МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. А. Н. КОСЫГИНА (ТЕХНОЛОГИИ. ДИЗАЙН. ИСКУССТВО)»
(ФГБОУ ВО «РГУ им. А. Н. Косыгина»)

Тюрин М. П., Бородина Е. С.

ТЕОРИЯ И ПРАКТИКА ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ

ПРАКТИКУМ

Часть 1

Учебное пособие

Допущено к изданию редакционно-издательским советом университета в качестве учебного пособия для подготовки бакалавров и магистров по направлениям

13.03.01, 13.04.01 Теплоэнергетика и теплотехника,

20.03.01, 20.04.01 Техносферная безопасность

Москва

ФГБОУ ВО РГУ им. А. Н. Косыгина, 2020

УДК 519.67
Т 98

Т98 Тюрин М.П., Бородина Е.С.
Теория и практика экспериментальных исследований. Практикум. Часть 1: Учебное пособие – М.: ФГБОУ ВО «РГУ им. А.Н. Косыгина», 2020.- 117 с.

Рецензенты:

Ивашин А. Д. – директор ООО "ПО ТЕХНОВАК", канд. техн. наук;
Кочетов О. С. – проф. кафедры энергоресурсоэффективных технологий, промышленной экологии и безопасности ФГБОУ ВО «РГУ им. А.Н. Косыгина», д-р техн. наук

В учебном пособии «Теория и практика экспериментальных исследований. Практикум. Часть 1» приведены теоретические материалы и задания для выполнения лабораторных и практических работ по применению свободно распространяемого программного обеспечения Octave. Рассмотрено решение различных математических и инженерных задач, которые могут возникать при планировании эксперимента и теоретической обработке экспериментальных данных.

Учебное пособие предназначено для обучающихся по направлениям подготовки 20.03/04.01 Техносферная безопасность, 13.03/04.01 Теплоэнергетика и теплотехника очной и заочной форм обучения и будет использовано при изучении дисциплин «Теория и практика проведения экспериментальных исследований», «Оценка погрешности результатов экспериментальных исследований», «Инженерный эксперимент. Анализ и обработка результатов эксперимента. Теория подобия и размерностей», «Постановка эксперимента в теплотехнике и оценка погрешности результатов экспериментальных исследований. Математическая обработка результатов экспериментальных исследований», «Теория подобия и физическое моделирование в промышленной теплоэнергетике».

УДК 519.67

Подготовлено к печати на кафедре энергоресурсоэффективных технологий, промышленной экологии и безопасности

Печатается в авторской редакции

Содержание

ЛАБОРАТОРНАЯ РАБОТА №1.
ОСНОВЫ РАБОТЫ В OCTAVE

Ниже описываются элементарные математические операции, переменные, встроенные функции системы и создание собственных, работа с массивами данных, что является основой работы в системе Octave.

1.1. ЭЛЕМЕНТАРНЫЕ МАТЕМАТИЧЕСКИЕ ВЫРАЖЕНИЯ

Простейшие арифметические операции в Octave выполняются с помощью следующих операторов:

• + сложение;
• − вычитание;
• * умножение;
• / деление слева направо;
• \ деление справа налево;
• ^ возведение в степень.

Вычислить значение арифметического выражения можно вводом его в командную строку и нажатием клавиши ENTER:

```
>> 13.5/(0.2+4.2)^2-2.3
ans = −1.6027
```

При вводе вещественных чисел для отделения дробной части используется точка.

Если вычисляемое выражение слишком длинное, перед нажатием клавиши ENTER следует набрать три или более точки. Это будет означать продолжение командной строки:

```
>>1+2∗3-4....
>>+5/6+7....
>>−8+9
ans=11.833
```

если в конце выражения указан символ точки с запятой «;», то результат вычислений не выводится на экран, а вместо этого активизируется следующая командная строка:

```
>>2-1;
>>2-1
ans=1
```

1.2. ТЕКСТОВЫЕ КОММЕНТАРИИ

Правилом хорошего тона в программировании всегда считался легко читаемый программный код, снабжённый комментариями. Пэтому в дальнейшем будут применяться текстовые комментарии.

Текстовый комментарий в Octave - это строка, начинающаяся с символа %. Использовать текстовые комментарии можно как в командной строке, так и в тексте программы. Строка после символа % не воспринимается как команда, и нажатие клавиши ENTER приводит к активации следующей командной строки.

Примеры использования текстовых комментариев приведены в следующем разделе.

1.3 ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННОГО ЧИСЛА

Числовые результаты могут быть представлены с плавающей (например, - 3.2E-6, 6.42E+2), или с фиксированной точкой (например, 4.12, -6.05, 17.5489). Числа в формате с плавающей точкой представлены в экспоненциальной форме mE±p, где m - мантисса (целое или дробное число с десятичной точкой), p _ порядок (целое число).

Для того, чтобы перевести число в экспоненциальной форме к обычному представлению с фиксированной точкой, необходимо умножить мантиссу (m) на десять в степени порядок (p). Например,

$6.42E+2=6.42*10^2 = 642$

$-3.2E - 6 = -3.2 \cdot 10-6 = -0.0000032$

Ниже приведён пример ввода вещественного числа.

```
>>0.987654321
ans=0.98765
```

Нетрудно заметить, что число знаков в дробной части числа в строке ввода больше чем в строке вывода. Происходит это потому, что результат вычислений выводится в виде, который определяется предварительно установленным форматом представления чисел. Команда, с помощью которой можно установить формат числа имеет вид:

format формат числа;

В Octave предусмотрены следующие форматы чисел:

• Short - краткая запись, применяется по умолчанию;
• Long - длинная запись;

```
>> format short
>> pi
ans=3.1416
>>format long
>>pi
ans=3.14159265358979
```

• Short E (Short e) - краткая запись в формате с плавающей точкой;
• Long E (Long e) - длинная запись в формате с плавающей точкой;

```
>> format short E
>> pi
ans = 3.1416E+00
>> format long E
>> pi
ans = 3.14159265358979E+00
```

• Short G (Short g) = вторая форма краткой записи в формате с плавающей точкой;
• Long G (Long g) - вторая форма длинной записи в формате с плавающей точкой;

```
>> format short G
>> pi
ans=3.1416
>> format long G
>> pi
ans=3.14159265358979
```

• Free _ запись без форматирования, чаще всего этот формат применяют для представления комплексного числа (подробно о комплексных числах см. п. 5.2);

```
>> format short
>> 3.1234+2.9876*i
ans = 3.1234 + 2.9876 i
>> format free
>>> 3.1234+2.9876*i
ans=(3.123, 2.988)
```

• Compact _ запись в формате, не превышающем шесть позиций, включая десятичную точку, если целая часть числа превышает четыре знака, число будет записано в экспоненциальной форме.

```
>> format compact
>> 123.123456
ans=123.12
>> 1234.12345
ans=1234.1
>> 12345.123
ans=1.2345 e+04
```

Формат Short установлен по умолчанию. Вызов команды format с другим числовым форматом означает, что теперь вывод чисел будет осуществляться в установленном формате.

1.4 ПЕРЕМЕННЫЕ

В Octave можно определять переменные и использовать их в выражениях. Для определения переменной необходимо набрать имя переменной, символ «=» и значение переменной. Здесь знак равенства - это оператор присваивания, действие которого не отличается от аналогичных операторов языков программирования.

Имя переменной не должно совпадать с именами встроенных процедур, функций и встроенных переменных системы. Система различает большие и малые буквы в именах переменных. А именно: ABC, abc, Abc, aBc - это имена разных переменных.

Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или символьным выражением. Если речь идёт о символьной или строковой переменной, то выражение в правой части оператора присваивания следует брать в одинарные кавычки.

```
>> x=1.2 % Определение переменной
x = 1.2000
>> y=3.14
y = 3.1400
>> z=x+y % Использование переменных в арифметическом
выражении
z = 4.3400
% Определение строковой переменной
```

```
>> str=’Посадил дед репку. Выросла репка большая,
пребольшая’
str = Посадил дед репку. Выросла репка большая, пребольшая
```

Ниже приведены примеры присвоения значений переменным:

```
>> a=1;b=2; c=a*b;d=c^2
d = 4
```

Поскольку символ «;» в конце выражения отсутствует, то в качестве результата выводится имя переменной и её значение. Наличие символа «;», как уже указывалось, передаёт управление следующей командной строке. Это позволяет использовать имена переменных для записи промежуточных результатов в память компьютера.

Ниже приведён листинг, который содержит пример использования в выражении не определённой ранее переменной. При этом выводится на экран сообщение об ошибке.

```
% Сообщение об ошибке. Переменная не определена.
>> t/(a+b)
error: ’t’ unde fined near line 37 column 1
```

Если команда не содержит знака присваивания, то по умолчанию вычисленное значение присваивается специальной системной переменной «ans». Причём полученное значение можно использовать в последующих вычислениях, но важно помнить, что значение ans изменяется после каждого вызова команды без оператора присваивания.

Примеры использования системной переменной ans:

```
>> 25−3
ans=22 % Значение системной переменной равно 22
>>2*ans
ans=44 % Значение системной переменной равно 44
>> x=ans^0.5
x = 6.6332
>> ans % Значение системной переменной не изменилось и
равно 44
ans = 44
```

Кроме переменной ans в Octave существуют и другие системные переменные:

• ans - результат последней операции без знака присваивания;

• i, j - мнимая единица ($\sqrt{-1}$);

• pi - число π (3.141592653589793);
• e - число Эйлера (e = 2.71828183);
• inf - машинный символ бесконечности (∞);
• NaN - неопределённый результат ($\frac{0}{0}, \frac{\infty}{\infty}, 1^{\infty}$ т.п.);
• realmin - наименьшее число с плавающей точкой (2.2251e−308);
• realmax _ наибольшее число с плавающей точкой (1.7977e+308).

Все перечисленные переменные можно использовать в математических выражениях.

Если речь идёт об уничтожении определения одной или нескольких переменных, то можно применить команду: clear *имя_переменной*.

Пример применения команды clear:

```
>> a=5.2; b=a^2;
>> a, b
a = 5.2000
b = 27.040
>> clear a
>> a, b
error: ’a’ unde fined near line 126 column 1
>> b
b = 27.040
```

1.5 ФУНКЦИИ

Все функции, используемые в Octave, можно разделить на два класса встроенные и определённые пользователем. В общем виде обращение к функции в Octave имеет вид:

имя переменной = имя функции(аргумент)

или

имя функции(аргумент)

Если имя переменной указано, то ей будет присвоен результат работы функции. Если же оно отсутствует, то значение вычисленного функцией результата присваивается системной переменной ans.

Например:

```
>> x=pi/2; % Определение значения аргумента
>> y=sin(x)% Вызов функции
```

y=1
>> cos(pi/3) % Вызов функции
ans=0.50000

Рассмотрим элементарные встроенные функции Octave. С остальными будем знакомиться по мере изучения материала.

1.5.1 Элементарные математические функции

Ниже приведены элементарные математические функции Octave.

sin(x) синус числа x
cos(x) косинус числа x
tan(x) тангенс числа x
cot(x) котангенс числа x
sec(x) секанс числа x
csc(x) косеканс числа x
asin(x) арксинус числа x
acos(x) арккосинус числа x
atan(x) арктангенс числа x
acot(x) арккотангенс числа x
asec(x) арксеканс числа x
acsc(x) арккосеканс числа x

Примеры работы с тригонометрическими функциями:

```
>> x=pi/7
x=0.44880
>> sin(x)
ans=0.43388
>>(1−cos(x)^2)^0.5
ans=0.43388
>> tan(x)/(1+tan(x)^2)^0.5
ans = 0.43388
>> (sec(x)^2−1)^0.5/sec(x)
ans = 0.43388
>> 1/csc (x)
ans=0.43388
>> asin(x)
ans=0.46542
>> acos((1−x^2)^0.5)
ans=0.46542
```

```
>> atan(x/((1−x^2)^0.5))
ans=0.46542
```

Экспоненциальные функции

exp(x) Экспонента числа x
log(x) Натуральный логарифм числа x
log10(x) Десятичный логарифм числа x
Применение экспоненциальных функций:

```
>> x=1
x = 1
>> exp(x)
ans = 2.7183
>> log(x)
ans=0
>> log(e^2)
ans=2
```

Гиперболические функции

sinh(x) гиперболический синус числа x
cosh(x) гиперболический косинус числа x
tanh(x) гиперболический тангенс числа x
coth(x) гиперболический котангенс числа x
sech(x) гиперболический секанс числа x
csch(x) гиперболический косеканс числа x

Листинг ниже содержит примеры работы с гиперболическими
функциями.

```
>> cosh(x)^2−sinh(x)^2
ans = 1
>> tanh(x)∗coth(x)
ans = 1
```

Целочисленные функции

fix(x) округление числа x до ближайшего целого в сторону нуля
floor(x) округление числа x до ближайшего целого в сторону отрицательной бесконечности

ceil(x) округление числа x до ближайшего целого в сторону положительной бесконечности
round(x) обычное округление числа x до ближайшего целого

Примеры работы с тригонометрическими функциями:

```
>> pi
ans = 3.1416
>> fix(pi)
ans = 3
>> floor (pi)
ans = 3
>> floor(-pi)
ans = −4
>> ceil(pi)
ans = 4
>> ceil(-pi)
ans = -3
>> round(pi)
ans = 3
>> pi/2
ans = 1.5708
>>round(pi/2)
ans = 2
```

Другие элементарные функции

sqrt(x) корень квадратный из числа x
abs(x) модуль числа x
log10(x) десятичный логарифм от числа x
log2(x) логарифм по основанию два от числа x
pow2(x) возведение двойки в степень x
rats(x) представление числа x в виде рациональной дроби

Примеры работы с приведёнными функциям.

```
>> x=9;
>> sqrt(x)
ans = 3
>> abs(-x)
ans = 9
```

```
>> abs(x)
ans = 9
>> x=10;
>> log10(x)
ans = 1
>> log10(10∗x)
ans = 2
>> x=4;
>> log2(x)
ans = 2
>> pow2(x)
ans = 16
```

1.5.2 Комплексные числа. Функции комплексного аргумента

Реализация комплексной арифметики в Octave. Как было отмечено ранее, для обозначения мнимой единицы зарезервировано два имени - i, j, поэтому ввод комплексного числа производится в формате: действительная часть + i * мнимая часть или действительная часть + j * мнимая часть.

Пример ввода и вывода комплексного числа:

```
>> 3+i∗5
ans = 3+5i
>> -2+3∗i
ans = -2 + 3 i
>> 7+2∗j
ans = 7 + 2i
>> 0+7*i
ans = 0 + 7i
>> 6+0∗j
ans = 6
```

Кроме того, к комплексным числам применимы элементарные арифметические операции: +, −, *, \, /, ^; например:

```
>> a=−5+2i;
>> b=3−5∗i;
>> a+b
ans = −2 − 3 i
>> a−b
```

```
ans = −8 + 7 i
>> a*b
ans = −5 + 31 i
>> a/b
ans = −0.73529 − 0.55882 i
>> a^2+b^2
ans = 5 − 50 i
```

Функции для работы с комплексными числами.
real(Z) выдаёт действительную часть комплексного аргумента Z
imag(Z) выдаёт мнимую часть комплексного аргумента Z

1.6 ОПЕРАЦИИ ОТНОШЕНИЯ

Операции отношения выполняют сравнение двух операндов и определяют, истинно выражение или ложно (таблица 1.1). Результат операции отношения _ логическое значение. В качестве логических значений в Octave используются 1 («истина») и 0 («ложь»).

Таблица 1.1.

Операция	Описание операции
<	меньше
>	больше
==	равно
~=	не равно
<=	меньше или равно
>=	больше или равно

1.7 ЛОГИЧЕСКИЕ ВЫРАЖЕНИЯ

Логическое выражение может быть составлено из операций отношения и логических операций (операторов). Логические выражения выполняются над логическими данными. В таблице 1.2 представлены основные логические выражения: .и., .или., .не.

Таблица 1.2 – Логические операции

A	B	«не» A	A «и» B	A «или» B	A «исключающее или» B
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	0	1	0

В Octave существует возможность представления логических выражений в виде логических операторов и логических операций (таблица 2.9). В таблице 2.9 A и B - логические выражения (или целочисленные значения 1 и 0).

Таблица 1.3. – Виды логических выражений

Тип выражения	Выражение	Логический оператор	Логическая операция
Логическое «и»	`A and B`	`and(A, B)`	`A & B`
Логическое «или»	`A or B`	`or(A, B)`	`A | B`
Исключающее «или»	`A xor B`	`xor(A,B)`	
Отрицание	`not A`	`not (A)`	`~ A`

Логические операторы (выражения) определены и над массивами (матрицами) одинаковой размерности и выполняются поэлементно над элементами. В итоге формируется результирующий массив (матрица) логических значений, каждый элемент которого вычисляется путём выполнения логической операции над соответствующими элементами исходных массивов.

При одновременном использовании в выражении логических и арифметических операций возникает проблема последовательности их выполнения. В Octave принят следующий приоритет операций.

1. Логические операторы.
2. Логическая операция ~.
3. Транспонирование матриц, операции возведения в степень, унарный + и −.
4. Умножение, деление
5. Сложение, вычитание.
6. Операции отношения.
7. Логическая операция «и» - &
8. Логическая операция «или» - |

Ниже представлены примеры использования логических операций над скалярными и матричными значениями.

```
>> A=3; B=4; C=2*pi;
```

```
>> P=~(((A+B)*C)>A^B) | (A+B)==((B−A)+A^2)
P = 1
>> X=[2 1 6 ] ;Y=[1 3 5 ] ;
>> Z=~or((X>2*Y), (X>Y)) | and((X>2*Y), (X<Y))
Z = 0 1 0
```

1.8. ПРОГРАММИРОВАНИЕ

1.8.1 Алгоритмы, их структура и описание

Любому программированию на ЭВМ предшествует процесс разработки алгоритма решения данной конкретной задачи.

Алгоритм - понятное и точное предписание совершить определенную последовательность действий, направленных на достижение указанной цели или решение поставленной задачи.

Алгоритм содержит несколько шагов, которые должны выполняться в определенной последовательности. Каждый шаг алгоритма может состоять из одной или нескольких простых операций.

Для алгоритма важен не только набор действий, но и то, в каком порядке они выполняются. Понятие алгоритма в информатике является фундаментальным. Таким же, какими являются понятия точки, прямой и плоскости в геометрии, вещества в химии, пространства и времени в физике и т.д.

Виды алгоритмов и способы их описания

При составлении блок-схемы алгоритма блоки записываются последовательно друг за другом и соединяются линиями потока информации, которые показывают направление движения по блок-схеме. Каждый блок алгоритма должен иметь вход и выход (исключение составляют блоки начала и конца алгоритма). При этом может быть несколько входящих в блок линий потока информации и только один выходящий поток (исключения составляют логический блок и блок модификации). Несколько линий потока могут объединяться в одну линию, но одна линия потока информации не может разветвляться на несколько потоков. В блок-схеме любой путь движения из блока «Начало» алгоритма должен привести в блок «Конец» алгоритма.

Графические объекты блок-схем

Название блока	Вид блока	Назначение блока
Прерывание		Начало и конец алгоритма
Передача данных		Ввод или вывод информации
Процесс		Арифметический блок, определяющий действие, которое необходимо выполнить
Принятие решения		Логический блок, проверяющий истинность или ложность некоторого условия
Подготовка		Задание массива, начало цикла

Рисунок 1.1 – Основные блок схемы описания программ

В общем случае любой алгоритм может состоять из трех частей: ввод исходных данных, вычисление требуемых величин и вывод полученных результатов. При этом каждый блок в блок-схеме должен быть пронумерован.

Существует три основных типовых структуры алгоритма:

1. Линейный вычислительный процесс.
2. Разветвляющийся вычислительный процесс.
3. Циклический вычислительный процесс.

Любой алгоритм сложной структуры может быть получен путем комбинированного использования типовых структур.

В линейном вычислительном процессе все действия выполняются в строгой последовательности друг за другом. Таким образом, существует только один путь, по которому можно пройти из блока «Начало» в блок «Конец» алгоритма, т.е. выполнить алгоритм.

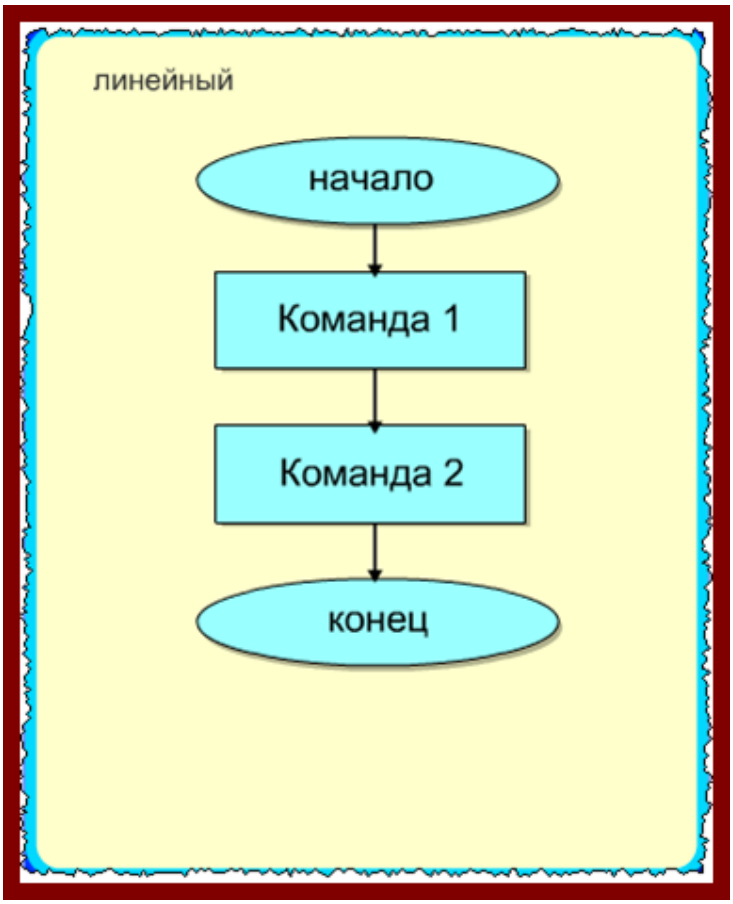


Рисунок 1.2 – Линейный вычислительный процесс

Разветвляющийся алгоритм, содержит хотя бы одну проверку условия, в результате которой обеспечивается переход на один из возможных вариантов решения;

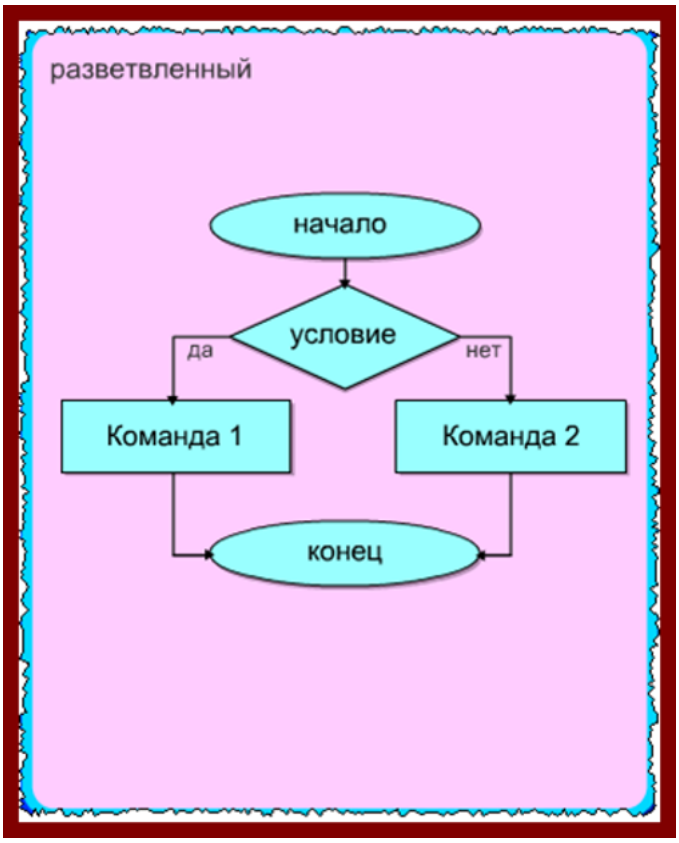


Рисунок 1.3 – Разветвляющийся алгоритм

Циклический алгоритм, предусматривает многократное повторение одной и той же последовательности действий. Количество повторений обусловливается исходными данными или условием задачи.

Рисунок 1.4 – Циклический алгоритм

Любая алгоритмическая конструкция может содержать в себе другую конструкцию того же или иного вида, т. е. алгоритмические конструкции могут быть вложенными.

Рассмотрим следующие способы описания алгоритма: словесное описание, псевдокод, блок-схема, программа.

Словесное описание представляет структуру алгоритма на естественном языке. Например, любой прибор бытовой техники (утюг, электропила, дрель и т.п.) имеет инструкцию по эксплуатации, т.е. словесное описания алгоритма, в соответствии которому данный прибор должен использоваться. Никаких правил составления словесного описания не существует. Запись алгоритма осуществляется в произвольной форме на естественном, например, русском языке. Этот способ описания не имеет широкого распространения, так как строго не формализуем (под «формальным» понимается то, что описание абсолютно полное и учитывает все возможные ситуации, которые могут возникнуть в ходе решения); допускает неоднозначность толкования при описании некоторых действий; страдает многословностью.

Например:

Алгоритм "Погода".

Начало
1. определить температуру воздуха
2. если температура ниже 0, то надеть шубу, иначе надеть куртку
Конец.

Псевдокод - описание структуры алгоритма на естественном, частично формализованном языке, позволяющее выявить основные этапы решения задачи, перед точной его записью на языке программирования. В псевдокоде используются некоторые формальные конструкции и общепринятая математическая символика. Строгих синтаксических правил для записи псевдокода не существует. Это облегчает запись алгоритма при проектировании и позволяет описать алгоритм, используя любой набор команд. Однако в псевдокоде обычно используются некоторые конструкции, присущие формальным языкам, что облегчает переход от псевдокода к записи алгоритма на языке программирования. Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором используемых слов и конструкций.

Блок-схема - описание структуры алгоритма с помощью геометрических фигур с линиями-связями, показывающими порядок выполнения отдельных инструкций. Этот способ имеет ряд преимуществ. Благодаря наглядности, он обеспечивает «читаемость» алгоритма и явно отображает порядок выполнения отдельных команд. В блок-схеме каждой формальной конструкции соответствует определенная геометрическая фигура или связанная линиями совокупность фигур.

