МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«Пензенский государственный университет» (ПГУ)

Работа с MATLAB и Simulink

Методические указания
к лабораторным работам

Составители:
В. В. Регеда,
О. Н. Регеда

Пенза
Издательство ПГУ
2014

УДК 001.8 (083.95)
Р13

Р е ц е н з е н т
доктор технических наук, профессор,
академик Российской метрологической академии,
заместитель директора Государственного регионального центра
стандартизации, метрологии и испытаний в Пензенской области
А. А. Данилов

Р13 **Работа с MATLAB и Simulink** : метод. указания к лабораторным работам / сост.: В. В. Регеда, О. Н. Регеда. – Пенза : Изд-во ПГУ, 2014. – 72 с.

Рассмотрены основные приемы решения средствами пакета MATLAB математических задач, возникающих в инженерной практике.

Издание подготовлено на кафедрах «Электротехника и транспортное электрооборудование» и «Информационно-измерительная техника» Пензенского государственного университета и предназначено для лабораторных и индивидуальных занятий студентов вузов, обучающихся по направлениям «Электроэнергетика и электротехника» и «Приборостроение», изучающих дисциплины «Компьютерные технологии» и «Компьютерные технологии в приборостроении».

УДК 001.8 (083.95)

СОДЕРЖАНИЕ

Введение

В настоящее время для технических расчетов и интерактивной разработки алгоритмов наряду с программной средой MathCAD [1] широко используется программная среда MATLAB. Ее достоинством является наличие большого числа стандартных пакетов прикладных программ. Наиболее часто используемое приложение – пакет Simulink, предназначенный для моделирования линейных и нелинейных динамических систем.

MATLAB представляет собой основу всего семейства продуктов компании MathWorks. Она используется для моделирования динамических объектов, разработки систем управления и коммуникационных систем, обработки сигналов и изображений, измерения сигналов и тестирования сложных систем и т.д.

Название системы MATLAB (сокращение от англ. «Matrix Laboratory») связано с тем, что ее работа основана на расширенном представлении и применении матричных операций для реальных, комплексных и аналитических типов данных.

Ядро MATLAB содержит встроенные функции линейной алгебры, быстрого преобразования Фурье, функции для работы с полиномами, функции базовой статистики и численного решения дифференциальных уравнений. Кроме того, MATLAB позволяет визуализировать математические функции и экспериментальные данные, реализовывать вычислительные алгоритмы, конструировать графический интерфейс пользователя для решения специфических задач, а также через специальные интерфейсы взаимодействовать с другими языками программирования и программами [2, 3].

В данных методических указаниях описывается версия MATLAB R2013a, интерфейс которой содержит *окно редактирования* и *интерфейсную ленту*, на *вкладках* которой сгруппированы связанные команды. С целью сохранения преемственности с более ранними версиями MATLAB в данных методических указаниях после указания пути доступа к командам в скобках указывается альтернативный путь доступа для интерфейса с меню, используемым в ранее созданных версиях пакета.

Лабораторная работа № 1

Знакомство со средой и основными объектами MATLAB

Цель работы: знакомство с интерфейсом и основными правилами работы в MATLAB при вычислении алгебраических выражений с использованием встроенных функций.

Рабочее задание

1 При запуске программы MATLAB открывается ее рабочая среда (рисунок 1.1), содержащая ленту, на вкладках которой сгруппированы связанные команды, а также несколько окон, главное из которых *командное окно* среды MATLAB – Command Window. Все символы команд, которые пользователи набирают с клавиатуры, результаты выполнения этих команд и информация об ошибках отображаются в командном окне.

Интерфейс программы можно настроить с помощью команды **Home⇒Layout** (для интерфейса с меню – это раздел меню **View**), подключая требуемые окна.

Выберите команду **HOME⇒Layout⇒Default** (для интерфейса с меню – укажите требуемые окна в разделе меню **View**).

Приглашением к вводу в командном окне является знак fx >> и мигающий вертикальный курсор, после которого вводятся требуемые символы с клавиатуры.

В MATLAB вычисление выражений в *режиме калькулятора*, а также выполнение команд в *программном режиме* осуществляются после нажатия на клавишу Enter.

Сеанс работы в MATLAB называют *сессией*. Входящие в сессию определения переменных и функций, которые расположены в рабочей области памяти, можно записать на диск в формате файла .mat с помощью команды **Save Workspace** и считать с диска с помощью команды **Import Data**. Эти команды можно также ввести в окне команд, набрав с клавиатуры соответственно save или load.

При выполнении работы следует копировать в текстовый файл все вычисления и результаты по каждому пункту из командного окна для продолжения работы при составлении отчета.

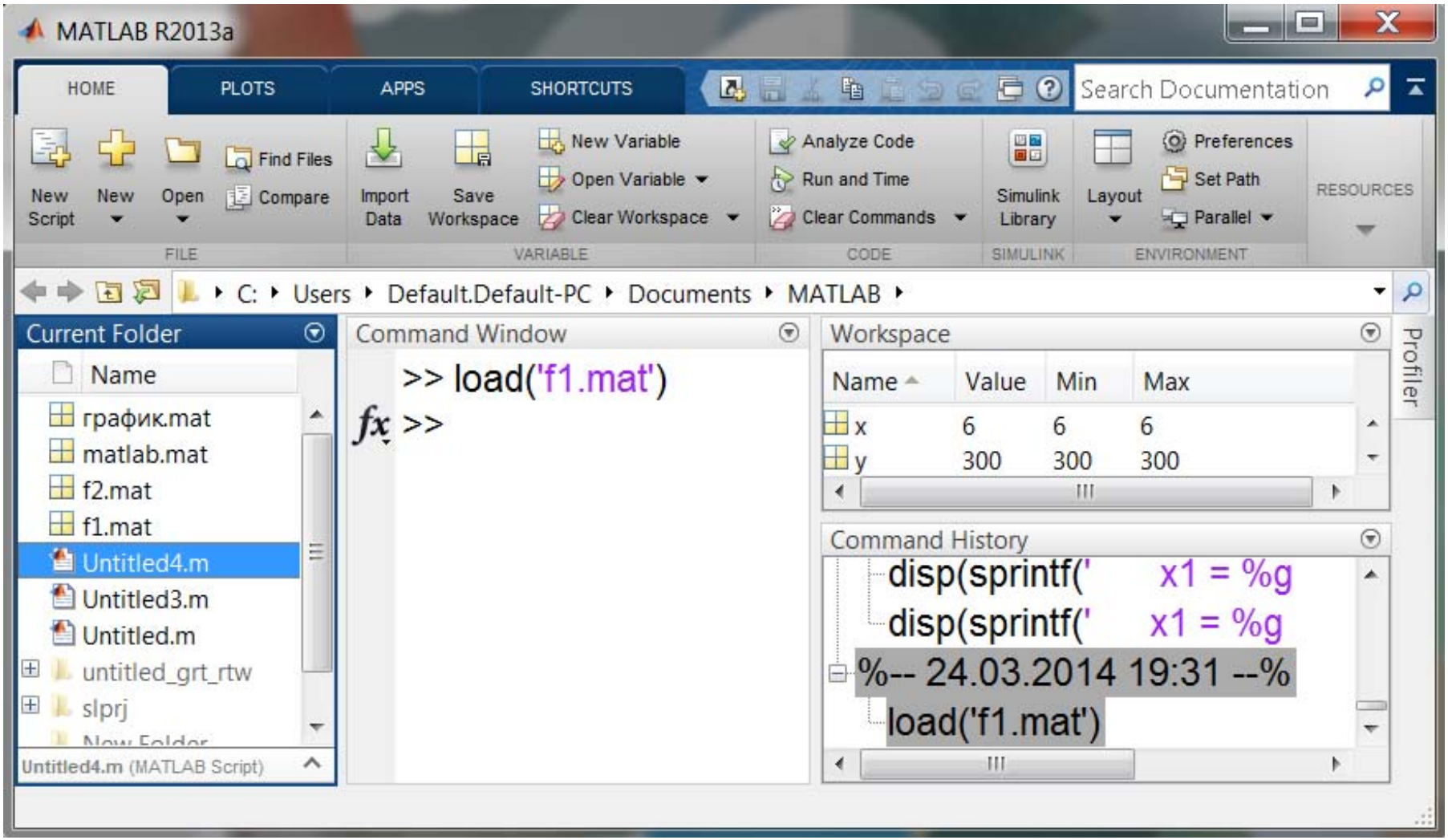

MATLAB R2013a
HOME
PLOTS
APPS
SHORTCUTS
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
RESOURCES
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
C: ▸ Users ▸ Default.Default-PC ▸ Documents ▸ MATLAB ▸
Current Folder
Name
график.mat
matlab.mat
f2.mat
f1.mat
Untitled4.m
Untitled3.m
Untitled.m
untitled_grt_rtw
slprj
Untitled4.m (MATLAB Script)
Command Window
>> load('f1.mat')
>>
Workspace
Name
Value
Min
Max
x
6
6
6
y
300
300
300
Command History
disp(sprintf(' x1 = %g
disp(sprintf(' x1 = %g
%-- 24.03.2014 19:31 --%
load('f1.mat')
Profiler

Рисунок 1.1

2 Основным понятием любой математической системы является *математическое выражение*, которое строится на основе *чисел, констант, переменных, операторов, функций* и различных *спецзнаков*.

Простейший объект системы MATLAB – это *число*, которое может быть целым, дробным с фиксированной и плавающей точкой (вместо запятой при обычной записи), а также комплексным.

Для ввода *действительных чисел* используются общие правила для языков программирования высокого уровня:

– целая часть отделяется от дробной с помощью десятичной точки;

– в показательной форме записи мантисса числа отделяется от его показателя символом e без пробелов.

Введите в командном окне десятичную константу 0,0000123, используя вместо запятой точку для разделения целой и дробной частей, и нажмите клавишу Enter, в результате ее значение присваивается системной переменной ans, значение которой в формате по умолчанию снова выводится в командное окно (рисунок 1.2).

Последнее значение системной переменной ans может быть использовано в последующих операторах вычисления путем указания ее имени. Ниже ответа расположена командная строка с мигающим курсором, позволяющая вводить новые выражения и находить их значения.

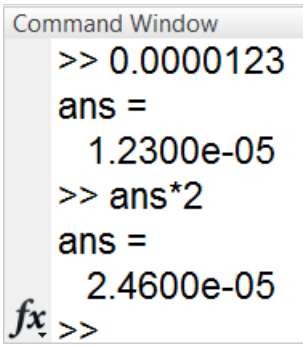


Рисунок 1.2

Введенные значения и результаты всех вычислений сохраняются в памяти компьютера с относительной погрешностью порядка $2 \cdot 10^{-16}$.

3 Требуемый формат вывода задается в окне **Preferences**, которое вызывается с помощью команды **HOME⇒Preferences** (либо **File⇒Preferences**). По умолчанию применяется краткая форма записи в формате с фиксированной точкой **Short** (**default**), при котором на экране отображаются только четыре цифры после десятичной точки. Однако при отображении слишком большого или слишком малого числа, не укладывающегося в формат **Short**, результат выводится в экспоненциальном формате **Short E**.

Отчет

Ознакомиться с содержимым командного окна и включить в отчет по лабораторной работе описание форматов вывода, используемых в MATLAB на примере числа, равного

$$1/(N + 50),$$

где N – номер варианта по указанию преподавателя.

4 В MATLAB могут использоваться *комплексные числа*, содержащие вещественную и мнимую части. Мнимая часть имеет множитель *i* или *j*. При работе с комплексным числом *Z* используются следующие функции:

– **real(*Z*)** и **imag(*Z*)** возвращают соответственно действительную и мнимую части;

– **abs(*Z*)** и **angle(*Z*)** возвращают соответственно модуль и фазу комплексного числа.

Ввести в командном окне комплексное число

$$Z = N + i(N + 10),$$

Отчет

где N – номер варианта. Включить в отчет по лабораторной работе программу, вычисляющую действительную и мнимую части числа *Z*, а также модуль и фазу числа *Z*.

5 Для записи промежуточных результатов в памяти компьютера в MATLAB можно применять *переменные*. Имя переменной может содержать до 30 латинских символов и должно начинаться с буквы. MATLAB различает регистр в именах переменных. Кроме того, имя переменной не должно совпадать с именем функций, процедур и системных переменных MATLAB.

В таблице 1.1 приводится список системных переменных MATLAB. При необходимости системные переменные могут переопределяться, однако в отличие от простых переменных они никогда не могут быть неопределенными. Их значения по умолчанию задаются сразу после загрузки системы.

Отчет

Определить наименьшее и наибольшее числа, с которыми может работать MATLAB. Включить полученные значения в отчет по лабораторной работе.

Таблица 1.1

Имя	Значение
i (j)	Мнимая единица (корень квадратный из –1)
pi	Число π = 3,1415926
eps	Погрешность операций над числами с плавающей точкой (2^{-52})
realmin	Наименьшее число с плавающей точкой
realmax	Наибольшее число с плавающей точкой
inf	Значение машинной бесконечности
NaN	Указание неопределенного результата (например, 0/0, ∞/∞)

6 Введите в командном окне строку x = 2*3 и нажмите клавишу Enter, в результате значение переменной x выведется в командном окне (рисунок 1.3). Если командную строку завершить символом «точка с запятой», то после нажатия клавиши Enter значение переменной не выводится в командное окно (на рисунке 1.3 переменная y). После точки с запятой в той же строке можно разместить и следующую команду MATLAB.

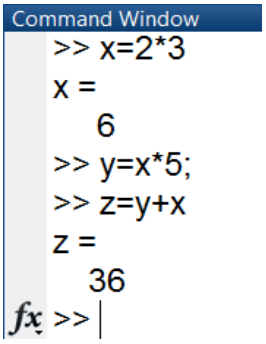


Рисунок 1.3

Если вводимое математическое выражение больше длины одной строки, то его можно перенести на следующую строку с помощью трех и более символов «точка».

7 Любые символы, стоящие после символа **%**, являются *комментариями*. Комментарии являются неисполняемыми операторами. Они делают программу более читаемой.

Отчет

Ввести в командной строке *текстовый комментарий* «Работа выполнена ФИО» (указать свою фамилию, имя и отчество).

8 При работе с MATLAB в командном режиме действует простейший строчный редактор. При этом MATLAB не позволяет редактировать ранее введенную команду после ее завершения путем простой установки курсора в нужную строку.

Однако можно выделить введенную ранее команду и либо перетащить ее в командную строку, либо нажать правую клавишу мыши и, выбрав из контекстного меню команду **Copy**, скопировать ее в буфер обмена. Затем перевести курсор в командную строку и с помощью команды **Paste** в контекстном меню скопировать в нее содержимое буфера обмена, при необходимости отредактировать команду и нажать клавишу Enter для ее выполнения.

В MATLAB все введенные команды загружаются в стек: клавиша ↑ позволяет последовательно вызывать предыдущие команды в командную строку, а клавиша ↓ позволяет вызывать команды в обратной последовательности, кроме того, при необходимости отредактировать в MATLAB одну или несколько введенных ранее команд, в том числе во время других сессий, их можно выделить в окне **Command History** и переместить в командное окно.

Для редактирования нескольких введенных ранее команд удобно выделить их с помощью клавиш Shift и ↓ в окне **Command History** и, нажав правую клавишу мыши, выбрать в контекстном меню

команду **Create Script**. В результате откроется окно редактирования **Editor**, в котором можно отредактировать программный код, выделить его мышью и переместить в командное окно (рисунок 1.4).

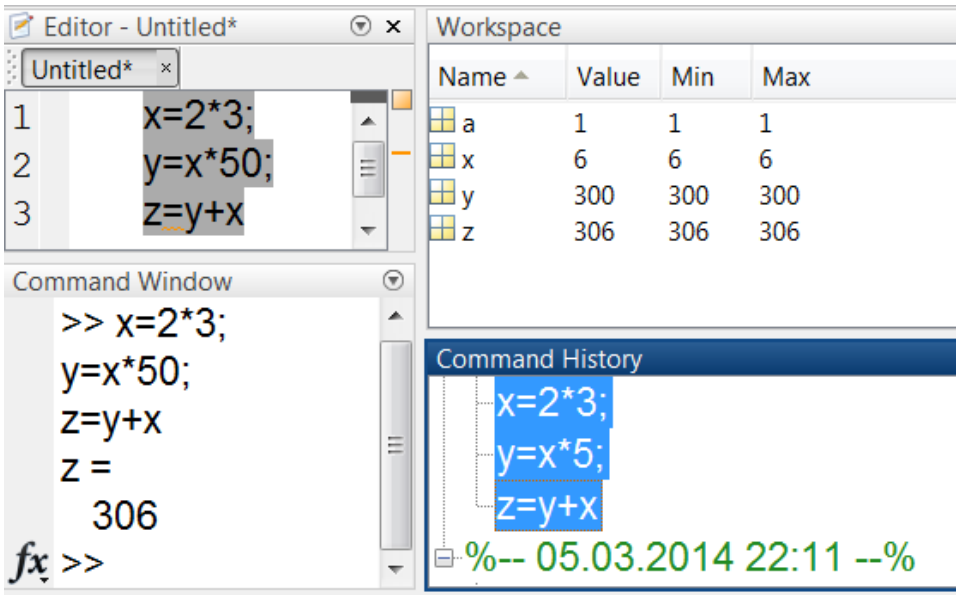


Рисунок 1.4

При необходимости можно очистить окно команд и истории с помощью команды **Home⇒Clear Commands** (либо выбрать требуемую команду в меню **Edit**).

Отчет

Изменить во второй строке окна редактирования множитель 5 на 50 (см. рисунок 1.4), вставить три строки из окна редактирования в окно команд и нажать клавишу Enter. Обратить внимание, как изменятся значения переменных y и z в окне команд. Включить в отчет по лабораторной работе отредактированные строки и полученные результаты их выполнения.

9 Значения переменных можно посмотреть в окне **Workspace** в виде таблицы. Двойной щелчок по строке, соответствующей каждой переменной, приводит к отображению ее содержимого в отдельном окне в виде электронной таблицы. Это удобно при работе с массивами. При необходимости лишние переменные можно удалить, например, с помощью контекстного меню.

10 Сохраните значения всех переменных в файле f1.mat, вызвав команду **HOME⇒Save Workspace** (либо **File⇒Save to Mat-File**). Укажите в появившемся окне путь доступа и имя файла. Расширение файла .mat будет присвоено ему по умолчанию.

11 Закончите сеанс работы в MATLAB, закрыв его окно.

12 Начните следующий сеанс работы в MATLAB и с помощью команды **HOME⇒Import Date** (либо с помощью команды **Load**) восстановите значения переменных, использующихся в предыдущем сеансе. Сохранение и восстановление переменных рабочей среды можно выполнить и в командном окне с помощью команд **save** и **load** соответственно.

Очистить рабочую область можно с помощью меню

HOME⇒Clear Workspace.

13 Система MATLAB предназначена для выполнения операций с векторами, матрицами и полиномами. Даже обычные числа и переменные рассматриваются в ней как матрицы размером 1 × 1.

На рисунке 1.5 приведен пример программы, в которой сначала поэлементно вводятся значения элементов для вектора-строки **v1** и **v2**, а затем из них с помощью операции объединения формируется вектор-строка **v = [v1 + v2]**, значения которой выводятся в командном окне при нажатии клавиши Enter.

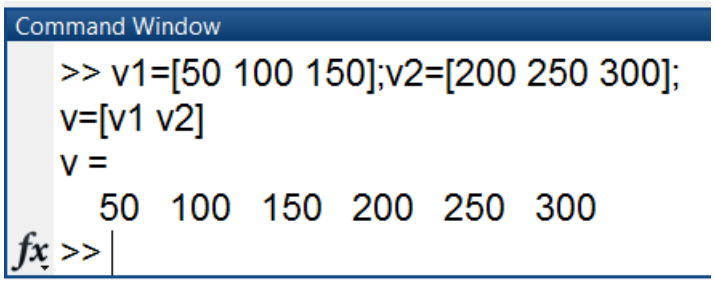


Рисунок 1.5

На рисунке 1.6 приведен пример программы, в которой формируется матрица **A** размером 2 × 3. Обратите внимание, что значения элементов строк матрицы **A** разделены знаком «**;**». Полученная матрица выводится в командном окне при нажатии клавиши Enter.

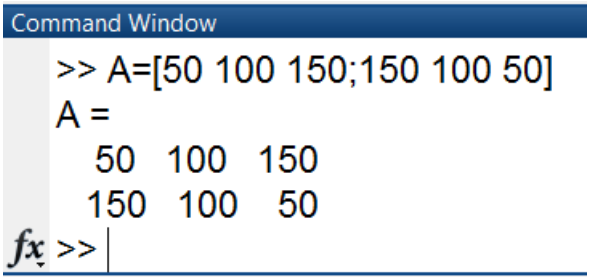


Рисунок 1.6

Отчет

Сформировать два вектора **v1** и **v2**, а также матрицу **A** размером 2 × 3, значения элементов которых в зависимости от номера варианта *N* равны:

$$\mathbf{v1} = [N\ 2N\ 3N],\ \mathbf{v2} = [4N\ 5N\ 6N],\ \mathbf{A} = \begin{bmatrix} N & 2N & 3N \\ 3N & 2N & N \end{bmatrix}.$$

Сформировать из векторов **v1** и **v2** с помощью операции объединения вектор-строку **v** = [**v1** + **v2**]. Включить в отчет по лабораторной работе полученное содержимое рабочего окна.

14 Вычисления математических выражений, содержащих операторы и функции, составляют главную цель любой системы, предназначенной для численных расчетов. *Оператор* представляет собой специальное обозначение для определенной операции над данными (операндами). Полный список операторов MATLAB выводится командой **help ops**. *Функции* – это объекты с уникальными именами, которые выполняют определенные преобразования своих аргументов и при этом обязательно *возвращают* результаты на место вызова. Функции бывают *встроенными* и *внешними*. В данной лабораторной работе рассмотрим встроенные функции.

В таблице 1.2 приведен список арифметических операторов MATLAB, соответствующих им функций.

Таблица 1.2

Знак операции	Функция	Название
+	**Plus**	Сложение
–	**Minus**	Вычитание
*	**Mtimes**	Матричное умножение
.*	**Times**	Поэлементное умножение
/	**Mrdivide**	Матричное деление слева направо
./	**Rdivide**	Поэлементное деление слева направо
\	**Mlrdivide**	Матричное деление справа налево
.\	**Ldivide**	Поэлементное деление справа налево
^	**Mpower**	Возведение матрицы в степень
.^	**Power**	Поэлементное возведение массива в степень

Так же как и в математических выражениях, операторы MATLAB имеют определенный *приоритет исполнения.* Для изменения приоритета операций должны использоваться круглые скобки.

