блоке информации происходит путем выбора интересующего поля, например даты создания (%Created), с последующим нажатием на кнопку переноса функции  на панель отображения. Таким образом, любую

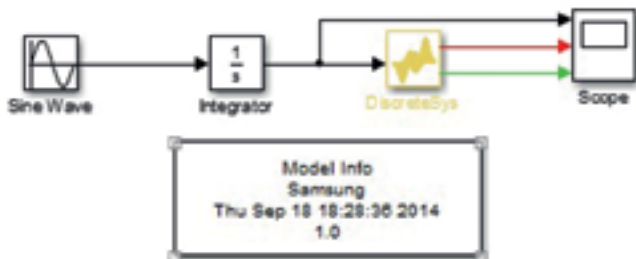
созданную S-модель можно персонифицировать, а также отразить ее версию и множество других полезных параметров.



**Рис. 4.15.** Редактор информации S-модели (Model Info)

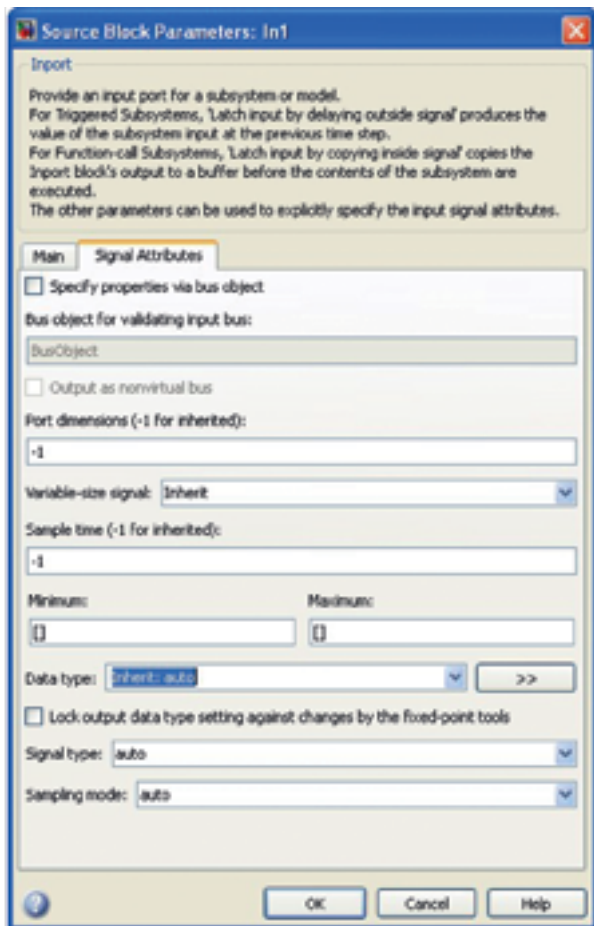
В результате оформления модели ее внешний вид изменится следующим образом:

#### Дискретизация модифицированного гармонического сигнала




Периодически возникают задачи, связанные с моделированием тех или иных алгоритмов с учетом их реализации на устройствах с ограниченной разрядностью в формате с фиксированной точкой. В этом случае требуется редактировать параметры блоков S-модели, отвечающих за разрядность представления чисел. Чаще всего такие параметры представлены во многих блока не в главной вкладке (Main), а в

разделе свойств сигналов (Signal Attributes). Рассмотрим эту вкладку подробнее на примере входного порта In1 подсистемы DiscreteSys созданной S-модели. Окно с описанием редактируемых полей входного порта представлено на рис. 4.16.



**Рис. 4.16.** Окно входного порта In1

Для работы в формате с фиксированной точкой с разрядностью, равной 16, установим тип данных (Data type) равным `fixdt(1,16,0)`. Размах сигнала с такой разрядностью ограничим минимальным и максимальным значениями, установив в поля Minimum и Maximum числа  $-10$  и  $10$  соответственно. Далее, нажав на кнопку расширения опций , подбираем оптимальное количество битов, приходящихся на

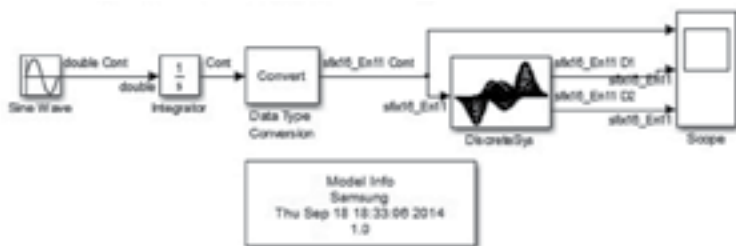
целую и дробную части числа, нажатием кнопки Calculate Best Precision Scaling. В результате этого параметр «Дробная часть» (Fractional length) становится равным 11. Подробности заданного формата представления чисел изображены на рис. 4.17.