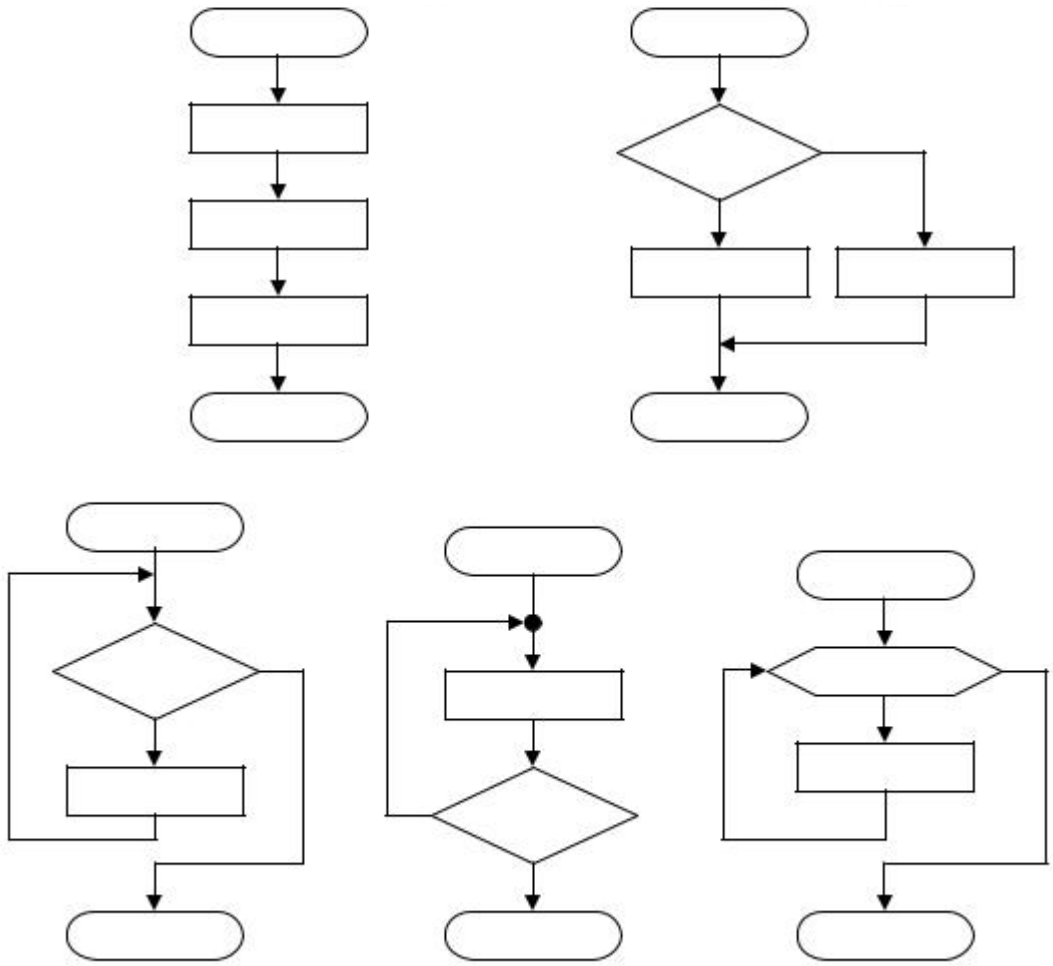


Рисунок 1.5. – Основные алгометрические структуры

Описания алгоритма в словесной форме, на псевдокоде или в виде блок-схемы допускают некоторый произвол при изображении команд. Вместе с тем они настолько достаточны, что позволяет человеку понять суть дела и исполнить алгоритм. На практике исполнителями алгоритмов выступают компьютеры. Поэтому алгоритм, предназначенный для исполнения на компьютере, должен быть записан на «понятном» ему языке, такой формализованный язык называют языком программирования.

Ниже приведена блок – схема решения квадратного уравнения, в которой наглядно представлен алгоритм решения данной задачи. Написание программы решения данной задачи на каком – либо языке программирования, используя приведённый алгоритм не составит труда.

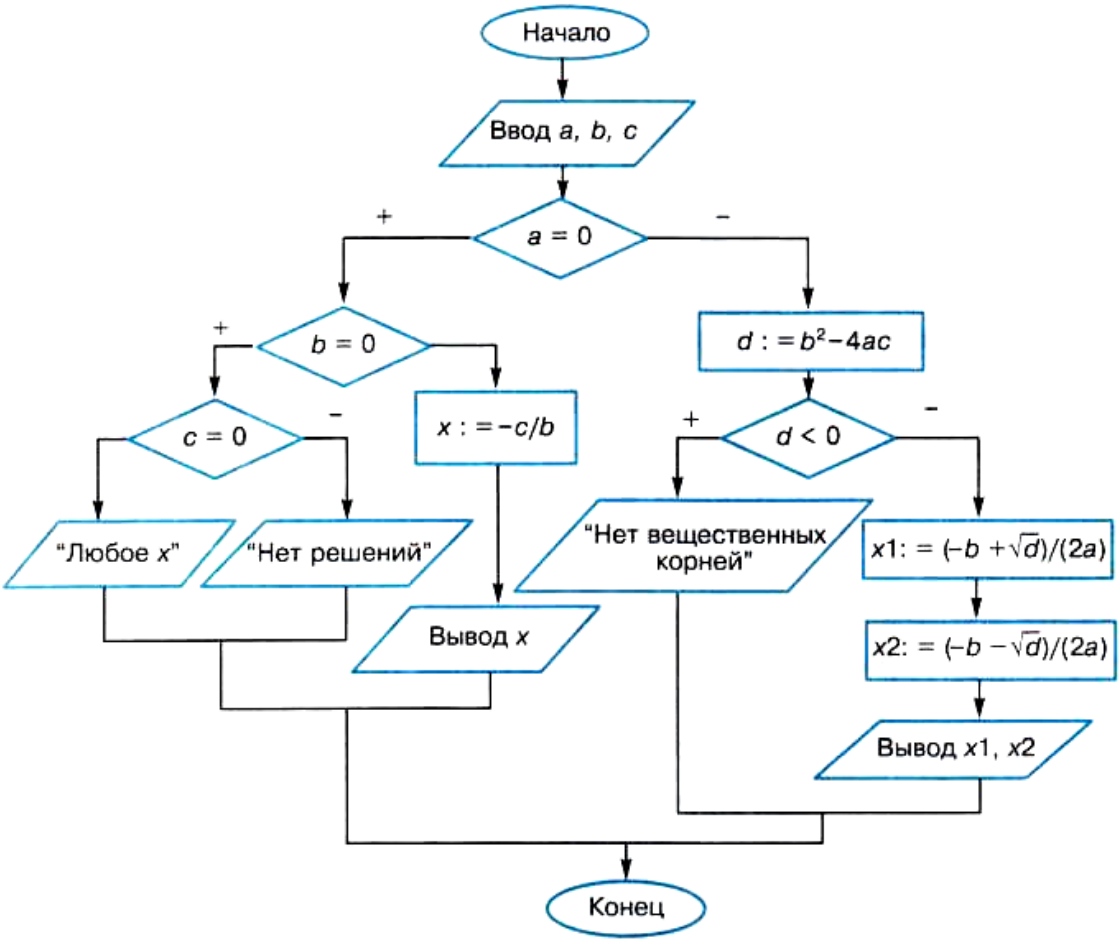


Рисунок 1.6 – Блок – схема алгоритма решения квадратного уравнения

Описание алгоритмов с помощью программ - алгоритм, записанный на языке программирования, называется программой.

Словесная и графическая формы записи алгоритма предназначены для человека. Алгоритм, предназначенный для исполнения на компьютере, записывается на языке программирования (языке, понятном ЭВМ). В нашем случае мы будем пользоваться языком программирования, заложенным в программном продукте Octave, являющимся одним из аналогов программного продукта Matlab.

1.8.2 Основные операторы языка программирования

Рассмотренные ранее группы команд, состоящие из операторов присваивания и обращения к встроенным функциям, представляют

собой простейшие программы Octave. Если такая программа хранится в файле с расширением .m (.M), то для её выполнения достаточно в командной строке Octave ввести имя этого файла (без расширения). В Octave встроен достаточно мощный язык программирования. Ниже приведены основные операторы этого языка и примеры их использования.

3.2.1 Оператор присваивания

Оператор присваивания служит для определения новой переменной (п. 2.4). Для того, чтобы определить новую переменную, достаточно присвоить ей значение:

имя_переменной = значение_выражения

Следует помнить, что любую переменную Octave воспринимает как матрицу. В простейшем случае матрица может состоять из одной строки и одного столбца:

```
>> m=pi
m = 3.1416
>> m
m = 3.1416
>> m(1)
ans = 3.1416
>> m(1,1)
ans = 3.1416
>> m(1,2)
error: A(I): Index exceeds matrix dimension.
>> m(3)
error: A(I): Index exceeds matrix dimension .
>> M=e; M(3,3 )=e/2;
>> M
M =
2.71828 0.00000 0.00000
0.00000 0.00000 0.00000
0.00000 0.00000 1.35914
```

Организация простейшего ввода и вывода в диалоговом режиме

При разработке программ возникает необходимость ввода исходных данных и вывода результатов. Если для вывода результатов на экран можно просто не ставить «;» после оператора, то для ввода

исходных данных при разработке программ, работающих в диалоговом режиме, следует использовать функцию input
имя_переменной = input(’подсказка’);

Если в тексте программы встречается оператор input, то выполнение программы приостанавливается, Octave выводит на экран текст подсказки и переходит в режим ожидания ввода. Пользователь вводит с клавиатуры значение и нажимает клавишу Enter. Введённое пользователем значение будет присвоено переменной, имя которой указано слева от знака присваивания.

Для вывода результатов можно использовать функцию следующей структуры:

disp(’строка_символов’) или disp(имя_переменной)

Пример 1.1 Создать программу для вычисления значения y по формуле y = sin(x), при заданном значении x.
Текст программы и результаты её работы показаны в листинге 2.1.

Листинг 1.1

Решение к примеру 1.1.

```
x=input(’Введите значение x=’); y=sin(x);
disp(’Значение y=’); disp(y);
% Результат работы программы
Введите значение x= pi/4
Значение y= 0.70711
```

3.2.3 Условный оператор

Одним из основных операторов, реализующим ветвление в большинстве языков программирования, является условный оператор. Существует обычная, сокращённая и расширенная формы этого оператора в языке программирования Octave.

Обычный условный оператор имеет вид:

```
if условие
операторы 1
else
операторы 2
end
```

Здесь условие - логическое выражение, операторы 1, операторы 2 - операторы языка или встроенные функции Octave.

Обычный оператор if работает по следующем алгоритму: если условие истинно, то выполняются операторы-1, если ложно - операторы_2.

Пример 1.2. Даны вещественные числа x и y. Определить принадлежит ли точка с координатами (x; y) заштрихованной части плоскости (рис. 1.1).

Как показано на рис. 1.7, фигура на плоскости ограничена линиями x = -1, x = 3, y = -2 и y = 4. Значит точка с координатами (x; y) будет принадлежать этой фигуре, если будут выполняться следующие условия: $x \geq -1$, $x \leq 3$, $y \geq -2$ и $y \leq 4$. Иначе точка лежит за пределами фигуры.

Ниже приведён текст программы и результаты её работы.

Листинг 1.2. Решение к примеру 1.2.

```
x=input('x='); y=input('y=');
if(x>=-1) & (x<=3) & (y>=-2) & (y<=4)
    disp('Точка принадлежит фигуре')
else
    disp('Точка не принадлежит фигуре');
end
% Результаты работы программы
x= 3
y= 3
Точка принадлежит фигуре
%————————————————————————————
x= 4
y= 4
Точка не принадлежит фигуре
```

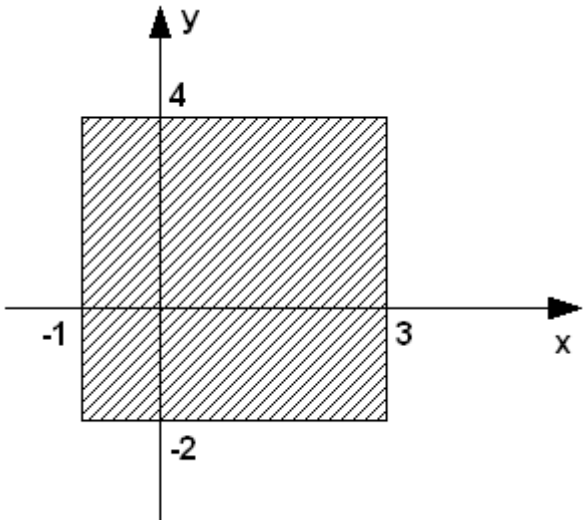


Рисунок 1.7 – Графическое представление задачи из примера 1.2

Сокращённый условный оператор выглядит так:

```
if условие
    операторы
end
```

Работает этот оператор следующим образом. Если условие истинно, то выполняются операторы, в противном случае управление передаётся оператору, следующему за оператором if:

```
z=0;
x=input('x='); y=input('y=');
if(x~=y)
z=x+y;
end;
disp ('Значение Z='); disp(z);
% Результаты работы программы
x= 3
y= 5
Значение Z= 8
x= 3
y= 3
Значение Z= 0
```

Расширенный условный оператор применяют когда одного условия для принятия решения недостаточно:

```
if условие 1
    операторы 1
elseif условие 2
```

```
    операторы 2
elseif условие 3
    операторы 3
    …
elseif условие n
    операторы n
else
    операторы
end
```

Расширенный оператор if работает следующим образом. Если условие 1 истинно, то выполняются операторы 1, иначе проверяется условие 2, если оно истинно, то выполняются операторы 2, иначе проверяется условие 3 и т.д. Если ни одно из условий по веткам elseif не выполняется, то выполняются операторы по ветке else.

Ниже приведён пример использования расширенного условного оператора.

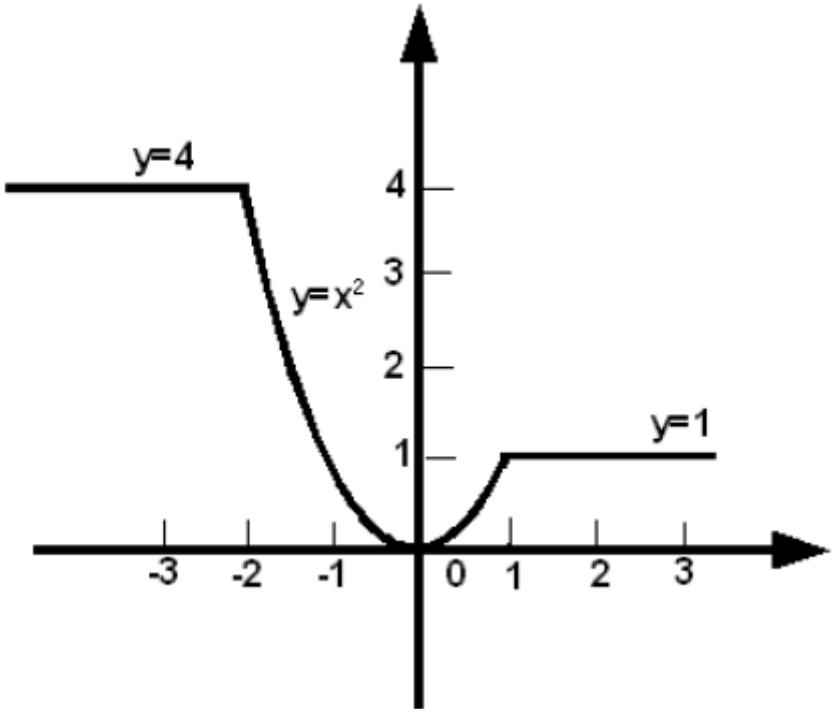
Пример 1.3 Дано вещественное число x. Для функции, график которой приведён на рис. 2.2 вычислить y = f(x).

Аналитически функцию, представленную на рис. 1.8, можно записать так:

$$y(x) = \begin{cases} 4, & x \leqslant -2 \\ 1, & x \geqslant 1 \\ x^2, & -2 < x < 1 \end{cases}$$

Составим словесный алгоритм решения этой задачи:

1. Начало алгоритма.
2. Ввод числа x (аргумент функции).
3. Если значение x меньше либо равно -2, то переход к п. 4, иначе
переход к п. 5.
4. Вычисление значения функции: y = 4, переход к п. 8.
5. Если значение x больше либо равно 1, то переход к п. 6, иначе переход к п. 7.
6. Вычисление значения функции: y = 1, переход к п. 8.
7. Вычисление значения функции: y = x2.
8. Вывод значений аргумента x и функции y.
9. Конец алгоритма.



Рисцнок 1.8 Графическое представление задачи из примера 1.3

Текст программы будет иметь следующий вид:

Листинг 1.3. Решение к примеру 1.3.

```
x=input(’x=’);
if x<=-2
    y=4;
elseif x>=1
    y=1;
else
    y=x^2;
end;
disp (’y=’); disp(y);
% Результаты работы программы
x= 2
y= 1
%————————————————————————
x= −3
y= 4
%————————————————————————
x= 0. 5
y= 0.25000
```

Оператор альтернативного выбора

Ещё одним способом организации разветвлений является оператор альтернативного выбора следующей структуры:

```
switch параметр
        case значение 1
            операторы 1
        case значение 2
            операторы 2
        case значение 3
            операторы 3
        …
          …
        otherwise
            операторы
end
```

Оператор switch работает следующим образом: если значение параметра равно значениие 1, то выполняются операторы1, иначе если параметр равен значениие 2, то выполняются операторы 2. В противном случае, если значение параметра совпадает со значение 3, то выполняются операторы3 и т.д. Если значение параметра не совпадает ни с одним из значений в группах case, то выполняются операторы, которые идут после служебного слова otherwise.

Конечно, любой алгоритм можно запрограммировать без использования switch, используя только if, но использование оператора альтернативного выбора switch делает программу более компактной.

Ниже приведён пример использования оператора switch.

Пример 1.4 Вывести на печать название дня недели, соответствующее заданному числу D, при условии, что в месяце 31 день и первое число - понедельник.

Для решения задачи воспользуемся функцией mod, позволяющей вычислить остаток от деления двух чисел. Если в результате остаток от деления заданного числа D на семь будет равен единице, то это понедельник (по условию первое число - понедельник), двойке - вторник, тройке - среда, и так далее. Следовательно, при построении алгоритма необходимо использовать семь условных операторов. Решение задачи станет значительно проще, если при написании

программы воспользоваться оператором альтернативного выбора (листинг 1.4).

Листинг 1.4 Решение к примеру 1.4.

```
D=input('Введите число от 1 до 31');
% Вычисление остатка от деления D на 7, сравнение его с
числами от 0 до 6.
switch mod(D,7)
        case 1
             disp('ПОНЕДЕЛЬНИК')
       case 2
           disp('ВТОРНИК')
       case 3
           disp('СРЕДА')
        case 4
            disp('ЧЕТВЕРГ')
        case 5
            disp('ПЯТНИЦА')
        case 6
            disp('СУББОТА')
        otherwise
             disp('ВОСКРЕСЕНЬЕ')
end
```

Условный циклический оператор

Оператор цикла с предусловием в языке программирования Octave имеет вид:

```
while выражение
      операторы
end
```

Работает цикл с предусловием следующим образом. Вычисляется значение условия выражение. Если оно истинно, выполняются операторы.

В противном случае цикл заканчивается, и управление передаётся оператору, следующему за телом цикла. Выражение вычисляется перед каждой итерацией цикла. Если при первой проверке выражение ложно, цикл не выполнится ни разу. Выражение должно быть переменной или логическим выражением.

Пример 1.5 Дано натуральное число N. Определить количество цифр в числе.

Для того, чтобы подсчитать количество цифр в числе, необходимо определить, сколько раз заданное число можно разделить на десять нацело. Например, пусть N = 12345, тогда количество цифр kol = 5.

Результаты вычислений сведены в таблицу 1.4.

Текст программы, реализующей данную задачу, можно представить следующим образом:

Листинг 1.5 Решение к примеру 1.5.

```
N=input('N=');
M=N; % Сохранить значение переменной N.
kol=1; % Число содержит хотя бы одну цифру.
while round(M/10)> 0 % Выполнять тело цикла, пока частное от деления
% M на 10, округлённое до целого, больше 0.
    kol=kol+1; % Счётчик количества цифр
    M=round(M/10); % Изменение числа.
end
disp('kol='); disp(kol);
% Результат работы программы
N= 12345678
kol= 8
```

Таблица 1.4. Определение количества цифр числа

kol	N
1	12345
2	12345 div10=1234
3	1234 div 10=123
4	123 div 10=12
5	12 div 10=1
5	1 div 10=0

Оператор цикла с известным числом повторений

Для записи цикла с известным числом повторений применяют оператор for

```
for параметр = начальное значение: шаг: конечное значение
    операторы
end
```

Выполнение цикла начинается с присвоения параметру цикла начального_значения. Затем следует проверка, не превосходит ли параметр цикла конечное значение. Если результат проверки утвердительный, цикл считается завершённым, и управление передаётся следующему за телом цикла оператору. В противном случае выполняются операторы в цикле. Далее параметр увеличивает своё значение на значение шага и снова производится проверка - не превзошло ли значение параметра цикла конечное значение. В случае положительного ответа алгоритм повторяется, в противном - цикл завершается.

Если шаг цикла равен 1, то оператор записывают так:

```
for параметр = начальное значение: конечное значение
    операторы
end
```

Пример 1.6. Дано натуральное число N. Определить K - количество делителей этого числа, не превышающих его. Например, для N = 12 делители 1, 2, 3, 4, 6. Количество делителей K = 5.

Для решения поставленной задачи нужно реализовать следующий алгоритм: в переменную K, предназначенную для подсчёта количества делителей заданного числа, поместить значение, которое не влияло бы на результат, т.е. нуль. Далее организовать цикл, в котором изменяющийся параметр i выполняет роль возможных делителей числа N. Если заданное число делится нацело на параметр цикла, это означает, что i является делителем N, и значение переменной K следует увеличить на единицу. Цикл необходимо повторить только N/2 раз[1].

Листинг 1.6. Количество делителей числа (к примеру 1.6).

```
N=input('N=');
K=0; % Количество делителей числа
for i=1: N/2
   if mod(N,i) == 0 % Если N делится нацело на i, то
         K=K+1; % увеличить счётчик на единицу.
   end
end
disp ('K = '); disp (K);
% Результат работы программы
N = 12
K = 5
```

Операторы передачи управления

Операторы передачи управления принудительно изменяют порядок выполнения команд. В языке программирования Octave таких операторов два. Операторы break и continue используют только внутри циклов. Так оператор break осуществляет немедленный выход из циклов while, for и управление передаётся оператору, находящемуся непосредственно за циклом. Оператор continue начинает новую итерацию цикла, даже если предыдущая не была завершена.

[1]Если делитель числа меньше самого числа, значит он не превосходит его половины (Прим. редактора).

Пример 1.7. Дано натуральное число N. Определить, является ли оно простым. Натуральное число N называется простым, если оно делится нацело без остатка только на единицу и N. Число 13 - простое, так как делится только на 1 и 13, 12 не является простым, так как делится на 1, 2, 3, 4, 6 и 12.

Алгоритм решения этой задачи заключается в том, что число N делится на параметр цикла i, изменяющийся в диапазоне от 2 до N/2. Если среди значений параметра не найдётся ни одного числа, делящего заданное число нацело, то N - простое число, иначе оно таковым не является. Разумно предусмотреть в программе два выхода из цикла. Первый - естественный, при исчерпании всех значений параметра, а второй - досрочный, с помощью оператора break. Нет смысла продолжать цикл, если будет найден хотя бы один делитель из указанной области изменения параметра.

Текст программы приведён в листинге 1.7.

Листинг 1.7. Является ли число простым (к примеру 1.7)?

```
N=input('Введите число');
pr=1; % Предполагаем, что число N является простым (pr=1).
for i=2:N/2 % Перебираем все возможные делители числа N от 2
до N/2.
    if mod(N,i)==0 % Если N делится на i,
        pr=0; % то число N не является простым (pr=0)
        break; % и прерывается выполнение цикла .
    end
end
if pr==1 % Если pr равно 1, то N - простое число.
    disp('ПРОСТОЕ ЧИСЛО')
```

```
else % Если pr равно 0, то N - не является простым.
    disp('НЕ ЯВЛЯЕТСЯ ПРОСТЫМ ЧИСЛОМ')
end
% Результаты работы программы
% Вводим сначала число 12, затем 13
Введите число 12
НЕ ЯВЛЯЕТСЯ ПРОСТЫМ ЧИСЛОМ
%————————————————————————————
Введите число 13
ПРОСТОЕ ЧИСЛО
```

1.8.3 Функции, определённые пользователем

Функция, как и программа, предназначена для неоднократного использования, но она имеет входные параметры и не выполняется без их предварительного задания. Функция имеет заголовок вида

function name1[, name2, ...]=fun(var1[,var2, ...]),

где name1[,name2, ...] - список выходных параметров, то есть переменных, которым будет присвоен конечный результат вычислений, fun - имя функции, var1[, var2, ...] - входные параметры. Таким образом простейший заголовок функции выглядит так:

function name = fun(var)

Все имена переменных внутри функции, а также имена из списка входных и выходных параметров воспринимаются системой как локальные, то есть эти переменные считаются определёнными только внутри функции.

Программы и функции в Octave могут быть созданы при помощи текстового редактора и сохранены в виде файла с расширением .m или .M. Но при создании и сохранении функции следует помнить, что её имя должно совпадать с именем файла.

Вызов программ в Octave осуществляется из командной строки. Программу можно запустить на выполнение, указав имя файла, в котором она сохранена. Обращение к функции осуществляется так же, как и к любой другой встроенной функции системы, то есть с указанием входных и выходных параметров. Вы можете вызвать функцию из командной строки или использовать её как один из операторов программы.

Пример 1.8 Создать функцию для решения кубического уравнения.

Кубическое уравнение

$$ax^3 + bx^2 + cx + d = 0 \qquad (1.1)$$

после деления на а принимает канонический вид:

$$x^3 + rx^2 + sx + t = 0 \qquad (1.2)$$

где $r = \frac{b}{a}$; $s = \frac{c}{a}$; $t = \frac{d}{a}$.

В уравнении (1.2) сделаем замену $x = y - \frac{r}{3}$ и получим следующее приведённое уравнение:

$$y^3 + py + q = 0 \qquad (1.3)$$

где $p = \frac{3s - r^2}{3}$; $q = \frac{2r^3}{27} - \frac{rs}{3} + t.$

Число действительных корней приведённого уравнения (1.2) зависит от знака дискриминанта $D = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^3$ (табл. 1.5).