Вычислите в режиме калькулятора значение числового выражения

$$\left\{\frac{8{,}8077}{20-\left[28{,}2:\left(13{,}333\cdot 0{,}3+0{,}0001\right)\right]\cdot 2{,}004}+4{,}9\right\}\cdot\frac{N}{32},$$

где N – номер варианта по указанию преподавателя.

Отчет

Включить в отчет по лабораторной работе полученное выражение и результат вычисления.

15 В таблице 1.3 приведен список операторов и функций отношения, используемых в MATLAB. При этом операторы <, <=, > и >= при комплексных операндах используются для сравнения только действительных частей операндов, а мнимые части операндов отбрасываются.

Таблица 1.3

Оператор	Функция	Название
= =	**Eq**	Равно
~ =	**Ne**	Не равно
<	**Lt**	Меньше чем
>	**Gt**	Больше чем
<=	**Le**	Меньше или равно
>=	**Ge**	Больше или равно

Отчет

Включить в отчет по лабораторной работе примеры, приведенные на рисунке 1.7, а также примеры применения других операторов и функций отношения для любых своих вещественных и комплексных операндов.

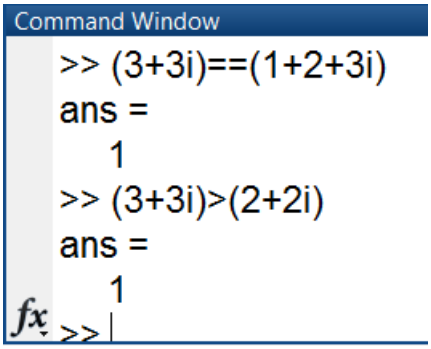


Рисунок 1.7

16 В таблице 1.4 приведены логические операторы и соответствующие им функции MATLAB, которые служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов.

Таблица 1.4

Оператор	Функция	Название операции
&	**And**	Логическое «И» (AND)
|	**Or**	Логическое «ИЛИ» (OR)
~	**Not**	Логическое «НЕ» (NOT)

На рисунке 1.8 приведены примеры применения логической операции «И» для ранее определенных векторов **v1** и **v2**.

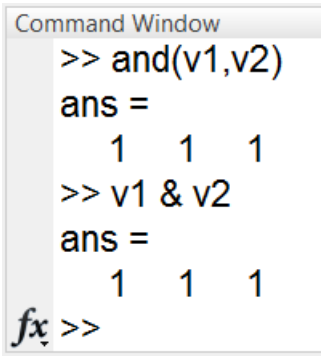


Рисунок 1.8

Если оба операнда операции «И» истинные (не равны 0), то результат этой операции равен 1 («истина»), во всех остальных случаях результат равен 0 («ложь»).

Операция «ИЛИ» возвращает 0 только тогда, когда оба операнда равны нулю.

Операция «НЕ» инвертирует «ложь» на «истину».

Отчет

Проверить работу логических операторов для двух векторов **v1** и **v2**, определенных ранее. Включить в отчет по лабораторной работе полученные выражения.

17 В таблице 1.5 приведены некоторые стандартные функции вещественного аргумента MATLAB.

Таблица 1.5

Функция	Название
1	2
a^x	Степенная функция
x^a	Показательная функция
sqrt(x)	Квадратный корень
exp(x)	Экспонента
log(x)	Натуральный логарифм

Продолжение таблицы 1.5

1	2
log10(x)	Десятичный логарифм
abs(x)	Модуль
fix(x)	Отбрасывание дробной части числа
round(x)	Обычное округление
mod(x,y)	Остаток от деления x на y с учетом знака
sign(x)	Знак числа
factor(x)	Разложение числа x на простые множители
sin(x)	Синус
asin(x)	Арксинус
cos(x)	Косинус
acos(x)	Арккосинус
tan(x)	Тангенс
atan(x)	Арктангенс

Отчет

Даны $x = 1{,}5$; $y = 2$; $z = 3$. Вычислить a, b из таблицы 1.6 для варианта, указанного преподавателем. Включить в отчет по лабораторной работе полученные результаты.

Таблица 1.6

Номер варианта	a	b
1	2	3
1	$a = \dfrac{z + y/(x^2+4)}{e^{-x-2}/(x^2+4)}$	$b = \dfrac{x}{y}(\operatorname{arctg} z + 1/6)$
2	$a = \dfrac{3{,}5 + \mathrm{e}^{y-1}}{1 + x^2\lvert y - \operatorname{tg} z\rvert}$	$b = \dfrac{(y-x)^2}{2} + \dfrac{\lvert y-x\rvert^3}{3}$
3	$a = \dfrac{\sqrt{\lvert x-1\rvert} - \sqrt[3]{\lvert z\rvert}}{1 + \dfrac{x^2}{2{,}5} + \dfrac{y^2}{4}}$	$b = \dfrac{1+\cos(y-2)}{\dfrac{x^2}{2} + \sin^2 z}$
4	$a = z + \dfrac{x}{y^2 + \left\lvert \dfrac{x^2}{y + x^3/1{,}3} \right\rvert}$	$b = 1 + \operatorname{tg}^3\left(\dfrac{z}{2x+2y}\right)$
5	$a = \dfrac{2{,}3\cos(x - 1/6)}{1/2 + \sin^2 y} + z$	$b = x^y + \dfrac{z^2}{3 + z^2/5}$
6	$a = \dfrac{1{,}5 + \sin^2 z}{\lvert x - 2x/(1 + x^2y^2)\rvert}$	$b = \cos^2\left(\operatorname{arctg}\dfrac{y}{z}\right) + \sin^2 x$

Продолжение таблицы 1.6

1	2	3
7	$a = \ln\left\|(y)\dfrac{1{,}5y}{z + x^2/4}\right\|$	$b = x - \dfrac{x^2}{3!} + \dfrac{x^5}{5!}$
8	$a = \dfrac{\sin\left(x^2 - 2y + z\right)}{2{,}6x^y}$	$b = \cos^2\left(x^2 + \dfrac{y}{z}\right)$
9	$a = \dfrac{5\cos(x - 1/6)}{0{,}5 + \sin^2 x}$	$b = \dfrac{y^2}{3 + z^2/7} - 3x$
10	$a = \dfrac{3{,}5 + \operatorname{tg}(x^2 + y)}{\left\|x - 4x/(1 + xy^2)\right\|}$	$b = \sin^2\left(\operatorname{arctg}\dfrac{1}{z}\right)$
11	$a = \dfrac{2{,}6 + \operatorname{tg}(x - y)}{\left\|x - 2x/(x^2 + y^2)\right\|}$	$b = x(\operatorname{tg} z + \mathrm{e}^{x-3})$
12	$a = \ln\left\|\dfrac{y + x^2/4}{5z}\right\|$	$b = 1{,}5 + \dfrac{(y - x)^3}{2} + \dfrac{\left\|y - x\right\|}{x}$
13	$a = \dfrac{\cos\left(x^3 + 2y - 2z\right)}{\operatorname{tg} y - 1{,}5}$	$b = \dfrac{1 + \sin(y - 2)}{x^2 \dfrac{1}{2 + \sin^2 x}}$
14	$a = \dfrac{5\sin(x + 1/3)}{1/2\cos x + 1}$	$b = \left(1 + \operatorname{tg}^2 \dfrac{z}{y + 2}\right)$
15	$a = \dfrac{3 + \sin^3(x^2 + y)}{2{,}5 + \left\|x - 4x/(1 + x^2 y^2\right\|}$	$b = 1 + \dfrac{z^2}{3 + z^2/5}$
16	$a = \dfrac{\sqrt{\left\|x - 1\right\|} - \sqrt[3]{\left\|y\right\|}}{1{,}5 + x^2 + y^2}$	$b = \cos^2\left(\operatorname{arctg}\dfrac{1}{z + 1}\right)$
17	$a = \dfrac{1{,}5 - \mathrm{e}^{2-y}}{2x - \sqrt{y - \operatorname{tg} z}}$	$b = x + \dfrac{x^2}{3 + z} + \dfrac{x^2}{5 + z}$
18	$a = \dfrac{3{,}3 + y^2 + (x^2 + 2)}{\mathrm{e}^{-0{,}5} + 1/(x^2 + 4)}$	$b = \cos^2\left(x^2 + \dfrac{y}{1 + z}\right)$
19	$a = y + \dfrac{3{,}5x}{y^2 - \sqrt{\dfrac{x^2}{2y + x^2}}}$	$b = 2 + \dfrac{y^2}{3 + \dfrac{z^2}{1 + x}} - 3x$
20	$a = \dfrac{2\cos(x - 1/6)}{1/2 + \sin^2 y}$	$b = \sqrt{\sin\left(\operatorname{arctg}\dfrac{1}{x + z}\right)}$

Контрольные вопросы

1 Перечислите основные окна в MATLAB и объясните их назначение.

2 Каким образом формируется очередная команда в MATLAB?

3 Как вызвать предыдущую команду в MATLAB?

4 Каким образом можно редактировать программы в MATLAB?

5 Чем определяются форматы представления чисел при выводе результатов вычислений в MATLAB?

6 Какие системные переменные MATLAB вы знаете?

7 По каким правилам формируются имена переменных в MATLAB?

8 Каким образом сохраняются значения переменных в файле и как можно восстановить значения переменных, используемых в предыдущих сеансах?

9 Как вводится комментарий в MATLAB?

10 Какие операции и встроенные функции применяются в MATLAB?

Лабораторная работа № 2
Матричные операции в среде MATLAB

Цель работы: знакомство с основными операциями над векторами и матрицами в MATLAB, в том числе позволяющими решать системы линейных уравнений.

Рабочее задание

1 Присваивать значения элементам вектора в MATLAB можно поэлементно с клавиатуры, указывая их внутри квадратных скобок через пробел или запятую, в конце строк необходимо указать знак точка с запятой. Значения элементов матриц могут задаваться в виде арифметических выражений с функциями MATLAB либо в комплексной форме (рисунки 2.1, 2.2).

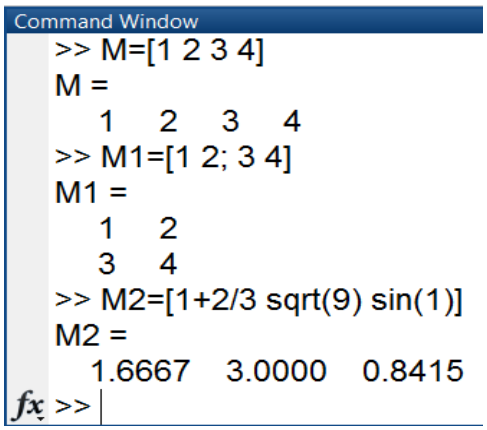


Рисунок 2.1

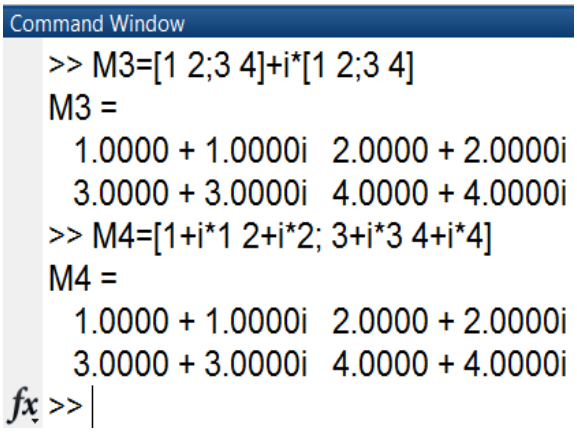


Рисунок 2.2

Если для отображения матрицы размера командного окна недостаточно, то она будет выведена по столбцам (рисунок 2.3, матрица **M4**).

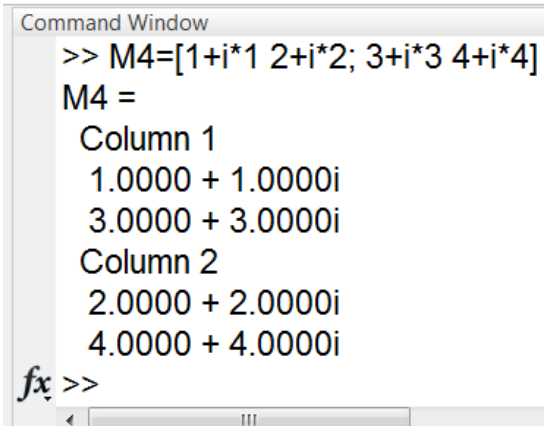


Рисунок 2.3

Отчет

Ввести в командном окне значения элементов матриц **M**, **M1**, **M2**, **M3**, **M4**, указанные на рисунках 2.1, 2.2. Включить в отчет по лабораторной работе содержимое рабочего окна.

2 Для формирования упорядоченных числовых последовательностей в MATLAB используется оператор «**:**» (двоеточие) в следующем формате:

Начальное _значение : Шаг : Конечное значение

При этом шаг может быть как положительным, так и отрицательным. Если шаг не указан, то по умолчанию значение шага принимается равным единице.

Отчет

Создать вектор, значения элементов которого представляют собой арифметическую прогрессию вида

$$N\,(N-1)\,(N-2)\,\ldots\,0,$$

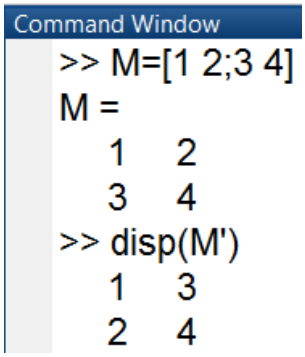
где N – номер варианта по указанию преподавателя. Включить в отчет по лабораторной работе содержимое рабочего окна.

3 Для формирования векторов и матриц определенного вида в MATLAB можно использовать следующие функции, приведенные в таблице 2.1.

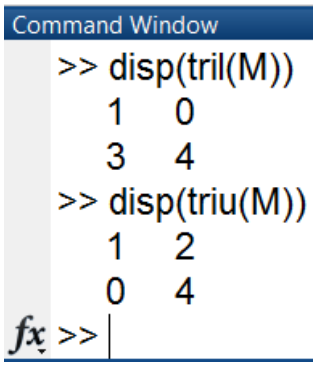
Таблица 2.1

Функция	Описание
'	Транспонирование
tril(M)	Создание нижней треугольной матрицы **M**
triu(M)	Создание верхней треугольной матрицы **M**
fliplr(M)	Вращение матрицы **M** относительно вертикальной оси
flipud(M)	Вращение матрицы **M** относительно горизонтальной оси
rot90(M).	Поворот матрицы **M** на 90° против часовой стрелки
rot90(M,-1)	Поворот матрицы **M** на 90° по часовой стрелке
diag(diag(M))	Создание диагональной матрицы по матрице **M**
diag(P)	Создание диагональной матрицы из массива **P**
rand(i,j)	Задание матрицы размерностью **i** × **j** по случайному равномерному закону
randn(i,j)	Задание матрицы размерностью **i** × **j** по случайному нормальному закону

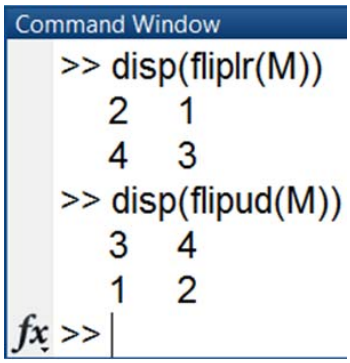
На рисунках 2.4,*а–е* приведены примеры применения некоторых функций из таблицы 2.1 для матрицы **M** и вектора-строки **P**. Обратите внимание на команду **disp**, которая осуществляет вывод значений указанной переменной или текста в командное окно без указания имени переменной.



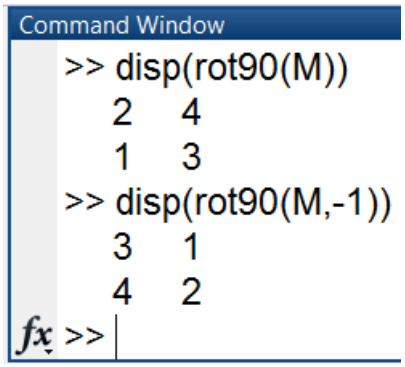
а)



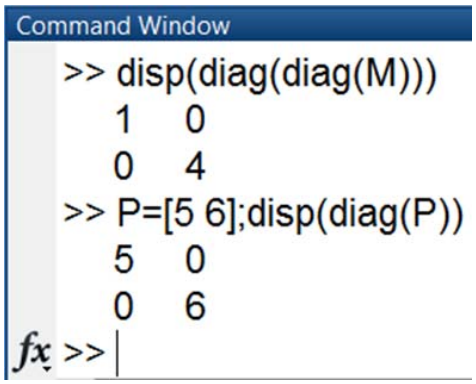
б)



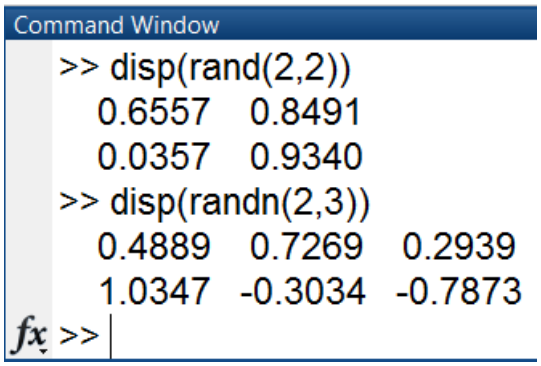
в)



г)



д)



е)

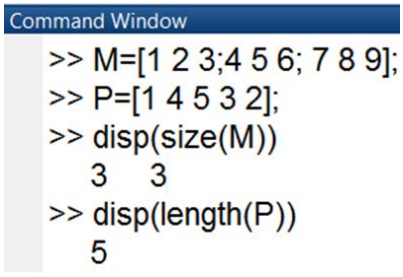
Рисунок 2.4

4 В таблице 2.2 приведены некоторые функции MATLAB, которые можно использовать для обработки векторов и матриц и получения информации о них.

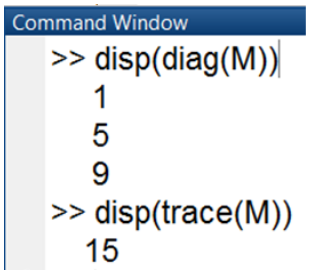
Таблица 2.2

Функция	Описание
size(M)	Размерность матрицы **M**
length(P)	Длина вектора **P**
diag(M)	Извлечение диагонали заданной матрицы **M**
trace(M)	Вычисление следа матрицы **M** (сумма элементов главной диагонали)
sum(M)	Суммирование элементов столбцов матрицы **M**
sum(P)	Суммирование элементов вектора **P**
prod(M)	Формирование произведения элементов столбцов матрицы **M**
prod(P)	Формирование произведения элементов вектора **P**
max(M)	Нахождение максимального значения матрицы **M** по столбцам
max(P)	Нахождение максимального значения вектора **P**
min(M)	Нахождение минимального значения матрицы **M** по столбцам
min(P)	Нахождение минимального значения вектора **P**
det(M)	Вычисление главного определителя матрицы **M**
sort(M)	Сортировка значений матрицы **M** по столбцам
sort(P)	Сортировка значений вектора **P**

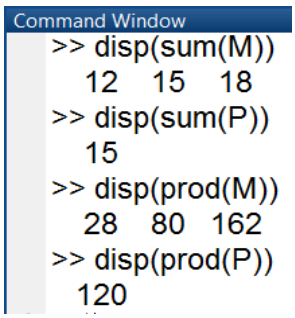
На рисунках 2.5,*а–д* приведены примеры применения функций из таблицы 2.2 для матрицы **M**, вектора-строки **P**.



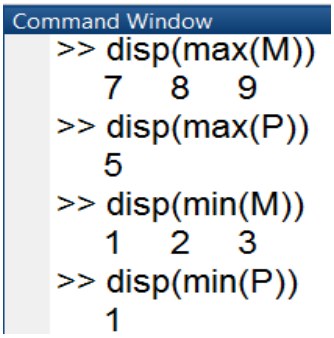
а)



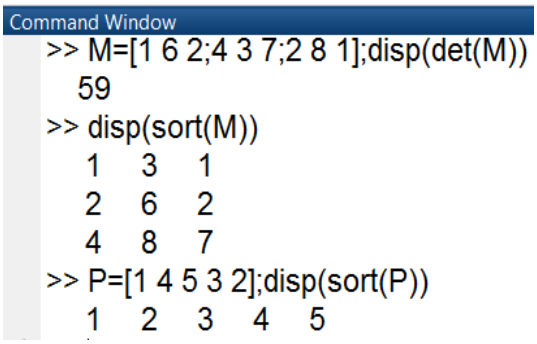
б)



в)



г)



д)

Рисунок 2.5

5 Создайте вектор-строку **P**, значения элементов которой равны N, $(N+10)$, $(N-20)$, $(N-30)$, $(N+4)$, а также матрицу **M1**, значения элементов которой равны

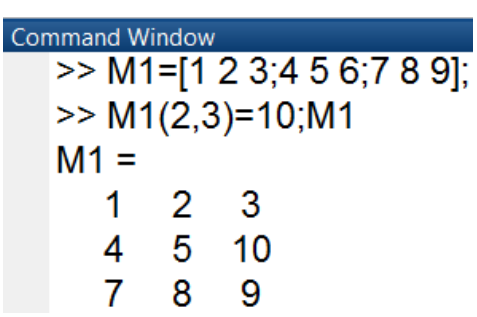
$$\mathbf{M1} = \begin{bmatrix} N-2 & N+5 & N-3 \\ N-20 & N+10 & N-1 \\ N+2 & N-12 & N-5 \end{bmatrix},$$

где N – номер варианта по указанию преподавателя.

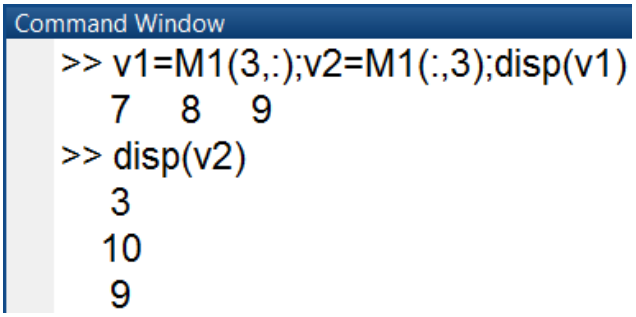
Отчет

Ввести в командном окне примеры, приведенные на рисунках 2.4 и 2.5 для созданных массивов **P** и **M1**. Включить в отчет по лабораторной работе содержимое командного окна.