**Рис. 4.17.** Параметры заданного формата для фиксированной точки с разрядностью представления данных, равной 16

После внесения изменений во входной порт In1 подсистемы DiscreteSys изменим параметры отображения типов данных S-модели. Для этого необходимо в пункте меню «Форматирование» (Format) редактора S-моделей выбрать подпункт отображения сигналов и портов (Port/Signal Displays) и нажать на поле отображения типов данных (Port Data Types). На следующем шаге изменяем атрибуты выходных портов подсистемы по аналогии с входным портом In1. Ускорение процедуры редактирования блоков можно получить, если перейти от редактирования отдельных блоков к изменению совокупности блоков, имеющих одинаковые параметры, с использованием браузера S-модели (Model Explorer). Для того чтобы модель стала работать корректно, в нее потребуется дополнительно добавить блок преобразования типов данных (Simulink\Signal Attributes\Data Type Conversion) и изменить один параметр множителя, отвечающий за преобразование типа входных данных (Output data type), на наследование от предыдущего блока (Inherit via back propagation). В результате рассмотренных изменений конечная версия модели примет вид.




## Дискретизация модифицированного гармонического сигнала



Таким образом, были рассмотрены основные принципы практической работы в редакторе моделей среды Simulink.

## 7. Практическая работа по созданию и моделированию простейшей S-модели

1. Создать новую S-модель в среде Simulink из главного окна MATLAB любым из доступных вам способов.
2. Открыть окно библиотеки блоков среды визуального моделирования Simulink (Simulink Library Browser).
3. Создать простейшую S-модель, для чего разместить в окне модели следующие блоки с параметрами, которые предварительно необходимо вычислить в соответствии со своим вариантом задания. Установка параметров в буквенном виде предполагает их предварительное определение (вычисление) в рабочей области MATLAB через командное окно либо специально написанную программу. Блоки, параметры которых не указаны, должны остаться без изменения.

1		Генератор случайных чисел (Simulink/Sources/Uniform Random Number) Период следования импульсов (Sample Time) $T_b = 1/F_b$
2		Определитель знака (Simulink/Math Operations/Sign)
3		Передискретизатор (Simulink/Discrete/Zero-Order Hold) Период дискретизации (Sample Time) $T_{base} = 1/F_{base}$
4		Виртуальный осциллограф (Simulink/Sinks/Scope) Настроить количество отображаемых данных таким образом, чтобы выводилась вся информация на интервале моделирования (величина интервала моделирования — ниже по тексту)



После установки параметров необходимо соединить блоки последовательно четырьмя разными способами и пронаблюдать результат моделирования при каждом из них на экране виртуального осциллографа, сохранив результат в свою рабочую папку в виде нескольких .jpg-рисунков. Последовательности соединения должны быть следующими 1-4, 1-2-4 и 1-2-3-4. Пример одного из вариантов соединения выглядит так:





Для запуска системы необходимо установить параметры моделирования следующим образом:

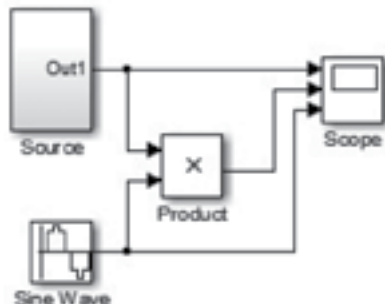
- выбрать пункт меню Simulation/Configuration Parameters;
- в диалоговом окне Configuration Parameters на закладке Solver установить следующие параметры: время начала симуляции (Start time) – 0; время окончания симуляции (Stop time) –  $1e-4$ ; тип шага (Type) – Variable-step), после чего, нажав кнопку «ОК», можно приступить к процессу моделирования и сохранения результатов.

4. Создать подсистему источника информации. Для этого необходимо отметить первые три блока текущей S-модели и выбрать пункт меню Edit/Create Subsystem. После этого необходимо изменить название полученной S-модели с Subsystem на Source.

5. Добавить в S-модель из библиотеки следующие блоки с параметрами:



5		Генератор гармонических колебаний (Simulink/Sources/Sine Wave) Частота (Frequency) - $2 \cdot \pi \cdot F_c$ ( $F_c = 3 \cdot F_b$ ) Период дискретизации (Sample Time) - Tbase
6		Перемножитель (Simulink/Math Operations/Product)

6. Соединить имеющиеся блоки следующим образом, предварительно увеличив количество входных портов виртуального осциллографа до трех, установив параметр Number of Axis=3:

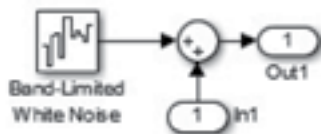


7. Пронаблюдать и сохранить в .jpg-файл результаты моделирования из блока Scope в рабочую директорию.