Таблица 1.5 – Количество корней кубического уравнения

Дискриминант	Количество действительных корней	Количество комплексных корней
D≥0	1	2
D<0	3	-

Корни приведённого уравнения могут быть рассчитаны по формулам Кардано:

$$y_1 = u + \upsilon,\ y_2 = -\frac{u+\upsilon}{2} + \frac{u-\upsilon}{2} i\sqrt{3},$$
$$y_3 = -\frac{u+\upsilon}{2} - \frac{u-\upsilon}{2} i\sqrt{3} \qquad (1.4)$$

где

$$u = \sqrt[3]{\frac{-q}{2} + \sqrt{D}},\ \upsilon = \sqrt[3]{\frac{-q}{2} - \sqrt{D}}\ .$$

Ниже приведён листинг программы, реализующий представленный способ решения кубического уравнения:

Листинг 1.8. Нахождение корней кубического уравнения (пример 1.8).

```
function[x1,x2, x3]=cub(a,b,c,d)
r=b/a; s=c/a; t=d/a;
```

```
p=(3*s-r ^2)/3;
q=2*r^3/27-r*s/3+t;
D=(p/3)^3+(q/2)^2;
u=(-q/2+sqrt(D))^(1/3);
v=(-q/2−sqrt (D))^(1/3);
y1=u+v;
y2=-(u+v)/2+(u-v)/2*i*sqrt(3);
y3=-(u+v)/2-(u-v)/2*i*sqrt(3);
x1=y1-r/3;
x2=y2-r/3;
x3=y3-r/3;
end (endfunction)
% Вычисляем корни уравнения 3x3 − 2x2 − x − 4 = 0
[x1,x2,x3]=cub(3,-2,-1,-4)
x1 = 1.4905
x2 = −0.41191 + 0.85141 i
x3 = −0.41191 − 0.85141 i
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №1

Задание 1.1

Рассчитать в соответствии с заданием цикл двигателя внутреннего сгорания и построить его в T,s и P,v – диаграммах.

Порядок выполнения:

1. Рассчитать параметры (p, v, T) в узловых точках цикла и заполнить таблицу №1 параметров цикла.
2. Рассчитать работы расширения и сжатия цикла, подведённую и отведённую в цикле теплоту и термический коэффициент полезного действия.
3. Отобразить цикл графически в P,v и– T,s диаграммах.
4. Представить отчёт по результатам работы.

Расчёты и графические работы выполнить в программном продукте Octave.

В отчёте привести:

1. Графический алгоритм решения задачи.
2. Листинг программы решения задачи в программном комплексе Octave.

3. Графическое отображение цикла, полученное в программном комплексе Octave.

Таблица № 3.1. – Параметры цикла

№ точки	P	V	T
1			
2			
3			
4			

Варианты задания.

1. Цикл двигателя внутреннего сгорания с подводом теплоты при V=const. Начальные параметры P_1 = 100 кПа; t_1 = 20 °С.

Варианты

№	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ε	**3,6**	**4**	**5**	**6**	**7**	**7,5**	**8**	**8,5**	**9**	**4,3**	**5,5**	**6,5**
λ	**3,33**	**3**	**4**	**4**	**4,5**	**4**	**4,5**	**5**	**5**	**3,5**	**4,5**	**4,5**

2. Цикл двигателя внутреннего сгорания с подводом теплоты при P=const. Начальные параметры P_1 = 100 кПа; t_1 = 20 °С.

Варианты

№	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ε	**6**	**7**	**8**	**8,5**	**9**	**9,5**	**10**	**10,5**	**11**	**7**	**8**	**9**
ρ	**1,5**	**1,6**	**1,7**	**1,8**	**1,85**	**1,85**	**1,9**	**1,95**	**2**	**2**	**1,5**	**1,6**

Теплоёмкости и показатель адиабаты считать постоянными

$$C_V = 0,723\frac{кДж}{кг \cdot К}; \quad C_P = 1,012\frac{кДж}{кг \cdot К}; \quad К = 1,4.$$

Задание 1.2

Написать листинг программы в соответствии с блок схемой решения квадратного уравнения (рис. 1.6)

Задание 1.3

Найти значения синуса в интервале от 0 градусов до 360 градусов с интервалом 10 градусов. Написать листинг программы.

Задание 1.4

Ввести случайным образом число от 0 до 100 и определить: сколько раз оно на цело делится пополам.

ЛАБОРАТОРНАЯ РАБОТА № 2.
ПОСТРОЕНИЕ ГРАФИКОВ

Цель работы: научится строить графики с помощью программного продукта Octave. Первый раздел посвящён работе с двумерными графиками. Во втором разделе рассмотрено создание различных трёхмерных графиков. Затем описаны анимационные возможности GNU Octave. Последний параграф посвящён описанию графических объектов.

2.1. ПОСТРОЕНИЕ ДВУМЕРНЫХ ГРАФИКОВ

Двумерным будем считать такой график, в котором положение точки определяется двумя величинами. Двумерные графики наиболее часто строят в декартовой и полярной системах координат.

2.1.1. Построение графиков в декартовой системе координат

Декартова, или прямоугольная система координат задаётся двумя перпендикулярными прямыми, называемыми осями координат. Горизонтальная прямая X ось абсцисс, а вертикальная Y ось ординат. Точку пересечения осей называют началом координат. Четыре угла, образованные осями координат, носят название координатных углов. Положение точки в прямоугольной системе координат определяется значением двух величин, называемых координатами точки. Если точка имеет координаты x и y, то x абсцисса точки, y ордината. Уравнение, связывающее координаты x и y является (называется) уравнением линии, если координаты любой точки этой линии удовлетворяют ему.

Величина y называется функцией переменной величины x, если каждому из тех значений, которые может принимать x, соответствует одно или несколько определённых значений y. При этом переменная величина x называется аргументом функции y = f (x). Говорят также, что величина y зависит от величины x. Функция считается заданной, если для каждого значения аргумента существует соответствующее значение функции. Чаще всего используют следующие способы задания функций:

табличный числовые значения функции уже заданы и занесе-ны в таблицу; недостаток заключается в том, что таблица может не содержать все нужные значения функции;

графический значения функции заданы при помощи линии (графика), у которой абсциссы изображают значения аргумента, а ординаты соответствующие значения функции;

аналитический функция задаётся одной или несколькими формулами (уравнениями); при этом, если зависимость между x и y выражена уравнением, разрешённым относительно y, то говорят о явно заданной функции, в противном случае функция считается неявной.

Совокупность всех значений, которые может принимать в условиях поставленной задачи аргумент x функции y = f(x), называется областью определения этой функции. Совокупность значений y, которые принимает функция f(x), называется множеством значений функции.

Далее будем рассматривать построение графиков в прямоугольной системе координат на конкретных примерах.

Пример 2.1. Построить график функции y = sin(x) + sin(3x)/3 + sin(5x)/5 на интервале [−10; 10].

Для того, чтобы построить график функции f(x) необходимо сформировать два массива x и y одинаковой размерности, а затем обратиться к функции plot.

Решение этой задачи представлено в листинге 2.1.

Листинг 2.1. Построение графика (пример 2.1).

```
x=−10:0.1:10; % Формирование массива x.
y=sin(x)+sin(3∗x)/3+sin(5∗x)/5; % Формирование массива y
plot (x, y) % Построение графика функции.
```

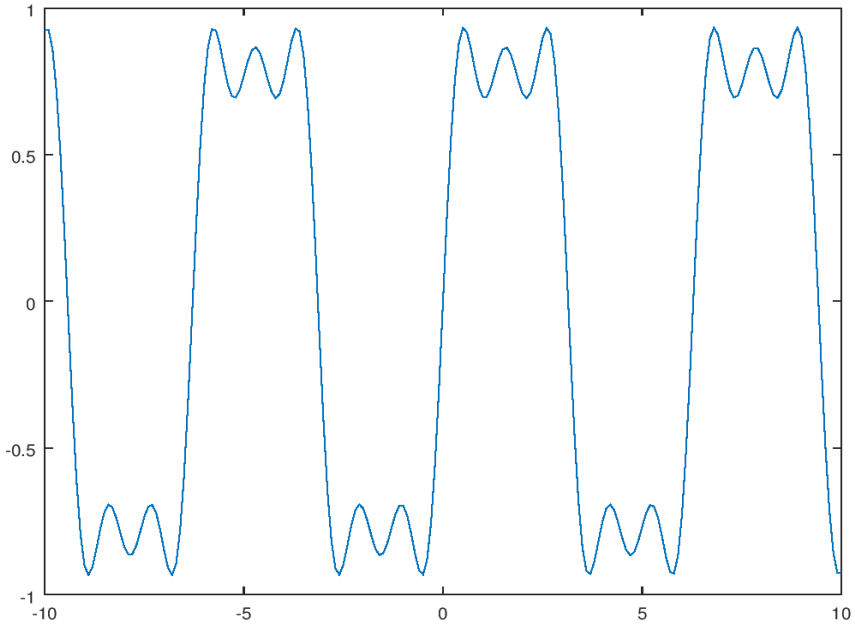


Рисунок 2.1 График функции y=sin(x)+sin(3x)/3+sin(5x)/5

В результате обращения к функции plot(x,y) будет создано окно с

именем Figure 1, в котором будет построен график функции y = sin(x) + 1/3(sin(3x)) + 1/5(sin(5x)) (рис. 2.1).

График формируется путём соединения соседних точек прямыми линиями. Чем больше будет интервал между соседними точками (чем меньше будет точек), тем больше будет заметно, что график представляет из себя ломанную линию.

Если повторно обратиться к функции plot, то в этом же окне будет стёрт первый график и нарисован второй. Для построения нескольких графиков в одной системе координат можно поступить одним из следующих способов:

Обратиться к функции plot следующим образом plot(x1, y1, x2, y2, . . . , xn, yn), где x1, y1 массив абсцисс и ординат первого графика, x2, y2 массив абсцисс и ординат второго графика, . . . , xn, yn массив абсцисс и ординат n-ого графика.

Каждый график изображать с помощью функции plot(x, y), но перед обращением к функциям plot(x2, y2), plot(x3, y3), . . . , plot(xn, yn) вызвать команду hold on[1], которая блокирует режим очистки окна.

Рассмотрим построение нескольких графиков этими способами на примере решения следующей задачи.

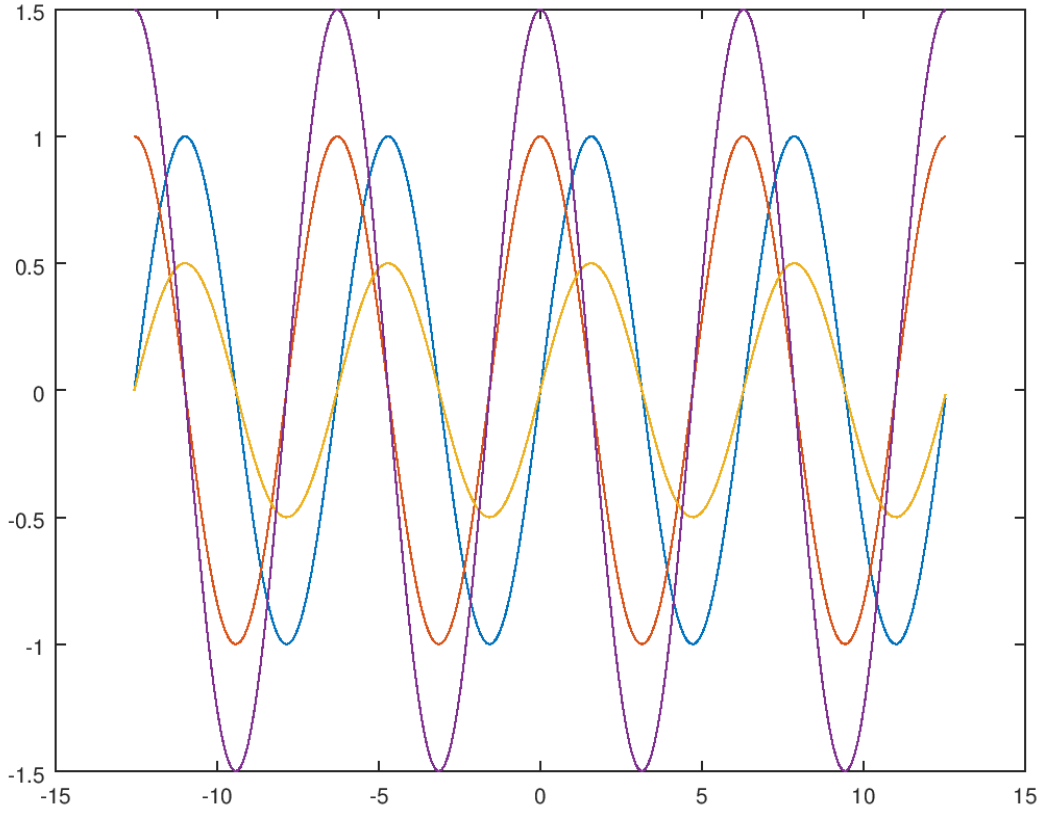


Рисунок 2.2 – Графики функций v = sin(x), w = cos(x), r = sin(x2) , p = 32cos(x)

Пример 2.2. Построить графики функций v = sin x, w = cos x, r = 32 cos x на интервале [−4π; 4π].

Построение графиков функций первым и вторым способом представлено в листинге 2.2. Получившиеся графики функций представлены на рис. 2.2.

Листинг 2.2. Два способа построения графика (пример 1.2).

Способ первый

```
x=−4∗pi:0.1:4∗pi;
v=sin(x); w=cos(x); r=sin(x)/2; p=1.5∗cos (x);
plot(x,v, x,w, x,r, x,p);
```

Способ второй

```
x=−4∗pi: 0.1: 4∗pi;
v=sin(x); plot(x ,v);
hold  on;
w=cos(x); plot(x, w);
r=sin(x)/2; plot(x, r);
p =1.5∗cos(x); plot(x, p);
```

При построении графиков первым способом Octave автоматически изменяет цвета изображаемых в одной системе координат графиков. Однако управлять цветом и видом каждого из изображаемых графиков может и пользователь, для чего необходимо воспользоваться полной формой функции plot: plot(x1, y1, s1, x2, y2, s2, . . . , xn, yn, sn), где x1, x2, . . . , xn - массивы абсцисс графиков; y1, y2, . . . , yn - массивы ординат графиков; s1, s2, . . . , sn - строка форматов, определяющая параметры линии и при необходимости, позволяющая вывести легенду.

В строке могут участвовать символы, отвечающие за тип линии, маркер, его размер, цвет линии и вывод легенды. Попробуем разобраться с этими символами. За сплошную линию отвечает символ « - ». За маркеры отвечают следующие символы (табл. 2.1).

Цвет линии определяется буквой латинского алфавита (табл. 2.2), можно использовать и цифры, но на взгляд авторов использование букв более логично (их легче запомнить по английским названиям цветов).

Таблица 2.1. Символы маркеров

Символ маркера	Изображение маркера
.	точка
*	✳
x	×
+	+
o	⊙
s	■
d	◆
v	▼
^	▲
<	▽
>	△
p	⊡
h	◇

Таблица 2.2. Цвета линии

Символ	Цвет линии
y	желтый
m	розовый
c	голубой
r	красный
g	зеленый
b	синий
w	белый

При определении строки, отвечающей за вывод линии, следует учитывать следующее:

не важен порядок символа цвета и символа маркера;

если присутствует символ «-», то линия всегда будет сплошная, при этом, если присутствует символ маркера, то все изображаемые точки ещё будут помечаться маркером, если символа маркера нет, то соседние точки просто будут соединяться линиями;

если символ маркера «-» отсутствует, то линия может быть, как сплошная, так и точечная; это зависит от наличия символа маркера, если символа маркера нет, то будет сплошная линия, иначе точечная.

Если необходима легенда для графика, то её следует включить в строку форматов, заключённую в символы «;». Например, команда plot(x = −pi: 0.1: pi, sin(x),” − k; sin(x);”) выведет на экран график функции y = sin(x) чёрного цвета на интервале [−π; π] с легендой «sin(x)» (см. рис. 2.3)

Пользователь может управлять и величиной маркера, для этого после строки форматов следует указать имя параметра markersize (размер маркера) и через запятую величину целое число, определяющее размер маркера на графике. Например, команда plot(x=-pi:0.1:pi, sin(x), ”-ok; sin(x);”, ”markersize”, 4); выведет на экран график, представленный на рис. 2.3.

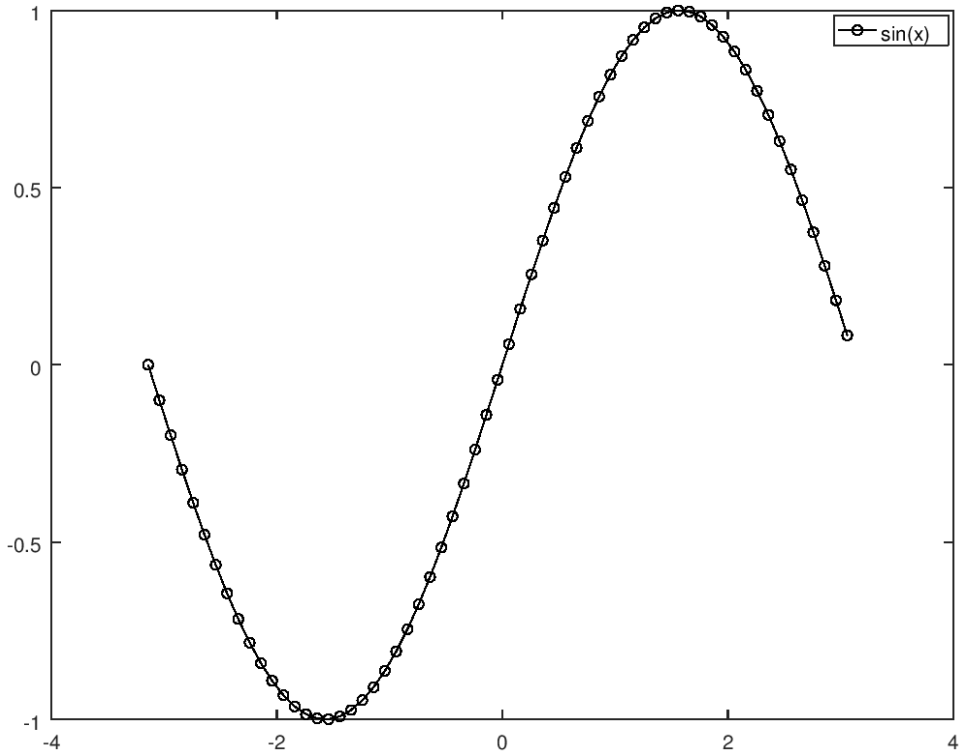


Рисунок 2.3 – Результаты работы функции
plot(x=−pi: 0.1: pi, sin(x),”− k; sin(x);”, ”markersize”, 4);

Для того, чтобы чтобы вывести график в новом окне, перед функцией plot, следует вызвать функцию figure().

Внимание! При работе с графиками в Octave необходимо понимать следующее: щелчок по кнопке закрытия окна с графиками приводит не к уничтожению (закрытию) окна, а к его скрытию. При повторном вызове команды рисования графиков происходит восстановление окна, в котором и изображаются графики. Корректное закрытие графического окна можно осуществить в Octave только программно.

Графическое окно создаётся функцией figure: h =figure(). Здесь h переменная, в которой будет храниться дескриптор (номер) окна. Для

дальнейших операций с окном надо будет использовать именно переменную, в которой хранится дескриптор. Уничтожение (закрытие) окна осуществляется с помощью функции delete(h), где h имя дескриптора закрываемого окна.

В Octave есть функция pause(n), которая приостанавливает выполнение программы на n секунд. Её логично вставлять перед функцией закрытия окна.

Octave представляет дополнительные возможности для оформления графиков:

команда grid on наносит сетку на график, grid off убирает сетку с графика;

функция axis[xmin, xmax, ymin, ymax] выводит только часть графика, определяемую прямоугольной областью xmin ≤ x ≤xmax, ymin ≤ y ≤ ymax;

функция title(’Заголовок’) предназначена для вывода заголовка графика;

функции xlabel(’Подпись под осью x’), ylabel(’Подпись под осью y’) служат для подписей осей x и y соответственно;

функция text(x,y,’текст’) выводит текст левее точки с координатами (x, y);

функция legend(’легенда 1’, ’легенда 2’, . . . , ’легенда n’, m) выводит легенды для каждого из графиков, параметр m определяет месторасположение легенды в графическом окне: 1 в правом верхнем углу графика (значение по умолчанию); 2 в левом верхнем углу графика; 3 в левом нижнем углу графика; 4 в правом нижнем углу графика.

При выводе текста с помощью функций xlabel, ylabel, title, text можно выводить греческие буквы[2] (см. табл. 1.3), использовать символы верхнего и нижнего индекса. Для вывода текста в верхнем индексе используется символ «^», в нижнем символ «_». Например, для вывода $e^{COS(x)}$ необходимо будет ввести текст: e^{cos(x)}, а для вывода x_{min} текст: x_{min}. При работе с текстом можно также использовать синтаксис TEX.

После описания основных возможностей по оформлению графиков рассмотрим ещё несколько примеров построения графиков.

Таблица 2.3. Греческие буквы и специальные символы

Команда	Символ	Команда	Символ
\alpha	α	\upsilon	υ
\beta	β	\phi	φ
\gamma	γ	\chi	χ
\delta	δ	\psi	ψ
\epsilon	ε	\omega	ω
\zeta	ζ	\Gamma	Γ
\eta	η	\Delta	Δ
\theta	θ	\Theta	Θ
\iota	ι	\Lambda	Λ
\kappa	κ	\Xi	Ξ
\lambda	λ	\Pi	Π
\mu	μ	\Sigma	Σ
\nu	ν	\Upsilon	Υ
\xi	ξ	\Phi	Φ
\pi	π	\Psi	Ψ
\rho	ρ	\Omega	Ω
\sigma	σ	\forall	∀
\varsigma	ς	\exists	∃
\tau	τ	\approx	≈
\int	∫	\in	Π
\wedge	^	\sim	~
\vee	v	\leq	≤
\pm	±	\leftrightarrow	↔
\geq	≥	\leftarrow	←
\inftyo	∞	\uparrow	↑
\partial	∂	\rightarrow	→
\neq	≠	\downarrow	↓
\nabla	Δ	\circ	°

Пример 2.3. Последовательно вывести в графическое окно графики функций y = sin x, y = sin(34x), y = cos x, y = cos x3 с задержкой 5 секунд.

Текст решения задачи с комментариями приведён в листинге 2.3.

Листинг 2.3. Построение графиков (пример 2.3).

okno1=figure(); % Создаём графическое окно с дескриптором okno1.

```
x=-6*pi():pi()/50:6*pi(); % Определяем аргумент на интервале [−6π; 6π]
y=sin(x); % Вычисляем значение функции SIN(x).
plot(x, y, 'k'); % Выводим график функции SIN(x) чёрного цвета.
grid on; % Выводим линии сетки.
title('Plot y = sin(x)'); % Выводим заголовок графика.
pause(5); % Приостанавливаем выполнение программы на 5 секунд.
y=sin(0.75*x);
plot (x,y, 'b'); % Выводим график функции SIN(0.75x) голубого цвета.
grid on; % Выводим линии сетки.
title('Plot y=sin(0.75x)');% Выводим заголовок графика.
pause(5); % Приостанавливаем выполнение программы на 5 секунд.
y=cos(x);
plot(x,y,'r');  % Выводим график функции COS(x) красного цвета.
grid on; % Выводим линии сетки.
title('Plot y = cos(x)'); % Выводим заголовок графика.
pause(5); % Приостанавливаем выполнение программы на 5 секунд.
y=cos(x/3);
plot(x,y,'g'); % Выводим график функции COS(x/3) зелёного цвета.
grid on; % Выводим линии сетки.
title('Plot y = cos(x/3)'); % Выводим заголовок графика.
pause(5); % Приостанавливаем выполнение программы на 5 секунд.
delete(okno1); % Закрываем окно с дескриптором okno1.
```

При запуске программы будет создано графическое окно, в котором будет выведен график функции sin(x) чёрного цвета с линиями сетки и заголовком. Это окно будет находиться на экране в течении 5 секунд, после чего очистится. Будет выведен график функции sin(0.75x) голубого цвета с линиями сетки и заголовком, которое будет на экране в течении 5 секунд. Далее аналогично будут с задержками 5 секунд выведены графики функций cos(x) и cos(x/3). После этого окно автоматически закроется. Для понимания механизма работы с графическими окнами в Octave, авторы рекомендуют читателю при выводе графика sin(x) попытаться закрыть окно и посмотреть, что из этого получится.

Пример 2.4. Построить графики функций $y = e^{\sin x}$, $u = e^{\cos(x/3)}$, $v = e^{\sin(x/2)}$ на интервале $[-3\pi; 3\pi]$.

Рассмотрим два варианта построения графиков (см. листинг 2.4).

Листинг 2.4. Построение графиков (пример 2.4).

Способ первый

```
x=−3*pi(): pi()/20:3*pi();% Формируем массив x.
y=exp(sin(x)); % Формируем массив y.
u=exp(cos(x/3));% Формируем массив u.
```

v=exp(sin(x/2));% Формируем массив v.

% Строим график функции y(x), сплошная чёрная линия, без маркера,
% в качестве легенды выводим $e^{sin(x)}$.
plot(x, y,"k; e^{ sin(x)};")
hold on; % Блокируем режим очистки окна.
% Строим график функции u(x), сплошная чёрная линия, с маркером
% треугольником, размер маркера 4, в качестве легенды выводим
% $e^{cos(x/3)}$.
plot(x,u,"->k;e^{ cos(x/3)};","markersize",4)
% Строим график функции v(x), сплошная чёрная линия, с маркером
% окружностью, размер маркера 4, в качестве легенды выводим
% $e^{sin(x/2)}$.
plot(x,v, "-ok; e^{sin(x/2)};","markersize",4);

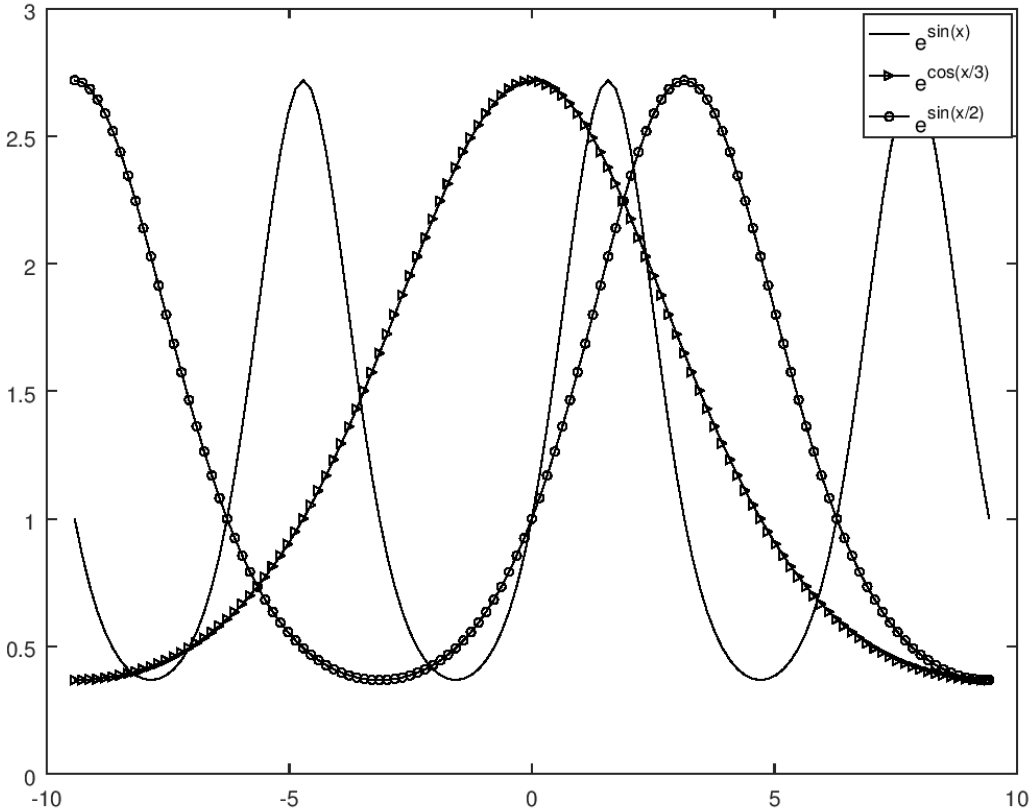


Рисунок 2.4 – Графики функций $y = e^{sinx}$, $u = e^{cos(x/3)}$, $v = e^{sin(x/2)}$

Способ второй

x=-3*pi():pi()/20:3∗pi();
y=exp(sin(x));
u=exp(cos(x/3));
v=exp(sin(x/2));
% Отличие вывода трёх графиков состоит в том, вместо 3-х функций

%plot и двух hold on используется одна функция plot в которой указаны

% те же параметры вывода графиков, что и в листинге 2.4

plot(x,y, "k;e^{sin(x)};",x,u,"->k;e^{cos(x/3)};",…

"markersize", 4, x, v, "-ok; e ^{sin (x/2)};","markersize",4)

Оба способа формируют один и тот же график (см. рис. 2.4).

Пример 2.5. Построить график функции $y(x)=1-\frac{0.4}{x}+\frac{0.05}{x^2}$ на интервале [−2; 2].

В связи с тем, что функция не определена в точке x = 0, будем строить её как графики двух функций y(x) на полуинтервале [−2; 0)] и y(x) на полуинтервале (0; 2]. Текст программы на Octave с комментариями приведён в листинге 2.5, график функции на рис. 2.5.

Листинг 2.5. Построение графика (пример 2.5).

a=1; b=-0.4; c=0.05;

x1=-2:0.01:-0.1; % Определяем аргумент (массив x) на [−2; −0.1].

x2=0.1:0.01:2; % Определяем аргумент (массив x) на [0.1; 2].

y1=a+b./x1+c./x1.^2; % Вычисляем значение функции y(x) на [−2; −0.1].

y2=a+b./x2+c./(x2.*x2); % Вычисляем значение функции y(x) на [0.1; 2].

% Строим график как график двух функций, на интервалах [−2; −0.1],

% [0.1; 2],

% цвет графика чёрный, легенда .

plot(x1,y1,’k; f(x)=a+b/x+c/x^2;’, x2,y2,’k’);title(‘y=f(x)’);

% Подпись над графиком.

xlabel (’X’); % Подпись оси X.

ylabel(’Y’) ;% Подпись оси Y.

grid on; % Рисуем линии сетки.

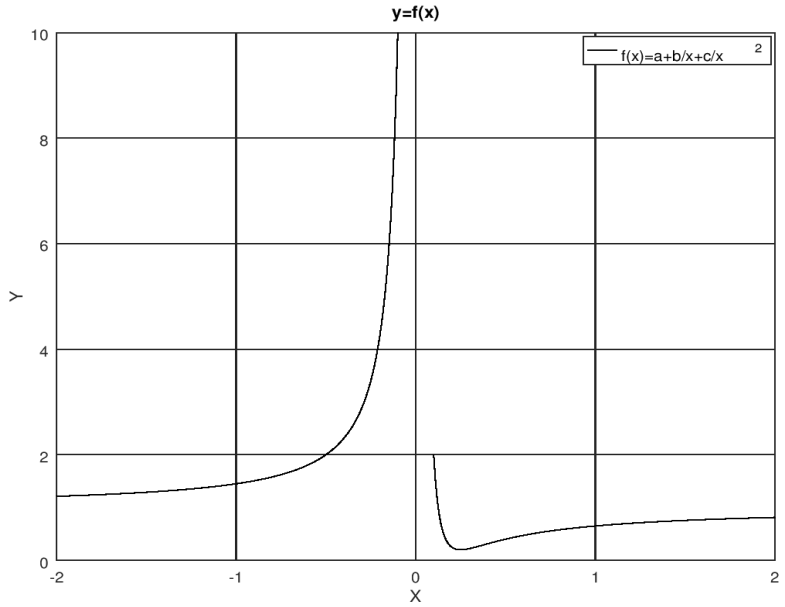


Рисунок 2.5 – График функции y(x)

Пример 2.6. Построить график функции $y(x)=\frac{1}{x^2-2x-3}$ на интервале [−5; 7].

Уравнение $y(x)=\frac{1}{x^2-2x-3}$ имеет корни −1, 3. Поэтому наша функция y(x) будет иметь разрывы в этих точках x = −1, x = 3. Будем строить её, как графики трёх функций, на трёх интервалах [-5; -1.1], [-0.9; 2.9], [3.1; 7]. Листинг 2.6 демонстрирует решение примера 2.6, а на рис. 2.6 изображён график функции $y(x)=\frac{1}{x^2-2x-3}$ как результат работы этой программы.

Листинг 2.6. Построение графика (пример 2.6).