6 В системе MATLAB для обращения к любому элементу заданной матрицы **M1** необходимо указать в скобках после имени матрицы номера соответствующих ему строки и столбца через запятую. На рисунке 2.6,*а* приведен пример обращения к элементу матрицы **M1** размером 3 × 3, расположенному на пересечении второй строки и третьего столбца, которому присваивается значение, равное 10.



а)



б)

Рисунок 2.6

Операция обращения к конкретным строкам или столбцам матрицы выполняется в MATLAB с использованием символа двоеточие «:». На рисунке 2.6,*б* приведены примеры обращения к элементам третьей строки и третьего столбца матрицы **M1** и присвоения их значений векторам **v1** и **v2** соответственно.

Отчет

Сформировать в командном окне MATLAB матрицу **M2** размерностью 5 × 5 с помощью функции **randn**. Присвоить в зависимости от номера варианта (таблица 2.3) элементу матрицы, находящемуся на пересечении *n*-й строки и *m*-го столбца, значение, равное номеру варианта *N*; найти для матрицы **M2** минимальное значение для *n*-й строки и максимальное значение для *m*-го столбца; выполнить для четных вариантов сортировку *n*-й строки, а для нечетных вариантов – сортировку *m*-го столбца; найти главный определитель для матрицы **M2**. Включить в отчет по лабораторной работе содержимое командного окна.

Таблица 2.3

№ строки (*n*)	№ варианта (*N*)				
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15
4	16	17	18	19	20
5	21	22	23	24	25
	1	2	3	4	5
	№ столбца (*m*)				

7 MATLAB позволяет вставить меньшую матрицу так, чтобы она стала определенной частью матрицы большего размера и, наоборот, создать из большей матрицы вектор или другую матрицу меньшего размера. На рисунках 2.7,*а*,*б* приведены соответствующие примеры.

```
Command Window
>> B=[10 11;12 13]
B =
   10   11
   12   13
>> M1=[1 2 3;4 5 6;7 8 9]
M1 =
    1    2    3
    4    5    6
    7    8    9
```

а)

```
Command Window
>> M1(1:2,1:2)=B;disp(M1)
   10   11    3
   12   13    6
    7    8    9
>> C=M1(1:3,1:2)
C =
   10   11
   12   13
    7    8
```

б)

Рисунок 2.7

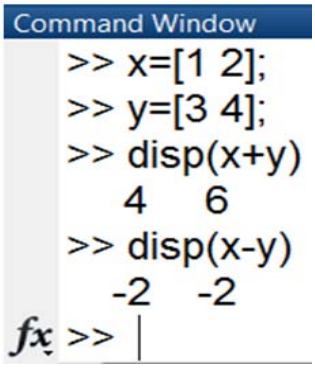
Отчет

Вставить в центр созданной в п. 6 матрицы **M2** матрицу **M1**, определенную в п. 5. Включить в отчет по лабораторной работе содержимое командного окна.

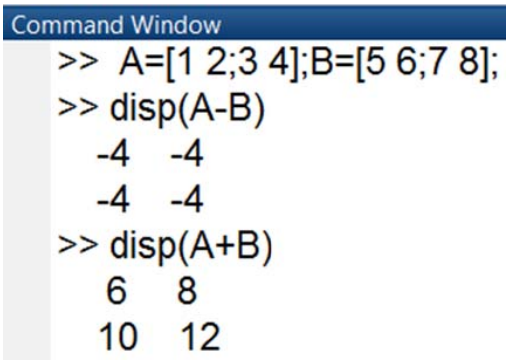
8 В MATLAB могут выполняться как традиционные действия над векторами и матрицами, предусмотренные векторным вычислением в математике, так и их поэлементные преобразования.

8.1 Для векторного *сложения* или *вычитания* векторов и матриц с помощью обычных знаков арифметических действий оба вектора должны быть либо вектором-столбцом, либо вектором-строкой и иметь одинаковую размерность, а обе матрицы должны иметь одинаковую размерность.

На рисунке 2.8,*а*,*б* приведены соответствующие примеры.



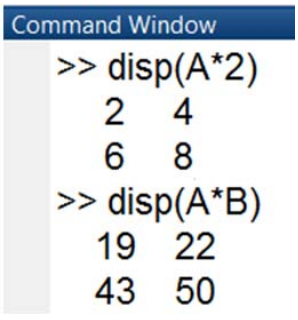
а)



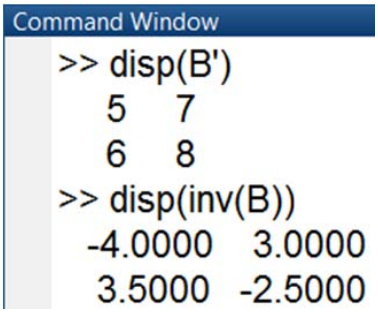
б)

Рисунок 2.8

Для векторного *умножения* векторов и матриц количество столбцов первого вектора или матрицы должно совпадать с количеством строк второго вектора или матрицы. На рисунке 2.9,*а*,*б* приведены соответствующие примеры, а также пример транспонирования и обращения матрицы **B**.



а)



б)

Рисунок 2.9

Обращение матрицы используется для решения системы линейных уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \qquad\qquad \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

при задании ее в матричной форме как

$$\mathbf{AX} = \mathbf{B}, \qquad (2.1)$$

где $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$; $\mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix}$; $\mathbf{A} = \left[a_{ij} \right]$; $i, j = \overline{1,n}$.

Система (2.1) имеет единственное решение, равное

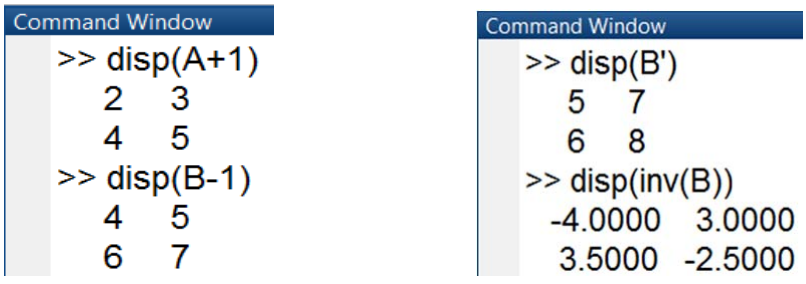
$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B},$$

если ее главный определитель отличен от нуля

$$\det(\mathbf{A}) \neq 0.$$

Получить обратную матрицу можно с помощью функции **inv(A)**, либо с помощью оператора **A**^–1, что означает **A** в степени –1.

8.2 Для поэлементного преобразования векторов и матриц в MATLAB можно использовать все арифметические операции и алгебраические функции, приведенные в лабораторной работе № 1 таблицах 1.2 и 1.5 соответственно). Они формируют матрицу того же размера, что и исходная матрица, у которой каждый элемент вычисляется как значение указанной функции от соответствующего элемента заданной матрицы. На рисунке 2.10 приведены примеры поэлементного преобразования матриц. Обратите внимание, что знак операции «поэлементное сложение и вычитание» не содержит точки.



а) *б*)

Рисунок 2.10

Создайте два вектора-строки **x1** = (*N*, *N* + 1) и **y1** = (*N*, *N* + 2), а также матрицу **A1** и **B1**, значения элементов которых равны

$$\mathbf{A1} = \begin{bmatrix} N-2 & N+5 & N-3 \\ N-20 & N+10 & N-1 \\ N+2 & N-12 & N-5 \end{bmatrix}; \quad \mathbf{B1} = \begin{bmatrix} 1 & 4 & 3 \\ 2 & 5 & 2 \\ 3 & 6 & 1 \end{bmatrix}.$$

Отчет

Ввести в командном окне примеры, приведенные на рисунках 2.8–2.10 для массивов **x1**, **y1**, **A1** и **B1**. Включить в отчет по лабораторной работе содержимое рабочего окна.

Контрольные вопросы

1 Каким образом в MATLAB могут задаваться значения элементов матриц?

2 Каким образом в MATLAB формируются упорядоченные матрицы?

3 Какие функции можно использовать для формирования векторов и матриц в MATLAB?

4 Какие функции MATLAB можно использовать для обработки векторов и матриц и получения информации о них?

5 Какие способы обращения к элементам, строкам и столбцам заданной матрицы существуют в системе MATLAB?

6 Как в MATLAB вставить меньшую матрицу так, чтобы она стала определенной частью матрицы большего размера и, наоборот, создать из большей матрицы вектор или другую матрицу меньшего размера?

7 Как в MATLAB могут выполняться как традиционные действия над векторами и матрицами, предусмотренные векторным вычислением в математике, так и поэлементные преобразования векторов и матриц?

8 Как в MATLAB решается система линейных уравнений, заданная в матричной форме?

Лабораторная работа № 3

Графические построения в среде MATLAB

Цель работы: знакомство с простейшими графическими средствами MATLAB.

Рабочее задание

1 Для построения в системе MATLAB графика функции одной переменной y(x) в декартовой (прямоугольной) системе координат используется команда

plot (X1, Y1, S1, X2, Y2, S2, …)

где **X*i***, **Y*i*** – заданные векторы одинакового размера, элементы которых являются координатами точек *i*-й кривой; **S*i*** – необязательные символьные переменные, относящиеся к *i*-й кривой. Любая из **S*i*** может содержать до трех специальных символов, определяющих тип линии, соединяющей отдельные точки графика (таблица 3.1), тип точки графика (таблица 3.2) и цвет линии (таблица 3.3).

Таблица 3.1

Обозначение	Тип линии	Обозначение	Тип линии
–	Сплошная	**– .**	Штрих-пунктир
:	Двойной пунктир	**– –**	Штриховая

Таблица 3.2

Обозначение	Тип точки	Обозначение	Тип точки
.	Точка	**V**	Треугольник (вниз)
0	Окружность	**A**	Треугольник (вверх)
X	Крест	**<**	Треугольник (влево)
+	Плюс	**>**	Треугольник (вправо)
*****	Звездочка	**P**	Пятиугольник
S	Квадрат	**H**	Шестиугольник
D	Ромб		

Таблица 3.3

Обозначение	Цвет линии	Обозначение	Цвет линии
Y	Желтый	**G**	Зеленый
M	Фиолетовый	**B**	Синий
C	Голубой	**W**	Белый
R	Красный	**K**	Черный

В случае, если значения переменных **S** не указано, то, по умолчанию, принимаются следующие параметры графика: тип линии графика – сплошная; тип точки графика – пиксель; цвет каждого последующего графика устанавливается в следующем порядке: синий, зеленый, красный, голубой, фиолетовый, желтый, черный.

Рассмотрим пример построения функции $Y = 10\sin(X + \pi/4)$ для значений аргумента, изменяющихся от нуля до 2π с шагом, равным $\pi/100$. Соответствующий программный код приведен на рисунке 3.1.

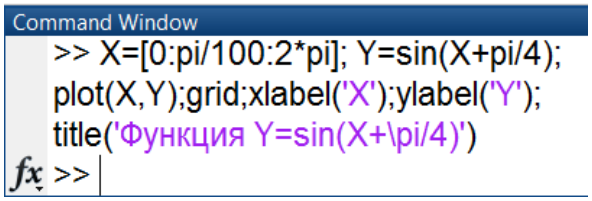


Рисунок 3.1

Сначала сформируем вектор значений аргумента **X** (при этом греческую букву «π» зададим как «pi»), затем вычислим вектор соответствующих значений функции **Y** и, наконец, построим график зависимости $Y(X)$ с помощью функции **plot**.

Функция **grid** позволяет добавить на график сетку из координатных линий, названия осей задаются с помощью функции **xlabel** и **ylabel**, а заголовок графика выводится с помощью процедуры **title**. Обратите внимание на то, что для вывода в заголовке графика греческой буквы «π» перед символами «**pi**» добавлена обратная косая черта «\». Аналогично можно вывести и другие прописные и строчные греческие буквы \alpha, \beta, \gamma, \Gamma, \Sigma, \Pi и т.д.

В результате на экране появится окно, в котором будет построен заданный график (рисунок 3.2).

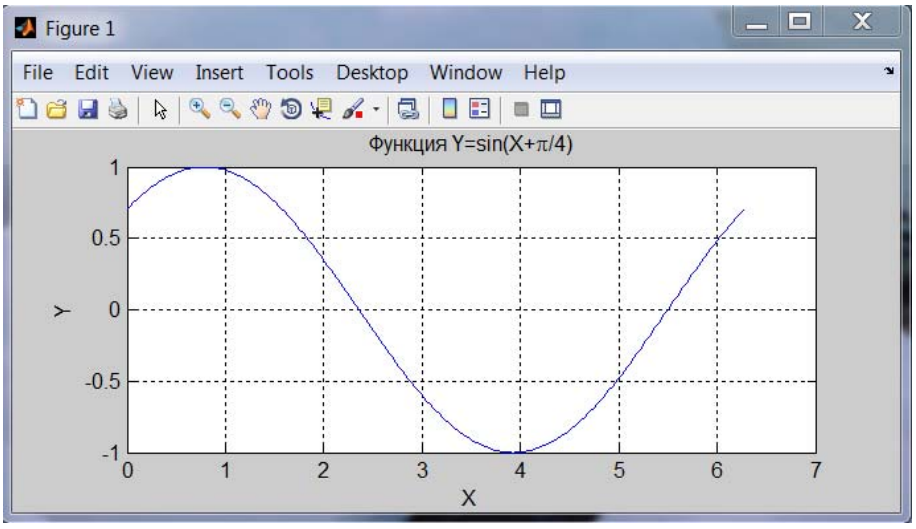


Рисунок 3.2

Для получения на графике гладких кривых необходимо выбрать достаточно большое количество точек на графике при формировании вектора аргумента. Если при обращении к функции **plot** вектор аргумента не указан явно, то MATLAB, по умолчанию, в качестве аргумента принимает номера элементов вектора функции (рисунок 3.3).

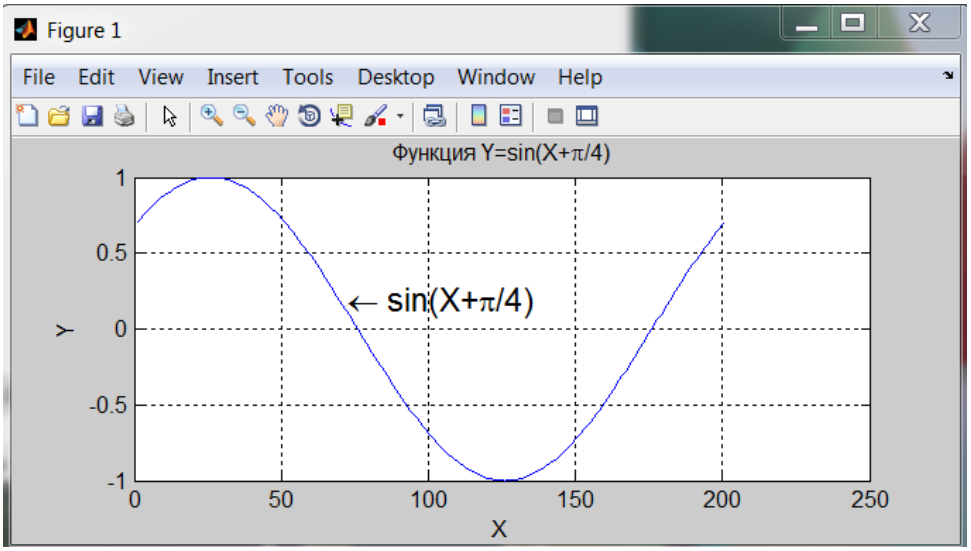


Рисунок 3.3

Для добавления текста в определенное место графика, например для обозначения кривой графика, используется команда

text(*x*,*y*,'string')

где ***x***, ***y*** определяют координаты точки начала текста; **'string'** – строковая константа, определяющая вводимый в график текст.

На рисунке 3.3 приведен пример текста, вставленного на графике в точку с координатами 70; 0,2. Соответствующий программный код приведен на рисунке 3.4.

```
Command Window
>> plot(Y);grid;xlabel('X');ylabel('Y');
title('Функция Y=sin(X+\pi/4)')
text(70,0.2,' \leftarrow sin(X+\pi/4)','FontSize',14)
fx >>
```

Рисунок 3.4

Обратите внимание на то, что строковая константа начинается с обратной косой черты «\» и команды **leftarrow**, которая задает стрелку, направленную влево. Аналогично можно добавить и другие виды стрелок (команды **\uparrow**, **\rightarrow** и т.д.), а также знаки «больше или равно», «меньше или равно», «не равно» (команды **\geq**, **\leq**, **\neq**).

В программе рисунок 3.4 для задания требуемого размера шрифта в пунктах используется строковая константа **'FontSize'**. В MATLAB имеется возможность задавать и требуемый тип с помощью команд **\fontname**{имя шрифта}, а также начертание шрифта с помощью следующих команд:

\bf – для жирного текста;

\it – для курсива;

\rm – для обычного текста.

MATLAB позволяет выводить с помощью команды **text** достаточно сложные математические формулы, включающие в себя индексы, дроби, корни, интегралы, суммы и т.д. Подробнее информацию о функции **text** можно посмотреть с помощью команды **doc text**.

Для задания индексов используется символ «^» перед верхним индексом или символ «_» перед нижним индексом. Так как в программе рисунка 3.5 верхний индекс содержит больше одного символа, то он должен быть полностью заключен в фигурные скобки.

```
Command Window
>> X=[0:0.01:1];Y=X.^(X+1);
plot(Y);grid;xlabel('X');ylabel('Y');
title('Функция Y=X^{(X+1)}')
text(20,0.5,'Y=X','FontSize',18)
text(35,0.55,'^{(X+1)}','FontSize',14)
fx >>
```

Рисунок 3.5

Результаты выполнения данного программного кода приведены на рисунке 3.6.

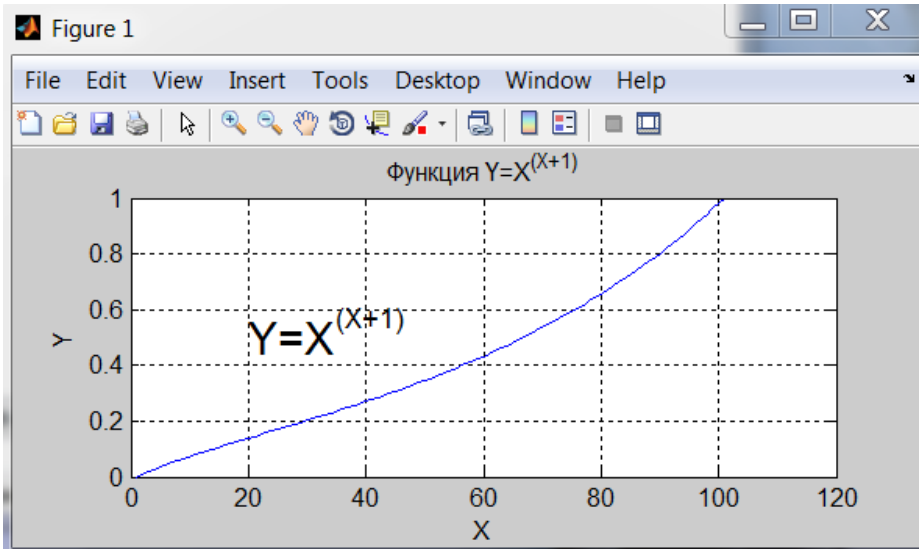


Рисунок 3.6

Координаты точки ввода текста в график удобно задавать с помощью мыши, используя команду **gtext('string')**. При исполнении этой команды на графике появляется большое перекрестие, установив которое в нужное место графика, достаточно нажать любую клавишу или любую кнопку мыши, и на этом месте появится надпись.

Если при неявном указании аргументов команды **plot** вектор функции **Y** содержит комплексные элементы (рисунок 3.7), то при выполнении команды **plot(Y)** в качестве аргументов функции **Y** берется вещественная часть ее элементов, а в качестве значений функции – мнимая часть ее элементов. Во всех других случаях мнимая часть данных игнорируется.

Результаты выполнения данного программного кода (см. рисунок 3.7) приведены на рисунке 3.8.

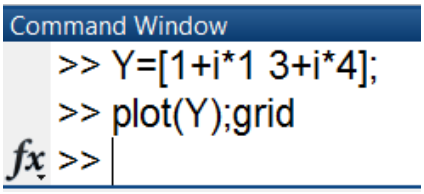


Рисунок 3.7

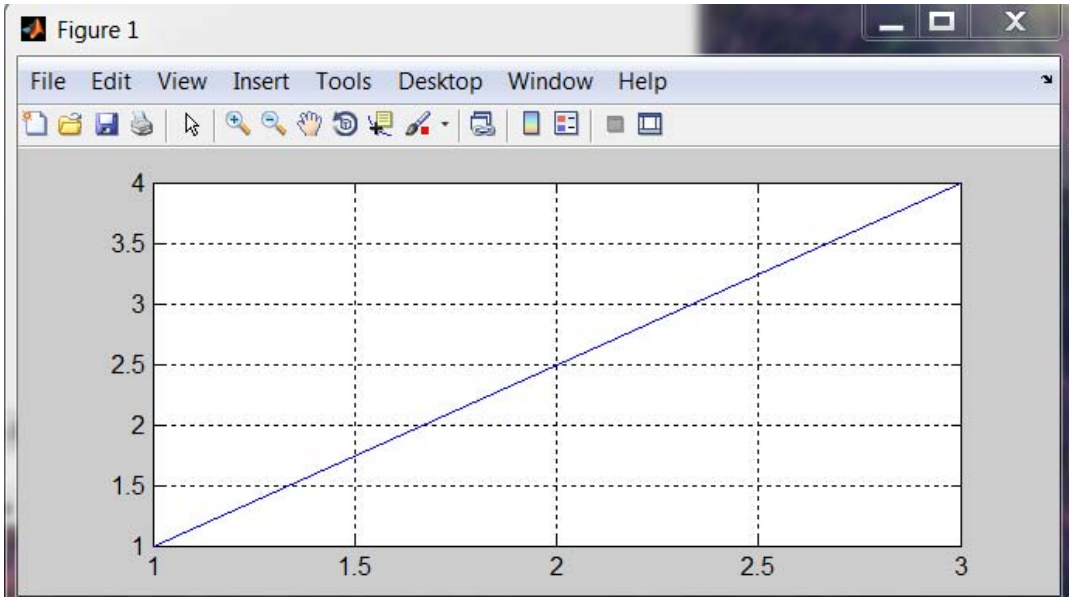


Рисунок 3.8

Отчет

Построить график функции для заданного варианта задания (таблица 3.4). Добавить к графику сетку из координатных линий, названия осей и заголовок. Включить в отчет по лабораторной работе содержимое командного окна и полученный график функции.