8. Построить простейшую систему, вносящую искажения в передаваемую информацию. Для этого необходимо добавить в модель следующие блоки с параметрами:





7		Генератор шума (Simulink/Sources/Band-limited White Noise) Мощность шума (Noise Power) – Tbase (здесь учитывается только число, размерность и смысловое значение переменной не передается) Период дискретизации (Sample Time) – Tbase
8		Сумматор (Simulink/Math Operations/Sum)

9. Объединить блоки в подсистему с названием Channel вида (при открытии подсистемы)

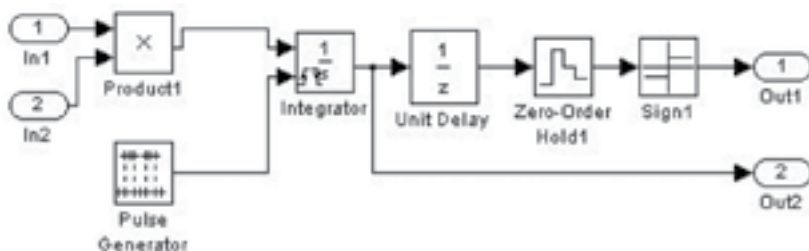


10. Собрать простейший приемник на основе блоков перемножения и интегрирования с управляемым сбросом. Для этого потребуются:

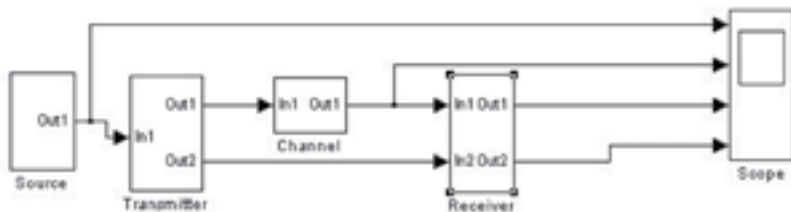
9		Перемножитель (Simulink/Math Operations/Product)
10		Интегратор (Simulink/Continuous/Integrator) Добавить внешний вход для сброса

11		Генератор прямоугольных импульсов (Simulink/Sources/Pulse Generator) Установить параметры таким образом, чтобы интегрирование выполнялось на интервале, равном $1/(2 \cdot F_b)$
12		Задержка сигнала (Simulink/Discrete/Unit Delay)
13		Передискретизатор (Simulink/Discrete/Zero-Order Hold) Период дискретизации (Sample Time) $1/F_b$
14		Определитель знака (Simulink/Math Operations/Sign)

Объединить блоки в подсистему с названием Receiver и добавить дополнительный блок Simulink/Sincs/Out. Если в результате создания подсистемы блоков количество входов и выходов получилось отличным от двух, то необходимо увеличить или уменьшить их количество до цифры 2. Полученная в результате соединения блоков подсистема Receiver должна иметь вид:



11. Окончательная система принимает вид:



12. Запустить процесс моделирования итоговой системы. Выполняя масштабирование горизонтальной оси в открытом блоке Scope, выделить фрагменты каждого из четырех сигналов длительностью от 10 до 20Tb. Сохранить полученные графики в рабочей директории.

Варианты заданий представлены в табл. 4.3.

## Варианты заданий

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fb, Мбит/с	30	25	10	12	30	40	0,1	20	25	10	15	35	8	5	45
Fbase, Гц	0,9	0,8	0,85	0,96	1,8	0,8	0,1	0,6	0,8	0,7	0,4	1	0,1	0,1	1
№	16	17	18	19	20										
Fb, Мбит/с	4	3	9	13	14										
Fbase, Гц	0,92	0,93	1,1	0,61	0,5										

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Солонина А.И.* Цифровая обработка сигналов. Моделирование в Simulink [Текст]: учебное пособие/ А.И. Солонина. — СПб.: БХВ-Петербург, 2012. — 432 с.: ил.
2. *Дьяконов В.П.* Simulink 5/6/7[Текст]: самоучитель. — М.: ДМК-Пресс, 2008.
3. *Черных И.В.* Simulink: среда создания инженерных приложений/ И.В. Черных [Текст]. — М.: ДИАЛОГ-МИФИ, 2004.
4. *Сергиенко А.Б.* Цифровая обработка сигналов [Текст]/ А.Б. Сергиенко — 3-е изд. — СПб.: БХВ-Петербург, 2010.
5. *Солонина А.И.* Цифровая обработка сигналов. Моделирование в MATLAB [Текст]/ А.И. Солонина, С.М. Арбузов. — СПб.: БХВ-Петербург, 2008.
6. [www.mathworks.com](http://www.mathworks.com) - официальный сайт компании разработчика программных продуктов MATLAB и Simulink.
7. [www.matlab.ru](http://www.matlab.ru) — российское представительство MATLAB.
8. [www.matlab.exponenta.ru](http://www.matlab.exponenta.ru) — главный русскоязычный форум, посвященный MATLAB и Simulink.

*Учебное издание*

*Алексей Анатольевич Овинников*

**ОСНОВЫ РАБОТЫ В СРЕДАХ  
MATLAB И SIMULINK**

*Учебное пособие*

Оригинал-макет подготовлен в Издательстве «КУРС»

Подписано к использованию 02.11.2022.

ООО Издательство «КУРС»

127273, Москва, ул. Олонекская, д. 17А, офис 104. Тел.:  
(495) 203-57-83.

E-mail: [kursizdat@gmail.com](mailto:kursizdat@gmail.com)      <http://www.kursizdat.ru>

## ЗАМЕТКИ

## ЗАМЕТКИ



## ЗАМЕТКИ