```
% Определяем аргументы на интервалах [-5; -1.1], [-0.9; 2.9], [3.1; 7].
x1=-5:0.01:-1.1;
x2=-0.9:0.01:2.9;
x3=3.1:0.01:7;
Вычисляем значение y(x) на соответствующих интервалах.
y1=1./(x1.*x1-2*x1-3);
y2=1./(x2.*x2-2*x2-3);
y3=1./(x3.*x3-2*x3-3);
% Строим график чёрного цвета, как график 3-х функций.
plot(x1,y1,'k', x2,y2,'k', x3,y3,'k'); title('y=f(x)');
% Подпись над графиком. xlabel('X'); % Подпись оси X.
ylabel ('Y'); % Подпись оси Y.
legend('f(x)=1/(x^2-2 x-3)',4); % Вывод легенды.
grid on; % Рисуем линии сетки.
```

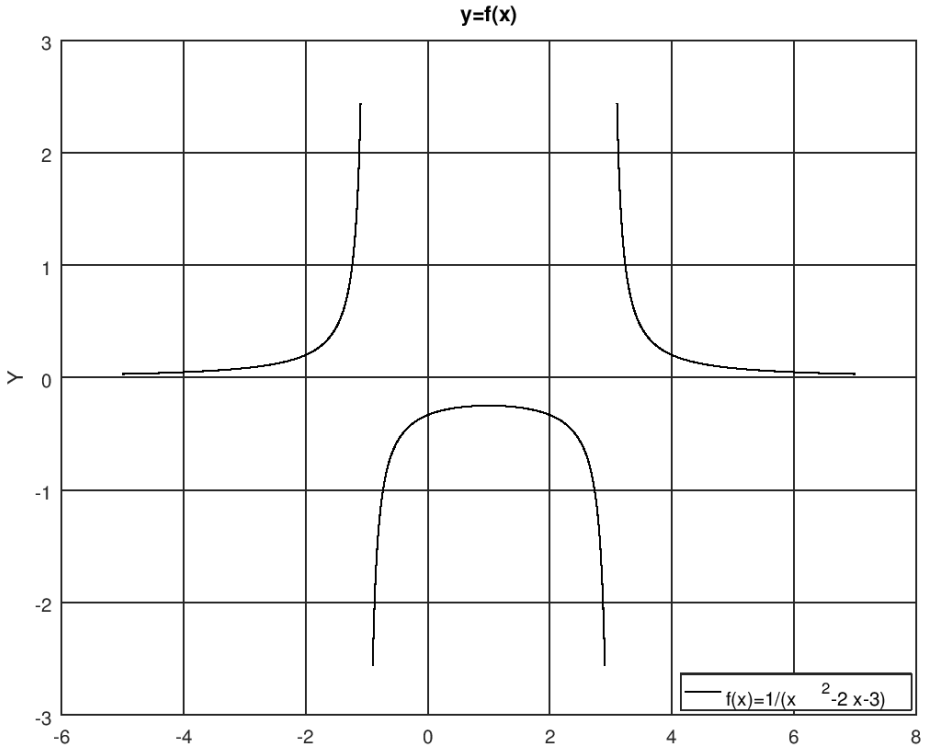


Рисунок 2.6 – График функции $y(x)=\frac{1}{x^2-2x-3}$

Octave предоставляет возможность построить несколько осей в графическом окне и вывести на каждую из них свои графики. Для этого следует использовать функцию subplot: subplot(row, col, cur).

Параметры row и col определяют количество графиков по вертикали и горизонтали соответственно, cur определяет номер текущего графика. Повторное обращение к функции subplot с теми же значениями row и col позволяет просто изменяет номер текущего графика может использоваться для переключения между графиками. Рассмотрим использование функции subplot при решении следующей задачи.

Пример 2.7. Построить графики функции $y=2e^{-0.15x^2}$, $z=e^{0.7x-0.25x^2}$, $u=0.5e^{-0.33x}\sin\left(2x+\frac{\pi}{3}\right)$, $k=3\sin(x-0.22x^2)$, $v=\cos x$, $w=e^{\cos x}$

Решение задачи представлено в листинге 2.7, полученное графическое окно на рис. 2.7.

Листинг 2.7. Построение графиков (пример 2.7).

```
% Формируем массивы x, y, z, u, k, v, w.
x=-6: 0.2: 6; y=2*exp(-0.15*x.^2); z=exp(0.7*x-0.25*x.^2);
u=0.5*exp(-0.33*x).*sin(2*x+pi()/3); k=3*sin(x-0.22.*x.^2);
v=cos(x); w=exp(cos(x));
```

% Делим графическое окно на 6 частей и объявляем первый график
% текущим. subplot(3,2,1);
plot(x,y); % Строим график y(x) с линиями сетки и подписями.
grid on; title(’Ploty=2 e^{-0.15x^2}’); xlabel(’x’); ylabel(’y’);
subplot(3,2,2); % Второй график объявляем текущим.
plot(x,z); % Строим график z(x) с линиями сетки и подписями
grid on; title(’Plot z=cos^2(x)’); xlabel(’x’); ylabel(’z’);
subplot(3,2,3); % Третий график объявляем текущим.
plot(x,u); % Строим график u(x) с линиями сетки и подписями.
grid on; title(’Plot u=0.5 e^{-0.33 x}sin(2 x+pi/3)’); xlabel(’x’); ylabel(’u’);
subplot(3,2,4); % Четвёртый график объявляем текущим.
plot(x,k); % Строим график k(x) с линиями сетки и подписями.
grid on; title(’Plot k=3 sin(x-0.22 x^2)’); xlabel(’x’); ylabel(’k’);
subplot(3,2,5); % Пятый график объявляем текущим.
plot(x,v); % Строим график v(x) с линиями сетки и подписями.
grid on; title(’Plot v=cos(x)’); xlabel(’x’); ylabel(’v’);
subplot(3,2,6); % Шестой график объявляем текущим.
plot(x,w); % Строим график w(x) с линиями сетки и подписями.
grid on; title(’Plot w=e^{cos(x)}’); xlabel(’x’); ylabel(’w’);

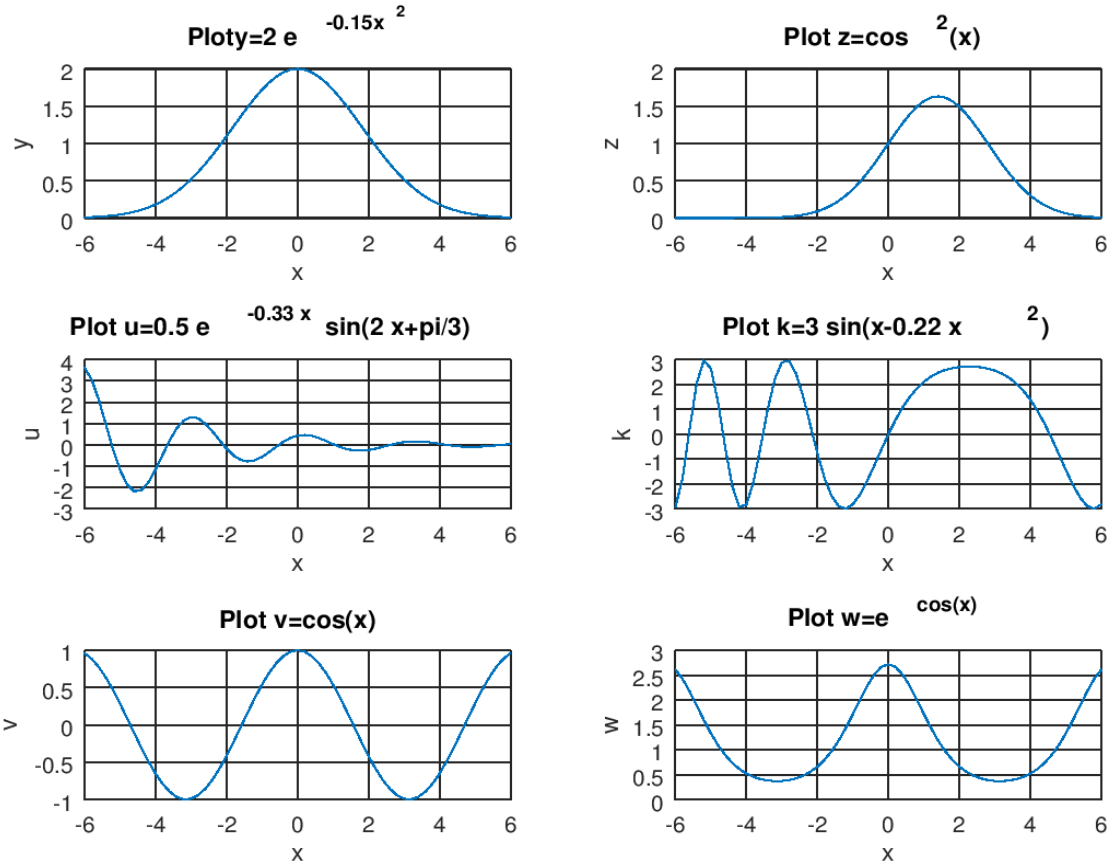


Рисунок 2.7 –Графики функций *y(x), z(x), u(x), k(x), v(x), w(x)*

Для построения графика функции можно использовать функцию **fplot** следующей структуры: fplot(@f, [xmin, xmax], s).

Здесь f имя функции (стандартной функции Octave или функции, определённой пользователем), [xmin, xmax] интервал, на котором будет строиться график, s строка формата, определяющая только параметры линии (но не легенду). Легенда графика формируется автоматически функцией fplot без использования функции legend. Не позволяет формировать легенду и третий параметр функции fplot, в отличии от функции plot.

Пример 2.8. Используя функцию fplot, построить графики функций $e^{\sin x}$, $e^{\cos x}$, sinx, cosx на интервале [−3π; 2π].

Текст программы на языке Octave с комментариями приведён в листинге 2.8.

Листинг 2.8. Построение графиков с помощью fplot (пример 2.8).

```
% Определяем функцию f=exp(cos(x))
function y=f(x)
    y=exp(cos(x));
end
% Определяем функцию g = exp(sin(x))
function z=g(x)
    z=exp(sin(x));
end
h=figure();
% Делим графическое окно на 4 части и объявляем первый график
% текущим.
subplot(2,2,1);
% Строим график g = exp(cos(x)) на интервале [−3π; 2π]
% голубого цвета с маркером.
fplot(@g, [-3∗ pi(), 2∗pi()],'- pb');
title('Plot y=e^{cos(x)}'); % Подписываем график.
grid on; % Проводим линии сетки
subplot(2,2,2); % Второй график объявляем текущим.
% Строим график f=exp(sin(x)) на интервале [−3π; 2π]
% красного цвета с маркером.
fplot(@f, [−3∗ pi(), 2∗ pi()], '-or');
title('Plot z=e^{sin(x)}');% Подписываем график.
grid on; % Проводим линии сетки
```

```
subplot(2,2,3); % Третий график объявляем текущим.
% Строим график sin(x) на интервале [−3π; 2π] чёрного цвета
fplot(@sin, [-3∗ pi(), 2∗pi()], 'k');
title('Plot sin(x)'); % Подписываем график.
grid on; % Проводим линии сетки
subplot(2,2,4); % Четвёртый график объявляем текущим.
% Строим график COS(x) на интервале [−3π; 2π] зелёного цвета
fplot(@cos, [-3∗pi(), 2∗ pi()], 'g');
title('Plot cos(x)'); % Подписываем график.
grid on; % Проводим линии сетки
```

Функция bar предназначена для построения гистограммы. Функция bar(y) выводит элементы массива y в виде гистограммы, в качестве массива x выступает массив номеров элементов массива y. Функция bar(x, y) выводит гистограмму элементов массива y в виде столбцов в позициях, определяемых массивом x, элементы которого должны быть упорядочены в порядке возрастания.

Рассмотрим несколько примеров. Фрагмент y=[5; 6; 7; 8; 9; 8; 7; 6; 4; 3]; bar(y); строит гистограмму, представленную на рис. 2.8.

Фрагмент x1=[−2,−1,0,1,2,3,4]; y1=exp(sin(x1)); bar(x1,y1); строит гистограмму, представленную на рис. 2.9.

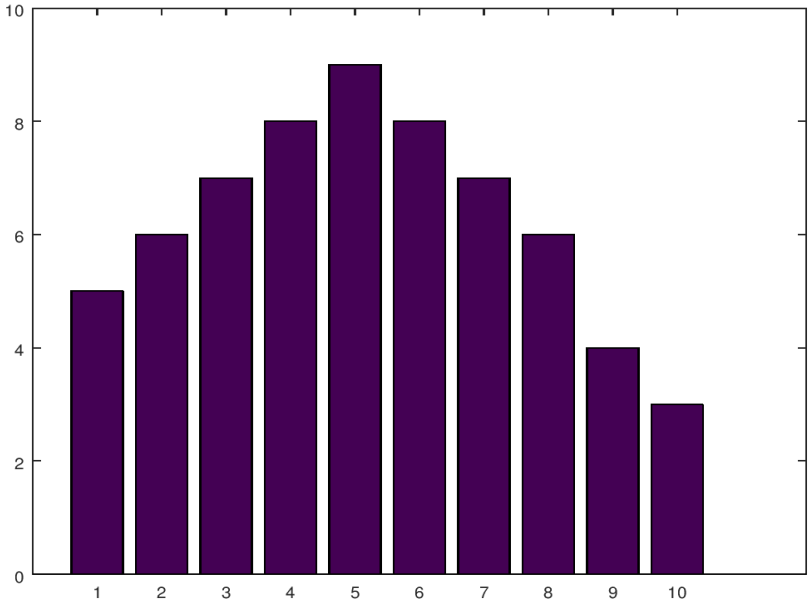


Рисунок 2.8 – Пример использования функции bar(y)

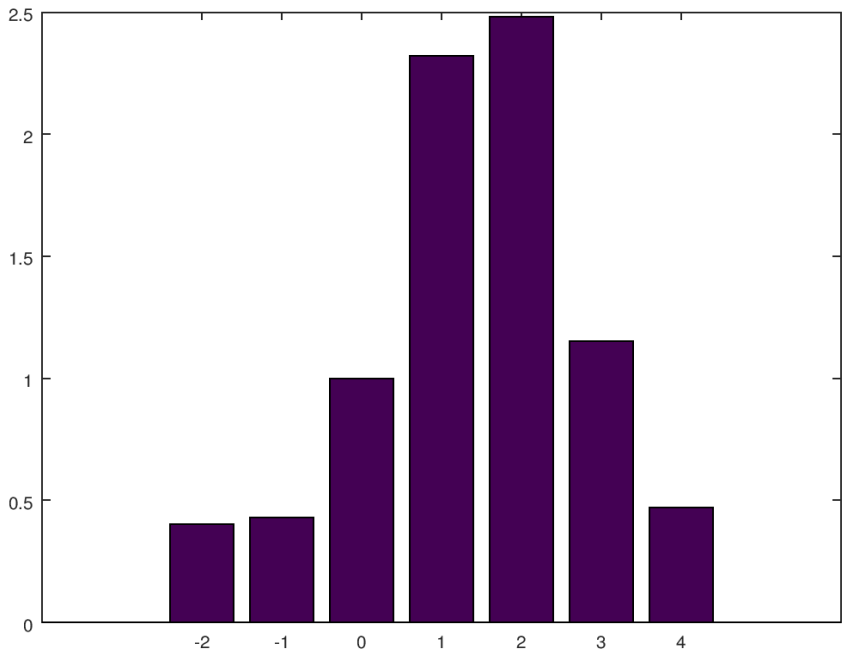


Рисунок 2.9 – Пример использования функции bar(x,y)

2.2. ПОСТРОЕНИЕ ГРАФИКОВ ПОВЕРХНОСТЕЙ

Для построения графика функции двух переменных z=f(x,y) необходимо выполнить следующие действия.

Сформировать в области построения графика прямоугольную сетку, проводя прямые, параллельные осям y = yj и x = xi, где

xi=x0+ih, h = xn−nx0, i=0, 1, 2, . . . , n, yj=y0+jδ, δ= k−ky0,

j = 0, 1, 2, . . . , k.

Вычислить значения $z_{i,j}$=f(xi, yj) во всех узлах сетки.

Обратиться к функции построения поверхности, передавая ей в качестве параметров сетку и матрицу Z={zi,j} значений в узлах сетки.

Для формирования прямоугольной сетки в Octave есть функция meshgrid. Рассмотрим построение трёхмерного графика на следующем примере.

Пример 2.9. Построить график функции

z(x, y) = 3x2-2sin2y, x ∈ [−2, 2], y ∈ [−3, 3].

Листинг 2.9.

```
% Для формирования сетки воспользуемся функцией meshgrid.
[x y]=meshgrid(-2:2 ,-3:3);
% После формирования сетки вычислим значение функции во всех
% узловых точках
z=3∗x.∗ x-2∗sin(y).^2
% Для построения каркасного графика следует обратиться к функции
% mesh
```

mesh(x, y, z);

После это будет создано графическое окно с трёхмерным графиком (рис. 2.10). Как видно, полученный график получился грубым, для получения менее грубого графика следует сетку сделать более плотной (листинг 2.10 и рис. 2.11).

Листинг 2.10. Построение графика поверхности (пример 2.9).

```
[x y]=meshgrid(-2:0.1:2,-3:0.1:3);
z=3*x.*x-2*sin(y).^2;
mesh(x,y,z);
```

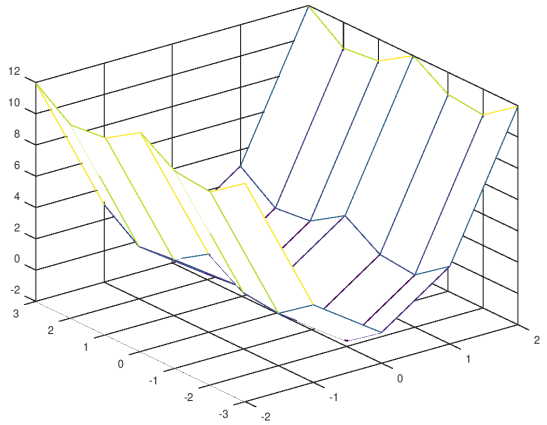


Рисунок 2.10 – График функции $z(x, y) = 3x^2 - 2\sin^2 y$

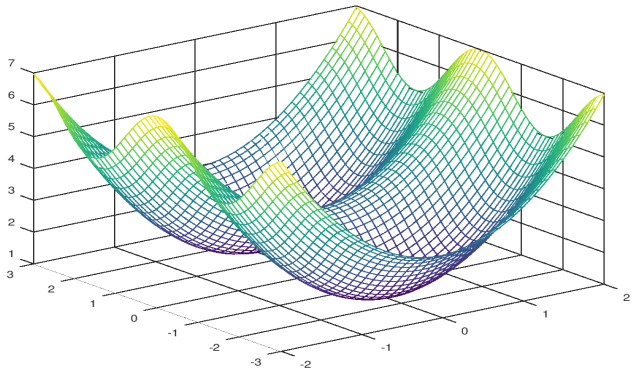


Рисунок 2.11 – График функции $z(x, y) = 3x^2 - 2\sin^2 y$

Любой трёхмерный график можно вращать, используя мышку. Для построения поверхностей, кроме функции mesh построения каркасного графика, есть функция **surf,** которая строит каркасную поверхность, заливая её каждую клетку цветом, который зависит от значения функции в узлах сетки.

Пример 2.10. С использованием функции surf построить график функции $z(x,y) = \sqrt{sin^2 x + \cos^2 y}$

На листинге 2.11 представлено решение задачи, а на рис. 2.12 изображён получившийся график.

Листинг 2.11. Построение графика поверхности (пример 2.10).

```
[x y]=meshgrid(-2:0.2:2, 0:0.2:4);
z=sqrt(sin(x).^2+cos(y).^2);
surf(x, y, z);
```

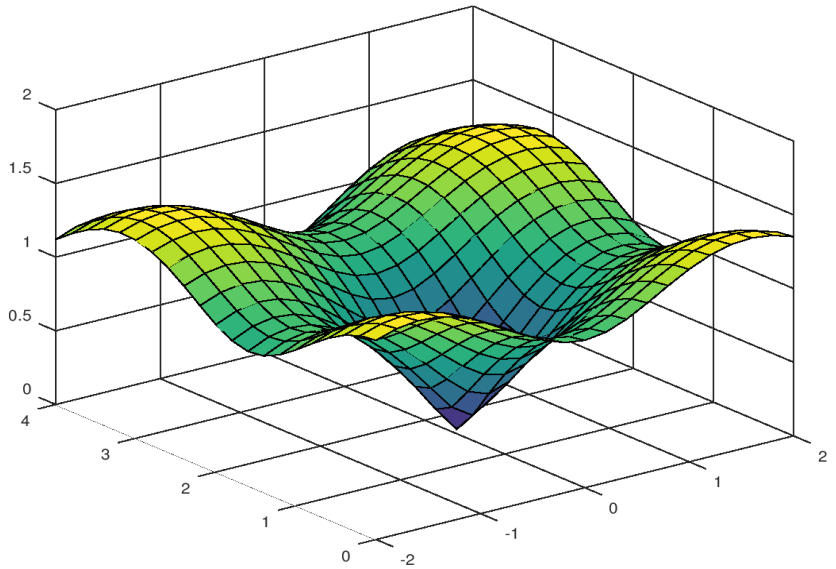


Рисунок 2.12 – График функции

$$z(x,y) = \sqrt{sin^2 x + \cos^2 y}$$

В Octave можно построить графики двух поверхностей в одной системе координат, для этого, как и для плоских графиков, следует использовать команду hold on, которая блокирует создание второго нового окна при выполнении команд surf или mesh.

Пример 2.11. Построить в одной системе координат графики функций $z(x, y)=\pm(2x^2 + 3y^4) - 1$.

Решение задачи с использованием функции surf представлено в листинге 1.12, полученный график изображён на рис. 2.13.

Листинг 2.12.

```
h=figure();
[x y]=meshgrid(-2:0.1:2,-3:0.1:3); z=2*x.^2+3*y.^4-1;
z1=−2*x.^2-3*y .^4-1;
surf(x, y, z);
hold on
surf(x, y, z1);
```

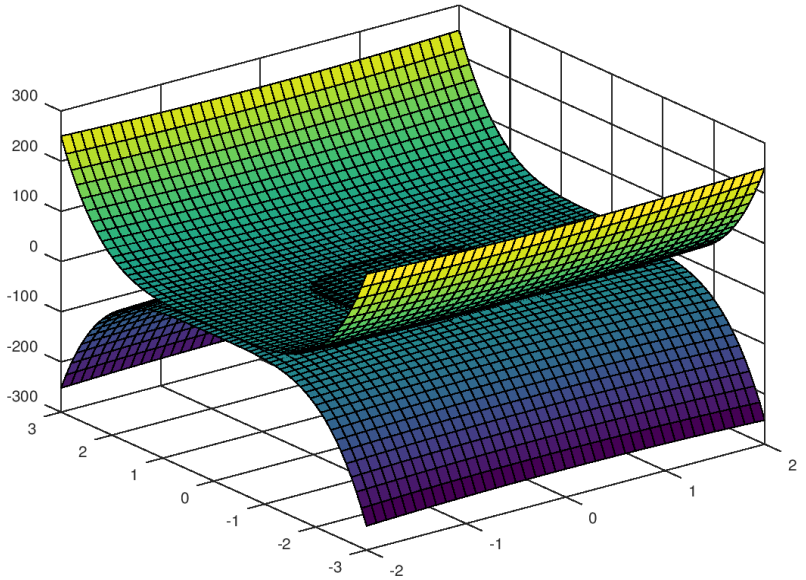


Рисунок 2.13 – Изображение двух поверхностей в одной системе координат с использованием функции surf

Листинг 2.13.