Таблица 3.4

Номер варианта	Функция	Диапазон изменения x	Шаг изменения x
1	$Y = -\sin(x/2)$	0°÷360°	10°
2	$Y = 13x^2$	0 ÷ 24	8
3	$Y = \cos(4x - 60°)$	0°÷180°	5°
4	$Y = \frac{1}{x+4}$	–1 ÷ 3	0,5
5	$Y = \frac{x+44}{x-23}$	–8 ÷ 8	2
6	$Y = (3x^2 + 43)$	0 ÷ 40	4
7	$Y = \mathrm{tg}(x) - 1$	0°÷90°	2°
8	$Y = \frac{x}{x-31}$	–10 ÷ 30	5
9	$Y = (x^3 - x)$	0 ÷ 200	20
10	$Y = \sin(4x - 45°)$	0–180°	2°
11	$Y = \sin(3x + 60°)$	0–360°	4°
12	$Y = 1 - e^{5x}$	0–5	0,1
13	$Y = \sin(5x)$	0–360°	4°
14	$Y = \mathrm{tg}(x + 30°)$	0°÷90°	2°
15	$Y = \cos(5x - 15°)$	0–180°	1°
16	$Y = \frac{2x}{x+100}$	–80 ÷ 80	10
17	$Y = 1 - e^{2x}$	0–5	0,1
18	$Y = 1 - x^2$	0–1	0,01
19	$Y = 1 + e^{4x}$	0–5	0,1
20	$Y = \cos(4x)$	0–360°	2°

Для сохранения и открытия графиков в MATLAB используются стандартные команды **Save As** и **Open** из меню **File** графического окна. Изучите самостоятельно эти команды и сохраните все свои графики в файле с расширением .fig.

Закройте графическое окно и снова загрузите его с помощью команды **Home⇒Open** (для интерфейса с меню из главного меню **File** системы MATLAB).

2 При построении в одних осях двух графиков одного аргумента значения функции можно задать либо в виде соответствующих векторов (**Y1** и **Y2** – на рисунке 3.9), либо в виде соответствующих строк матрицы (**Y** – на рисунке 3.10).

```
Command Window
>> X=[0:pi/100:2*pi];Y1=sin(X);Y2=sin(X+pi/4);
plot(X,Y1,X,Y2), grid
title('Функция Y1=sin(X) Y2=sin(X+\pi/4)')
fx >>
```

Рисунок 3.9

```
Command Window
>> X=[0:pi/100:2*pi];Y=[sin(X);sin(X+pi/4)];
plot(X,Y),grid
title('Функции Y1=sin(X) Y2=sin(X+\pi/4)')
fx >>
```

Рисунок 3.10

В результате выполнения обеих программ (см. рисунки 3.9 и 3.10) на экране появится окно с графиками двух функций в одних осях (рисунок 3.11).

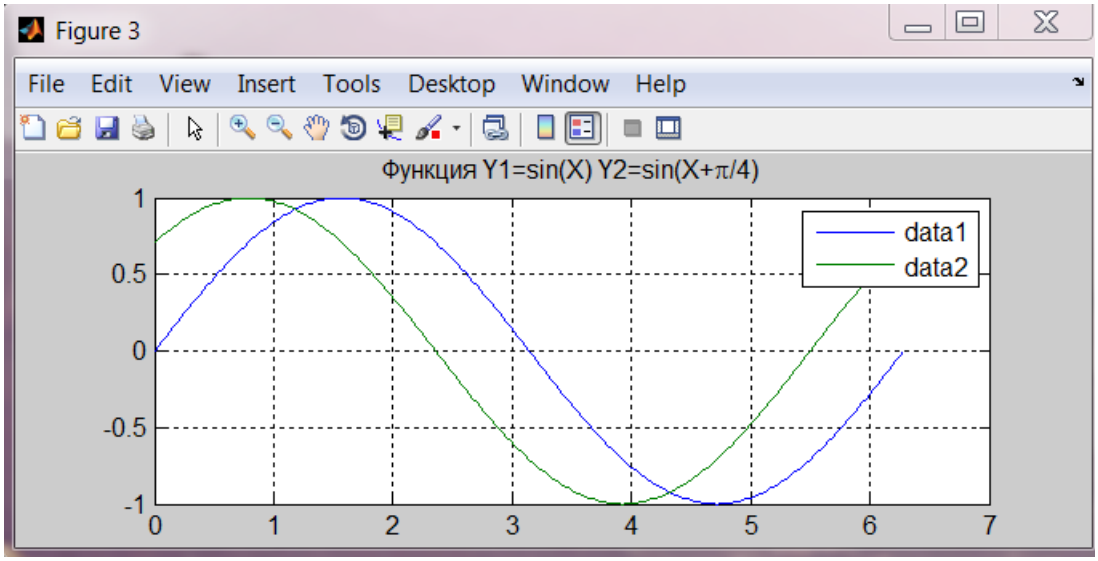


Рисунок 3.11

В правом верхнем углу графического окна добавлены пояснения в виде разноцветных отрезков линий со справочными надписями, называемые легендой. Легенду можно добавить путем нажатия на кнопку «**Insert Legend**» на панели инструментов окна графиков. Команда **legend(stringl, string2, ...)** добавляет к текущему графику легенду в виде строк, указанных в списке параметров. Для переноса ле-

генды в другое место достаточно установить на нее курсор, нажать левую кнопку мыши и перетащить легенду в необходимую позицию.

3 Задайте для графиков функций **Y1** и **Y2** (рисунок 3.12) различный стиль представления: для **Y1** – тип линии сплошная, тип точки звездочка, а цвет по умолчанию синий; для **Y2** – тип линии штрихпунктир, тип точки квадрат, а цвет линии красный. В результате выполнения программного кода (см. рисунок 3.12) на экране появится окно с графиками (рисунок 3.13). Обратите внимание, что количество точек на этом графике в десять раз меньше, чем было на графике рисунка 3.3, так как был задан в десять раз больший шаг.

```
Command Window
>> X=[0:pi/10:2*pi];Y1=sin(X);Y2=sin(X+pi/4);
plot(X,Y1,'-*',X,Y2,'-.SR'), grid
fx >>
```

Рисунок 3.12

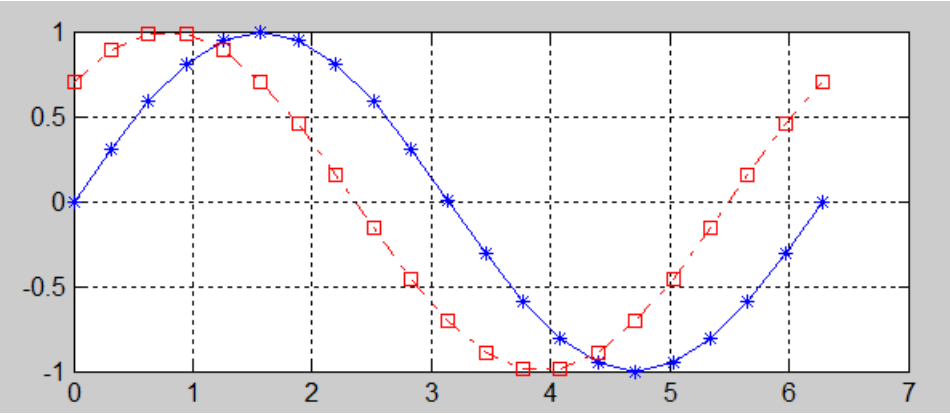


Рисунок 3.13

Постройте в одних осях два графика функции для одного аргумента. Значения аргумента и заданной функции первого графика взять в соответствии с вариантом задания из таблицы 3.1, а значения функции второго графика должны быть в два раза больше соответствующих значений первой функции. Задайте для графиков различный стиль, добавьте к графикам сетку из координатных линий, название оси «*X*», легенду и заголовок.

Отчет

Включить в отчет по лабораторной работе содержимое командного окна и полученные графики функции.

4 Для построения векторных диаграмм, например, при расчете гармонических цепей удобно использовать команду

compass(VR,VI)

где **VR**, **VI** – векторы, определяющие действительную и мнимую части каждого из радиус-векторов. Радиус-векторы строятся в виде стрелок, исходящих из начала координат и имеющих угол и длину, определяемые действительной и мнимой частями комплексных чисел, представляющих их.

В результате выполнения программного кода (рисунок 3.14) на экране появится окно с векторами (рисунок 3.15,*а*).

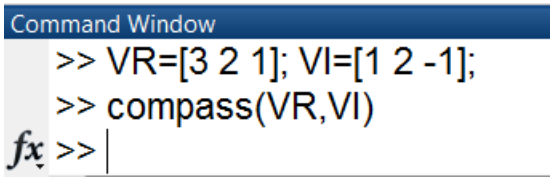
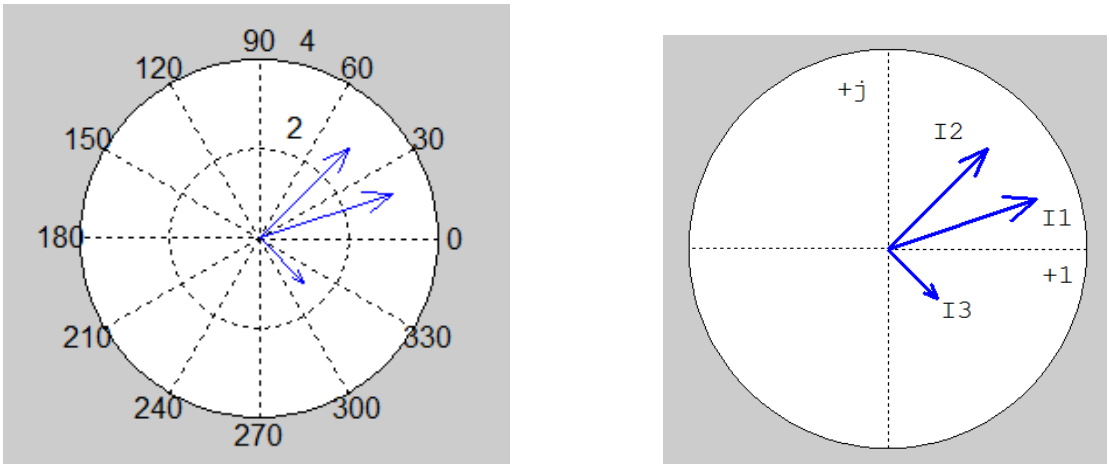


Рисунок 3.14



а) *б*)

Рисунок 3.15

Для придания диаграмме большей наглядности нанесем на нее буквенные обозначения векторов и осей с помощью команды **gtext('строка')**, а также уберем из нее лишние элементы (см. рисунок 3.15,*б*).

Для настройки изображения графика можно воспользоваться средствами редактирования графического окна, подключив его либо с помощью команды **Edit⇒Figure Properties**, либо с помощью

команды **View⇒Properties⇒Edit**. На рисунках 3.16 и 3.17 приведены примеры окна редактирования **Properties Edit** при редактировании текста и линии соответственно.

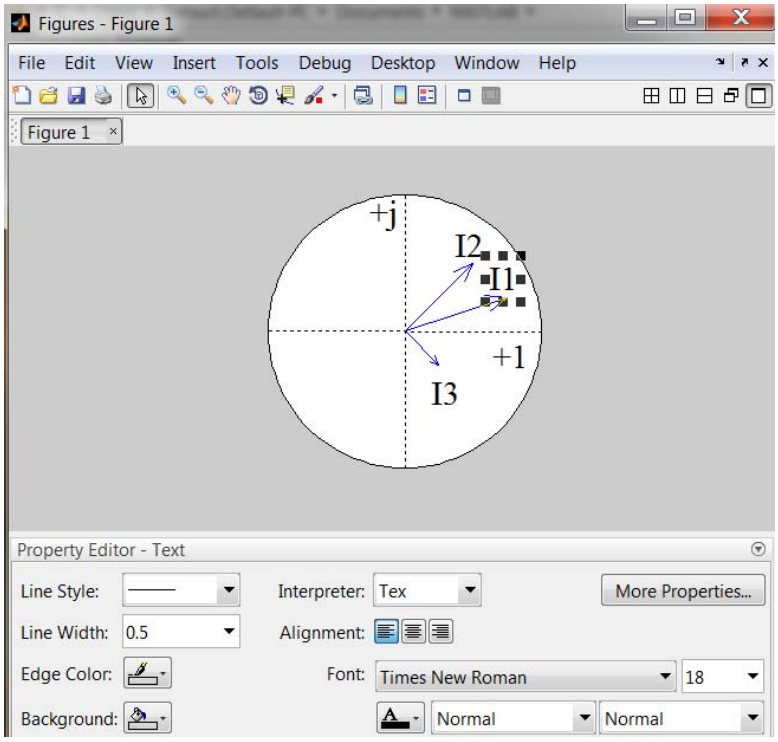


Рисунок 3.16

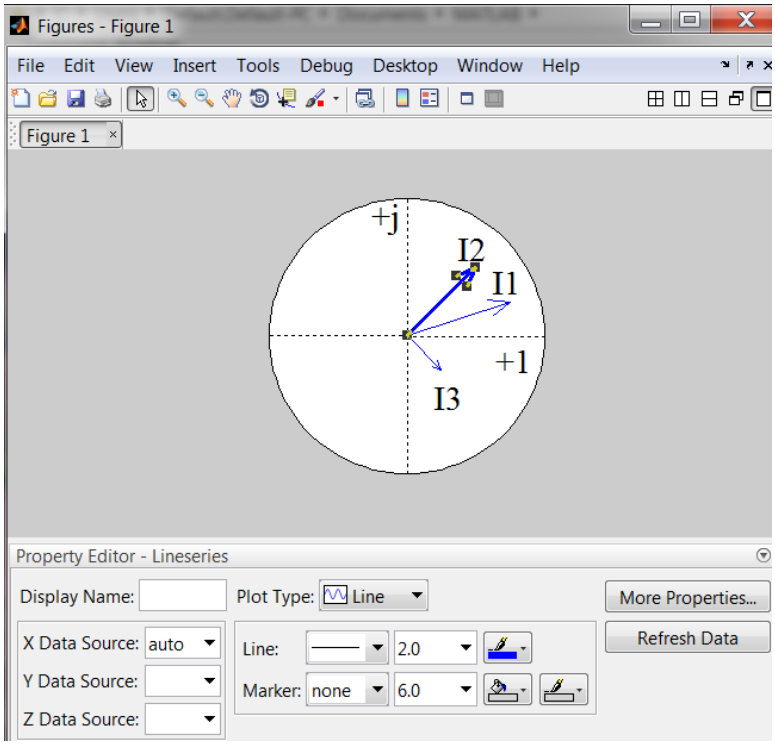


Рисунок 3.17

Отчет

Ввести в командном окне два комплексных числа $z1 = N + i(N + 10)$ и $z2 = N + i\,N$, где N – номер варианта. Построить векторную диаграмму для трех векторов $z1$, $z2$ и векторной суммы $z1 + z2$. Нанести на векторной диаграмме буквенные обозначения векторов и осей, а также убрать из нее лишние элементы. Включить в отчет по лабораторной работе содержимое командного окна и окна с векторами.

5 В MATLAB для построения графиков функций со значениями X и Y, изменяющимися в широких пределах, используют команду **loglog(X,f(X))**, которая позволяет построить график функции $f(X)$ в логарифмическом масштабе для координатных осей X и Y. В результате выполнения программного кода (рисунок 3.18) на экране появится окно с графиком (рисунок 3.19). Диапазон изменения функции в логарифмическом масштабе (по основанию 10) задается с помощью функции **logspace**, а командой **grid on** строится координатная сетка.

```
Command Window
>> X=logspace(0,2);
>> loglog(X,exp(X))
>> grid on
fx >>
```

Рисунок 3.18

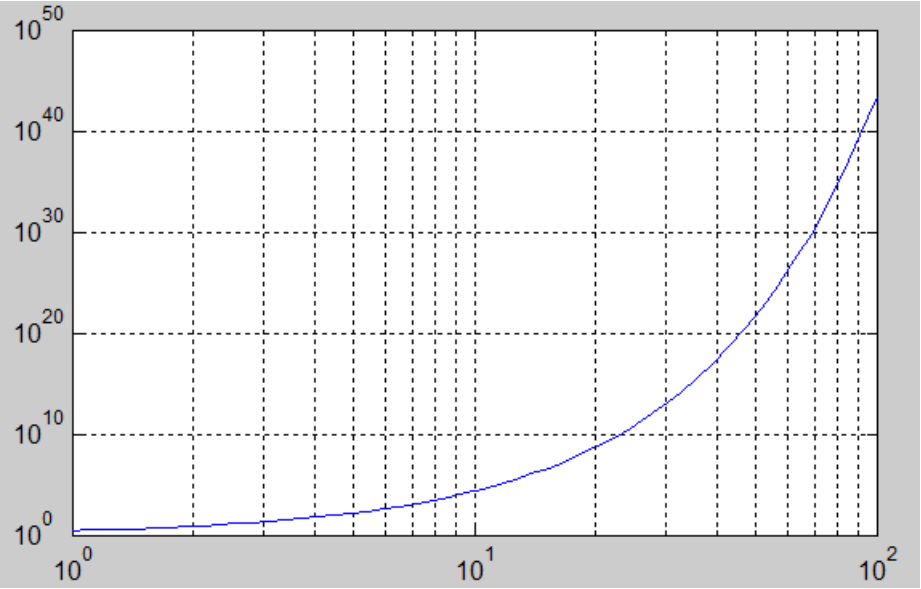


Рисунок 3.19

На рисунках 3.20 и 3.21 приведены окна графиков, в которых используется логарифмический масштаб только по одной оси. Команда **semilogx(X,exp(X))** строит график экспоненты в логарифмическом масштабе по оси *X* и линейном по оси *Y*. Команда **semilogy(X,exp(X))** строит график экспоненты в логарифмическом масштабе по оси *Y* и линейном по оси *X*.

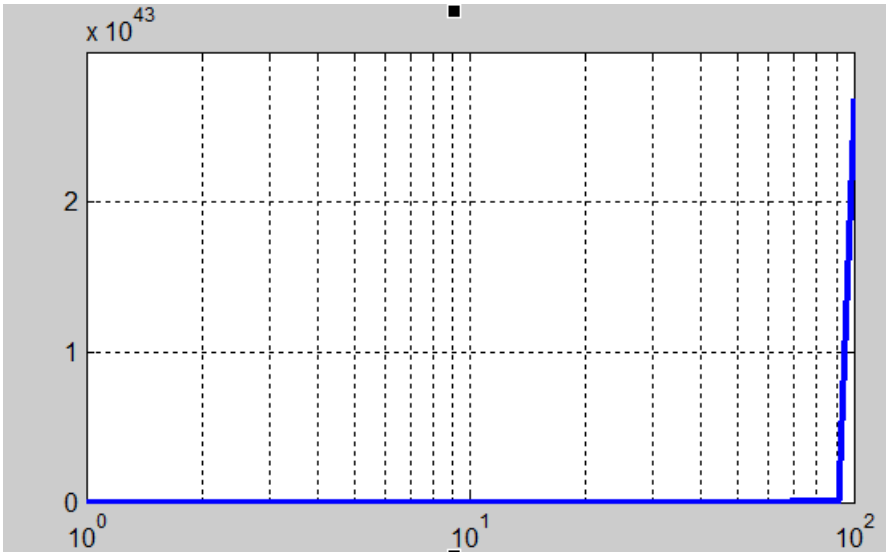


Рисунок 3.20

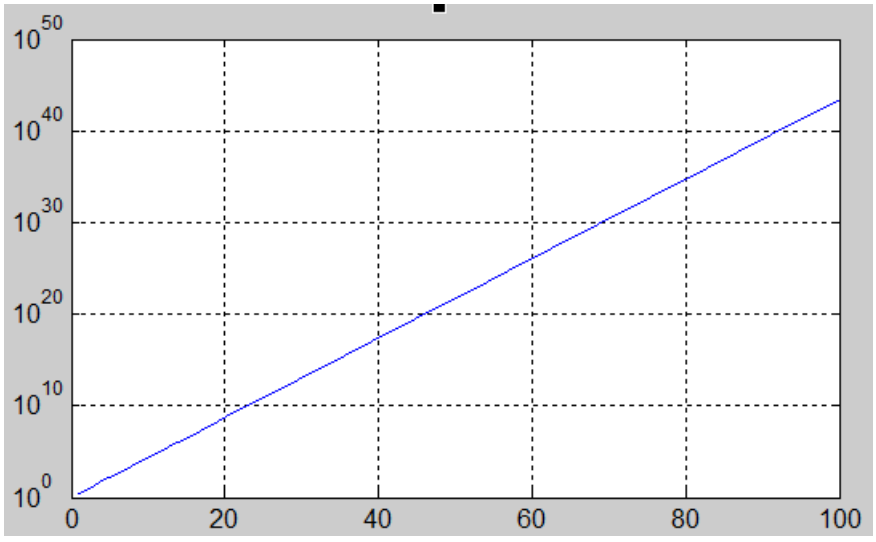
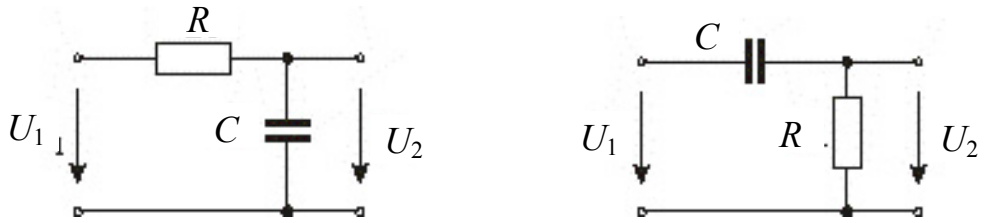


Рисунок 3.21

6 Рассчитать АЧХ и ФЧХ *RC*-фильтра нижних частот для четных вариантов (рисунок 3.22,*а*), или *RC*-фильтра верхних частот для нечетных вариантов (рисунок 3.22,*б*) при $R = 50$ Ом и емкости конденсатора (C) в микрофарадах, равной номеру варианта.