```
h=figure();
[x y]=meshgrid(-2:0.1:2,-3:0.1:3);
z=2*x.^2+3*y .^4-1;
z1=-2*x.^2-3*y.^4-1;
mesh(x, y, z);
hold on
mesh(x , y, z1);
```

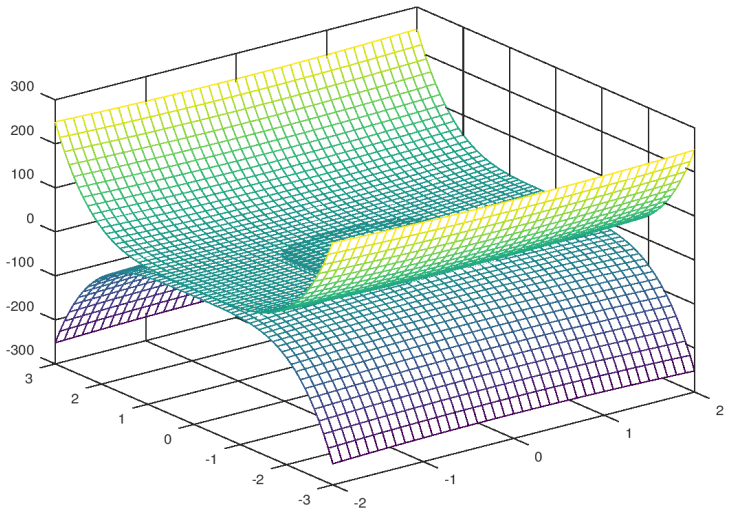


Рисунок 2.14 – Изображение двух поверхностей в одной системе координат с использованием функции mesh

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №2

Задание 2.1. Построить график функции f (x) и приблизительно определить один из корней уравнения. Решить уравнение f (x) = 0 с точностью ε = 10^{-4}.

Вариант	$f(x)$	Вариант	$f(x)$
1	$3\sin(\sqrt{x})+0{,}35x-3{,}8$ $x\in[2;3]$	**9**	$e^{x}-e^{-x}-2$ $x\in[0;1]$
2	$x-\dfrac{1}{3+\sin(3{,}6x)}$ $x\in[0;1]$	**10**	$\sqrt{1-x}-tgx$ $x\in[0;1]$
3	$\arccos x-\sqrt{1-0{,}3\,x^{3}}$ $x\in[0;1]$	**11**	$\sqrt{2x^{2}+1{,}2-\cos x}-1$ $x\in[0;1]$
4	$\sqrt{1-0{,}4x^{2}}-\arcsin x$ $x\in[0;1]$	**12**	$\cos(2/x)-2\sin(1/x)+1/x$ $x\in[1;2]$
5	$3x-14+e^{x}-e^{-x}$ $x\in[1;3]$	**13**	$0{,}1x^{2}-x\ln x$ $x\in[1;2]$
6	$0{,}25x^{3}+x-2$ $x\in[0;2]$	**14**	$1-x+\sin x-\ln(1+x)$ $x\in[0;2]$
7	$\arccos\left((1-x^{2})/(1+x^{2})\right)-x$ $x\in[2;3]$	**15**	$e^{x-1}-x^{3}-x$ $x\in[0;1]$
8	$3x-4\ln x-5$ $x\in[2;4]$	**16**	$5\cos(\sqrt{x})+0{,}95x-2{,}1$ $x\in[2;3]$

Задание 2.2. Построить график функции $y=\dfrac{10}{a\cdot x^{2}+b\cdot x+c}$ **на интервале [m; n]**

N	a	b	c	m	n	N	a	b	c	m	n
1	2	6	4	-3	0	9	2	2	-3	-2,5	2
2	2	-6	4	0	3	10	2	-2	-3	-2	2,5
3	1	0	-4	-3	3	11	3	3	-4,5	-2,5	2
4	3	9	6	-3	0	12	3	-3	-4,5	-2	2,5
5	3	-9	6	0	3	13	3	-3	-6	-2	3
6	2	-2	-4	-2	3	14	3	3	-6	-3	2
7	2	2	-4	-3	2	15	1	-2	-1,25	-1,5	3,5

8	1	0	-9	-4	4	16	1	2	-1,25	-3,5	1,5

Задание 2.3. Решить приближённо систему уравнений графическим методом (Найти один из возможных корней)

№	1	2	№	1	2
1	$y=\frac{2+x}{1+x^2}$	$y=\ln(2x)+0,1x$	9	$y=\frac{1+2x}{1+1,5x^2}$	$y=\ln(1,5x)+0,2x$
2	$y=2\sin(x)+\cos(2$	$y=-1.2+0,5x+0,6$	10	$y=1+0,55x+0,69$	$y=0,2\sin(2x)+\cos($
3	$0.1x+\exp(1+0.5x)$	$y=\frac{8}{1+x}$	11	$y=\frac{8}{1+0.9*x}$	$y=0.2x+\exp(x)$
4	$y=2\cdot\exp(-0.5\cdot x)$	$y=0.2+1.5x^{1.2}$	12	$y=0.2+1.5x^{1.4}$	$y=3\cdot\exp(-0.8\cdot x)$
5	$y=-1+\exp(0.5x)$	$y=\frac{3}{1+2x}$	13	$y=\frac{2.3}{1+1.8x}$	$y=\exp(0.5x)-1$
6	$y=1.1\sin(1.3x)$	$y=1.1\cos(1.3x)$	14	$y=\cos(x)$	$y=\sin(x)$
7	$y=\exp(-x)$	$y=\log(0.2+x)$	15	$y=\log(0.3+x)$	$y=\exp(-1.1x)$
8	$y=1+\frac{1}{\log(x+1.2)}$	$y=\exp(x)-1$	16	$y=\exp(x)-2$	$y=1+\frac{1}{\log(x+1.1)}$

Задание 2.4. Отобразить в пространственной системе координат функцию z=f(x,y) с помощью команд **mesh** и **surf** в диапазоне **x, y [-2,2;-3,3)**

№		№	
1	$1.5\sqrt{\sin^2(x)+\cos^2(y)}$	9	$\sqrt{\sin^2(1,2\cdot x)+\cos^2(1,5y)}$
2	$x^2\cdot\cos^2(y)$	10	$y^2\cdot\cos^2(x)$
3	$\sqrt{2.*\sin^2(x)+1.5.\cos^2(y)}$	11	$\sqrt{2\cdot\sin^2(1.3\cdot x)+1.5\cdot\cos^2(1.2\cdot y)}$
4	$x^2\cdot\sin^2(y)$	12	$y^2\cdot\sin^2(x)$
5	$0.5\cdot x\cdot\sqrt{\sin^2(x)+\cos^2(y)}$	13	$0.5\cdot x\cdot\sqrt{2\cdot\sin^2(x)+1.5\cdot\cos^2(y)}$
6	$\frac{x^2\cdot\sin^2(y)}{y+3.5}$	14	$\frac{x^2\cdot\cos^2(y)}{y+3.5}$
7	$x\cdot\sqrt{\sin^2(x)+\cos^2(y)}$	15	$y\cdot\sqrt{\sin^2(x)+\cos^2(y)}$
8	$\frac{y^2\cdot\sin^2(x)}{x+2.5}$	16	$\frac{y^2\cdot\cos^2(x)}{x+2.5}$

В отчёте по лабораторной работе привести варианты своего задания, алгоритмы и листинги программ решения задач и полученные результаты в графическом исполнении.

ЛАБОРАТОРНАЯ РАБОТА № 3.
ЛИНЕЙНАЯ АЛГЕБРА

3.1. РЕШЕНИЕ НЕКОТОРЫХ ЗАДАЧ АЛГЕБРЫ МАТРИЦ

Напомним основные определения алгебры матриц [7]. Если m×n выражений расставлены в прямоугольной таблице из m строк и n столбцов, то говорят о матрице размера m×n:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Выражения a_{ij} называют элементами матрицы. Элементы a_{ii} (i=1..n), стоящие в таблице на линии, проходящей из левого верхнего угла в правый нижний угол квадрата n×n , образуют главную диагональ матрицы.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & \dots & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & \dots & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ii} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & \dots & \dots & a_{mn} \end{pmatrix}$$

Матрица размером $m \times n$ ($m \neq n$) называется *прямоугольной*. В случае если m=n , то матрицу называют квадратной матрицей порядка n.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

В частности матрица типа 1×n это вектор–строка: a_{11} a_{12} ... a_{1n} Матрица размером m×1 является вектором–столбцом:

$$\begin{pmatrix} a_{11} \\ a_{21} \\ \dots \\ a_{m1} \end{pmatrix}$$

Число (скаляр) можно рассматривать как матрицу типа 1×1 - a11. Квадратная матрица A={a_{ij}} размером n×n называется нулевой, если все ее элементы равны нулю:

$$\begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

Определителем (детерминантом) матрицы A является число detA или Δ, вычисляемое по правилу:

$$detA = \sum(-1)^{\lambda} a_{1i_1} a_{2i_2} \dots a_{ni_n},$$

где сумма распределена на всевозможные перестановки (i_1,i_2,..., i_n) элементов 1,2,…,n и, следовательно, содержит n! слагаемых, причем λ=0, если перестановка четная, и λ=1, если перестановка нечетная.

Квадратная матрица называется невырожденной, если ее определитель отличен от нуля

detA≠0. В противном случае det A=0 матрица называется вырожденной или сингулярной.

С матрицами можно проводить операции сравнения, сложения и умножения.

Две матрицы A={a_{ij}} и B={b_{ij}} считаются равными, если они одного типа, то есть имеют одинаковое число строк и столбцов, и соответствующие элементы их равны {a_{ij}}={b_{ij}}.

Суммой двух матриц A={a_{ij}} и B={b_{ij}} одинакового размера называется матрица C={c_{ij}} того же размера, элементы которой равны сумме соответствующих элементов матриц A={a_{ij}} и B={b_{ij}}: {c_{ij}}={a_{ij}+b_{ij}}.

Разность матриц A={a_{ij}} и B={b_{ij}} определяется аналогично: {c_{ij}}={a_{ij}−b_{ij}}.

Произведением числа h на матрицу A={a_{ij}} (или произведением матрицы на число) называется матрица, элементы которой получены умножением всех элементов матрицы A={a_{ij}} на число h: h A={h·a_{ij}}.

Произведением матриц A={a_{ij}} размерностью m×n и B={b_{ij}} размерностью p×s является матрица C={c_{ij}} размерностью m×s, каждый элемент которой можно представить формулой

C={c_{ij}}={a_{i1} b_{1j}+a_{i2} b_{2j}+...+a_{in} b_{nj}}, (i=1..m, j=1..s).

Таким образом, произведение матриц A={a_{ij}} и B={b_{ij}} имеет смысл тогда и только тогда, когда количество строк матрицы A={a_{ij}} совпадает с количеством столбцов матрицы B={b_{ij}}. Кроме того, произведение двух матриц не обладает переместительным законом, то есть A·B≠B·A. В тех случаях, когда A·B=B·A, матрицы A={a_{ij}} и B={b_{ij}} называются перестановочными.

Если в матрице A={a_{ij}} размерностью m×n заменить строки соответствующими столбцами, то получится транспонированная матрица

A^T={a_{ji}}.

В частности, для вектора–строки a={a_1 a_2 a_3... a_n} транспонированной матрицей является вектор–столбец:

$$a^T = \begin{pmatrix} a_{11} \\ a_{21} \\ \dots \\ a_{m1} \end{pmatrix}$$

Обратной матрицей по отношению к данной матрице A={a_{ij}} размерностью n×n , называется матрица A−1={A_{ij}} того же размера, которая, будучи умноженной как справа, так и слева на данную матрицу, в результате дает единичную матрицу E:

$$A \cdot A^{-1} = A^{-1} \cdot A = E$$

Нахождение обратной матрицы для данной называется обращением данной матрицы.

Всякая неособенная матрица имеет обратную матрицу

Перейдем к конкретным задачам.

ЗАДАЧА 3.1. Для матриц A, B и C проверить выполнение следующих тождеств:

- (A·B)·C=A·(B·C);
- (A^T+B)·C=A^T·C+B·C.

В листинге видно, что матрицы, получившиеся в результате вычисления левой и правой частей первого тождества, равны, следовательно, первое предположение истинно. Для исследования второго тождества из левой части равенства вычитаем правую и получаем нулевую матрицу, что так же приводит к выводу об истинности предположения.

Листинг 3.1.

```
>>% Исследование первого тождества
>> A=[1 -2 0; -3 0 4]
A =
      1 -2 0
      -3 0 4
>> B=[3 1;2 0;-1 1]
B =
     166
     3 1
     2 0
     -1 1
>> C=[1 2;-1 0]
C =
     1 2
     -1 0
>> (A*B)*C
```

```
ans =
      -2 -2
      -14 -26
>> A*(B*C)
ans =
      -2 -2
      -14 -26
>>% Исследование второго тождества
>> (A'+B)*C-(A'*C+B*C)
ans =
        0 0
        0 0
        0 0
```

ЗАДАЧА 3.2. Проверить является ли матрица ортогональной. Квадратная матрица называется ортогональной, если $|A|=\det A\neq 0$ и $A^T=A^{-1}$.

Для решения поставленной задачи необходимо вычислить определитель заданной матрицы, и убедиться в том, что он не равен нулю. Затем транспонировать исходную матрицу и найти обратную к ней. Если визуально сложно убедиться в том, что транспонированная матрица равна обратной, можно вычислить их разность. В результате должна получиться нулевая матрица (листинг 3.2).

Листинг 3.2.

```
>> A=[0.5 0.7071 0.5;0.7071 0 -0.7071;0.5 -0.7071 0.5]
A =
      0.50000 0.70710 0.50000
      0.70710 0.00000 -0.70710
      0.50000 -0.70710 0.50000
>> % Определитель матрицы A отличен от нуля
>> det(A)
    ans = -0.99998
```

В результате вычитания из транспонированной матрицы A обратной к ней матрицы получаем нулевую матрицу. Это означает, что A - ортогональная.

```
>> A'-inv(A)
ans =
        0.0000e+00 -1.3562e-05 0.0000e+00
        -1.3562e-05 0.0000e+00 1.3562e-05
        0.0000e+00 1.3562e-05 0.0000e+00
```

Пример 3.1. Решить матричные уравнения A·X=B и X·A=B, выполнить проверку.

Матричное уравнение — это уравнение вида A·X =B или X·A=B, где X - это неизвестная матрица. Если умножить матричное уравнение на матрицу обратную к A, то оно примет вид:

$A^{-1}\cdot A\cdot X = A^{-1}\cdot B$ или $X\cdot A\cdot A^{-1}=B\cdot A^{-1}$.

Так как $A^{-1}\cdot A=A\cdot A^{-1}=E$, а $E\cdot X=X\cdot E=X$, то неизвестную матрицу X можно вычислить так: $X = A^{-1}\cdot B$ или $X = B\cdot A^{-1}$. Понятно, что матричное уравнение имеет единственное решение если A и B – квадратные матрицы n-го порядка и определитель матрицы A не равен нулю. Как решить матричное уравнение в Octave, показано в листинге 3.3.

Листинг 3.3.

```
>> A=[2 3;-2 6]
A =
      2 3
      -2 6
>> B=[2 5;2/3 5/3]
B =
      2.00000 5.00000
      0.66667 1.66667
>> % Решение матричного уравнения A·X=B
>> % Первый способ
>> X=A\B
X =
0.55556 1.38889
0.29630 0.74074
>> % Второй способ
>> X=inv(A)*B
X =
      0.55556 1.38889
      0.29630 0.74074
>> % Проверка A·X-B=0
>> A*X-B
ans =
        0.0000e+00 0.0000e+00
        -3.3307e-16 -6.6613e-16
>> % Решение матричного уравнения X A=B
>> % Первый способ
>> X=B/A
X =
      1.222222 0.222222
```

```
        0.407407 0.074074
>> % Второй способ
>> X=B*inv(A)
X =
        1.222222 0.222222
        0.407407 0.074074
>> % Проверка X A-B=0
>> X*A-B
ans =
          0.0000e+00 0.0000e+00
          1.1102e-16 0.0000e+00
```

3.2. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

Система m уравнений с n неизвестными вида

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & = & b_m \end{cases}$$

называется системой линейных алгебраических уравнений (СЛАУ), причём xj - неизвестные, aij - коэффициенты при неизвестных, bi - свободные коэффициенты (i = 1 ... m, j = 1 ... n).

Кроме того, система из m линейных уравнений с n неизвестными может быть описана при помощи матриц: A·x=b , где x = {xj} - вектор неизвестных, A = {aij} = матрица коэффициентов при неизвестных или матрица системы, b = {bi} - вектор свободных членов системы или вектор правых частей (i = 1 . . .m, j = 1 . . . n).

Матрица (A|b), которая формируется путём приписывания к матрице коэффициентов A столбца свободных членов b, называется расширенной матрицей системы.

Если все bi = 0, то речь идёт об однородной системе линейных уравнений, иначе говорят о неоднородной системе. Совокупность всех решений системы (x1, x2 ... xn), называется множеством решений или просто решением системы. Две системы уравнений называются эквивалентными, если они имеют одинаковое множество решений.

Однородные системы линейных уравнений Ax = 0 всегда разрешимы, так как последовательность (x1 = 0, x2 = 0, ..., xn = 0) удовлетворяет всем уравнениям системы. Такое решение называют тривиальным. Вопрос о решении однородных систем сводится к вопросу о том, существуют ли кроме тривиального другие, нетривиальные решения.

Система линейных уравнений может не иметь ни одного решения тогда она называется несовместной, например, в системе:

$$\begin{cases} x_1 + x_2 = 1 \\ x_1 + x_2 = 3 \end{cases}$$

левые части уравнений совпадают, а правые различны, поэтому никакие значения x1 и x2 не могут удовлетворить обоим уравнениям сразу.
не могут удовлетворить обоим уравнениям сразу.
Если же система линейных уравнений обладает решением, то она называется совместной. Совместная система называется определенной, если она обладает одним единственным решением, и неопределенной, если решений больше чем одно. Так, система

$$\begin{cases} x_1 + 2x_2 = 7 \\ x_1 + x_2 = 6 \end{cases}$$

определена и имеет единственное решение x1=5, x2=1 , а система уравнений

$$\begin{cases} 3x_1 - x_2 = 1 \\ 6x_1 - 2x_2 = 2 \end{cases}$$

неопределена, так как имеет бесконечное множество решений вида x1=k, x2=3k−1, где число k произвольно.

Совокупность всех решений неопределенной системы уравнений называется ее общим решением, а какое–то одно конкретное решение - частным. Частное решение, полученное из общего при нулевых значениях свободных переменных, называется базисным.

При определении совместности систем уравнений важную роль играет понятие ранга матрицы. Пусть дана матрица A размером n×m. Вычеркиванием некоторых строк или столбцов из нее можно получить квадратные матрицы k-го порядка, определители которых называются минорами порядка k матрицы A. Наивысший порядок не равных нулю миноров матрицы A называют рангом матрицы и обозначают r(A). Из определения вытекает, что r(A) $r(\mathrm{A}) \le \min(\mathrm{n,m})$, r(A)=0, только если матрица нулевая и r(A)=n для невырожденной матрицы n-го порядка. При элементарных преобразованиях (перестановка строк матрицы, умножение строк на число отличное от нуля и сложение строк) ранг матрицы не изменяется. Итак, если речь идет об исследовании системы на совместность, следует помнить, что система n линейных уравнений с m неизвестными:

• несовместна, если $r(\mathrm{A|b}) > r(\mathrm{A})$;

• совместна, если $r(\mathrm{A|b}) = r(\mathrm{A})$, причем при $r(\mathrm{A|b}) = r(\mathrm{A}) = m$ имеет единственное решение, а при $r(\mathrm{A|b}) = r(\mathrm{A}) < m$ имеет бесконечно много решений.

Существует не мало методов для практического отыскания решений систем линейных уравнений. Эти методы разделяют на точные и приближенные. Метод относится к классу точных, если с его помощью можно найти решение в результате конечного числа арифметических и логических операций. В этом разделе на конкретных примерах будут рассмотрены только точные методы решения систем.

Пример 3.2. Решить систему линейных уравнений при помощи правила Крамера.

Правило Крамера заключается в следующем. Если определитель det(A) матрицы системы A·x=b из n уравнений с n неизвестными отличен от нуля то система имеет единственное решение (x1, x2, ... , xn), определяемое по формулам Крамера $x_i = \frac{\det_i}{\det}$, где $\det_i$ – определитель матрицы, полученной из матрицы системы A заменой i–го столбца столбцом свободных членов b.

Итак, для решения поставленной задачи необходимо выполнить следующие действия:

• представить систему в матричном виде, то есть сформировать матрицу системы A и вектор правых частей b;

• вычислить главный определитель det(A);

• сформировать вспомогательные матрицы для вычисления определителей $\det_i$;

• вычислить определители $\det_i$;

• найти решение системы по формуле $x_i = \frac{\det_i}{\det}$.

Листинг 3.4 содержит решение поставленной задачи.

Листинг 3.4.

```
disp('Решение СЛАУ методом Крамера');
disp('Матрица системы:');
A=[2 -1 5;3 2 -5;1 1 -2]
disp('Вектор свободных коэффициентов:');
b=[0;1;4]
disp('Главный определитель:');
D=det(A)
disp('Вспомогательные матрицы:');
A1=A; A1(:,1)=b
A2=A; A2(:,2)=b
A3=A; A3(:,3)=b
disp('Вспомогательные определители:');
d(1)=det(A1);
d(2)=det(A2);
d(3)=det(A3);
```

```
d
disp('Вектор решений СЛАУ Ax=b');
x=d/D
disp('Проверка Ax-b=0');
A*x'-b
%-----------------------------------------------
% Решение СЛАУ методом Крамера
Матрица системы:
A =
      2 -1 5
      3 2 -5
      1 1 -2
Вектор свободных коэффициентов:
b =
    0
    1
    4
Главный определитель:
D = 6.0000
Вспомогательные матрицы:
A1 =
      0 -1 5
      1 2 -5
      4 1 -2
A2 =
      2 0 5
      3 1 -5
      1 4 -2
A3 =
      2 -1 0
      3 2 1
      1 1 4
Вспомогательные определители:
d =
    -17.000 91.000 25.000
Вектор решений СЛАУ Ax=b
x =
    -2.8333 15.1667 4.1667
Проверка Ax-b=0
ans =
        -4.4409e-15
```

3.5527e-15
8.8818e-16

Решение СЛАУ по формулам Крамера выглядит достаточно громоздко, поэтому на практике его используют довольно редко.

Пример 3.3. Решить систему линейных уравнений из задачи 3 **методом обратной матрицы**.

Метод обратной матрицы: для системы из n линейных уравнений с n неизвестными $A \cdot x=b$, при условии, что определитель матрицы A не равен нулю, единственное решение можно представить в виде $x=A^{-1} \cdot b$. Итак, для того, чтобы решить систему линейных уравнений методом обратной матрицы,

необходимо выполнить следующие действия:

• сформировать матрицу коэффициентов и вектор свободных членов заданной системы;

• решить систему, представив вектор неизвестных как произведение матрицы обратной к матрице системы и вектора свободных членов (листинг 3.5).

Листинг 3.5.

```
disp('Решение СЛАУ методом обратной матрицы');
disp('Матрица системы:');
A=[2 -1 5;3 2 -5;1 1 -2]
disp('Вектор свободных коэффициентов:');
b=[0;1;4]
disp('Вектор решений СЛАУ Ax=b');
x=A^(-1)*b
disp('Вектор решений СЛАУ Ax=b с помощью функции inv(A)');
x=inv(A)*b
disp('Проверка Ax=b');
R1=A*x
%----------------------------------------
>>Решение СЛАУ методом обратной матрицы
Матрица системы:
A =
2 -1 5
3 2 -5
1 1 -2
Вектор свободных коэффициентов:
b =
0
1
4
```

Вектор решений СЛАУ Ax=b
x =
-2.8333
15.1667
4.1667
Вектор решений СЛАУ Ax=b с помощью функции inv(A)
x =
-2.8333
15.1667
4.1667
Проверка Ax=b
ans =
-5.3291e-15
1.0000e+00
4.0000e+00

Решение систем линейных уравнений с помощью функции linsolve

linsolve(A,b) – встроенная функция Octave решает систему линейных алгебраических у равнений вида Ax=b.

Решение системы линейных уравнений
{x1+2x2=7; x1+x2=6}.
Свободные коэффициенты вводятся как вектор–столбец.
A=[1 2;1 1];b=[7;6];
x=linsolve(A,b)
x =
5.
1.

Результатом операции A*x-b является вектор достаточно близкий к нулю, это значит, что система решена верно.

A*x-b
ans =
1.0D-14 *
- 0.6217249
0.0888178

Решение системы {x1+x2–1=0; x1+x2–3=0}
A=[1 1;1 1]; b=[-1;-3];
Система не имеет решений:
linsolve(A,b)
warning: matrix singular to machine precision
warning: called from

```
   linsolve at line 107 column 7
ans =
        1.00000
        1.00000
```

Решение системы {3x1–x2=1; 6x1–2x2=2}.

В случае, когда система имеет бесконечное множество решений, Octave выдаст одно из них.

```
A=[3 -1;6 -2];
b=[1;2];
x=linsolve(A,b)
x =
    0.3
    - 0.1
```

Проверка не верна

```
A*x-b
ans =
        -1.1102e-016
        -2.2204e-016
```

Пример 3.4. Решить систему линейных уравнений методом Гаусса [6].

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 8 \\ x_1 - 3x_2 - 6x_4 = 9 \\ 2x_2 - x_3 + 2x_4 = -5 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = 0 \end{cases}$$

Решение системы линейных уравнений при помощи метода Гаусса основывается на том, что от заданной системы, переходят к эквивалентной системе, которая решается проще, чем исходная система.

Метод Гаусса состоит из двух этапов. Первый этап - это прямой ход, в результате которого расширенная матрица системы путем элементарных преобразований (перестановка уравнений системы, умножение уравнений на число отличное от нуля и сложение уравнений) приводится к ступенчатому виду:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right) \Rightarrow \left(\begin{array}{cccc|c} 1 & c_{12} & \dots & c_{1n} & d_1 \\ 0 & 1 & \dots & c_{2n} & d_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & d_n \end{array}\right)$$

На втором этапе (обратный ход) ступенчатую матрицу преобразовывают так, чтобы в первых n столбцах получилась единичная матрица:

$$\left(\begin{array}{cccc|c} 1 & 0 & \dots & 0 & x_1 \\ 0 & 1 & \dots & 0 & x_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & x_n \end{array}\right)$$

Последний, n+1 столбец этой матрицы содержит решение системы линейных уравнений.

Исходя из выше изложенного, порядок решения задачи в Octave (листинг 3.6) следующий:

• сформировать матрицу коэффициентов A и вектор свободных членов b заданной системы;

• сформировать расширенную матрицу системы, объединив A и b;

• используя функцию rref привести расширенную матрицу к ступенчатому виду;

• найти решение системы, выделив последний столбец матрицы, полученной в предыдущем пункте;

• выполнить вычисление A·x-b, если в результате получился нулевой вектор, задача решена верно.

Листинг 3.6.

```
disp('Решение СЛАУ методом Гаусса');
disp('Матрица системы:');
A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6]
disp('Вектор свободных коэффициентов:');
b=[8;9;-5;0]
disp('Расширенная матрица системы:');
C=rref([A b])
disp('Размерность матрицы C:');
n=size(C)
disp('Вектор решений СЛАУ Ax=b');
x=C(:,n(2))
disp('Проверка Ax-b');
A*x-b
%----------------------------------------
>> Решение СЛАУ методом Гаусса
Матрица системы:
A =
2 1 -5 1
1 -3 0 -6
0 2 -1 2
1 4 -7 6
Вектор свободных коэффициентов:
```

b =
8
9
-5
0
Расширенная матрица системы:
C =
1.00000 0.00000 0.00000 0.00000 3.00000
0.00000 1.00000 0.00000 0.00000 -4.00000
0.00000 0.00000 1.00000 0.00000 -1.00000
0.00000 0.00000 0.00000 1.00000 1.00000
Размерность матрицы C:
n =
4 5
Вектор решений СЛАУ Ax=b
x =
3.00000
-4.00000
-1.00000
1.00000
Проверка Ax-b
ans =
0.0000e+00
1.7764e-15
-8.8818e-16
-1.6653e-15

Пример 3.5. Исследовать систему на совместность и если возможно решить ее:

$$\text{а)}\ \begin{cases} x_1 + 2x_2 + 2x_3 = -9 \\ x_1 - x_2 + x_3 = 2 \\ 3x_1 - 6x_2 - x_3 = 25 \end{cases}$$

$$\text{б)}\ \begin{cases} x_1 - 5x_2 - 8x_3 + x_4 = 3 \\ 3x_1 + x_2 - 3x_3 - 5x_4 = 1 \\ x_1 - 7x_2 + 2x_4 = -5 \\ 11x_2 + 20x_3 - 9x_4 = 2 \end{cases}$$

$$\text{в)}\ \begin{cases} 4x_1 + x_2 - 3x_3 - x_4 = 0 \\ 2x_1 + 3x_2 + x_3 - 5x_4 = 0 \\ x_1 - 2x_2 - 2x_3 + 3x_4 = 0 \end{cases}$$

Для решения задачи введем исходные данные, то есть матрицу коэффициентов системы и вектор правых частей. Затем выполним вычисление рангов матрицы коэффициентов и расширенной матрицы системы.

В случае а) ранги матриц равны и совпадают с количеством неизвестных r(A|b)=r(A)=3, значит, система совместна и имеет единственное решение.

Вычисление рангов матрицы системы и расширенной матрицы системы б) показывает, что ранг расширенной матрицы больше ранга матрицы системы r(A|b)>r(A), что означает несовместность системы.

В процессе вычислений для системы в) выясняется, что ранг расширенной матрицы равен рангу матрицы системы r(A|b)=r(A)=3, но меньше, чем количество неизвестных системы r(A|b)=r(A)<4. Значит, система совместна, но имеет бесконечное множество решений.

Листинг 3.7

```
disp('Исследование системы на совместность');
disp('Введите матрицу системы:');
A=input('A=');
disp('Введите вектор свободных коэффициентов:');
b=input('b=');
disp('Размерность системы:');
[n,m]=size(A)
disp('Ранг матрицы системы:');
r=rank(A)
disp('Ранг расширенной матрицы:');
R=rank([A b])
if r==R
disp('Система совместна.');
if r==m
disp('Система имеет единственное решение.');
disp('Решение системы методом обратной матрицы:');
x=inv(A)*b
disp('Проверка Ax-b=0:');
A*x-b
else
disp('Система имеет бесконечно много решений.');
end;
else
disp('Система не совместна');
end;
%----------------------------------------
% Исследование системы а)
```

```
Исследование системы на совместность
Введите матрицу системы:
A= [1 2 5;1 -1 3; 3 -6 -1]
Введите вектор свободных коэффициентов:
b= [-9;2;25]
Размерность системы:
n = 3
m = 3
Ранг матрицы системы:
r = 3
Ранг расширенной матрицы:
R = 3
Система совместна.
Система имеет единственное решение.
Решение системы методом обратной матрицы:
x =
    2.0000
   -3.0000
   -1.0000
Проверка Ax-b=0:
ans =
    0.0000e+00
    8.8818e-16
    3.5527e-15
%----------------------------------------
% Исследование системы б)
Исследование системы на совместность
Введите матрицу системы:
A= [1 -5 -8 1;3 1 -3 -5;1 0 -7 2;0 11 20 -9]
Введите вектор свободных коэффициентов:
b= [3;1;-5;2]
Размерность системы:
n = 4
m = 4
Ранг матрицы системы:
r = 3
Ранг расширенной матрицы:
R = 4
Система не совместна
%----------------------------------------
% Исследование системы в)
```

Исследование системы на совместность
Введите матрицу системы:
A= [4 1 -3 -1;2 3 1 -5;1 -2 -2 4]
Введите вектор свободных коэффициентов:
b= [0;0;0]
Размерность системы:
n = 3
m = 4
Ранг матрицы системы:
r = 3
Ранг расширенной матрицы:
R = 3
Система совместна.
Система имеет бесконечное множество решений.

ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ №3

Решить систему линейных уравнений

Исследовать систему на совместность и если возможно решить ее: методами Крамера, Гаусса, обратной матрицы и спомощью функции linsolve

Вариант	Система линейных уравнений	Вариант	Система линейных уравнений
1	$\begin{cases} 2x_1+x_2+2x_3+3x_4=8 \\ 3x_1+3x_3=6 \\ 2x_1-x_2+3x_4=4 \\ x_1+2x_2-x_3+2x_4=4 \end{cases}$	**3**	$\begin{cases} 9x_1+10x_2-7x_3-x_4=23 \\ 7x_1-x_3-5x_4=37 \\ 5x_1-2x_3+x_4=22 \\ 4x_1+x_2+2x_3+3x_4=26 \end{cases}$
2	$\begin{cases} x_1+2x_2+3x_3+4x_4=22 \\ 2x_1+3x_2+x_3+2x_4=17 \\ x_1+x_2+x_3-x_4=8 \\ x_1-2x_3-3x_4=-7 \end{cases}$	**4**	$\begin{cases} 6x_1-x_2+10x_3-x_4=158 \\ 2x_1+x_2+10x_3+7x_4=128 \\ 3x_1-2x_2-2x_3-x_4=7 \\ x_1-12x_2+2x_3-x_4=17 \end{cases}$

Вариант	Система линейных уравнений	Вариант	Система линейных уравнений
5	$\begin{cases} x_1 - 2x_2 + 6x_3 + x_4 = 88 \\ 5x_1 + 2x_3 - 3x_4 = 88 \\ 7x_1 - 3x_2 + 7x_3 + 2x_4 = 181 \\ 3x_1 - 7x_2 + 5x_3 + 2x_4 = 99 \end{cases}$	**11**	$\begin{cases} 2x_1 - 8x_2 - 3x_3 - 2x_4 = -18 \\ x_1 - 2x_2 + 3x_3 - 2x_4 = 28 \\ x_2 + x_3 + x_4 = 10 \\ 11x_1 + x_3 + 2x_4 = 21 \end{cases}$
6	$\begin{cases} x_1 - 2x_2 - 8x_4 = 26 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = -8 \\ x_1 + x_2 - 5x_3 + x_4 = -10 \\ 2x_1 - x_2 + 2x_4 = 7 \end{cases}$	**12**	$\begin{cases} 2x_1 - x_2 + 4x_3 + x_4 = 66 \\ x_2 - 6x_3 + x_4 = -63 \\ 8x_1 - 3x_2 + 6x_3 - 5x_4 = 146 \\ 2x_1 - 7x_2 + 6x_3 - x_4 = 80 \end{cases}$
7	$\begin{cases} 2x_1 + 2x_2 + 6x_3 + x_4 = 15 \\ -x_2 + 2x_3 + x_4 = 18 \\ 4x_1 - 3x_2 + x_3 - 5x_4 = 37 \\ 3x_1 - 5x_2 + x_3 - x_4 = 30 \end{cases}$	**13**	$\begin{cases} 2x_1 - 3x_3 + 2x_4 = -16 \\ 2x_1 - x_2 + 13x_3 + 4x_4 = 213 \\ 3x_1 + x_2 + 2x_3 + x_4 = 72 \\ x_1 - 2x_3 - 5x_4 = -159 \end{cases}$
8	$\begin{cases} 4x_1 - 5x_2 + 7x_3 + 5x_4 = 165 \\ 2x_1 + x_2 - 3x_3 - x_4 = -15 \\ 9x_1 - 4x_3 - x_4 = 194 \\ x_1 - x_2 - 2x_3 - 3x_4 = -19 \end{cases}$	**14**	$\begin{cases} 7x_1 + 7x_2 - -7x_3 - 2x_4 = 25 \\ 3x_1 + 4x_2 + 5x_3 + 8x_4 = 60 \\ 2x_1 + 2x_2 + 2x_3 + x_4 = 27 \\ 2x_1 - 2x_3 - x_4 = -1 \end{cases}$
9	$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = -4 \\ x_1 - 3x_2 - 6x_4 = -7 \\ 2x_2 - x_3 + 2x_4 = 2 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = -2 \end{cases}$	**15**	$\begin{cases} 6x_1 - 9x_2 + 5x_3 + x_4 = 124 \\ 7x_2 - 5x_3 - x_4 = -54 \\ 5x_1 - 5x_2 + 2x_3 + 4x_4 = 83 \\ 3x_1 - 9x_2 + x_3 + 6x_4 = 45 \end{cases}$
10	$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 26 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 34 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 26 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 26 \end{cases}$	**16**	$\begin{cases} x_1 - 3x_2 - 8x_4 = 26 \\ x_1 + 4x_2 - 6x_3 + 6x_4 = -8 \\ x_1 + 2x_2 - 5x_3 + x_4 = -13 \\ 2x_1 - 2x_2 + x_4 = 7 \end{cases}$

ЛАБОРАТОРНАЯ РАБОТА № 4.
РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

НЕЛИНЕЙНЫЕ УРАВНЕНИЯ И СИСТЕМЫ

В общем случае аналитическое решение уравнения $f(x)=0$ можно найти только для узкого класса функций. Чаще всего приходится решать это уравнение численными методами.

Численное решение уравнения проводят в два этапа. На *первом этапе* отделяют корни уравнения, т.е. находят достаточно тесные промежутки, в которых содержится только один корень. Эти промежутки называют *интервалами изоляции корня*. Определить интервалы изоляции корня можно, например, изобразив график функции. Идея *графического метода* основана на том, что непрерывная функции $f(x)$ имеет на интервале $[a, b]$ хотя бы один корень, если она поменяла знак $f(a)\cdot f(b)<0$. Границы интервала a и b называют *пределами интервала изоляции*. На *втором этапе* проводят *уточнение* отделенных корней, т.е. находят корни с заданной точностью [9].

4.1. МНОГОЧЛЕНЫ И ДЕЙСТВИЯ НАД НИМИ

В общем виде многочлен (полиномом) может быть записан так:

$$a_0+a_1 x+a_2 x^2 +...+a_n x^n,$$

где x – переменная, a_i – коэффициенты полинома, n – его степень.

В Octave определить многочлен можно в виде вектора его коэффициентов $p=\{a_n, a_{n-1}, ... , a_1, a_0\}$. Например, полином $2x^5+3x^3-1$ задается вектором

```
>> p=[2,0,3,0,-1]
p = 2 0 3 0 -1
```

Построить многочлен по заданному вектору его корней позволяет функция poly(x), где x — вектор корней искомого полинома.

Пример 4.1. Записать алгебраическое уравнение, если известно, что его корни x1=−2, x2=3. Согласно листингу 4.1 решение задачи имеет вид: $x^2-x-6=0$

Листинг 4.1.

```
x=[-2 3];
poly(x)
>>ans = 1 -1 -6
```

Вычислить значение многочлена в заданной точке можно с помощью функции polyval(p, x), где p - многочлен степени n, x - значение, для которого нужно определить многочлен.

Пример 4.2. Вычислить значение многочлена $x^6-x^5+3x^4-8x^2+x-10$ в точках x1=−1, x2=1. Решение представлено в листинге 4.2.

Листинг 4.2.

```
p=[1 -1 3 0 -8 1 -10];
x=[-1,1];
polyval(p,x)
>>ans =
        -14 -14
```

4.2. РЕШЕНИЕ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Любое уравнение P(x)=0, где P(x) это многочлен (полиномом), отличный от нулевого, называется алгебраическим уравнением относительно переменных x. Всякое алгебраическое уравнение относительно x можно записать в виде

$$a_0+a_1 x+a_2 x^2+...+a_n x^n=0,$$

где, a_i – коэффициенты алгебраического уравнения n–й степени. Например, линейное уравнение - это алгебраическое уравнение первой степени, квадратное – второй, кубическое – третьей и так далее.

Пример 4.3. Записать алгебраическое уравнение, если известно, что его корни x1=−2, x2=3. Согласно листингу 4.3 решение задачи имеет вид:

Листинг 4.3.

```
x^2−x−6=0
x=[-2 3];
poly(x)
>>ans = 1 -1 -6
```

Решить алгебраическое уравнение P(x)=0 можно при помощи встроенной функции

roots(p),

где p - многочлен степени n.

Функция формирует вектор, элементы которого являются корнями заданного полинома.

Пример 4.4. Решить алгебраическое уравнение x2−x−6=0.

Из листинга 4 видно, что значения x1=−2, x2=3 являются решением уравнения.

Листинг 4.4.

```
p=[1 -1 -6];
roots(p)
>>ans =
        3
        -2
```

Пример 4.5. Найти корни полинома 2 x3−3 x2−12 x−5=0.

Листинг 4.5.

```
>> p=[2 -3 -12 -5];
x=roots(p)
>>x =
        3.44949
        -1.44949
        -0.50000
```

Графическое решение заданного уравнения показано в листинге 4.5 и на рис. 4.1.

Точки пересечения графика с осью абсцисс и есть корнями уравнения. Не трудно заметить, что графическое решение совпадает с найденным в листинге 4.5.

Листинг 4.6.

```
okno1=figure();
x=-2:0.1:5.5;
y=2*x.^3-3*x.^2-12*x-5;
pol=plot(x,y);
set(pol,'LineWidth',3,'Color','k')
set(gca,'xlim',[-2,4]);
set(gca,'ylim',[-30,30]);
225
set(gca,'xtick',[-2:0.5:4]);
set(gca,'ytick',[-30:5:30]);
grid on;
xlabel('x');
ylabel('y');
title('Plot y=2*x^3-3*x^2-12*x-5');
```

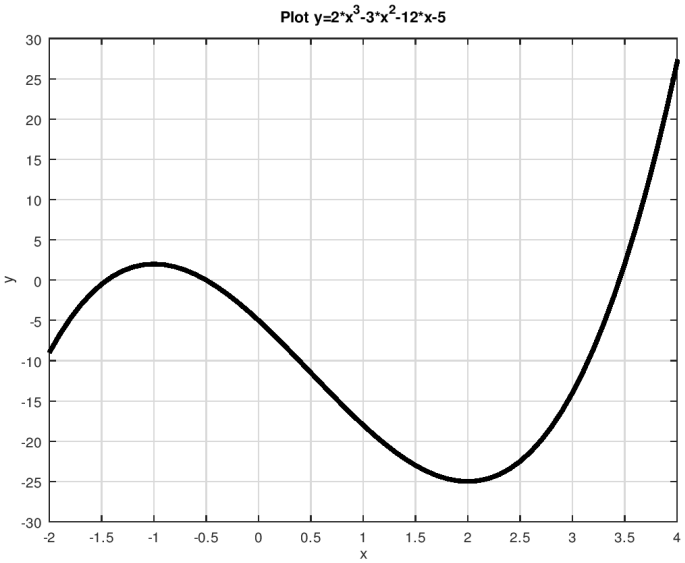


Рисунок 4.1 – Графическое решение примера 4.5

Пример 4.6. Найти решение уравнения $x^4+4x^3+4x^2-9=0$.

Графическое решение задачи было получено при помощи последовательности команд приведенных в листинге 4.7.

Листинг 4.7.

```
okno1=figure();
x=-4:0.1:2;
y=x.^4+4*x.^3+4*x.^2-9;
cla;
pol=plot(x,y);
set(pol,'LineWidth',3,'Color','k')
set(gca,'xlim',[-4,2]);
set(gca,'ylim',[-10,5]);
set(gca,'xtick',[-4:0.5:2]);
set(gca,'ytick',[-10:1:5]);
grid on;
xlabel('x');
ylabel('y');
title('Plot y=x^4+4*x^3+4*x^2-9');
```

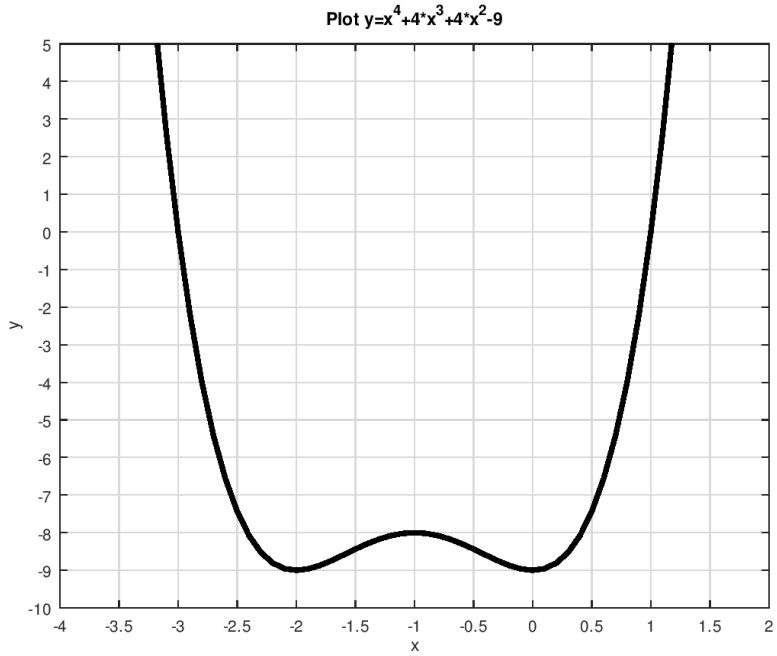


Рисунок 4.2

На рис. 2 видно, что заданное алгебраическое уравнение имеет два действительных корня. Решение задачи, представленное в листинге 8 показывает не только действительные, но и комплексные корни.

Листинг 4.8.

```
p=[1 4 4 0 -9];
x=roots(p)
>>x =
        -3.00000 + 0.00000i
        -1.00000 + 1.41421i
        -1.00000 - 1.41421i
        1.00000 + 0.00000i
```

4.3. РЕШЕНИЕ ТРАНСЦЕНДЕНТНЫХ УРАВНЕНИЙ

Уравнение, в котором неизвестное входит в аргумент трансцендентных функций, называется трансцендентным уравнением. К трансцендентным уравнениям принадлежат показательные, логарифмические, тригонометрические [9].

Для решения трансцендентных уравнений вида f(x)=0 в Octave существует функция fzero(name, x0) или fzero(name, [a, b]), где name – имя функции, вычисляющей левую часть уравнения, x0 – начальное приближение к корню или интервал изоляции корня [a, b].

Если функция вызывается в формате:

[x, y]= fzero(name, x0),

то здесь x – корень уравнения, y – значение функции в точке x.

Пример 4.7. Найти решение уравнения:

$$\sqrt[3]{(2x-3)^2} - \sqrt[3]{(x-1)^2} = 0.$$

Начнем решение данного трансцендентного уравнения с определения интервала изоляции корня. Воспользуемся для этого графическим методом. Построим график функции, указанной в левой части уравнения (листинга 4.9), создав предварительно функцию для ее определения.

Листинг 4.9.

```
%Функция для вычисления левой части уравнения f(x)=0
function y=f1(x)
y=((2*x-3).^2).^(1/3)-((x-1).^2).^(1/3);
end;
% Построение графика функции f(x)
okno1=figure();
x=-1:0.1:3;
y=f1(x);
cla;
pol=plot(x,y);
set(pol,'LineWidth',3,'Color','k')
set(gca,'xlim',[-1,3]);
```

```
set(gca,'ylim',[-1,1.5]);
set(gca,'xtick',[-1:0.5:3]);
set(gca,'ytick',[-1:0.5:1.5]);
grid on;
xlabel('x');
ylabel('y');
title('Plot y=(2x-3)^(2/3)-(x-1)^(2/3)');
```

На графике (рис. 3) видно, что функция f(x) дважды пересекает ось O x . Первый раз на интервале [1, 1.5], второй - [1.5, 2.5].

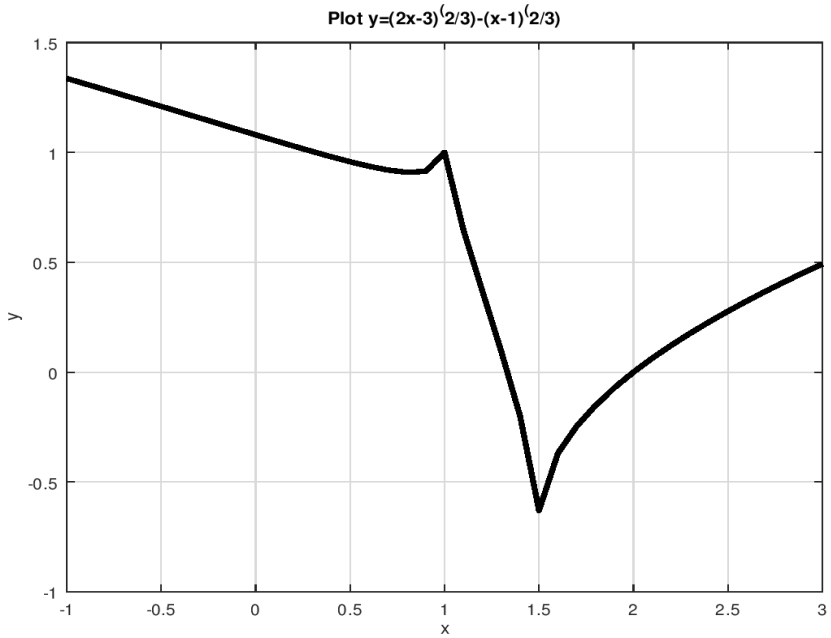


Рисунок 4.3

Уточним корни, полученные графическим методом. Создадим функцию, вычисляющуюлевую часть заданного уравнения (листинг 4.10) и обратимся к функции fzero, указав в качестве параметров имя созданной функции и число близкое к первому корню:

Листинг 4.10.

```
x1=fzero('f1', 1)
>>x1=1.3333
```

Теперь применим функцию fzero, указав в качестве параметров имя функции,

вычисляющей левую часть тождества и интервал изоляции второго корня:

Листинг 4.11.

```
x2=fzero('f1', [1.5 2.5])
>>x1= 2
```

Не трудно заметить, что и в первом и во втором случае функция fzero правильно нашла корни заданного уравнения.

Листинг 4.12 содержит пример некорректного обращения к функции fzero, здесь интервал изоляции корня задан неверно. На графике видно, что на концах этого интервала функция знак не меняет. Или, другими словами, выбранный интервал содержит сразу два корня.

Листинг 4.12.

```
fzero('f1', [1 3])
>>error: fzero: not a valid initial bracketing
error: called from:
error: /usr/share/octave/3.2.3/m/optimization/fzero.m at
line 137, column 5
```

В листинге 4.13 приведен пример обращения к функции fzero в полном формате:

Листинг 4.13.

```
[X(1),Y(1)]=fzero('f1', [1 1.5]);
[X(2),Y(2)]=fzero('f1', [1.5 2.5]);
X
Y
%Решение уравнения
>>X =
1.3333 2.0000
%Значения функции в точке X
>>Y =
-2.3870e-15 0.0000e+00
```

Пример 4.8. Найти решение уравнения $x^4+4x^3+4x^2-9=0$.

Как видим, левая часть уравнения представляет собой полином. Данное уравнение имеет четыре корня: два действительных и два мнимых.

Листинг 4.14 демонстрирует решение алгебраического уравнения при помощи функции fzero. Не трудно заметить, что результатом работы функции являются только действительные корни. Графическое решение подтверждает это: функция дважды пересекает ось абсцисс.

Листинг 4.14.

```
function y=f2(x)
y=x.^4+4*x.^3+4*x.^2-9;
end;
```

```
[X(1),Y(1)]=fzero('f2', [-4 -2]);
[X(2),Y(2)]=fzero('f2', [0 2]);
X
Y
>>X =
-3 1
>>Y =
0 0
```

4.4. РЕШЕНИЕ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Напомним, что если заданы m уравнений с n неизвестными и требуется найти последовательность из n чисел, которые одновременно удовлетворяют каждому из m уравнений, то говорят о системе уравнений.

Несложную систему элементарной подстановкой можно привести к нелинейному уравнению. Рассмотрим несколько примеров, в которых описан такой прием решения нелинейной системы.

Пример 4.9. Решить систему уравнений:

$$\begin{cases} x^2 + 2y^2 = 1 \\ x - y = 1 \end{cases}$$

Найдем графическое решение с помощью команд листинга 4.15. На рис.4.4 видно, что система имеет два решения.

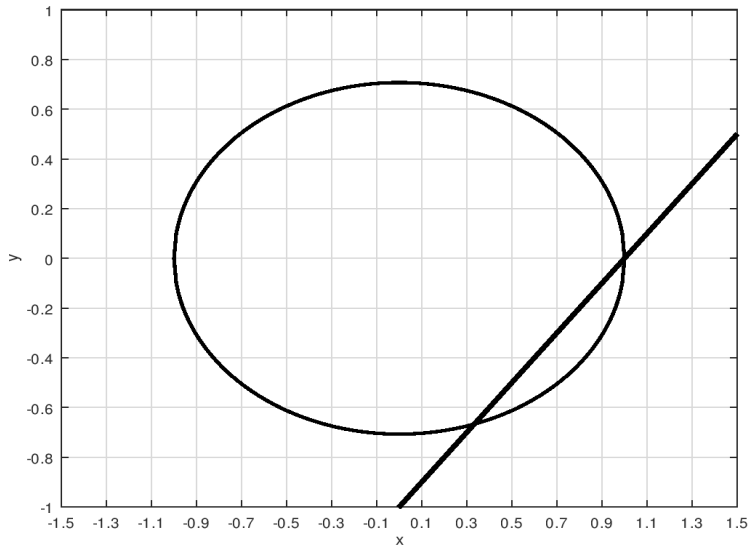


Рисунок 4.4 – Решение системы нелинейных уравнений из задачи 4.9

Листинг 4.15.

```
% Верхняя часть эллипса
function y=f1(x)
y=sqrt((1-x.^2)/2);
```

```
end;
%Нижняя часть эллипса
function y=f2(x)
y=-sqrt((1-x.^2)/2);
end;
% Прямая
function y=f3(x)
y=x-1;
end;
%Построение графика
okno1=figure();
x1=-1:0.01:1;
x2=-1.5:0.1:1.5;
cla;
L1=plot(x1,f1(x1),x1,f2(x1));
set(L1,'LineWidth',2,'Color','k')
hold on
L2=plot(x2,f3(x2));
set(L2,'LineWidth',3,'Color','k')
set(gca,'xlim',[-1.5,1.5]);
set(gca,'ylim',[-1,1]);
set(gca,'xtick',[-1.5:0.2:1.5]);
set(gca,'ytick',[-1:0.2:1]);
grid on;
xlabel('x');
ylabel('y');
```

Не сложно убедиться, что данная система легко сводится к одному уравнению:

$\{y=x-1,\ x^2+2\,y^2=1\} \Rightarrow x^2+2(x-1)^2-1=0 \Rightarrow 3\,x^2-4\,x+1=0.$

Решив это уравнение с помощью функции roots, найдем значения x. Затем подставим их в одно из уравнений системы, например во второе, и тем самым вычислим значения y.

Листинг 4.16 содержит решение данной задачи. Понятно, что система имеет два решения

x1=1, y1=0 и x2=0.333, y1=−0.666 (рис. 4.4). Графическое решение уравнения (листинг 4.17), к которому сводится система, показано на рис. 4.5.

Листинг 4.16.

```
>> p=[3 -4 1];
x=roots(p)
```

```
y=x-1
>>x =
        1.00000
        0.33333
>>y =
        -1.1102e-16
        -6.6667e-01
```

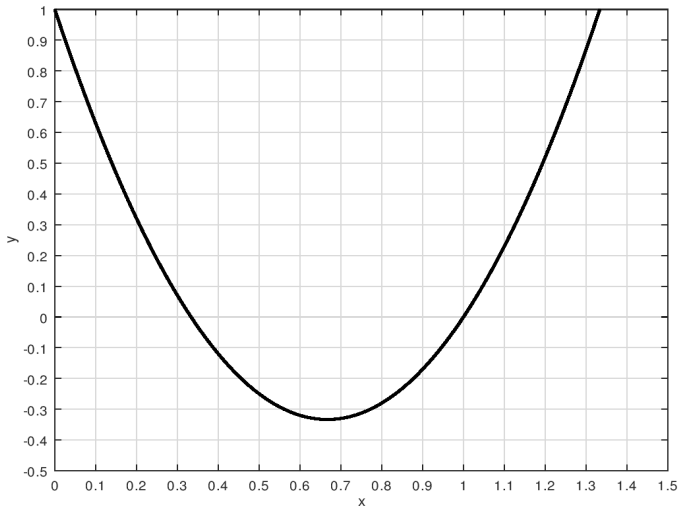


Рисунок 4.5 – Графическое решение задачи 4.17

Листинг 4.17.