а) *б*)

Рисунок 3.22

При расчете используйте известные соотношения для частоты среза

$$f_{\text{с}} = \frac{1}{2\pi RC},$$

модуля (K_u) и аргумента (φ) комплексного коэффициента передачи по напряжению для *RC*-фильтров нижних и верхних частот соответственно:

$$K_u(f) = \frac{1}{\sqrt{1+(2\pi f RC)^2}},\ \varphi(f) = -\text{arctg}\,(2\pi f RC)\ ,$$

$$K_u(f) = \frac{1}{\sqrt{1+\ 1\big/(2\pi fRC)^2}},\ \varphi(f) = \text{arctg}\frac{1}{2\pi f RC}.$$

Отметим, что АЧХ и ФЧХ четырехполюсника могут быть получены также из его комплексного коэффициента передачи K_u с помощью функций **abs(Ku)** и **angle(Ku)**, которые возвращают модуль и фазу комплексного числа.

Отчет

Включить в отчет по лабораторной работе содержимое командного окна и графические окна с АЧХ и ФЧХ рассчитанного фильтра в логарифмическом масштабе для диапазона изменения частоты от $0{,}1f_{\text{с}}$ до $10f_{\text{с}}$ с шагом, равным $0{,}1f_{\text{с}}$. Добавить в графики сетку, название осей и заголовок.

Контрольные вопросы

1 Какие команды используются в MATLAB для построения одного и нескольких графиков в одних осях?

2 Как задаются различные стили графиков?

3 Как добавить к графикам сетку из координатных линий, названия осей, легенду и заголовок?

4 Каким образом в MATLAB можно вводить верхние и нижние индексы с помощью команды **text**?

5 Каким образом строятся графики в логарифмическом масштабе?

6 Каким образом можно построить векторную диаграмму в MATLAB?

7 Как можно отредактировать созданный график в MATLAB?

8 Каким образом в MATLAB можно сохранить график в файле?

9 Как можно открыть график из файла в MATLAB?

Лабораторная работа № 4

Программирование в среде MATLAB

Цель работы: изучение возможностей MATLAB по составлению файлов-сценариев и файлов-функций, в том числе для организации разветвляющихся и циклических программ.

Рабочее задание

Работа в режиме калькулятора в среде MATLAB не позволяет выполнять циклические алгоритмы вычислений. Такие вычисления удобнее выполнять в виде программ. Для этого следует набирать команды в окне редактирования **Editor**, а затем выполнять их все сразу или частями, в том числе и из командной строки, а также сохранить их в файле с расширением .m и использовать в дальнейшем. Кроме того, в окне редактирования удобнее выполнять отладку программы.

Файлы, которые содержат в себе коды программы MATLAB, называются *M*-файлами. Они бывают двух типов: *файлы-сценарии* (**Script M-Files**) и *файлы-функции* (**Function M-Files**), описывающие функции, определяемые пользователем.

Основные программы, управляющие от начала до конца порядком всего вычислительного процесса, являются файлами-сценариями. Файлы-функции используют для оформления отдельных процедур и функций, при выполнении которых либо внутри файлов-сценариев, либо внутри других файлов-функций необходимо сначала задать значения их входных переменных.

1 Запустите программу MATLAB и настройте имя директории по умолчанию для хранения создаваемых файлов. Для этого выберите команду **Home⇒Set Path⇒Add Folder** (для интерфейса с меню – **File⇒Set Path**…**⇒Add Folder**). В появившемся окне (рисунок 4.1) укажите свою папку, находящуюся на диске H:, нажмите на клавишу Save и закройте окно, подтвердив запрос об использовании указанной папки для хранения создаваемых файлов по умолчанию.

Обратите внимание, что имя вашей папки должно содержать лишь латинские буквы или цифры без пробелов!

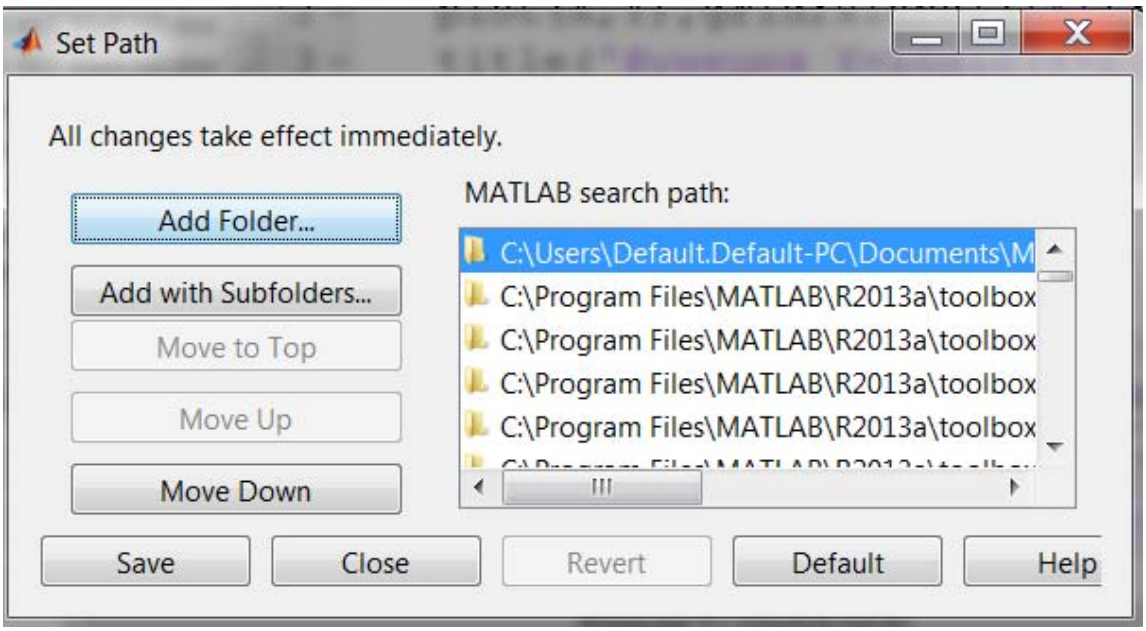


Рисунок 4.1

2 Создайте *M*-файл с помощью команды **Home⇒New⇒Script** (для интерфейса с меню – **File⇒New⇒M-file**) из основного окна MATLAB. В результате откроется окно редактора *M*-файлов **Editor**, в котором наберите команды для построения графика (рисунок 4.2).

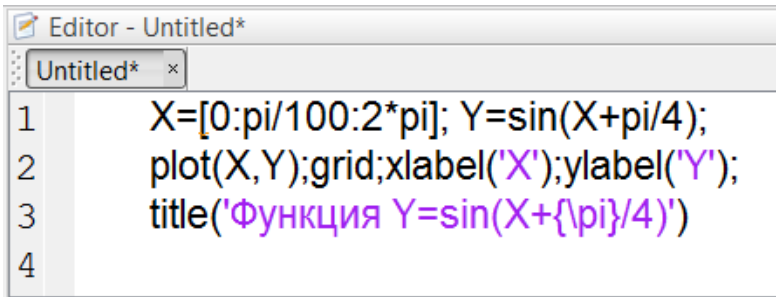


Рисунок 4.2

3 Сохраните *M*-файл в своей папке на диске H: под именем grafik1.m с помощью команды **Editor⇒Save As** (для интерфейса с меню – **File⇒Save As** в окне редактора *M*-файлов).

Обратите внимание, что имя файла должно содержать лишь латинские буквы или цифры без пробелов!

4 С помощью команды **Editor⇒Run** (для интерфейса с меню – **Debug⇒Run** в окне редактора *M*-файлов) или путем нажатия клави-

ши F5 выполните все команды из окна редактирования. В результате на экране появится графическое окно **Figure l** с графиком функции и заголовком.

5 Увеличьте в программе диапазон изменения аргумента X в два раза и запустите все команды снова. Посмотрите, как изменится график, и сделайте вывод.

6 Выделите с помощью мыши первые две строки программы в окне редактирования *M*-файлов, нажмите правую клавишу мыши и выберите в появившемся контекстном меню команду **Evaluate Selection** (или нажмите клавишу F9). Обратите внимание, что в графическом окне выведется только график без заголовка.

Отчет

Включить в отчет по лабораторной работе содержимое окон редактирования *M*-файлов и графики, полученные при выполнении пп. 2–6.

7 Закройте окно редактирования *M*-файлов и графическое окно. Откройте снова файл **figura1.m** с помощью команды **Home⇒Open** (для интерфейса с меню – **File⇒Open**).

8 В редакторе *M*-файлов может быть одновременно открыто несколько файлов. Переход между файлами осуществляется с помощью закладок с именами файлов. Создайте новый *M*-файл под именем **fazagrad.m** как файл-функцию и определите в нем новую *M*-функцию с помощью ключевого слова **function**. Формат заголовка *M*-функции имеет следующий вид:

Function [выходные переменные] = Имя (входные переменные)

Определите в окне редактора *M*-файлов **Editor** (рисунок 4.3) функцию под именем **fazagrad**, которая предназначена для перевода фазового сдвига между напряжением и током на участке электрической цепи, измеренного по осциллографу в секундах (**fazac**), в градусы (**fazag**).

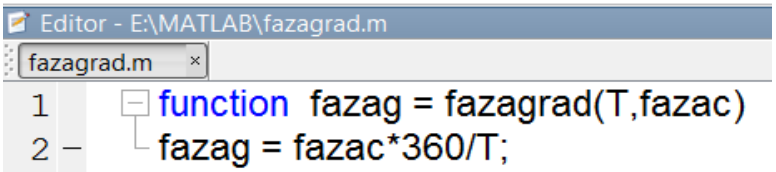


Рисунок 4.3

В качестве входных параметров данной функции выступают измеренные по осциллографу (в секундах) период гармонического сигнала *T* и фазовый сдвиг **fazac**. Данная функция возвращает свое значение (в градусах) переменной **fazag**.

9 Сохраните созданную *M*-функцию с помощью команды **File⇒Save** и закройте окно редактирования.

10 Обращение к созданной функции **fazag** может быть выполнено или в командном окне, или в окне редактора файла-сценария. На рисунке 4.4 представлен пример обращения к функции **fazagrad** из командного окна.

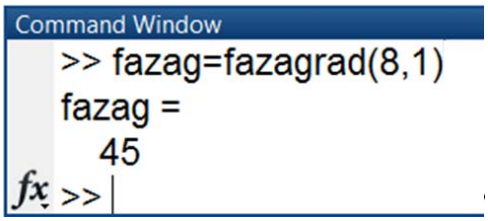


Рисунок 4.4

Отчет

Включить в отчет по лабораторной работе содержимое окна редактирования, полученное при выполнении п. 8, а также результаты обращения к функции для трех различных значений ее аргументов.

11 Как в файлах-сценариях, так и в файлах-функциях для организации разветвляющиеся и циклические вычисления используют такие операторы управления вычислительным процессом, которые начинаются со служебных слов **if**, **while** или **for** и заканчиваются служебным словом **end**. Между указанными служебными словами располагаются другие операторы.

Синтаксис оператора условного перехода приведен на рисунке 4.5.

```
if условие 1
<операторы 1>
elseif условие 2
< операторы 2>
. . .
else
<операторы>
end
```

Рисунок 4.5

При выполнении этого оператора сначала проверяется выполнение первого условия. Если оно ложно, то по порядку проверяется

условие в каждом из следующих за ним операторов **ElseIf** (их может быть сколько угодно). Как только находится истинное условие, выполняются соответствующие операторы, после чего выполнение блока **If** прекращается и управление передается на оператор, следующий за словом **End**. Если никакое из проверенных условий **ElseIf** не справедливо, то выполняются операторы, следующие за словом **Else**, если они есть. После чего выполнение блока **If…end** прекращается.

Пример программы с применением условного оператора для вычисления значения ступенчатой функции

$$Y = \begin{cases} 2x, & \text{при} \quad x = 1; \\ x^2, & \text{при} \quad x = 2; \\ x^3, & \text{при } x \neq 1 \text{ и } x \neq 2 \end{cases}$$

и результаты ее выполнения приведены на рисунке 4.6.

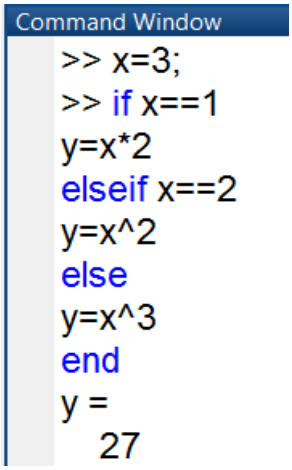


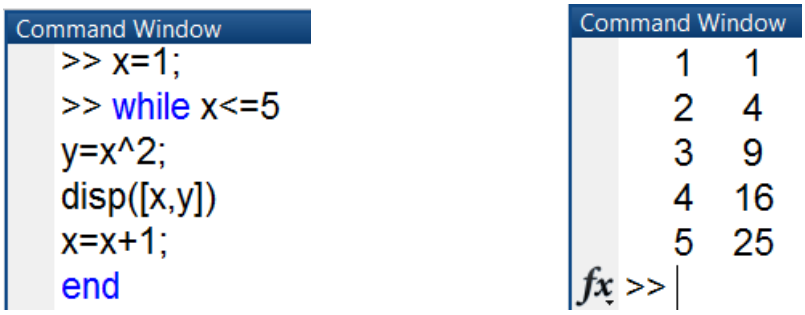
Рисунок 4.6

12 Циклы с неопределенным количеством повторений позволяет задавать конструкция **While…end**, в которой указывается условие продолжения цикла:

While условие
... [операторы]
end

Цикл повторяется до тех пор, пока условие не станет ложным. После чего все операторы до оператора **end** включительно пропускаются, и программа выполняется дальше.

На рисунке 4.7,*а* приведен пример программы с применением оператора **While**, а на рисунке 4.7,*б* – результаты ее работы.



а) *б*)

Рисунок 4.7

13 В MATLAB имеется команда **disp**, которая осуществляет вывод значений указанной переменной или указанного текста в командное окно. Обращение к ней имеет следующий вид:

disp (<переменная или текст в апострофах>)

Чтобы вывести значения нескольких переменных в одну строку (например, при создании таблиц данных), нужно создать единый объект, который содержал бы все эти значения. Это можно сделать, объединив соответствующие переменные в вектор, пользуясь операцией создания вектора строки:

x = [x1 x2 ...].

На рисунке 4.8 приведен пример программы, в результате выполнения которой в командное окно в одну строку выводятся значения четырех ранее определенных переменных через пробелы.

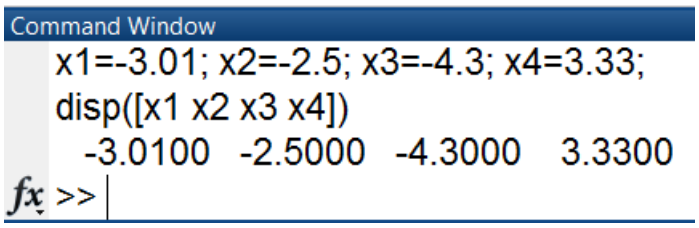


Рисунок 4.8

Для одновременного вывода символьной и цифровой информации в командное окно удобно использовать функцию **sprintf**. На рисунке 4.9 приведен пример соответствующей программы.

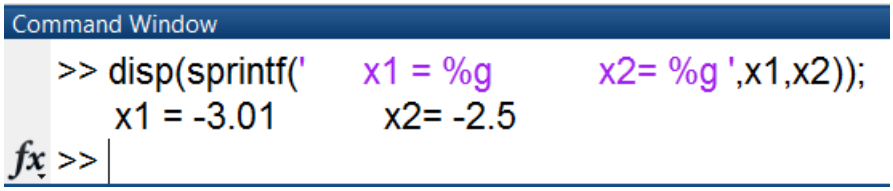


Рисунок 4.9

14 В MATLAB для организации цикла с заданным числом повторений используется оператор **For…end**, имеющий следующий формат:

for Имя = <Начальное значение>:<Шаг>:<Конечное значение>
. . . операторы
end

где **Имя** – имя управляющей переменной цикла. Если шаг не указан, то по умолчанию он равен единице.

Конструкция выполняет операторы, следующие за оператором **For**, пока в программе не встретится оператор **end**. Тогда к текущему значению переменной цикла прибавляется значение шага, и полученное значение сравнивается с конечным значением. Если значение переменной цикла больше конечного значения, то выполняется оператор, следующий за оператором **end**, иначе управление снова передается к оператору, находящемуся за оператором **For**.

На рисунке 4.10 приведен пример циклической программы, выводящей на дисплей значения переменной *i*, изменяющиеся от 0 до 30 с шагом, равным 10.

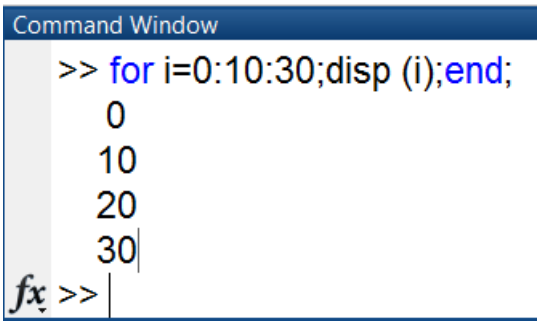


Рисунок 4.10

15 Запишите в среде MATLAB программу для вычисления и вывода на экран в виде таблицы значений x и функции $A(x)$ для заданного варианта из таблицы 1. Для всех вариантов значения x меняются

от –10 до 10 с шагом 5. Для всех вариантов составьте две программы: с использованием оператора **For…end** и оператора **While…end**.

Отчет

Включить в отчет по лабораторной работе содержимое окна редактирования и командного окна, полученное при выполнении пп. 11–15.

Таблица 4.1

Номер варианта	Функция	Номер варианта	Функция
1	$A = \sum_{n=1}^{5} \sqrt[4]{1000n + x^4}$	11	$A = \prod_{k=1}^{4} \sin\ (x + 3k)$
2	$A = \prod_{n=1}^{8} (x^2 - n)$	12	$A = \sum_{n=1}^{5} \sqrt{n^2 + 2x}$
3	$A = \prod_{k=1}^{6} \frac{\cos x}{2k}$	13	$A = \prod_{k=1}^{8} \sqrt{(xk + 2)^3}$
4	$A = \sum_{k=1}^{4} (x^4 + 4k)$	14	$A = \sum_{n=1}^{6} \frac{x^n}{n}$
5	$A = \prod_{k=1}^{5} (\mathrm{e}^x + kx)$	15	$A = \sum_{k=0}^{5} \mathrm{tg}\ (x + k)$
6	$A = \sum_{k=1}^{10} \frac{1-x}{k}$	16	$A = \sum_{i=1}^{10} \frac{1+x}{i+1}$
7	$A = \sum_{i=1}^{8} \frac{1+x}{i^3}$	17	$A = \prod_{k=1}^{5} \sqrt{(xk + 1)^2}$
8	$A = \prod_{k=1}^{6} \frac{\sin x}{2k+1}$	18	$A = \sum_{i=1}^{8} \frac{(x)}{i(i+1)}$
9	$A = \sum_{i=1}^{7} \frac{x}{(2i+1)^2}$	19	$A = \sum_{i=1}^{6} \frac{(x+2)}{i(i-1)}$
10	$A = \sum_{i=1}^{8} \frac{(-1)^x}{(x+1)(i+2)}$	20	$A = \sum_{i=1}^{5} \frac{(x)^2}{4^i + 5^i}$

Контрольные вопросы

1 Как задается место хранения рабочих файлов в MATLAB?

2 Чем отличаются файлы-сценарии и файлы-функции в среде MATLAB?

3 Как создаются, открываются, сохраняются и запускаются на исполнение *M*-файлы в среде MATLAB?

4 Как выполнить несколько строк из окна редактирования в среде MATLAB?

5 Какие операторы управления вычислительным процессом существуют в среде MATLAB и как они работают?

Лабораторная работа № 5

Визуальное моделирование динамических систем в среде MATLAB

Цель работы: изучение интерфейса и основных возможностей программного модуля **Simulink**; знакомство с разделами библиотеки **SimPowerSystems**, предназначенными для моделирования электроэнергетических объектов.

Рабочее задание

MATLAB позволяет проводить в диалоговом режиме визуальные исследования во времени (визуальное программирование) динамических характеристик нелинейных систем с помощью программного модуля **Simulink**. При этом модель исследуемой системы представляется в виде *S*-модели и сохраняется в файле с расширением .slx (для более ранних версий – с расширением .mdl).

Модели создаются по технологии **Drag-and-Drop** (перетяни и оставь) из отдельных блоков (модулей). Сами модули хранятся в библиотеках программного модуля **Simulink**, которые имеют иерархическую структуру и могут расширяться пользователем за счет разработки собственных блоков. Для наблюдения моделируемых процессов используются специальные блоки («обзорные окна»), входящие в состав библиотек **Simulink**.

1 Запустите программу MATLAB с помощью команды **Home⇒New⇒Simulink Model** (для версии с меню – это команда **File⇒New⇒Model**), в результате откроется новое пустое окно **untitled** (рисунок 5.1), в котором будет осуществляться сборка *S*-модели.

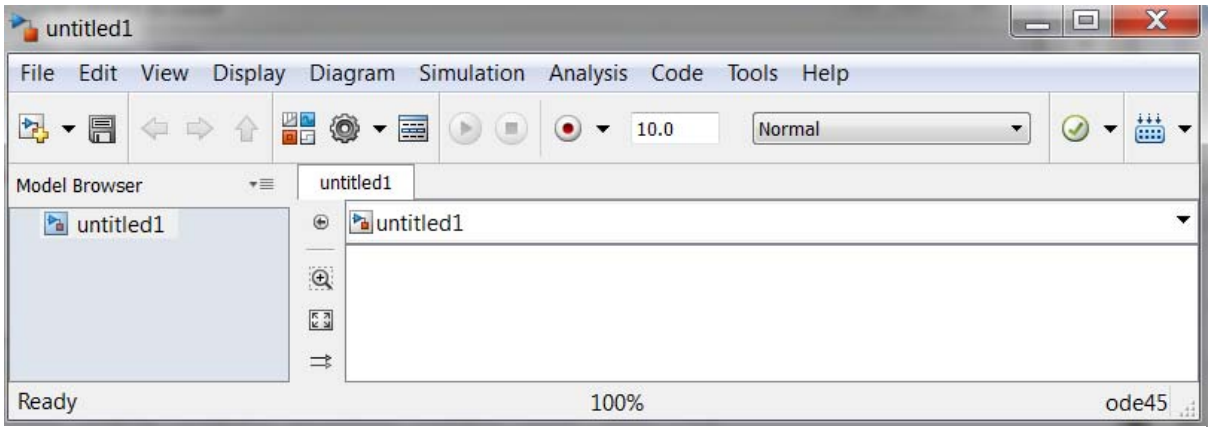


Рисунок 5.1

Выберите в открывшемся окне команду **File⇒Save As**... и сохраните файл под именем model1 на диске в своей папке.

2 С помощью кнопки  (для версии с меню – это кнопка ) откройте окно библиотеки **Simulink Library Browser** (рисунок 5.2), в левой части которого представлен перечень **Simulink**-библиотек, входящих в состав установленной конфигурации программного модуля **Simulink**. В правой части окна на закладке **Library**: **Simulink** в зависимости от выбранной библиотеки помещаются соответствующие пиктограммы ее разделов. Чтобы раскрыть перечень разделов какой-либо из библиотек следует выполнить двойной щелчок на ее имени.

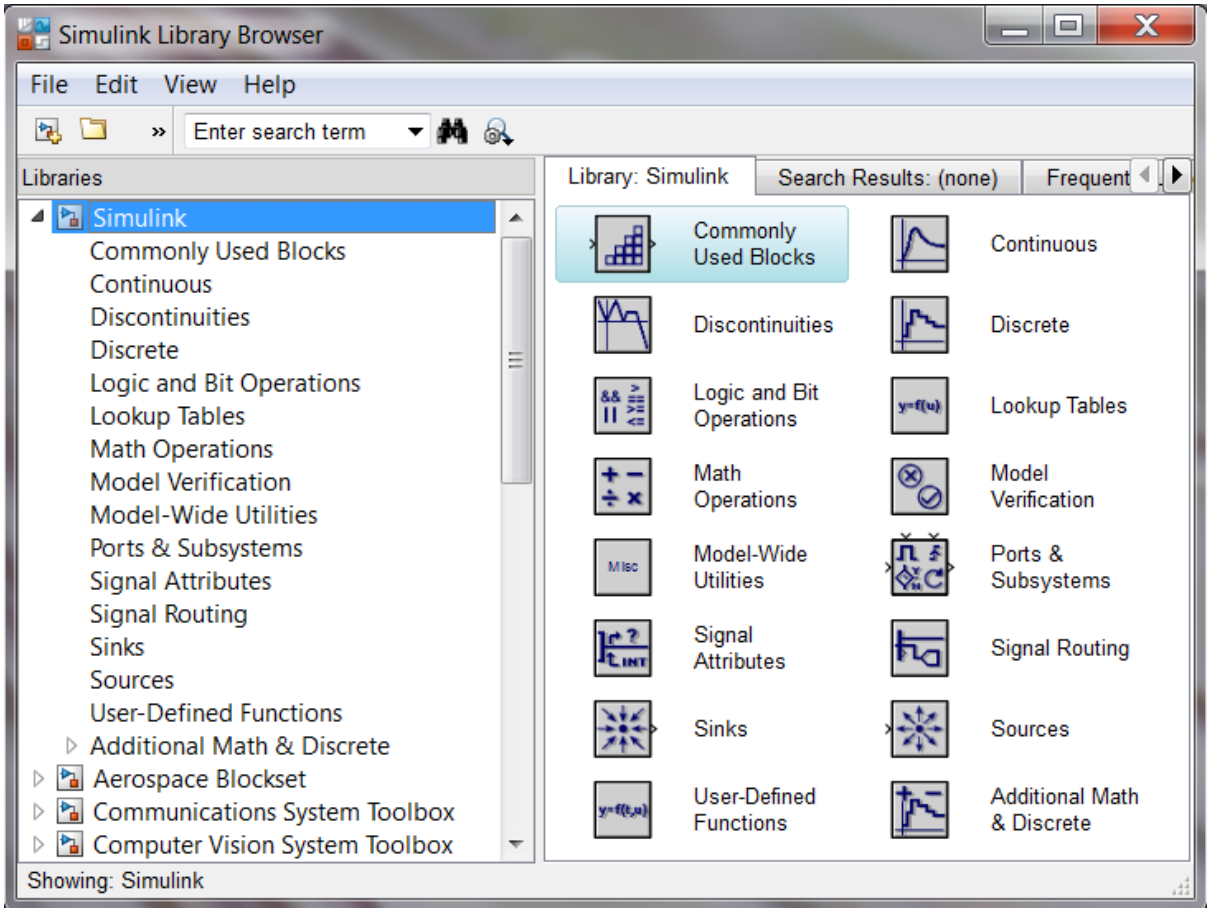


Рисунок 5.2

В зависимости от версии и выбранной конфигурации программного модуля **Simulink** окно библиотеки **Simulink Library Browser** может содержать разное число библиотек. Ядром пакета **Simulink** является одноименная библиотека **Simulink**, которая представляет собой набор визуальных объектов, предназначенных для по-

лучения функциональных блок-схем любого устройства. Остальные библиотеки при необходимости могут включаться пользователем в состав общей библиотеки.

3 Ознакомьтесь с основными блоками раздела **Sinks** (Приемники) библиотеки **Simulink**. На рисунке 5.3 показаны блоки раздела **Sinks**, которые используются как обзорные окна при моделировании. Они позволяют управлять процессом моделирования, обеспечивают сохранение промежуточных и исходных результатов моделирования. Блоки раздела **Sinks** имеют только входы и не имеют выходов.

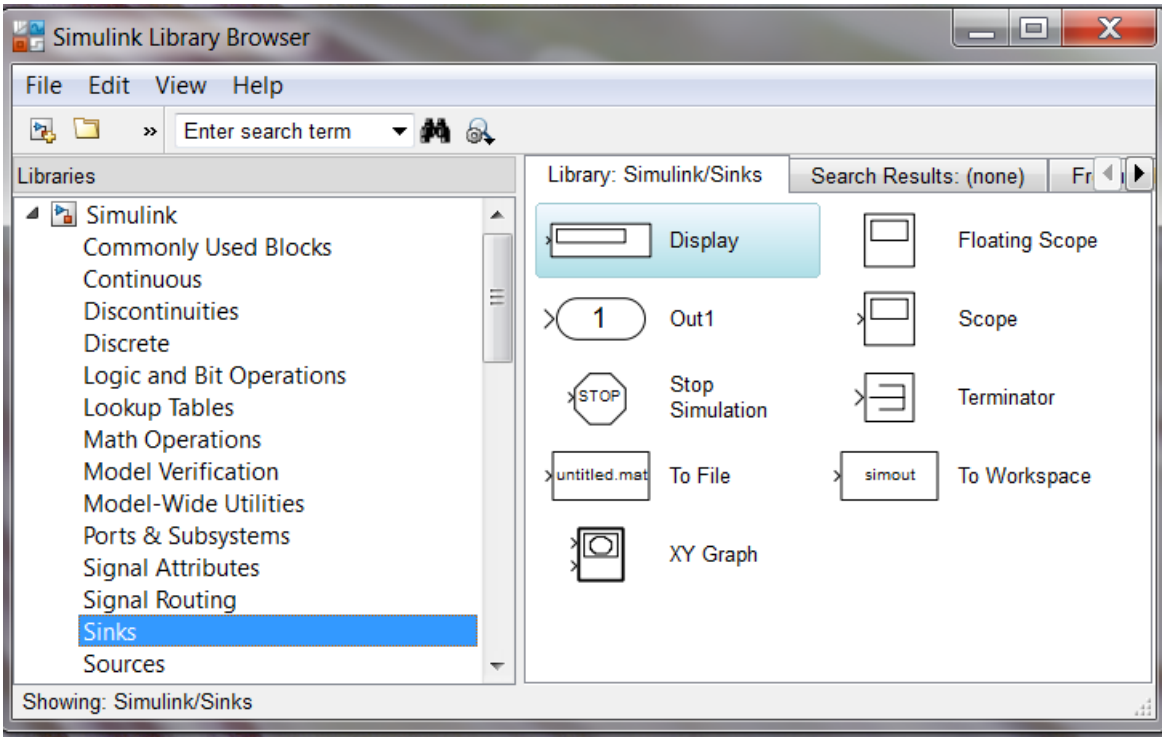


Рисунок 5.3

В качестве обзорных окон при моделировании выступают следующие блоки раздела **Sinks**:

– блок **Scope** с одним входом выводит в графическое окно график зависимости от времени подаваемой на его вход величины;

– блок **XY Graph** с двумя входами обеспечивает построение графика зависимости моделируемой величины, подаваемой на его нижний вход от величины, подаваемой на верхний вход;

– блок **Display** с одним входом предназначен для отображения численных значений входной величины.

Для сохранения результатов моделирования используются следующие блоки раздела **Sinks**:

– **To File** обеспечивает сохранение результатов моделирования на диске в MAT-файле;

– **To Workspace** сохраняет результаты моделирования в рабочем пространстве MATLAB.

Для прерывания процесса моделирования при выполнении тех или иных условий используется блок **Stop Simulation**, срабатывающий при поступлении на его вход ненулевого сигнала.

4 Ознакомьтесь с основными блоками раздела **Sources** (Источники) библиотеки **Simulink**. На рисунке 5.4 показаны блоки раздела **Sources**, предназначенные для формирования входных сигналов, которые обеспечивают работу *S*-модели в целом или отдельных ее частей при моделировании.

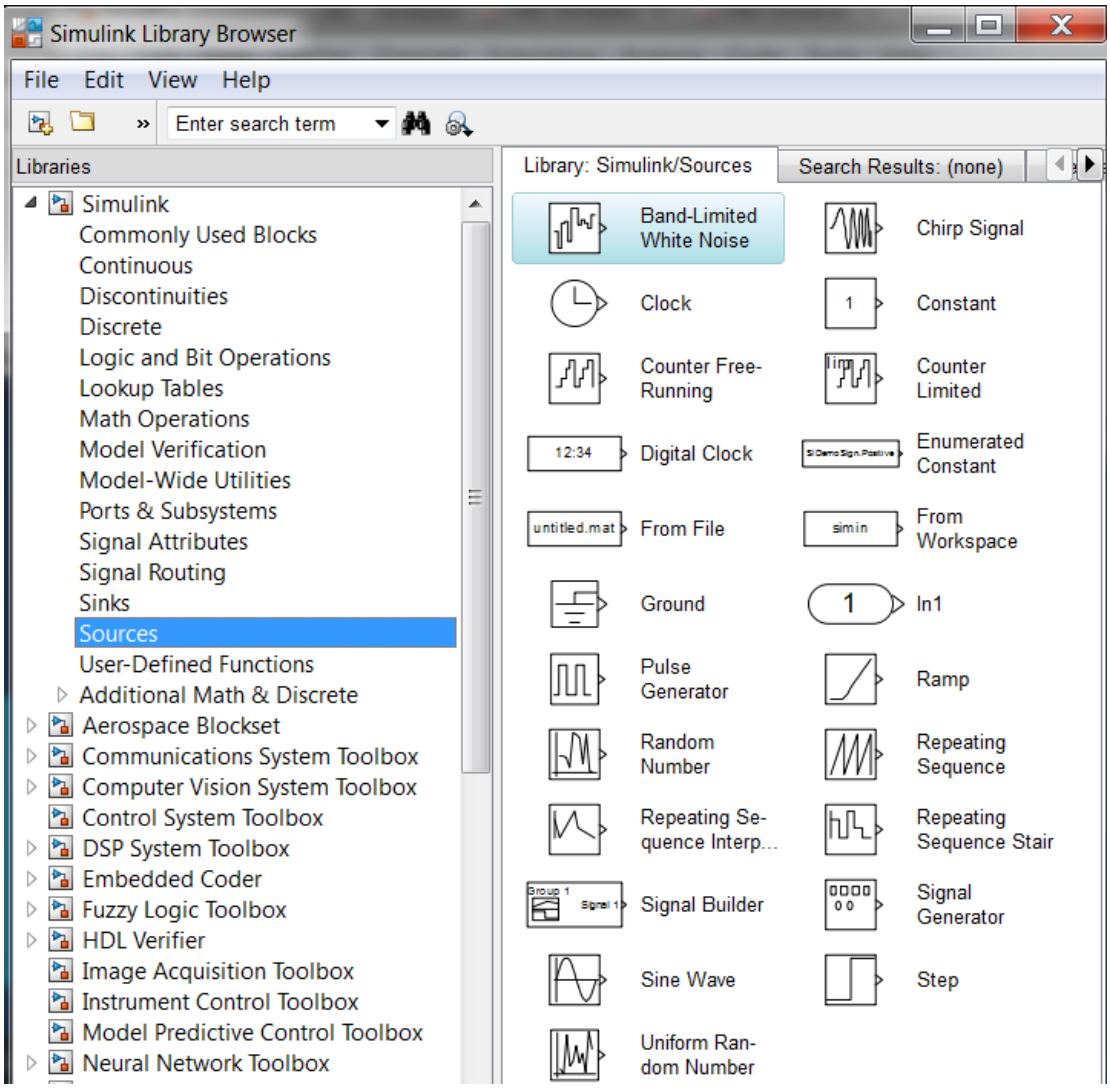


Рисунок 5.4

Они имеют по одному выходу, но не имеют входов.

В данной лабораторной работе будут использоваться следующие блоки-источники:

– **Sine Wave** генерирует гармонический сигнал;

– **Pulse Generator** – генератор непрерывных прямоугольных импульсов;

– **Ramp** создает линейно восходящий (или нисходящий) сигнал (пилообразный);

– **Signal Generator** создает непрерывный колебательный сигнал одной из волновых форм: синусоидальный, прямоугольный, треугольный или случайный;

– **Clock** (Часы) – источник непрерывного сигнала, пропорционального времени моделирования;

– **Random Number** – источник дискретного сигнала, значения которого являются случайной величиной, распределенной по нормальному (гауссовому) закону.

Блоки-источники могут настраиваться пользователем, за исключением блока **Clock**, работа которого основана на использовании аппаратного таймера компьютера.

5 Постройте блок-схему *S*-модели, содержащей блоки **Sine Wave** и **Scope**, перетянув блоки из соответствующих разделов библиотек. Установите указатель мыши в область выходного порта блока **Sine Wave** (при этом указатель принимает вид крестика) и, удерживая нажатой левую кнопку мыши, переместите указатель к входному порту блока **Scope**. После отпускания кнопки мыши появится соединительная линия со стрелкой на конце, указывающей направление передачи сигнала (рисунок 5.5).

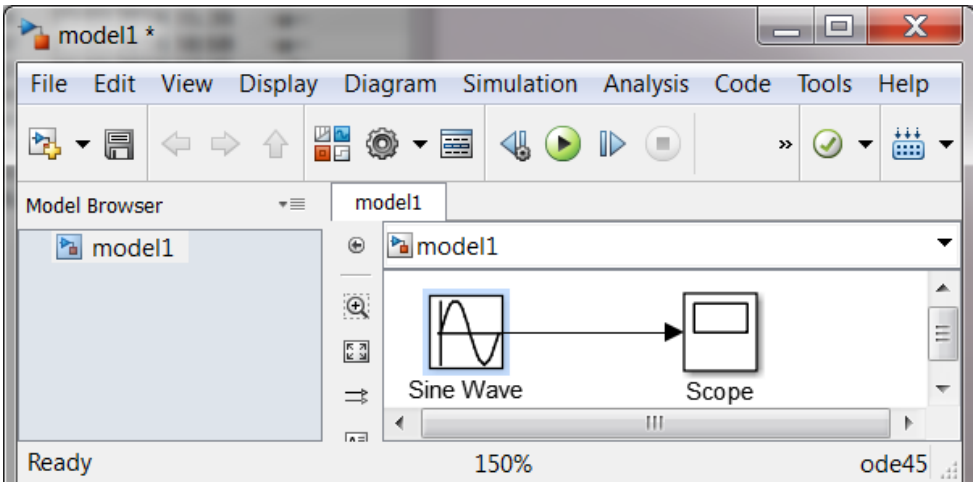


Рисунок 5.5

6 Выполните двойной щелчок по блоку **Sine Wave** и установите в появившемся окне (рисунок 5.6) значение амплитуды гармонического сигнала (**Amplitude**) равным трем, оставив остальные параметры без изменения. Закройте окно, нажав клавишу ОК.

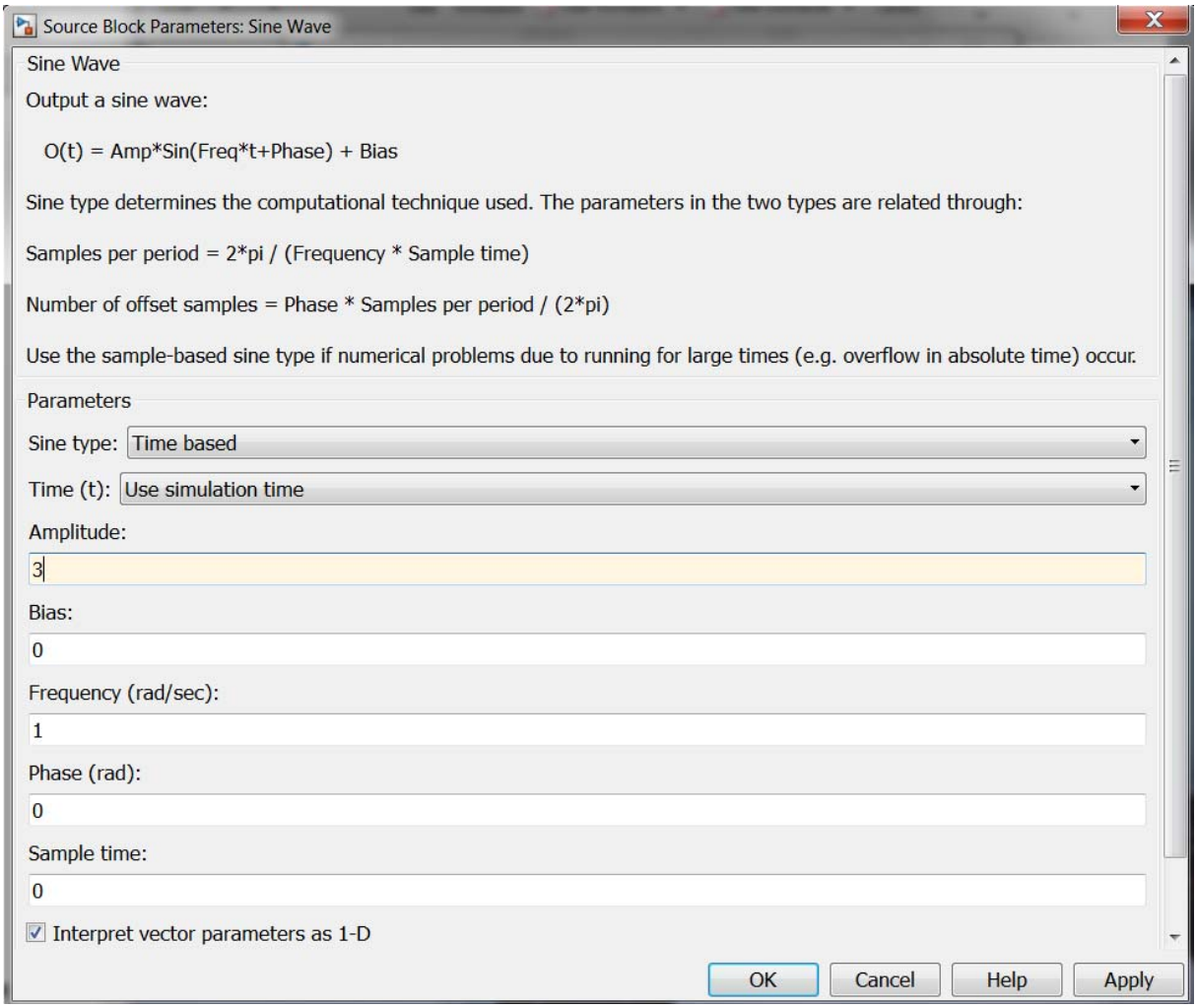


Рисунок 5.6

7 Установите параметры моделирования с помощью команды **Simulation⇒Model Cofiguration Parameters⇒Solve**. Задайте в области **Simulation time** значения начального (**Start time:**) и конечного (**Stop time:**) времени в соответствии с рисунком 5.7. В области **Solver options** (см. рисунок 5.7) выберите равномерный шаг дискретизации **Fixed step** и тип графика моделируемого процесса **ode1 (Euler)**. Задайте в области **Fixed step size** значение шага дискретизации равным «**0.01**».

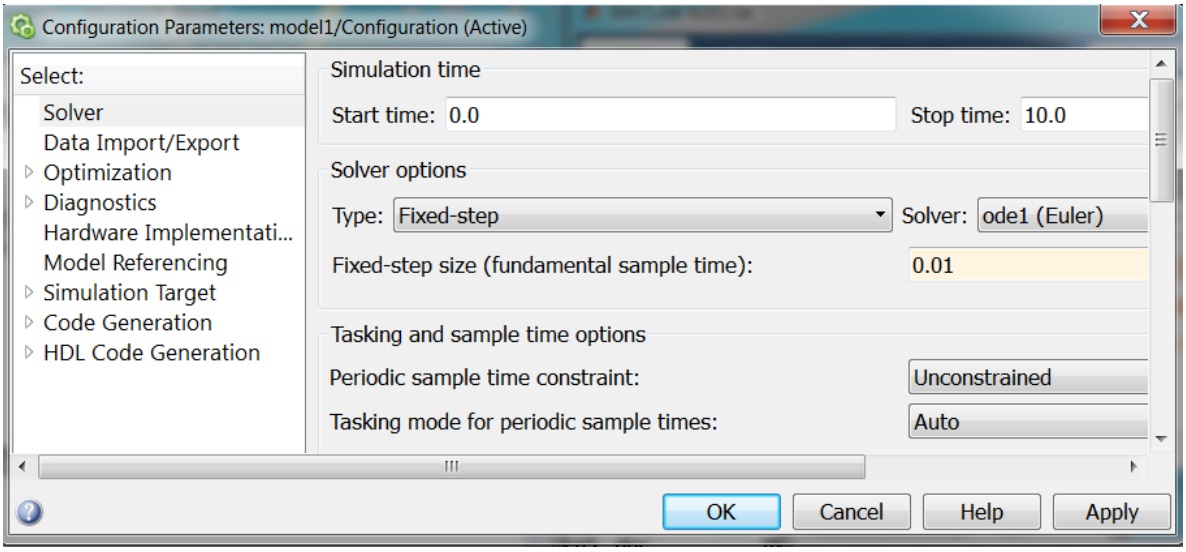


Рисунок 5.7

8 Начните моделирование с помощью команды **Simulation⇒Run**. Для получения изображения графика изменения во времени сигнала (рисунок 5.8) выполните двойной щелчок по блоку **Scope**.