```
function y=f(x)
y=3*x.^2-4*x+1;
end;
okno1=figure();
x=0:0.01:1.5;
cla;
L=plot(x,f(x));
set(L,'LineWidth',2,'Color','k')
231
set(gca,'xlim',[0,1.5]);
set(gca,'ylim',[-0.5,1]);
set(gca,'xtick',[0:0.1:1.5]);
set(gca,'ytick',[-0.5:0.1:1]);
grid on;
xlabel('x');
ylabel('y');
```

Пример 4.10. Решить систему уравнений:

$$\begin{cases} \sin(x+1)-y=1.2 \\ 2\,x+\cos(y)=2. \end{cases}$$

Проведем элементарные алгебраические преобразования и представим систему в виде одного уравнения:

{y=sin(x+1)-1.2, 2x+cos(y)-2=0}⇒2 x+cos(sin(x+1)-1.2)-2=0

Рис. 4.6 содержит графическое решение уравнения (листинг 4.18), к которому сводится система, его удобно использовать для выбора начального приближения функции fzero.

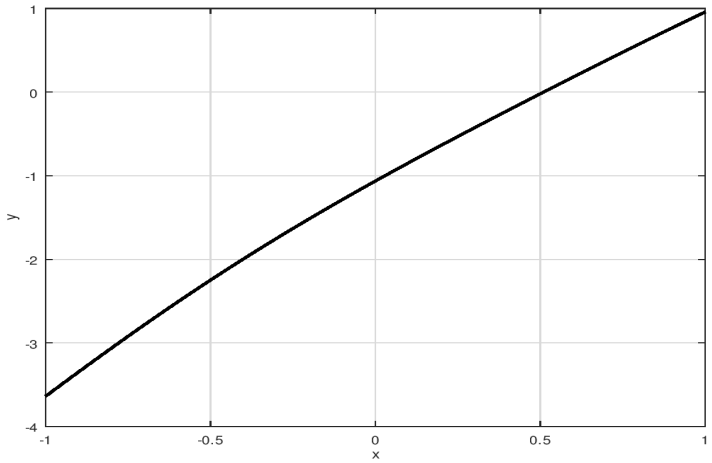


Рисунок 4.6 – Графическое решение задачи 10

Листинг 4.18

```
function y=fun(x)
z=sin(x+1)-1.2;
y=2*x+cos(z)-2;
end;
okno1=figure();
x=-1:0.01:1;
cla;
L=plot(x,fun(x));
set(L,'LineWidth',2,'Color','k');
grid on;
xlabel('x');
ylabel('y');
%Решение системы
X=fzero('fun',0)
Y=sin(X+1)-1.2
>>X = 0.51015
>>Y = -0.20184
```

Решить систему нелинейных уравнений, или одно нелинейное уравнение в Octave можно с помощью функции

fsolve(fun, x0),

где fun – имя функции, которая определяет левую часть уравнения f(x)=0 или системы уравнений F(x)=0 (она должна принимать на входе вектор аргументов и возвращать вектор значений), x0 – вектор приближений, относительно которого будет осуществляться поиск решения.

Листинг 4.19. Найти решение системы нелинейных уравнений:

$$\begin{cases} \cos(x) + 2y = 2 \\ \dfrac{x^2}{3} - \dfrac{y^2}{3} = 1. \end{cases}$$

Решим систему графически, для чего выполним перечень команд указанных в листинге 4.19. Результат работы этих команд показан на рис. 4.7. Понятно, что система имеет два корня.

Листинг 4.20

```
% Уравнения, описывающие линии гиперболы
function y=f1(x)
y=sqrt(x.^2-3);
end;
function y=f2(x)
y=-sqrt(x.^2-3);
end;
%Уравнение косинусоиды
function y=f3(x)
y=1-cos(x)/2;
end;
% Построение графика
okno1=figure();
x1=-5:0.001:-sqrt(3);
x2=sqrt(3):0.001:5;
x3=-5:0.1:5;
cla;
%Гипербола
L1=plot(x1,f1(x1),x1,f2(x1),x2,f1(x2),x2,f2(x2));
set(L1,'LineWidth',3,'Color','k')
hold on
%Косинусоида
L2=plot(x3,f3(x3));
set(L2,'LineWidth',3,'Color','k')
grid on;
xlabel('x');
```

```
ylabel('y');
```

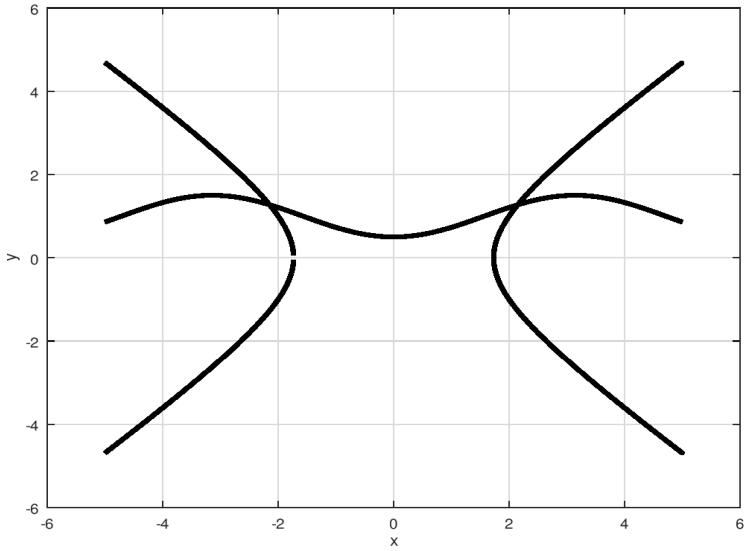


Рисунок 4.7 – Графическое решение задачи 4.10

Для численного решения задачи составим функцию, соответствующую левой части системы (листинг 4.20). Здесь важно помнить, что все уравнения должны иметь вид F(x)=0. Кроме того, обратите внимание, что x и y в этой функции — векторы (x — вектор неизвестных, y — вектор решений).

Листинг 4.21

```
function [y]=fun(x)
y(1)=cos(x(1))+2*x(2)-2;
y(2)=x(1)^2/3-x(2)^2/3-1;
end;
% Теперь решим систему, указав в качестве % начального
% приближения в начале вектор [−3,−1], затем [1, 3].
[X1_Y1]=fsolve('fun', [-3 -1])
[X2_Y2]=fsolve('fun', [1 3])
>>X1_Y1 =
            -2.1499 1.2736
>>X2_Y2 =
            2.1499 1.2736
```

Понятно, что решением задачи являются пары x1=−2.15 , y1=1.27 и x2=2.15 , y2=1.27 , что соответствует графическому решению (рис. 7).

Если функция решения нелинейных уравнений и систем имеет вид [x, f, ex] = fsolve(fun, x0), то здесь x – вектор решений системы, f – вектор значений уравнений системы для найденного значения x, ex

– признак завершения алгоритма решения нелинейной системы, отрицательное значение параметра ex означает, что решение не найдено, ноль – досрочное прерывание вычислительного процесса при достижении максимально допустимого числа итераций, положительное значение подтверждает, что решение найдено с заданной точностью.

Пример 4.11. Решить систему нелинейных уравнений:

$$\begin{cases} x_1^2 + x_2^2 + x_3^2 = 1 \\ 2x_1^2 + x_2^2 - 4x_3 = 0 \\ 3x_1^2 - 4x_2^2 + x_3^2 = 0. \end{cases}$$

Листинг 4.22 содержит функцию заданной системы и ее решение. Обратите внимание на выходные параметры функции fsolve. В нашем случае значения функции f для найденного решения x близки к нулю и признак завершения ex положительный, значит, найдено верное решение.

Листинг 4.22

```
function f=Y(x)
f(1)=x(1)^2+x(2)^2+x(3)^2-1;
f(2)=2*x(1)^2+x(2)^2-4*x(3);
f(3)=3*x(1)^2-4*x(2)+x(3)^2;
end
[x,f,ex]=fsolve('Y',[0.5 0.5 0.5])
>>x =
        0.78520 0.49661 0.36992
f =
     1.7571e-08 3.5199e-08 5.2791e-08
ex = 1
```

Пример 4.12. Решить систему:

$$\begin{cases} x^2 + y^2 = 1 \\ 2\sin(x-1) + y = 1. \end{cases}$$

Графическое решение системы (рис. 4.8) показало, что она корней не имеет. Рисунок был получен с помощью программы листинга 4.23.

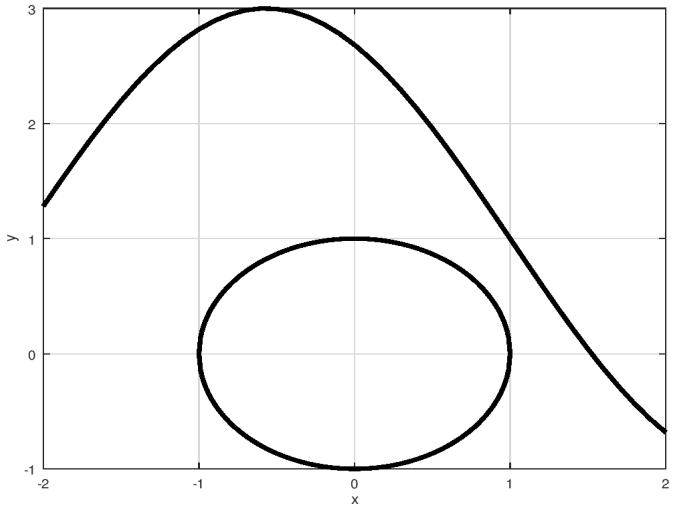


Рисунок 4.8 – Графическое решение задачи 4.12

Листинг 4.23

```
%Уравнения линий окружности
function y=f1(x)
y=sqrt(1-x.^2);
end;
235
function y=f2(x)
y=-sqrt(1-x.^2);
end;
%Уравнение синусоиды
function y=f3(x)
y=1-2*sin(x-1);
end;
okno1=figure();
x1=-1:0.01:1;
x3=-2:0.1:2;
cla;
%Окружность
L1=plot(x1,f1(x1),x1,f2(x1));
set(L1,'LineWidth',3,'Color','k')
hold on
%Синусоида
L2=plot(x3,f3(x3));
set(L2,'LineWidth',3,'Color','k')
grid on;
xlabel('x');
ylabel('y');
```

Однако применение к системе функции fsolve дает положительный ответ, что видно из листинга 4.23. Происходит это потому, что алгоритм, реализованный в этой функции, основан на минимизации суммы квадратов компонент вектор–функции. Следовательно, функция fsolve в этом случае нашла точку минимума, а наличие точки минимума не гарантирует существование корней системы в ее окрестности, т.е. необходима дополнительная проверка.

Листинг 4.24

```
function [y]=fun(x)
y(1)=x(1)^2+x(2)^2-1;
y(2)=2*sin(x(1)-1)+x(2)-1;
end;
[X1_Y1, f, ex]=fsolve('fun', [1 1])
X1_Y1 =
            1.04583   0.52344
f =
    0.36775  -0.38493
ex = 3
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 4

Решение нелинейных уравнений

Вариант	Функция	Вариант	Функция
1	$x - \sin x = 0{,}25$ $x^3 - 3x^2 + 9x - 8 = 0$	17	$x^2 \cos(2x + 1) = -1$ $2x^4 - 3x^2 + 15 = 0$
2	$\mathrm{tg}(0{,}2 + x) = x^3 + 3$ $x^3 + 2x^2 - 7x + 1 = 0$	18	$0{,}5^x - 1 = (x + 2)^2$ $3x^3 - 2x^2 + x + 1 = 0$
3	$(x + 1)^{1/3} - \cos(0{,}3 + 0{,}4x) = 2$ $x^3 + x^2 - 3x + 4 = 0$	19	$\cos(x + 2) - x + 2x + 1 = 0$ $x^4 + 2x^2 + 3x - 10 = 0$
4	$\mathrm{tg}(2{,}3 + 0{,}5x) = 3x + 2$ $2x^3 - 3x^2 + x + 1 = 0$	20	$x + \ln(2x + 3) = 0{,}5$ $-2^{x-1} - x = 0$
5	$2e^{x+1} + 3x + 1 = 0$ $3x^4 + 4x^3 - 12x - 1 = 0$	21	$x^2 + 4\sin(x + 1) = 0$ $3x^4 + 4x^3 - 12x^2 - 5 = 0$
6	$3x^2 + \cos(2x + 1) = 1$ $x^4 - 2x^3 - 10x^2 - 2 = 0$	22	$e^{2x+1} + 5x - 1 = 0$ $7x^3 - 2x^2 + 3x - 10 = 0$
7	$5x\sin(2x + 1) = 0{,}43$ $x^3 - 7x^2 + 2x - 1 = 0$	23	$2e^{x+1} - 3x + 1 = 0$ $x\lg(x^2 + 2x - 1) = 1$
8	$x\cos(x + 2) = x^2 - 3x + 1$ $x^3 + x^2 + x - 10 = 0$	24	$x^2 \cos(2x - 1) = 1$ $2x^4 - 3x^3 + 3x^2 - x + 1 = 0$
9	$(x - 3)\cos(x + 2) = 1$ $2x^4 - 3x^3 + x - 1 = 0$	25	$2x - \lg(x + 3) = 7$ $\mathrm{tg}^3(x) + x - 1 = 0$
10	$\sin(x + \pi/3) + 0{,}5x + 2 = 0$ $x4 - x - 1 = 0$	26	$(1 - x)e^{3x-1} = 0{,}5$ $3\sin^2(x + 1) - x^2 + x = 2$
11	$x\lg(x + 1) = 1$ $2x^3 - 9x^2 - 60x = 0$	27	$2\sin(x - \pi/6) = x^2 - 0{,}5$ $x^4 + 4x^3 - 8x^2 - 17 = 0$
12	$\mathrm{arctg}x - 1/(3x^3) = 0$ $2x^4 + x - 3 = 0$	28	$5\cos(x + 3) = x - 0{,}5$ $3^x + 2 - x = 0$
13	$\ln x + (x + 1)^3 = 0$ $x^2 - 2 + 0{,}5^x = 0$	29	$x = (\log(x + 2))^{1/2} - 1$ $x^4 - x^3 - 2x^2 + 3x - 3 = 0$
14	$\cos(x + 0{,}5) = x^3$ $2x^4 + x - 3 = 0$	30	$(x - 2)^3 \lg(x - 3) = 1$ $2x^3 - 9x^2 - 60x + 1 = 0$
15	$(x - 4)^2 \log_2(x - 3) = 1$ $3x^4 + 8x^3 + 2x - 1 = 0$	31	$(x^2 + 2x - 20)\sin(x + 1) = 1$ $e^x = (x + 1)^3$
16	$e^{-2x} - 2x + 1 = 0$ $x^4 + 4x^3 - 8x^2 - 17 = 0$	32	$3\cos(x + 1)^2 = 2x + 1$ $3x^4 + 4x^3 - 12x^2 + 1 = 0$

Решение системы нелинейных уравнений

Вариант	Система нелинейных уравнений	Вариант	Система нелинейных уравнений
1	$\begin{cases} \sin x + 2y = 2 \\ \cos(y-1) + x = 0{,}7 \end{cases}$	**9**	$\begin{cases} \sin y + x = -0{,}4 \\ 2y - \cos(x+1) = 0 \end{cases}$
2	$\begin{cases} \sin(x+0{,}5) - y = 1 \\ \cos(y-2) + x = 0 \end{cases}$	**10**	$\begin{cases} \sin(x+2) - y = 1{,}5 \\ \cos(y-2) + x = 0{,}5 \end{cases}$
3	$\begin{cases} \cos x + y = 1{,}5 \\ 2x - \sin(y-0{,}5) = 1 \end{cases}$	**11**	$\begin{cases} \cos(x+0{,}5) - y = 2 \\ \sin y - 2x = 1 \end{cases}$
4	$\begin{cases} \cos(x+0{,}5) + y = 0{,}8 \\ \sin y - 2x = 1{,}6 \end{cases}$	**12**	$\begin{cases} \cos(x-2) + y = 0 \\ \sin(y+0{,}5) - x = 1 \end{cases}$
5	$\begin{cases} \sin(x-1) = 1{,}3 - y \\ x\sin(y+1) = 0{,}8 \end{cases}$	**13**	$\begin{cases} \cos(x+0{,}5) + y = 1 \\ \sin(y+0{,}5) - x = 1 \end{cases}$
6	$\begin{cases} \cos(x+0{,}5) - y = 2 \\ \sin y - 2x = 1 \end{cases}$	**14**	$\begin{cases} \sin(x) - 2y = 1 \\ \cos(y+0{,}5) - x = 2 \end{cases}$
7	$\begin{cases} -\sin(x+1) + y = 0{,}8 \\ \sin(y-1) + x = 1{,}3 \end{cases}$	**15**	$\begin{cases} 2y - \sin(x-0{,}5) = 1 \\ \cos(y) + x = 1{,}5 \end{cases}$
8	$\begin{cases} \sin(x) - 2y = 1 \\ \sin(y-1) + x = 1{,}3 \end{cases}$	**16**	$\begin{cases} \cos(x+0{,}5) - y = 0{,}8 \\ \sin(y-2) + x = 1 \end{cases}$

ЛАБОРАТОРНАЯ РАБОТА № 5.
РЕШЕНИЕ ОПТИМИЗАЦИОННЫХ ЗАДАЧ

В данной лабораторной работе рассматриваются решения задач поиска минимума (максимума) в **Octave**. В первой части на примерах решения практических задач рассматривается функция sqp, предназначенная для поиска минимума функции одной или нескольких переменных с ограничениями. Вторая часть целиком посвящена задачам линейного программирования.

Изучение оптимизационных задач начнём с обычных задач поиска минимума (максимума) функции одной или нескольких переменных.

5.1. ПОИСК ЭКСТРЕМУМА ФУНКЦИИ

Для решения классических оптимизационных задач с ограничениями в Octave можно воспользоваться следующей функцией

[x, obj, info, iter] = sqp(x0, phi, g, h, lb, ub, maxiter, tolerance), которая предназначена для решения следующей оптимизационной задачи.

Найти минимум функции $\varphi(x)$ при следующих ограничениях $g(x) = 0$, $h(x) \geq 0$, $lb \leq x \leq ub$. Функция sqp при решении задачи оптимизации использует метод квадратичного программирования.

Аргументами функции sqp являются:

• x0 - начальное приближение значения x,

• phi - оптимизируемая функция $\varphi(x)$,

• g и h - функции ограничений $g(x) = 0$ и $h(x) \geq 0$,

• lb и ub - верхняя и нижняя границы ограничения $lb \leq x \leq ub$,

• maxiter - максимальное количество итераций, используемое при решении оптимизационной задачи, по умолчанию эта величина равна 100,

• tolerance - точность ε, определяющая окончание вычислений, вычисления прекращаются при достижении точности $\sqrt{\varepsilon}$.

Функция sqp возвращает следующие значения:

• x - точка, в которой функция, достигает своего минимального значения,

• obj - минимальное значение функции,

• info - параметр, характеризующий корректность решения оптимизационной задачи, (если функция sqp возвращает значение info = 101 (104), то задача решена правильно),

• iter - реальное количество итераций при решении задачи.

Рассмотрим несколько примеров использования функции sqp при решении задач поиска экстремума функции одной переменной без ограничений.

Пример 5.1.

Найти минимум функции φ(x) = x4 +3x3 −13x2 − 6x + 26

При решении задачи оптимизации с помощью функции sqp необходимо иметь точку начального приближения. Построим график функции φ(x) (см. рис. 1). Из графика видно, что функция имеет минимум в окрестности точки x = −4. В качестве точки начального приближения выберем x0 = −3. Решение задачи представлено в листинге 5.1.

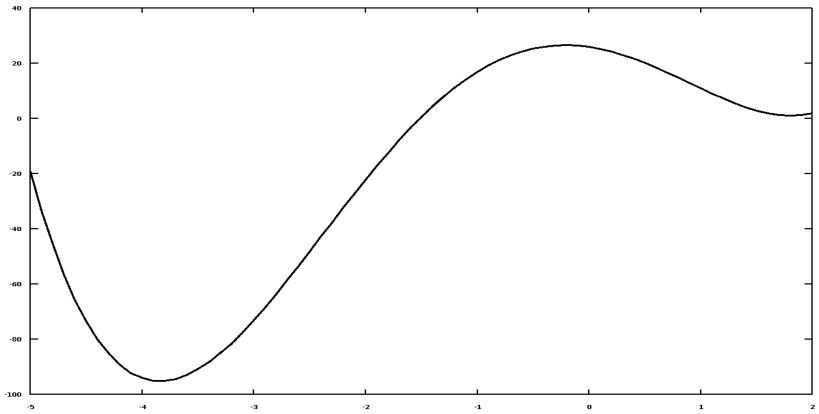


Рисунок 5.1. – График функции φ(x) = x4 + 3x3 − 13x2 − 6x + 26

Листинг 5.1. Поиск минимума функции (пример 5.1)

```
function obj = phi (x)
obj = x^4+3∗x^3−13∗x^2−6∗x+26;
end % (endfunction) Можно сокращённо
[x, obj, info, iter]=sqp (-3,@phi)
% Результаты решения
x = −3.8407
obj = −95.089
info = 104
iter = 5
```

Минимум функции φ(x) = −95.089 достигается в точке x = −3.8407, количество итераций равно 5, параметр info = 104 свидетельствует о корректном решении задачи поиска минимума φ(x) = x4 + 3x3 − 13x2 − 6x + 26.

Рассмотрим пример поиска минимума функции нескольких переменных.

Пример 5.2.

Найти минимум функции Розенброка1

$f(x, y) = N(y - x^2)^2 + (1 - x^2)^2$.

Построим график функции Розенброка для N = 20 (см. листинг 2). График полученной поверхности приведён на рис. 6.2.

Листинг 5.2.

График функции Розенброка для N = 20

```
[x y]=meshgrid(-2:0.1:2,2-0.1:-2);
z=20*(y−x.^2).^2+(1−x).^2;
surf(x, y, z);
```

Как известно, функция Розенброка имеет минимум в точке (1,1) равный 0. В виду своей специфики функция Розенброка является тестовой для алгоритмов минимизации. Найдём минимум этой функции с помощью функции sqp (см. листинг 5.3).

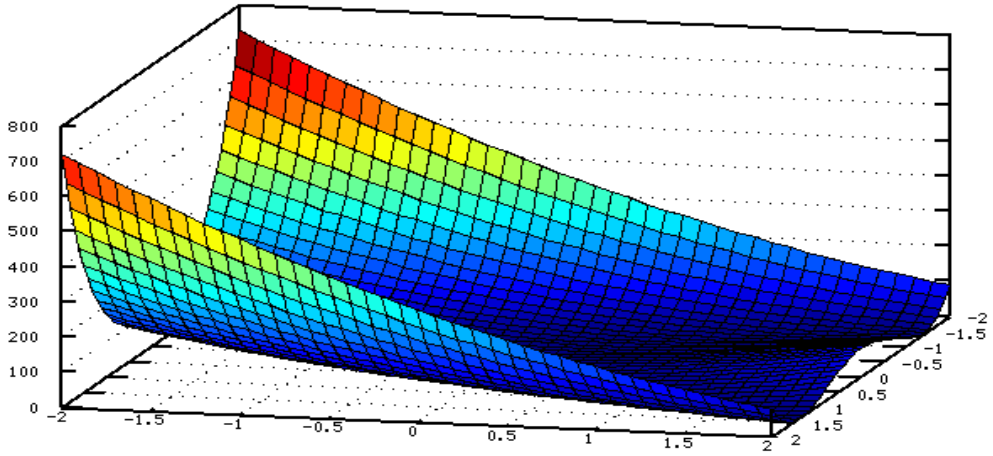


Рис. 5.2. График функции Розенброка

При решении задач на экстремум функций многих переменных следует учитывать особенности синтаксиса при определении оптимизируемой функции. Аргументом функции многих переменных (в нашем случае - её имя r) является массив x, первая переменная имеет имя x(1), вторая x(2) и т. д. Если имя аргумента функции многих переменных будет другим - допустим m, то изменятся и имена переменных: m(1), m(2), m(3) и т.д.

Листинг 5.3.

Вычисление минимума функции Розенброка (N = 20)

```
function y=r (x)
y=20*(x(2)−x(1)^2)^2+(1−x(1))^2;
end (endfunction)
x0=[0;0];
[x,obj,info,iter]=sqp(x0,@r)
```

Результаты вычислений

x =
1.00000
1.00000
obj = 7.1675 e−13
info = 101
iter = 14

Как и следовало ожидать, функция sqp нашла минимум в точке (1,1), само значение 0 найдено достаточно точно (7.2·10−13). Значение info = 101 говорит о корректном решении задачи, для нахождения минимального значения функции Розенброка потребовалось 14 итераций.

Таким образом, функция sqp предназначена для поиска минимума функций (как одной, так и нескольких переменных) с различными n ограничениями.

Рассмотрим несколько задач поиска экстремума с ограничениями/

Пример 5.3.

Найти максимум и минимум функции $F=(x-3)^2-(y-4)^2$ при ограничениях:

$$\begin{cases} 3x+2y \geq 7 \\ 10x-y \leq 8 \\ -18x+4y \leq 12 \\ x \geq 0 \\ y \geq 0 \end{cases}$$

В функции sqp все ограничения должны быть вида ≥ 0. Поэтому второе и третье ограничение умножим на −1, и перенесём всё в левую часть неравенств. В результате этих несложных преобразований система ограничений примет вид:

$$g(x) = \begin{cases} 3x+2y-7 \geq 0 \\ -10x+y+8 \geq 0 \\ 18x-4y+12 \geq 0 \\ x \geq 0 \\ y \geq 0 \end{cases}$$

Листинг 5.4.

Нахождение минимума функции (пример 5.3)

Последовательно рассмотрим задачу на минимум (листинг 5.4) и максимум (листинг 5.5).

```
% В задаче на минимум функция, в которой хранится F(x) будет такой
function y=f(x)
```

```
y=(x (1)−3)^2+(x(2)−4)^2;
endfunction
% Вектор-функцию ограничений g(x) можно записать так:
function r=g(x)
r=[3*x(1)+3*x(2)−7;−10*x(1)+x(2)+8;18*x(1)−4*x(2)+12;x(1);x(2)];
end
% Вычисляем с помощью функции sqp
x0=[0;0]; [x,obj,info,iter]=sqp(x0,@f,[],@g)
minimum=f(x)

% Результаты
x =
1.2178
4.1782
obj=3.2079
info = 101
iter = 5
mininum=3.2079
```

Минимум 3.2079 достигается в точке (1.2178, 4.1782), значение info = 101 (104) говорит о корректном решении задачи, для нахождения минимального значения потребовалось всего 5 итераций.

Теперь рассмотрим решение задачи на максимум. Функция sqp может искать только минимум. Поэтому вспомним, как задача на максимум сводится к задаче на минимум: max f(x) =−min f(−x). Это учтено в листинге 5.5.

Листинг 5.5. Нахождение максимума (пример 6.3)

```
function y=f(x)
y=(x(1)−3)^2+(x(2)−4)^2;
endfunction
% Определяем функцию, для которой минимум будет максимумом функции f
function y=f1(x)
y=−f(−x);
end
function r = g(x)
r=[3*x(1)+2*x(2)−7;−10*x(1)+x(2)+8;18*x(1)−4*x(2)+12;x(1);x(2)];
end
x0=[0;0];[x,obj,info,iter]=sqp(x0,@f1,[],@g)
```

```
maximum=f(x)
% Результаты работы программы
x =
2.0000
12.0000
obj=−281.00
info=101
iter=3
maximum=65.000
```

Максимум достигается в точке (2,12), его величина равна 65.

Пример 5.4.

План производства изделий трёх типов составляет 120 деталей (x_1 - количество изделий первого вида, x_2 – количество изделий второго вида, x_3 - количество изделий третьего вида). Изделия можно изготовить тремя способами. При первом технологическом способе производят изделия первого типа и затраты составляют $4x_1 + x_1^2$. Второй технологический способ предназначен для производства изделий второго типа и затраты составляют $8x_2 + x_2^2$. Третий способ позволяет производить изделия третьего типа и затраты в нём можно рассчитать по формуле x_3^2. Определить, сколько изделий каждого типа надо изготовить, чтобы затраты были минимальными [1].

Сформулируем эту задачу, как задачу оптимизации. Найти минимум функции $f(x_1, x_2, x_3) = 4x_1 + x_1^2 + 8x_2 + x_2^2 + x_3^2$ при следующих ограничениях $x_1 + x_2 + x_3 = 120,\ x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0.$

Листинг 5.6. Решение задачи оптимизации из примера 4.

```
% Оптимизируемая функция f
function y=f(x)
y=4∗x (1)+x(1)∗x(1)+8∗x(2)+x(2)∗x(2)+x(3)∗x(3);
end (endfunction)
% Функция ограничения g(x)=0
function z=g(x)
z=x(1)+x(2)+x(3)-120;
end
% Функция ограничения φ(x) ≥ 0
function u=fi(x)
fi =[x(1); x(2); x(3)];
```

```
end
x0=[0;0;0]; [x,obj,info,iter]=sqp(x0,@f,@g)

% Результаты решения
x =
   40.000
   38.000
   42.000
obj =  5272.0
info =  104
iter =  6
```

Минимальные затраты составят 5272 денежных единицы, при этом будет произведено 40 изделий первого вида, 38 - второго и 42 - третьего. Для решения задачи было проведено 8 итераций.