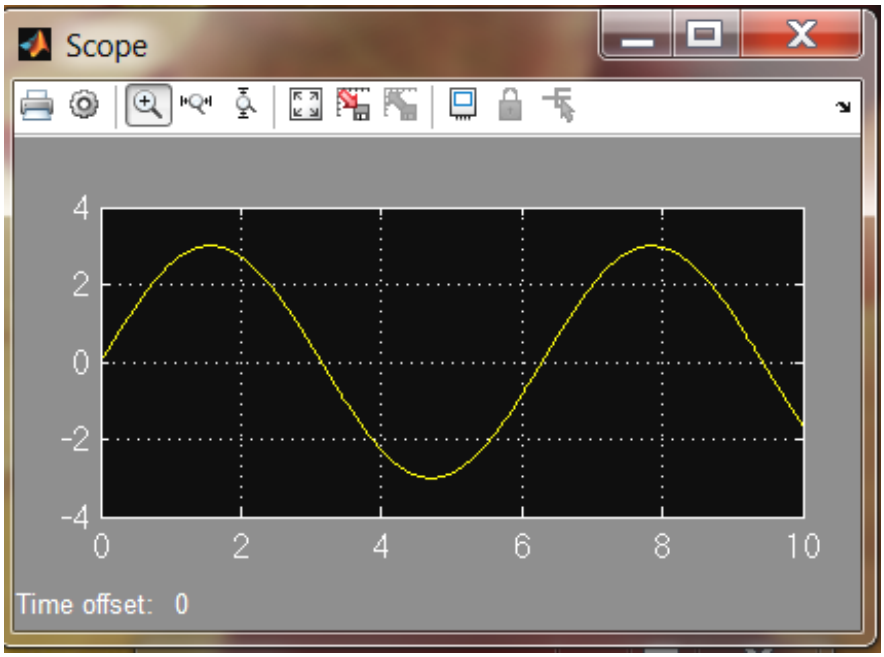


Рисунок 5.8

Параметры окна **Scope** можно настраивать с помощью кнопок, находящихся на панели инструментов в верхней части обзорного окна. Например, щелчок по второй кнопке приводит к появлению на экране диалогового окна **'Scope' parameters** (рисунок 5.9).

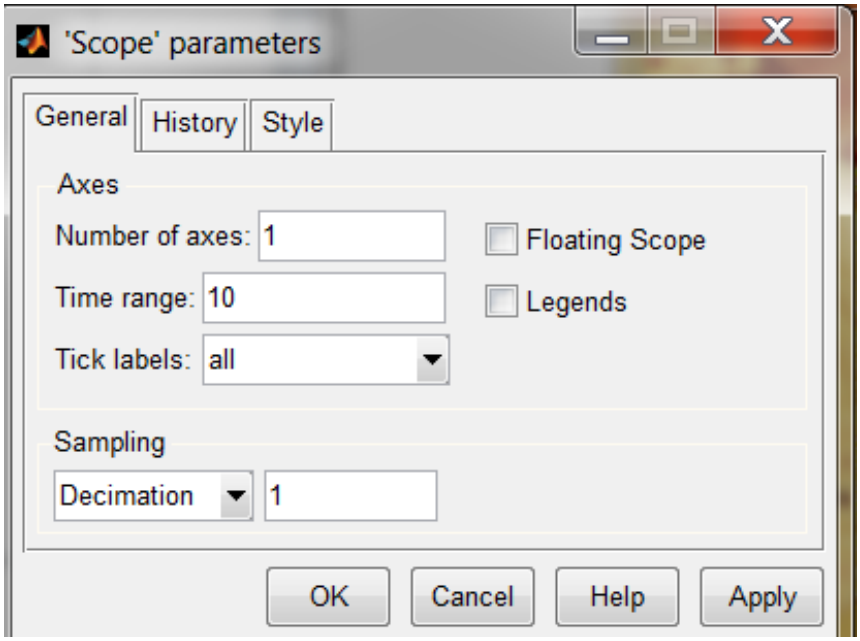


Рисунок 5.9

Установите в окне **'Scope' parameters** на вкладке **General**:

– количество графических полей (**Number of axes – 1**);

– интервал времени моделирования в секундах (**Time range – 10**);

– подписи для всех осей (**Tick Labels – all**).

Выберите из списка **Sumpling** в окне **'Scope' parameters** значение **Decimation** и задайте в соседнем правом окне значение «**1**», обозначающее количество интервалов дискретизации, через которые полученные данные моделирования будут использоваться для построения графиков.

Если выбрать из списка **Sumpling** в окне **'Scope' parameters** значение **Sumple time**, то в соседнем правом окне необходимо указать промежуток времени, кратный интервалу дискретизации, через который полученные данные моделирования будут использоваться для построения графиков.

Команды на вкладке **Data history** в окне **'Scope' parameters** позволяют задать максимальное количество элементов массива данных, используемых для построения графиков (рисунок 5.10).

Команды на вкладке **Style** в окне **'Scope' parameters** позволяют задать цвет для фигуры, фона и осей, тип, толщину и цвет линии, тип маркера в точках дискретизации, используемых для построения графиков в окне **Scope** (рисунок 5.11).

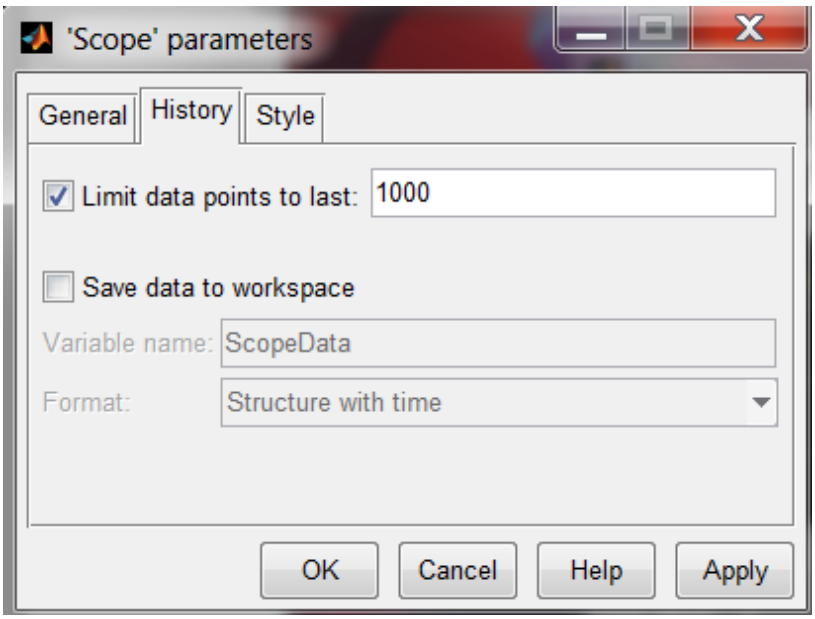


Рисунок 5.10

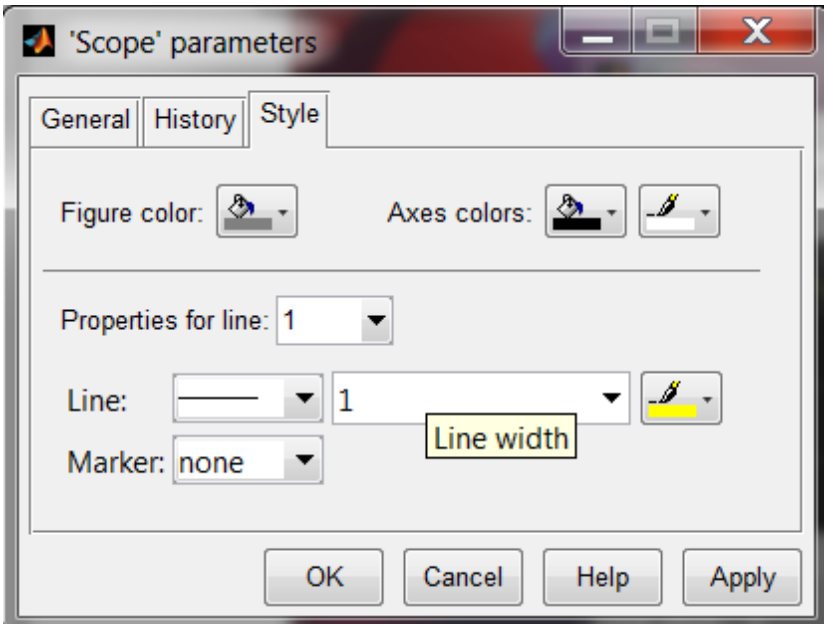


Рисунок 5.11

Средняя кнопка  на панели инструментов окна **Scope** позволяет автоматически устанавливать оптимальный масштаб осей.

9 Постройте блок-схему *S*-модели динамической системы, содержащей блок **Random Number** и блок **Scope** (рисунок 5.12) и посмотрите содержимое обзорного окна **Scope** (рисунок 5.13).

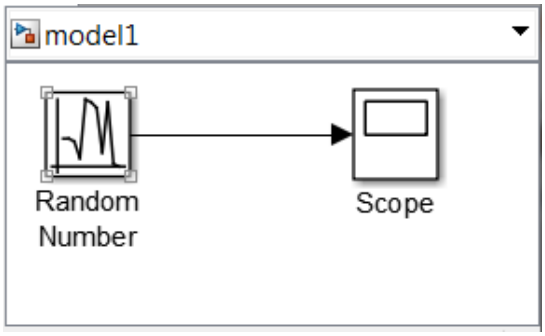


Рисунок 5.12

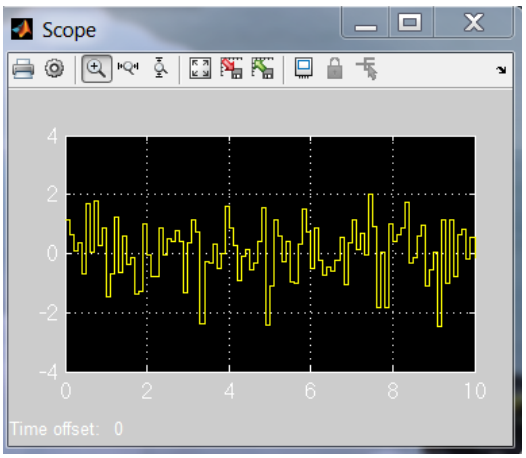


Рисунок 5.13

10 Постройте блок-схему *S*-модели динамической системы, содержащей блоки **Signal Generator**, **Clock** и **XY Graph** (рисунок 5.14).

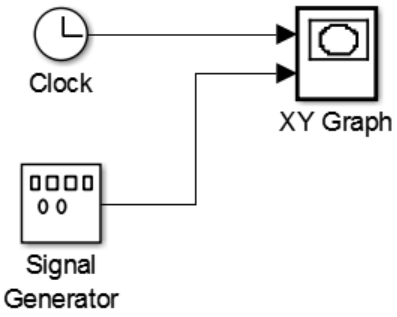


Рисунок 5.14

11 Откройте окно настройки параметров блока **Signal Generator**, выполнив по нему двойной щелчок (рисунок 5.15).

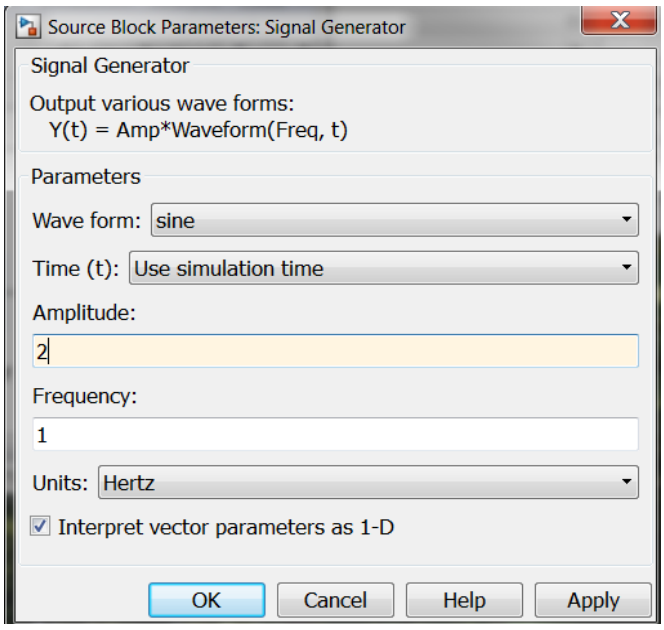


Рисунок 5.15

В области **Wave form**: задайте синусоидальный волновой сигнал (sine) с амплитудой (**Amplitude**), равной двум, и частотой (**Frequency**), равной 1 Гц.

12 Откройте окно настройки параметров блока **XY Graph**, выполнив по нему двойной щелчок. Задайте границы изменения обеих входных величин и интервал дискретизации (**Sample time:**) в соответствии с рисунком 5.16. Закройте окно, нажав клавишу ОК.

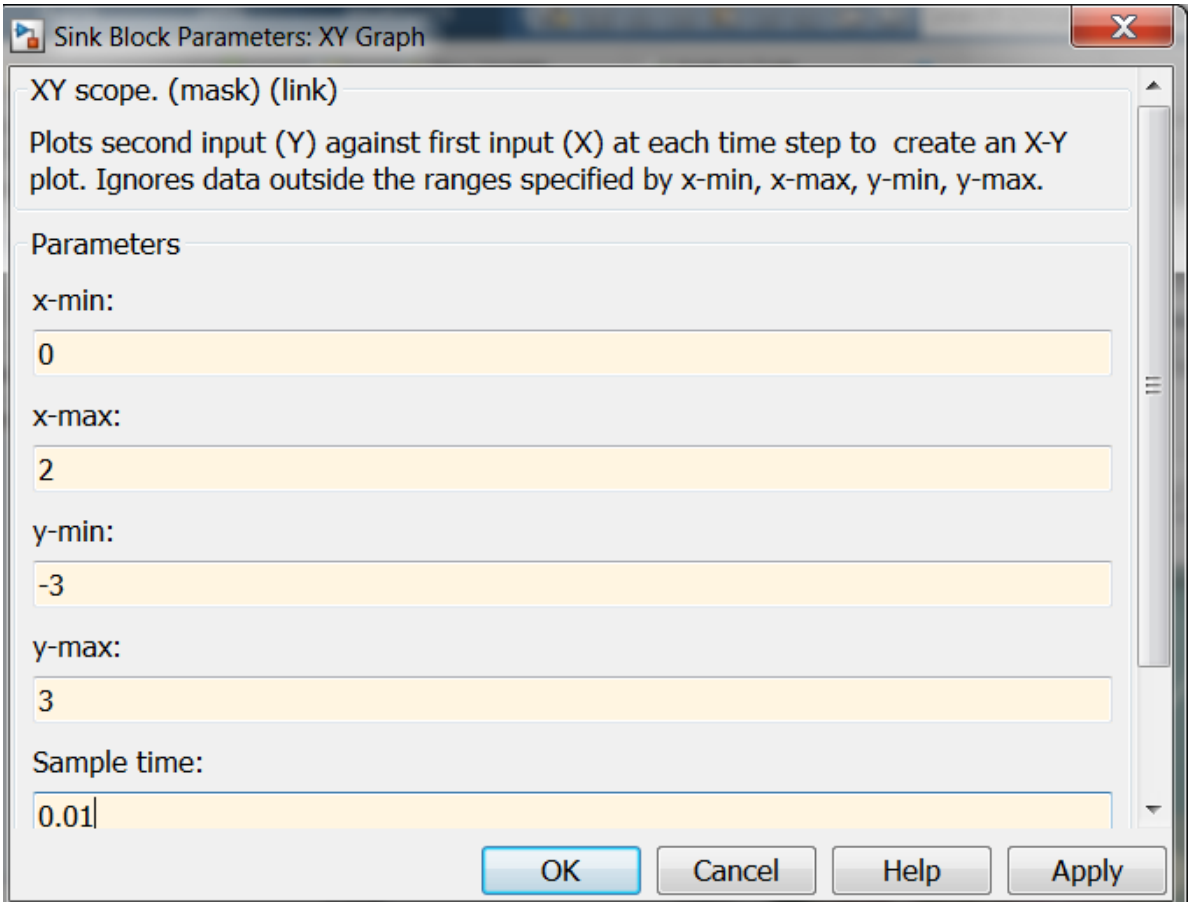


Рисунок 5.16

13 Начните моделирование с помощью команды **Simulation⇒Run**. В результате на экране появится новое окно блока **XY Graph** (рисунок 5.17).

Отчет

Включить в отчет по лабораторной работе содержимое рабочих окон с *S*-моделями, полученными при выполнении пп. 5–13, а также соответствующие обзорные окна блоков **Scope** и **XY Graph**.

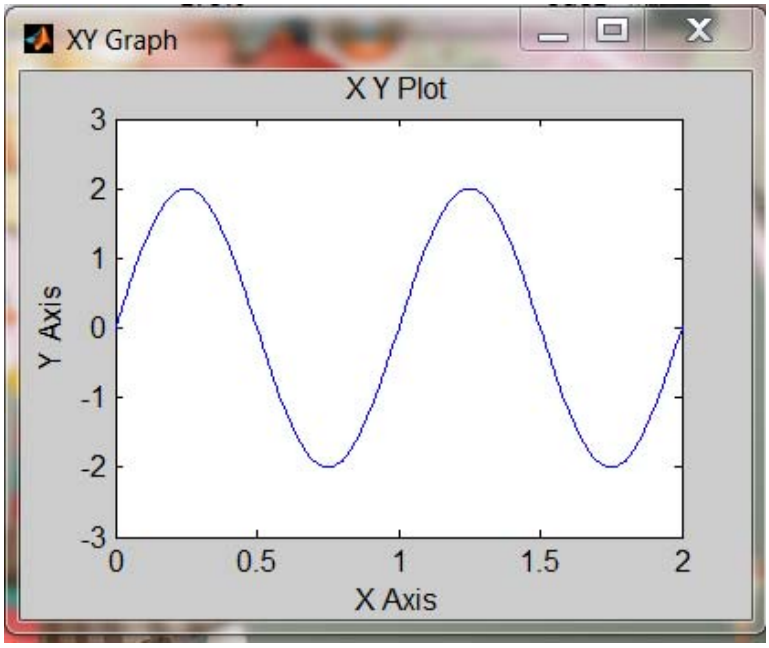


Рисунок 5.17

14 Ознакомьтесь с разделом библиотеки **SimPowerSystems**, которая представляет собой набор визуальных объектов для моделирования типовых устройств силовой электроэнергетики, таких как электрические двигатели, генераторы, трансформаторы, преобразователи, линии электропередач, а также элементы силовой электроники.

Элементы из разделов **Electrical Sources** и **Elements** библиотеки **SimPowerSystems** приведены на рисунках 5.18–5.19.

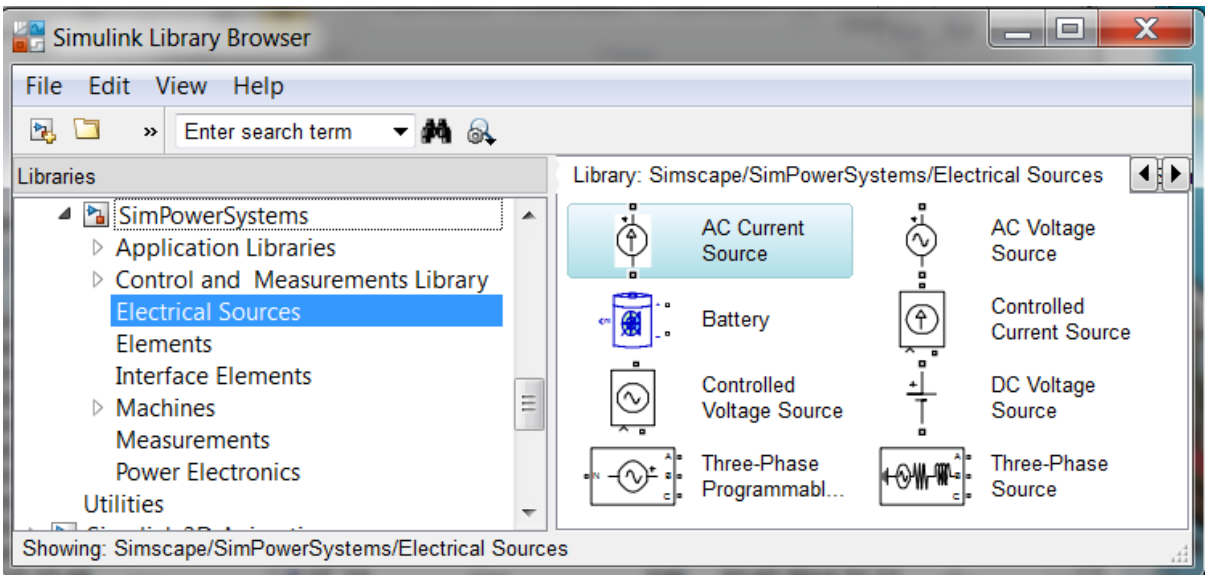


Рисунок 5.18

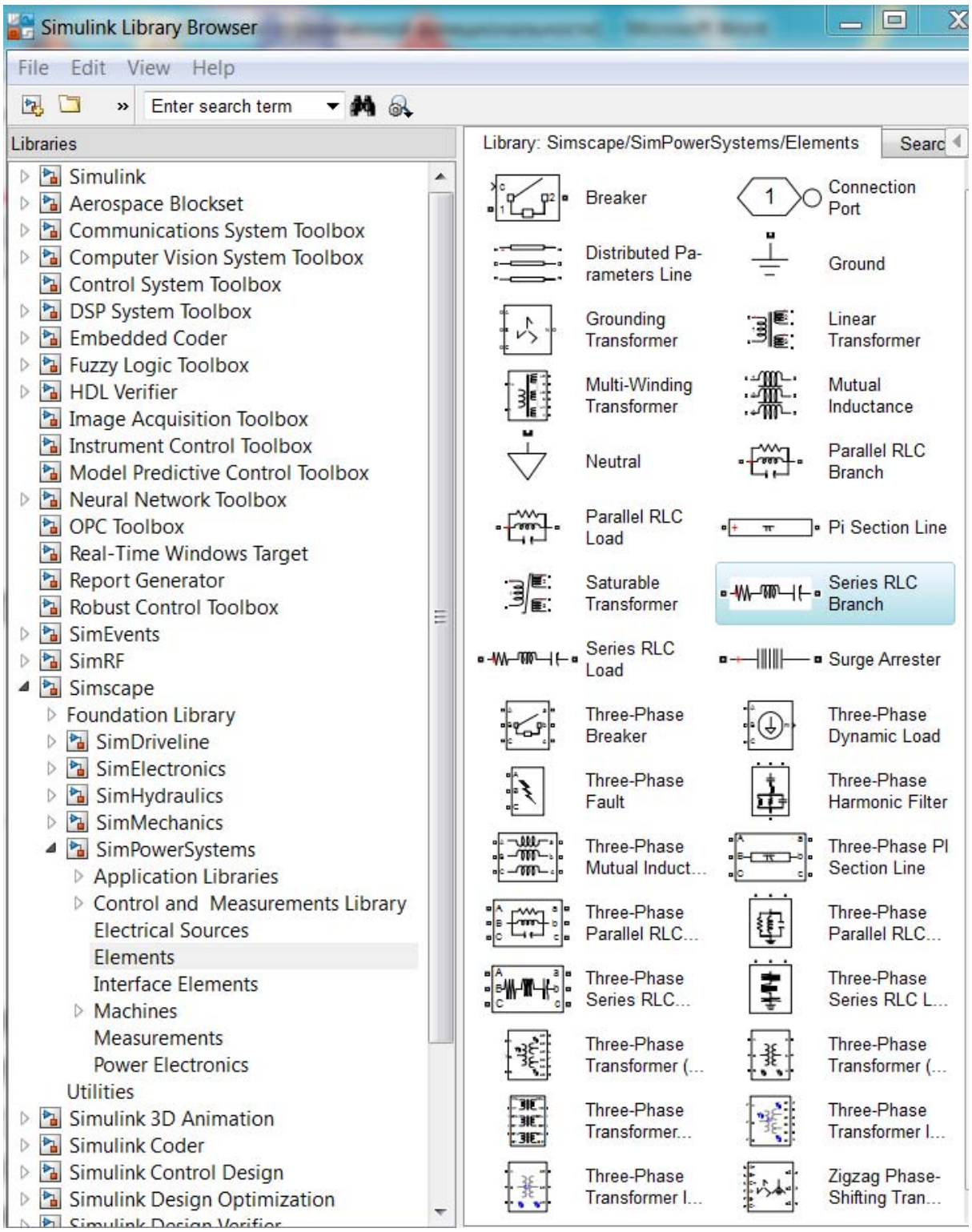


Рисунок 5.19

Обратите внимание, что модели в **SimPowerSystems** (*P*-модели с *p*-входами и *p*-выходами) имитируют процессы в электрических цепях и отличаются от обычных *S*-моделей (с *m*-входами и с *m*-выходами), в которых входные и выходные величины не имеют физиче-

ского содержания, а линии соединения переносят некоторый информационный сигнал. Поэтому *P*-блоки не могут непосредственно подключаться к *S*-блокам.

Для связи *P*-блоков с *S*-блоками используются лишь отдельные блоки библиотеки **SimPowerSystems**:

– в разделе **Measurements** (рисунок 5.20) размещаются блоки-измерители, имеющие *p*-входы и *m*-выходы (амперметры **Current Measurement**, вольтметры **Voltage Measurement** и т.д.). Эти блоки имеют *p*-входы для подключения измерителя к электрической цепи, а также по одному *m*-выходу (в блоке амперметра он обозначен «**i**», а в блоке вольтметра – «**v**»);

– в разделе **Electrical Sources** размещаются блоки источников электрических сигналов, имеющие *m*-входы и *p*-выходы, входы.

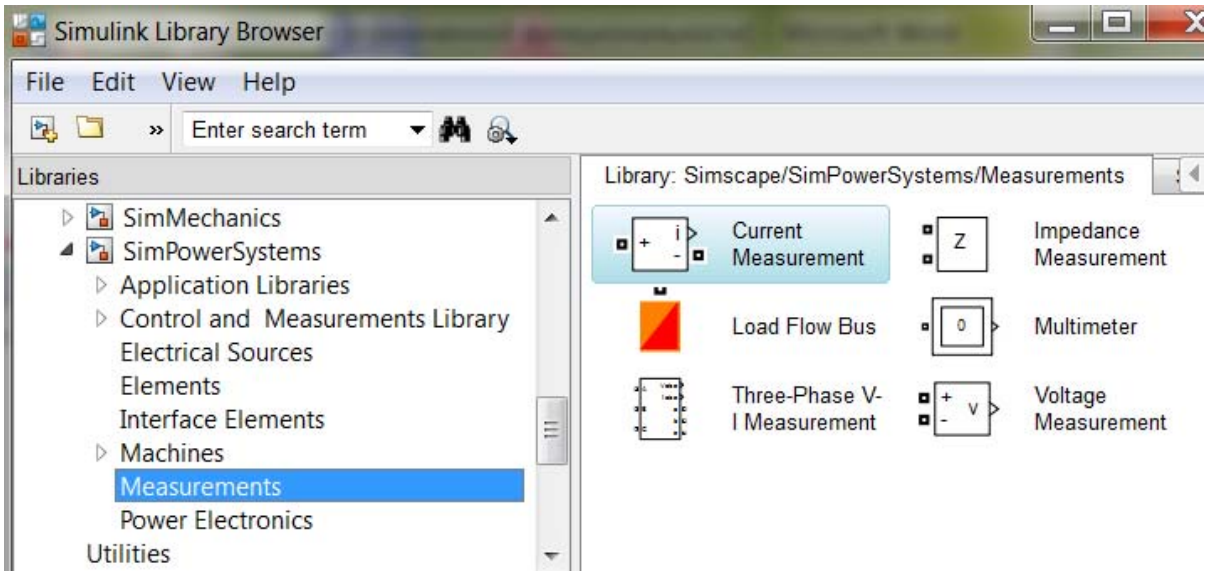


Рисунок 5.20

Другие блоки, предназначенные для преобразования, в данной работе не используются.

15 Создайте модель одноконтурной *RLC*-цепь (рисунок 5.21) с источником синусоидального напряжения **AC Voltage Source**. В качестве *RLC*-цепи используйте блок **Series RLC Branch**, представляющий собой последовательное соединение сопротивления, индуктивности и емкости, которым можно задать любые, в том числе нулевые и бесконечно большие значения параметров (системная константа inf).

Отредактируйте обозначения элементов на схеме в соответствии с рисунком 5.21.

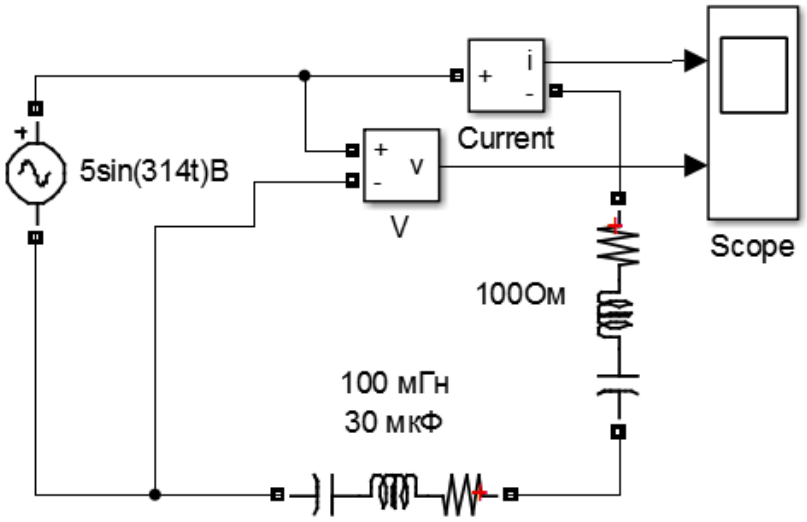


Рисунок 5.21

Для измерения сдвига по фазе между входным напряжением и током, протекающим в цепи, используйте двухканальный блок **Scope**, на входы которого подаются сигналы с *m*-выходов амперметра и вольтметра: блоки **Current Measurement** и **Voltage Measurement** соответственно. Задайте для блоков **AC Voltage Source** и **Series RLC Branch** значения параметров, указанные в соответствующем окне параметров (рисунок 5.22–5.24).

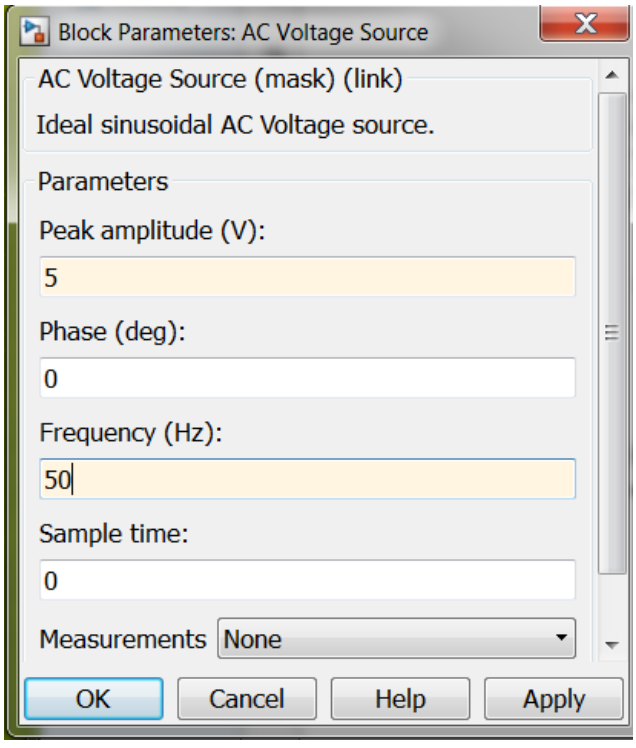


Рисунок 5.22

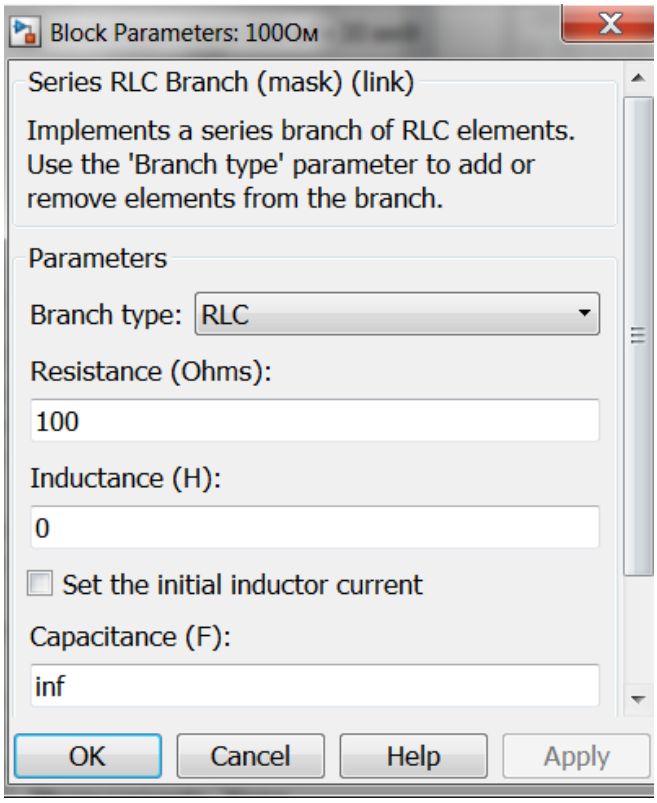


Рисунок 5.23

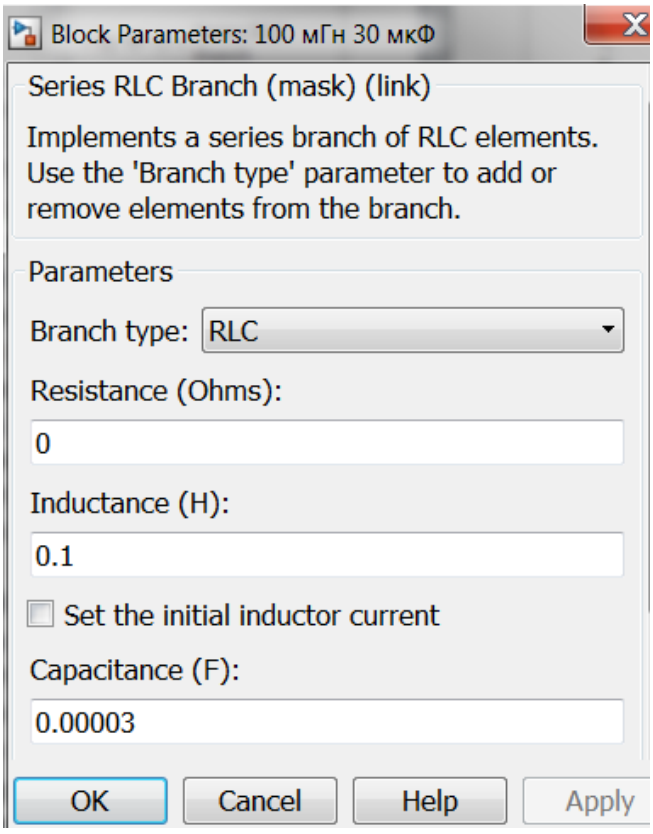


Рисунок 5.24

16 Установите требуемые параметры моделирования в соответствующем окне (см. рисунок 5.7). Начните моделирование с помощью команды **Simulation⇒Run**, а затем дважды щелкните на изображении блока **Scope**. В результате на экране появится новое окно **Scope** с изображениями графиков изменения во времени тока и напряжения (рисунок 5.25).

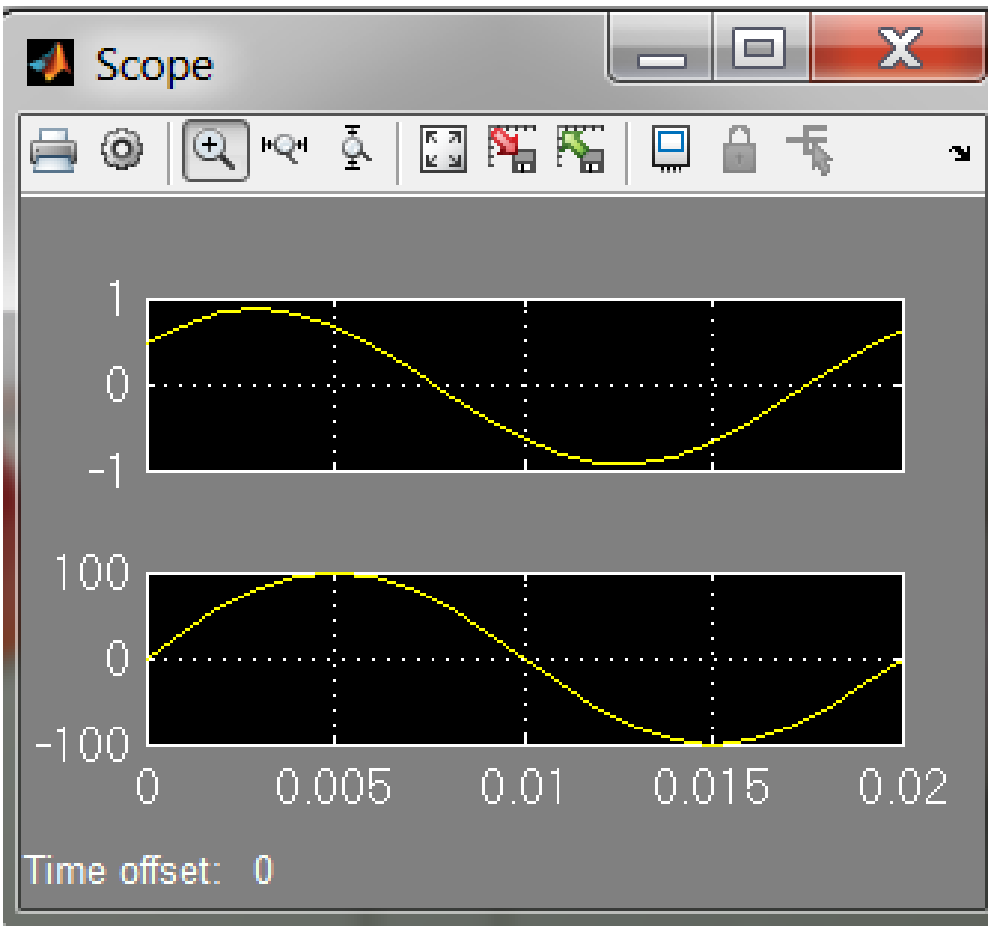


Рисунок 5.25

Отчет

Включить в отчет по лабораторной работе содержимое рабочих окон с моделями, собранными при выполнении пп. 14–16, а также соответствующие обзорные окна блоков **Scope**.

Контрольные вопросы

1 Каким образом строятся блок-схемы в программном модуле **Simulink**?

2 Какие настройки имеются у обзорных окон **Scope** и **XY Graph**?

3 Какие настройки имеются у блоков-источников **Sine Wave** и **Signal Generator**?

4 Какие настройки можно задать в окне **Cofiguration Parameters** перед запуском *S*-модели?

5 Чем отличаются *P*-модели из раздела **SimPowerSystems** от обычных *S*-моделей **Simulink** и каким образом осуществляется связь между ними?

6 Каким образом настраиваются параметры блока **Series RLC Branch** и какие значения они могут принимать?

Список литературы

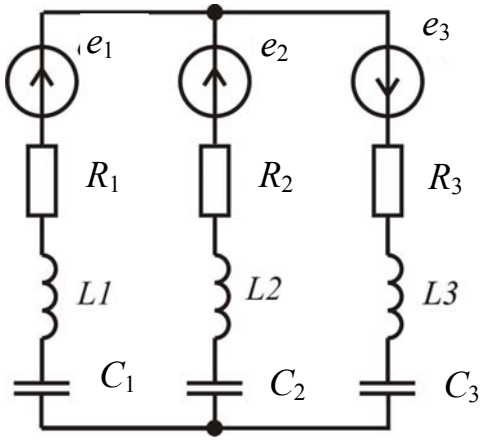
1. Малеев, Р. А. Компьютерные технологии : учеб. пособие / Р. А. Малеев, В. В. Регеда, О. Н. Регеда. – Пенза : Изд-во ПГУ, 2014.

2. URL: http://matlab.exponenta.ru/mltb/default.php – Сайт MATLAB. Exponenta

3. Лазарев, Ю. Моделирование процессов и систем в MATLAB : учеб. курс / Ю. Лазарев. – СПб. : Питер ; Киев : Изд. группа BUV, 2005.

Приложение А
Индивидуальное задание

1 Рассчитать в среде MATLAB электрическую цепь, приведенную на рисунке, в которой в зависимости от номера варианта отсутствуют элементы, указанные в таблице знаком «–».



2 Параметры цепи в зависимости от номера варианта «*N*» имеют следующие значения:

$$e = E_m \sin(\omega * t + \psi),$$

где амплитудные значения ЭДС в вольтах равны соответственно $E_{1m} = N$, $E_{2m} = 2N$; $E_{3m} = 3N$;

– начальные фазы ЭДС ($\psi_1 = \psi_2 = \psi_3$) равны 0°;

– частота всех источников ЭДС равна $f = N*50$ в Гц;

– $R_1 = R_3 = (N + 20)$ Ом; $R_2 = (N + 30)$ Ом;

– $L_1 = L_3 = 0{,}2/(N + 1)$ Гн; $L_2 = 0{,}3/(N + 1)$ Гн;

– $C_1 = 50/(N + 1)$ мкФ; $C_2 = C_3 = 100/(N + 1)$ мкФ.

К расчетным значениям параметров применить обычное округление с помощью функции **round**.

3 Построить по результатам расчета электрической цепи соответствующую векторную диаграмму токов для любого из узлов.

4 Смоделировать в системе **Simulink** рассчитанную электрическую цепь и получить в обзорном окне изображение напряжений, указанных в таблице в зависимости от номера варианта.

Номер варианта	e_1	e_2	e_3	L_1	L_2	L_3	C_1	C_2	C_3	Изображение в обзорном окне	
1	–					–	–			$u_{L2}(t)$	$u_{R1}(t)$
2		–			–			–		$u_{L3}(t)$	$u_{R2}(t)$
3			–	–					–	$u_{C1}(t)$	$u_{R3}(t)$
4	–			–					–	$u_{C2}(t)$	$u_{R1}(t)$
5		–				–	–			$u_{L1}(t)$	$u_{R3}(t)$
6			–		–			–		$u_{C1}(t)$	$u_{R2}(t)$
7	–				–			–		$u_{L1}(t)$	$u_{R2}(t)$
8		–		–					–	$u_{C2}(t)$	$u_{R1}(t)$
9			–			–	–		–	$u_{L2}(t)$	$u_{R3}(t)$
10	–				–	–				$u_{C2}(t)$	$u_{R2}(t)$
11		–		–			–			$u_{C3}(t)$	$u_{R3}(t)$
12			–					–	–	$u_{L3}(t)$	$u_{R1}(t)$
13	–							–	–	$u_{L1}(t)$	$u_{R3}(t)$
14		–			–	–				$u_{C1}(t)$	$u_{R1}(t)$
15			–	–			–			$u_{L2}(t)$	$u_{R2}(t)$
16	–						–	–		$u_{C3}(t)$	$u_{R3}(t)$
17		–				–			–	$u_{L2}(t)$	$u_{R2}(t)$
18			–	–	–					$u_{L3}(t)$	$u_{R1}(t)$
19	–			–	–					$u_{C1}(t)$	$u_{R1}(t)$
20		–					–	–		$u_{C3}(t)$	$u_{R2}(t)$
21			–			–			–	$u_{L1}(t)$	$u_{R3}(t)$
22	–									$u_{C1}(t)$	$u_{R1}(t)$
23		–								$u_{L2}(t)$	$u_{R3}(t)$
24			–							$u_{C2}(t)$	$u_{R2}(t)$

5 Составить отчет по индивидуальному заданию, включив в него титульный лист, содержимое командного окна MATLAB с результатами расчета, свою схему, смоделированную в системе **Simulink**, график с векторной диаграммой и содержимое обзорного окна.

Учебное издание

Работа с MATLAB и Simulink

Составители:
Регеда Владимир Викторович,
Регеда Ольга Николаевна

Редактор *О. Ю. Ецина*
Компьютерная верстка *Н. В. Ивановой*

Подписано в печать 21.08.2014.
Формат 60×84 1/16. Усл. печ. л. 4,18.
Тираж 100. Заказ № 805.

Издательство ПГУ
440026, Пенза, Красная, 40.
Тел./факс: (8412) 56-47-33; e-mail: iic@pnzgu.ru