Пример 5.5.

Найти максимум функции $f = -x_1^2 - x_2^2$ при ограничениях $(x_1-7)^2+(x_2-7)^2 \le 18$, $x1 \ge 0$, $x2 \ge 0$ [1].

В этой задаче необходимо свести задачу на максимум к задаче на минимум, а также путём умножения на −1 заменить знак в неравенстве. Текст программы-решения задачи в Octave представлен в листинге 5.7. Функция достигает своего максимального значения −32 в точке (4, 4).

Листинг 5.7.

```
Решение задачи из примера 5
function y=f(x)
y=−x(1)∗x(1)-x(2)∗x(2);
end
function y=f1(x)
y=−f(-x);
end
function u=fi(x)
u=[-(x(1)-7)^2-(x(2)-7)^2+18; x(1); x(2)];
end
x0=[0;0]; [xopt,obj,info,iter]=sqp(x0,@f1,[],@fi)
f(xopt)

% Результаты решения
xopt =
   4.0000
   4.0000
```

obj = 32.000
info = 104
iter = 8
ans = -32.000

Другим классом оптимизационных задач будут задачи линейного программирования (ЗЛП).

5.2. РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Эти задачи встречаются во многих отраслях знаний. Алгоритмы их решения хорошо известны. Эти алгоритмы реализованы во многих, как проприетарных, так и свободных, математических пакетах. Не является исключением и **Octave**. Но перед тем, как рассмотреть решение задач линейного программирования в Octave, давайте вспомним, что такое задача линейного программирования.

5.2.1. Задача линейного программирования

Знакомство с задачами линейного программирования начнём на примере задачи об оптимальном рационе. Задача об оптимальном рационе. Имеется четыре вида продуктов питания: П1, П2, П3, П4. Известна стоимость единицы каждого продукта c1, c2, c3, c4. Из этих продуктов необходимо составить пищевой рацион, который должен содержать не менее b1 единиц белков, не менее b2 единиц углеводов, не менее b3 единиц жиров. Причём известно, в единице продукта П1 содержится a_{11} единиц белков, a_{12} единиц углеводов и a_{13} единиц жиров и т.д. (см. таблицу 1).

Требуется составить пищевой рацион, чтобы обеспечить заданные условия при минимальной стоимости.

Пусть x1, x2, x3, x4 - количества продуктов П1, П2, П3, П4. Общая стоимость рациона равна

$$L = c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 = \sum_{i=1}^{4} c_ix_i \qquad (5.1)$$

Сформулируем ограничение на количество белков, углеводов и жиров в виде неравенств. В одной единице продукта П1 содержится a_{11} единиц белков, в x1 единицах - a_{11}x1, в x2 единицах продукта П2 содержится a_{21}x2 единиц белка и т.д. Следовательно, общее количество белков во всех

четырёх типов продукта равно $\sum_{j=1}^{4} a_{j1}x_j$ и должно быть не больше b_1. Получаем первое ограничение

$$a_{11}x_1 + a_{21}x_2 + a_{31}x_3 + a_{41}x_4 \leq b_1 \quad (5.2)$$

Аналогичные ограничения для жиров и углеводов имеют вид:

$$a_{12}x_1 + a_{22}x_2 + a_{32}x_3 + a_{42}x_4 \leq b_2 \quad (5.3)$$

$$a_{13}x_1 + a_{23}x_2 + a_{33}x_3 + a_{43}x_4 \leq b_3 \quad (5.4)$$

Принимаем во внимание, что x1, x2, x3, x4 положительные значения, получим ещё четыре ограничения

$$x_1 \geq 0, \qquad x_2 \geq 0, \qquad x_3 \geq 0, \qquad x_4 \geq 0 \quad (5.5)$$

Таким образом задачу о оптимальном рационе можно сформулировать следующим образом: найти значения переменных x_1, x_2, x_3, x_4, удовлетворяющие системе ограничений (5.2) – (5.5), при которых линейная функция (5.1) принимала бы минимальное значение.

Задача об оптимальном рационе является задачей линейного программирования, функция (5.1) называется функцией цели, а ограничения (5.2) - (5.5) системой ограничений задачи линейного программирования.

В задачах линейного программирования функция цели L и система ограничений являются линейными.

В общем случае задачу линейного программирования можно сформулировать следующим образом. Найти такие положительные значения x_1, x_2, ... , x_n, при которых функция цели L (5.6) достигает своего минимального значения и удовлетворяет системе линейных ограничений (5.7).

$$L = c_1x_1 + c_2x_2 + \cdots + c_nx_n = \sum_{i=1}^{n} c_ix_i \quad (5.6)$$

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \;\; i = 1, \dots m \quad (5.7)$$

Если в задачу линейного программирования добавляется ограничение целочисленности значений x, то мы получаем задачу целочисленного программирования.

Octave позволяет решать задачи линейной оптимизации с ограничениями в более общей формулировке.

Найти такие положительные значения x1, x2, ... , xn, при которых функция цели L (5.5) достигает своего минимального (максимального)

значения и удовлетворяет системе линейных ограничений. Система ограничений может быть представлена неравенствами (5.8) или (5.9). При этом значения x могут быть, как вещественными, так и целочисленными, как положительными, так и отрицательными.

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \tag{5.8}$$

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, \quad i = 1, \dots, m \tag{5.9}$$

5.2.2. Решение задач линейного программирования в Octave

Для решения задач линейного программирования в Octave существует функция [xopt,fmin,status,extra]=glpk(c, a, b, lb, ub, ctype, vartype, sense)

Здесь:

• c - вектор-столбец, включающий в себя коэффициенты при неизвестных функции цели, размерность вектора c равна количеству неизвестных n в задаче линейного программирования;

• a - матрица коэффициентов при неизвестных из левой части системы ограничений, количество строк матрицы равно количеству ограничений m, а количество столбцов совпадает с количеством неизвестных n;

• b - вектор-столбец содержит свободные члены системы ограничений, размерность вектора равна количеству ограничений m.

• lb - вектор-столбец размерности n, содержащий верхнюю систему ограничений (x > lb), по умолчанию lb - вектор столбец, состоящий из нулей;

• ub - вектор-столбец размерности n, содержащий нижнюю систему ограничений (x < ub), по умолчанию верхняя система ограничений отсутствует, подразумевается, что все значения вектора ub равны +∞;

• ctype - массив символов размерности n, определяющий тип ограничения (например, (5.8) или (5.9)), элементы этого вектора могут принимать одно из следующих значений:

– «F» - ограничение будет проигнорировано,

– «U» - ограничение с верхней границей ((A(i,:)*x <= b(i)),

– «S» - ограничение в виде равенства ((A(i,:)*x = b(i)),

– «L» - ограничение с верхней границей ((A(i,:)*x >= b(i)),

– «D» - двойное ограничение ((A(i,:)*x <= b(i) и (A(i,:)*x>= b(i));

• vartype - массив символов размерности n, который определяет тип переменной x_i «C» - вещественная переменная, «I» - целочисленная переменная;

• sense - значение, определяющее тип задачи оптимизации:

a. 1 - задача минимизации,
b. -1 - задача максимизации.

Функция glpk возвращает следующие значения:

• xopt - массив значений x, при котором функция цели L принимает оптимальное значение;

• fmin - оптимальное значение функции цели;

• status - переменная, определяющая как решена задача оптимизация, при status = 180 (0), решение найдено и задача оптимизации решена полностью[2];

• extra - структура, включающая следующие поля:

– lambda - множители Лагранжа;

– time - время в секундах, затраченное на решение задачи.

[2]Если status отлично от 180 (0), то полученному решению доверять нельзя.

Рассмотрим несколько примеров решения задач линейного программирования.

Пример 5.6.

Найти такие значения переменных x_1, x_2, x_3, x_4 при которых функция цели $L=-x_2-2x_3+4x_4$ достигает своего минимального значения и удовлетворяются ограничения:

$$3x_1 - x_2 \leq 2$$
$$x_2 - 2x_3 \leq -1$$
$$4x_3 - x_4 \leq 3$$
$$5x_1 + x_4 \geq 6$$
$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$$

Сформируем параметры функции gplk

$$c = \begin{bmatrix} 0 \\ -1 \\ -2 \\ 4 \end{bmatrix} - \text{коэффициенты при неизвестных функции цели,}$$

$$a = \begin{bmatrix} 3 & -1 & 0 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & 4 & -1 \\ 5 & 0 & 0 & 1 \end{bmatrix} - \text{матрица системы ограничений,}$$

$$b = \begin{bmatrix} 2 \\ -1 \\ 3 \\ 6 \end{bmatrix} - \text{свободные члены системы ограничений,}$$

ctype = ”UUUL” – массив символов, определяющий тип ограничения[15]

vartype = “CCCC” – массив, определяющий типа переменной, в данном случае все переменные вещественные,

sense = 1 – задача на минимум.

Текст программы решения задачи приведён в листинге 5.8.

Листинг 8.

```
%Решение задачи из примера 5.6
c=[0;-1;-2;4]; a=[3 −1 0 0 ; 0 1 −2 0 ; 0 0 4 −1;5 0 0 1 ];
b=[2; −1; 3 ; 6 ];
ctype="UUUL"; vartype="CCCC"; sense=1;
[xmin,fmin,status]=glpk(c,a,b,[0;0;0;0],[],ctype,vartype,sense)

% Результаты решения
xmin =
1.00000
1.00000
1.00000
1.00000
fmin = 1.00000
status = 0
```

Значение переменной status равно 0, что свидетельствует о корректном решении задачи линейного программирования.

Пример 5.7.

Найти такие значения переменных x1, x2, x3 при которых функция цели L = −5 + x1 − x2 − 3x3 достигает своего минимального значения и удовлетворяются ограничения:

$x_1 + x_2 \geq 2$

$x_1 - x_2 \leq 0$

$x_1 + x_3 \geq 2$

$x_1 + x_2 - x_3 \leq 3$

$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$

Сформируем параметры функции gplk

$c = \begin{bmatrix} 1 \\ -1 \\ -3 \end{bmatrix}$ – коэффициенты при неизвестных функции цели,

$a = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix}$ – матрица системы ограничений (три переменных и четыре ограничения),

$$b = \begin{bmatrix} 2 \\ 0 \\ 2 \\ 3 \end{bmatrix} -$$ свободные члены системы ограничений

ctype = ”LULU” – массив символов, определяющий тип ограничения[16]

vartype = “CCC” – массив, определяющий типа переменной, в данном случае все переменные вещественные,

sense = –1 – задача на максимум.

Текст программы решения задачи приведён в листинге 5.9.

Листинг 5.9.

```
% Решение задачи из примера 5.7
c=[1;-1;-3] ; a=[1 1 0;1 -1 0; 1 0 1;1 1 -1];b=[2;0;2;3];
ctype="LULU"; var type=" CCC"; sense=-1;
[xmax,fmax,status]=glpk(c,a,b,[],[],ctype,vartype,sense)
Lmax= -5+xmax(1)-xmax(2)-3*xmax(3)
% Можно Lmax=-5+fmax
% Результаты решения
xmax=

    1.66667
    1.66667
    0.33333

fmax=-1.0000
status = 0
Lmax = -6
```

Минимальное значение 6 L = −6 достигается при x = [1.66667; 1.66667; 0.33333].

Значение переменной status равно 0, что свидетельствует о корректном решении задачи линейного программирования.

Решим задачу 5.7, как задачу целочисленного программирования (см. листинг 5.10).

Листинг 5.10. Решение задачи из примера 5.7 в целочисленном варианте.

```
% Решение задачи из примера 5.7
c=[1;-1;-3] ; a=[1 1 0;1 -1 0; 1 0 1;1 1 -1];b=[2;0;2;3];
```

```
ctype="LULU"; var type="III"; sense=-1;
[xmax,fmax,status]=glpk(c,a,b,[],[],ctype,vartype,sense)
Lmax= -5+xmax(1)-xmax(2)-3*xmax(3)
% Можно Lmax=-5+fmax
% Результаты решения
xmax =
        2
        2
        1
fmax = -3
status = 0
Lmax = -8
```

Значение status = 0 (171) свидетельствует о корректном решении задачи целочисленного программирования, значение Lmax = −8 достигается при x=[2;2;1].

Пример 5.8.

Для изготовления четырёх видов изделий используется токарное, фрезерное, сверлильное, расточное шлифовального оборудование, а также комплектующие изделия. Сборка изделий требует сборочно-наладочных работ. В таблице 5.1 представлены:

нормы затрат ресурсов на изготовление различных изделий, наличие каждого из ресурсов, прибыль от реализации одного изделия, ограничения на выпуск изделий второго и третьего типа. Сформировать план выпуска продукции для достижения максимальной прибыли.

Таблица 5.1 – нормы затрат ресурсов.

Ресурсы	Нормы затрат на одно изделие				Общий объём ресурсов
	1	2	3	4	
Производительность оборудования (человеко-ч)					
токарного	550		620		64270
фрезерного	40	30	20	20	4800
сверлильного	86	110	150	52	22360
расточного	160	92	158	128	26240
шлифовального		158	30	50	7900
Комплектующие изделия (шт.)	3	4	3	3	520
Сборочно-наладочные работы (человеко-ч)	4.5	4.5	4.5	4.5	720
Прибыль от реализации одного изделия (тыс. руб.)	315	278	573	370	
Выпуск					
минимальный		40			
максимальный			120		

Пусть x1 - количество изделий первого вида, x2 - количество изделий второго вида, x3 и x4 - количество изделий третьего и четвёртого вида соответственно. Тогда прибыль от реализации всех изделий вычисляется по формуле

$$L = 315x_1 + 278x_2 + 573x_3 + 370x_4 \qquad (5.10)$$

Ограничения на фонд рабочего времени формируют следующие ограничения

$$L = 315x_1 + 278x_2 + 573x_3 + 370x_4 \qquad (5.11)$$

$$\begin{cases} 550x_1 + 620x_3 \leq 64270 \\ 30x_1 + 30x_2 + 20x_3 + 20x_4 \leq 4800 \\ 86x_1 + 110x_2 + 150x_3 + 128x_4 \leq 26240 \\ 158x_2 + 30x_3 + 50x_4 \leq 7900 \end{cases} \qquad (5.12)$$

Ограничение на возможное использование комплектующих изделий

$$3x_1 + 4x_2 + 3x_3 + 3x_4 \leq 520 \qquad (5.13)$$

Ограничение на выполнение сборочно-наладочных работ

$$4.5x_1 + 4.5x_2 + 4.5x_3 + 4.5x_4 \leq 720 \qquad (5.14)$$

Ограничения на возможный выпуск изделий каждого вида

$$x2 \geq 40,\ x3 \leq 120,\ x1 \geq 0,\ x3 \geq 0,\ x4 \geq 0 \qquad (5.15)$$

Сформулируем задачу линейного программирования.

Найти значения x1, x2, x3 и x4 при которых функция цели L (5.10) достигает своего максимального значения и выполняются ограничения (10.11) – (10.14).

Сформируем параметры функции gplk

$$c = \begin{bmatrix} 315 \\ 278 \\ 573 \\ 370 \end{bmatrix} - \text{коэффициенты при неизвестных функции цели,}$$

$$a = \begin{pmatrix} 550 & 0 & 620 & 0 \\ 40 & 30 & 20 & 20 \\ 86 & 110 & 150 & 52 \\ 160 & 92 & 158 & 128 \\ 0 & 158 & 30 & 50 \\ 3 & 4 & 3 & 3 \\ 4.5 & 4.5 & 4.5 & 4.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} -$$ матрица системы ограничений (четыре переменных и двенадцать ограничений),

$$b = \begin{pmatrix} 64270 \\ 4800 \\ 22360 \\ 26240 \\ 7900 \\ 520 \\ 720 \\ 40 \\ 120 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$ – свободные члены системы ограничений

ctype ="UUUUUUULULLL" - массив символов, определяющий тип ограничения[9],

vartype ="IIII" - массив, определяющий тип переменной, в данном случае все переменные целые (задача целочисленного программирования),

sense = −1 - задача на максимум.

Программа решения задачи в Octave представлена в листинге 5.11.

Листинг 5.11.

```
% Решение задачи из примера 5.8
c =[315;278;573;370];
a=[550 0 620 0;40 30 20 20;86 110 150 52;160 92 158 128;0 158 30 50;3 4 3 3;4.5 4.5 4.5 4.5;0 1 0 0; 0 0 1 0;1 0 0 0;0 0 1 0;0 0 0 1];
b=[64270;4800;22360;26240;7900;520;720;40;120;0;0;0];
ctype="UUUUUUULULLL";vartype="IIII";sense=-1;
[xmax,fmax,status]=glpk(c,a,b,[],[],ctype,vartype,sense)
```

% Результаты решения

xmax =

65

40

46

4

fmax = 59433

status = 171

Для получения максимальной прибыли fmax необходимо произвести 65 единиц изделий первого типа, 40 - второго, 46 - третьего и 4 - четвёртого. Значение параметра status = 171 говорит о корректности решения задачи линейного программирования.

[9]Первые три ограничения типа «меньше», четвёртое и пятое типа «равно».

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 5

Нахождение экстремумов функций.
Определить возможные экстремумы функций
(В случае множества экстремумов ограничиться тремя)

Вариант	Функция	Вариант	Функция
1	$x - \sin x = 0{,}25$ $x^3 - 3x^2 + 9x - 8 = 0$	17	$x^2 \cos(2x + 1) = -1$ $2x^4 - 3x^2 + 15 = 0$
2	$\mathrm{tg}(0{,}2 + x) = x^3 + 3$ $x^3 + 2x^2 - 7x + 1 = 0$	18	$0{,}5^x - 1 = (x + 2)^2$ $3x^3 - 2x^2 + x + 1 = 0$
3	$(x + 1)^{1/3} - \cos(0{,}3 + 0{,}4x) = 2$ $x^3 + x^2 - 3x + 4 = 0$	19	$\cos(x + 2) - x + 2x + 1 = 0$ $x^4 + 2x^2 + 3x - 10 = 0$
4	$\mathrm{tg}(2{,}3 + 0{,}5x) = 3x + 2$ $2x^3 - 3x^2 + x + 1 = 0$	20	$x + \ln(2x + 3) = 0{,}5$ $-2^{x-1} - x = 0$
5	$2e^{x+1} + 3x + 1 = 0$ $3x^4 + 4x^3 - 12x - 1 = 0$	21	$x^2 + 4\sin(x + 1) = 0$ $3x^4 + 4x^3 - 12x^2 - 5 = 0$
6	$3x^2 + \cos(2x + 1) = 1$ $x^4 - 2x^3 - 10x^2 - 2 = 0$	22	$e^{2x+1} + 5x - 1 = 0$ $7x^3 - 2x^2 + 3x - 10 = 0$
7	$5x\sin(2x + 1) = 0{,}43$ $x^3 - 7x^2 + 2x - 1 = 0$	23	$2e^{x+1} - 3x + 1 = 0$ $x\lg(x^2 + 2x - 1) = 1$
8	$x\cos(x + 2) = x^2 - 3x + 1$ $x^3 + x^2 + x - 10 = 0$	24	$x^2 \cos(2x - 1) = 1$ $2x^4 - 3x^3 + 3x^2 - x + 1 = 0$
9	$(x - 3)\cos(x + 2) = 1$ $2x^4 - 3x^3 + x - 1 = 0$	25	$2x - \lg(x + 3) = 7$ $\mathrm{tg}^3(x) + x - 1 = 0$
10	$\sin(x + \pi/3) + 0{,}5x + 2 = 0$ $x4 - x - 1 = 0$	26	$(1 - x)e^{3x-1} = 0{,}5$ $3\sin^2(x + 1) - x^2 + x = 2$
11	$x\lg(x + 1) = 1$ $2x^3 - 9x^2 - 60x = 0$	27	$2\sin(x - \pi/6) = x^2 - 0{,}5$ $x^4 + 4x^3 - 8x^2 - 17 = 0$
12	$\mathrm{arctg}x - 1/(3x^3) = 0$ $2x^4 + x - 3 = 0$	28	$5\cos(x + 3) = x - 0{,}5$ $3^x + 2 - x = 0$
13	$\ln x + (x + 1)^3 = 0$ $x^2 - 2 + 0{,}5^x = 0$	29	$x = (\log(x + 2))^{1/2} - 1$ $x^4 - x^3 - 2x^2 + 3x - 3 = 0$
14	$\cos(x + 0{,}5) = x^3$ $2x^4 + x - 3 = 0$	30	$(x - 2)^3 \lg(x - 3) = 1$ $2x^3 - 9x^2 - 60x + 1 = 0$
15	$(x - 4)^2 \log_2(x - 3) = 1$ $3x^4 + 8x^3 + 2x - 1 = 0$	31	$(x^2 + 2x - 20)\sin(x + 1) = 1$ $e^x = (x + 1)^3$
16	$e^{-2x} - 2x + 1 = 0$ $x^4 + 4x^3 - 8x^2 - 17 = 0$	32	$3\cos(x + 1)^2 = 2x + 1$ $3x^4 + 4x^3 - 12x^2 + 1 = 0$

Решение задач оптимизации

Решить задачу целочисленного программирования

1. $W = 2x_1 - x_2 + x_4 \to \min$
$$\begin{cases} x_1 + x_2 + x_3 - x_4 \leqslant 1 \\ x_1 - x_2 + x_3 - x_4 \leqslant 0 \\ 2x_1 + x_2 + x_3 - x_4 \geqslant 3 \end{cases}$$

2. $W = x_1 + x_3 \to \max$
$$\begin{cases} 2x_1 - 7x_2 + 22x_3 \leqslant 22 \\ 2x_1 - x_2 + 6x_3 \leqslant 6 \\ 2x_1 - 5x_2 + 2x_3 \leqslant 2 \\ -4x_1 + x_2 + x_3 \leqslant 1 \end{cases}$$

3. $W = 3 + 2x_2 + x_3 \to \max$
$$\begin{cases} x_1 - x_2 + 2x_3 + x_4 \geqslant 1 \\ 2x_1 - x_2 + x_3 - x_4 \geqslant 1 \\ x_1 - 2x_2 + x_3 - x_4 \geqslant -1 \\ x_1 + x_2 + x_3 + 2x_4 \leqslant 5 \end{cases}$$

4. $W = x_3 + 3x_4 \to \min$
$$\begin{cases} x_1 + x_2 - x_3 - x_4 \leqslant 2 \\ x_1 - x_2 - x_3 + x_4 \geqslant 0 \\ -x_1 - x_2 + 2x_3 - x_4 \geqslant -3 \\ x_1 \geqslant 1 \end{cases}$$

5. $W = -x_1 + x_2 \to \max$
$$\begin{cases} x_1 - 2x_2 \geqslant 2 \\ 2x_1 - x_2 \geqslant 2 \\ x_1 + x_2 \geqslant 5 \end{cases}$$

6. $W = x_1 - x_2 - 2x_4 \to \max$
$$\begin{cases} 2x_1 - x_2 + 2x_3 - x_4 \leqslant 4 \\ x_1 - 2x_2 + x_3 - 2x_4 \geqslant 2 \\ x_1 - x_4 \geqslant 1 \\ x_2 + x_3 \leqslant 1 \end{cases}$$

7. $W = x_1 - x_2 + 3x_3 + x_4 \to \max$
$$\begin{cases} x_1 - x_2 + x_4 \leqslant 1 \\ x_2 - x_3 + x_4 \leqslant 1 \\ x_1 + x_3 + 2x_4 \leqslant 2 \\ -2x_2 + x_4 \leqslant 0 \end{cases}$$

8. $W = -x_2 - 2x_3 + x_4 \to \min$
$$\begin{cases} 3x_1 - x_2 \leqslant 2 \\ x_2 - 2x_3 \leqslant -1 \\ 4x_3 - x_4 \leqslant 3 \\ 5x_1 + x_4 \geqslant 6 \end{cases}$$

9. $W = x_1 + x_2 + 3x_3 - x_4 \to \max$
$$\begin{cases} x_1 - 5x_2 + 4x_3 \leqslant 5 \\ x_1 - 2x_2 - 3x_3 \leqslant 4 \\ x_1 + 6x_2 + 5x_3 \leqslant 4 \\ x_2 + x_3 \leqslant 1 \end{cases}$$

10. $W = -4 - 2x_1 - x_2 - x_3 \to \min$
$$\begin{cases} x_1 - 2x_2 + 3x_3 - 4x_4 \geqslant -10 \\ x_1 + x_2 - x_3 - x_4 \leqslant -4 \\ x_1 - x_2 + x_3 - x_4 \geqslant -6 \\ x_1 + x_2 + x_3 + x_4 \leqslant 10 \end{cases}$$

11. $W = x_1 + x_2 + x_3 + 1 \to \min$
$$\begin{cases} x_1 + x_2 \geqslant 0 \\ x_1 + x_3 \geqslant 1 \\ x_2 - x_3 \geqslant 1 \\ x_1 + 2x_2 + 3x_3 \geqslant 0 \end{cases}$$

12. $W = 2 + 2x_2 - x_3 + 3x_4 \to \max$
$$\begin{cases} -x_1 + x_2 - 2x_4 \geqslant -1 \\ x_1 + x_3 + x_4 \geqslant 1 \\ x_2 + x_3 - x_4 \geqslant 1 \\ x_3 \leqslant 4; \qquad x_2 \leqslant 10 \end{cases}$$

13. $W = x_1 + x_2 + 3 \to \max$
$$\begin{cases} x_1 - x_2 \leqslant 1 \\ x_1 - 2x_2 \geqslant -2 \\ -x_1 + x_2 \geqslant -1 \\ 2x_1 + x_2 \geqslant -2 \end{cases}$$

14. $W = x_1 - 10x_2 + 100x_3 \to \max$
$$\begin{cases} x_1 + x_2 + x_3 \leqslant 1 \\ x_1 - x_2 - x_3 \leqslant 2 \\ -x_1 + 2x_3 \leqslant 0 \\ x_1 + 2x_3 \leqslant 5 \end{cases}$$

15. $W = -3 + x_1 + 3x_2 + 5x_3 \to \max$
$$\begin{cases} x_1 - x_2 + x_3 \leqslant 1 \\ 2x_1 + x_2 + x_3 \leqslant 1 \\ x_1 + 2x_2 + x_3 \leqslant 1 \\ x_1 + x_2 + 2x_3 \leqslant 1 \end{cases}$$

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Введение в Octave для инженеров и математиков: / Е. Р. Алексеев, О. В. Чеснокова — М.: ALT Linux, 2012. — 368 с.
2. Вержбицкий В.М. Основы численных методов. — М.:Высшая школа, 2002. – 840 с.
3. Эшби У.Р. Введение в кибернетику / Пер. с англ. / Под ред. В.А. Успенского. – М.: ИЛ, 1962.
4. Гроп Д. Методы идентификации систем. – М.: Мир, 1979. 1979. – 302 с.
5. Кафаров В.В. Методы кибернетики в химии и химической технологии. –М.: Химия, 1985. -448 с.
6. Ахназарова С.Л., Кафаров В.В. Методы оптимизации эксперимента в химической технологии. М.: Высшая школа, 1985. -327 с.

Учебная литература

Тюрин Михаил Павлович

Бородина Елена Сергеевна

ТЕОРИЯ И ПРАКТИКА ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ

ПРАКТИКУМ

Часть 1

Учебное пособие

Усл.печ.л.____Тираж 32 экз. Заказ № _____

Редакционно-издательский отдел ФГБОУ ВО «РГУ им. А.Н. Косыгина»

115035, Москва, ул. Садовническая, 33, стр.1

тел. 8-495-811-01-01 доб. 1099

e-mail: riomgudt@mail.ru

Отпечатано в РИО ФГБОУ ВО «РГУ им. А.Н. Косыгина»