**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)**

С. С. Владимиров

ТЕОРИЯ И ПРАКТИКА ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

Моделирование в системе GNU/Octave

**Учебно-методическое пособие
по выполнению лабораторных работ**

**Санкт-Петербург
2021**

УДК 621.391 (076)
ББК 32.811 я73
В 57

Рецензент
профессор кафедры сетей связи и передачи данных,
доктор технических наук, профессор
О. С. Когновицкий

Рекомендовано к печати редакционно-издательским советом СПбГУТ

Владимиров, С. С.

В 57 Теория и практика помехоустойчивого кодирования. Моделирование в системе GNU/Octave : учебно-методическое пособие по выполнению лабораторных работ / С. С. Владимиров ; СПбГУТ. — Санкт-Петербург, 2021. — 64 с.

Призвано ознакомить студентов старших курсов с принципами моделирования помехоустойчивых кодов и каналов передачи данных в системе компьютерной алгебры GNU/Octave. Представленный материал служит справочным и методическим пособием при выполнении лабораторных работ по дисциплинам «Теория и практика помехоустойчивого кодирования» и «Помехоустойчивое кодирование в инфокоммуникационных системах».

Предназначено для студентов, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника» и 11.03.02 «Инфокоммуникационные технологии и системы связи».

УДК 621.391 (076)
ББК 32.811 я73

СОДЕРЖАНИЕ

1. ПРОВЕДЕНИЕ РАСЧЕТОВ В СИСТЕМЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ OCTAVE

1.1. Краткая справка об Octave

Система компьютерной алгебры GNU Octave представляет собой кроссплатформенный свободный программный пакет для математических вычислений, использующий язык высокого уровня, совместимый с Matlab [1–4].

Работа с Octave проводится в интерактивном командном интерфейсе, который позволяет выполнять как отдельные команды и функции, так и целые сценарии. Язык Octave оперирует арифметикой вещественных и комплексных скаляров, векторов и матриц. Существуют подключаемые к Octave библиотеки функций, решающих различные задачи, такие как: цифровая обработка сигналов, обработка изображений, видео и звука, работа с базами данных, статистические вычисления, символьные вычисления и многое другое вплоть до взаимодействия с низкоуровневыми аппаратными интерфейсами.

Библиотеки функций Octave размещены на сайте Octave Forge[1]. На этом же сайте содержится полная документация по содержащимся в этих библиотеках функциям [5]. Для загрузки библиотек в Octave используется либо встроенный в нее пакетный менеджер pkg (разд. 1.2.3), либо пакетный менеджер операционной системы (в случае если репозитории операционной системы содержат соответстующие пакеты).

При необходимости функциональность Octave может быть расширена пользовательскими функциями, написанными либо в виде скриптов на встроенном языке Octave, либо в виде динамически загружаемых модулей на языках C/C++, Фортран и пр.

Существующая справка написана для ОС Debian Linux и версии GNU Octave, использующейся в лабораториях кафедры на момент написания пособия, для стандартного интерфейса пользователя.

Для приводимых функций указаны основные способы записи и применения, необходимые для выполнения лабораторных работ. Подробное описание каждой функции можно прочитать в документации Octave, на сайте Octave Forge либо вывести командой `help`, указав после нее имя нужной функции.

1.1.1. Запуск системы Octave в графическом режиме

Для запуска системы Octave в графическом режиме необходимо использовать соответствующий пункт главного меню.

Также ее можно запустить через терминал командой

```
user@host:[~]$ octave &
```

[1] `https://octave.sourceforge.io/`.

Окно терминала после этого закрывать нельзя.

Главное окно Octave показано на рис. 1.1.

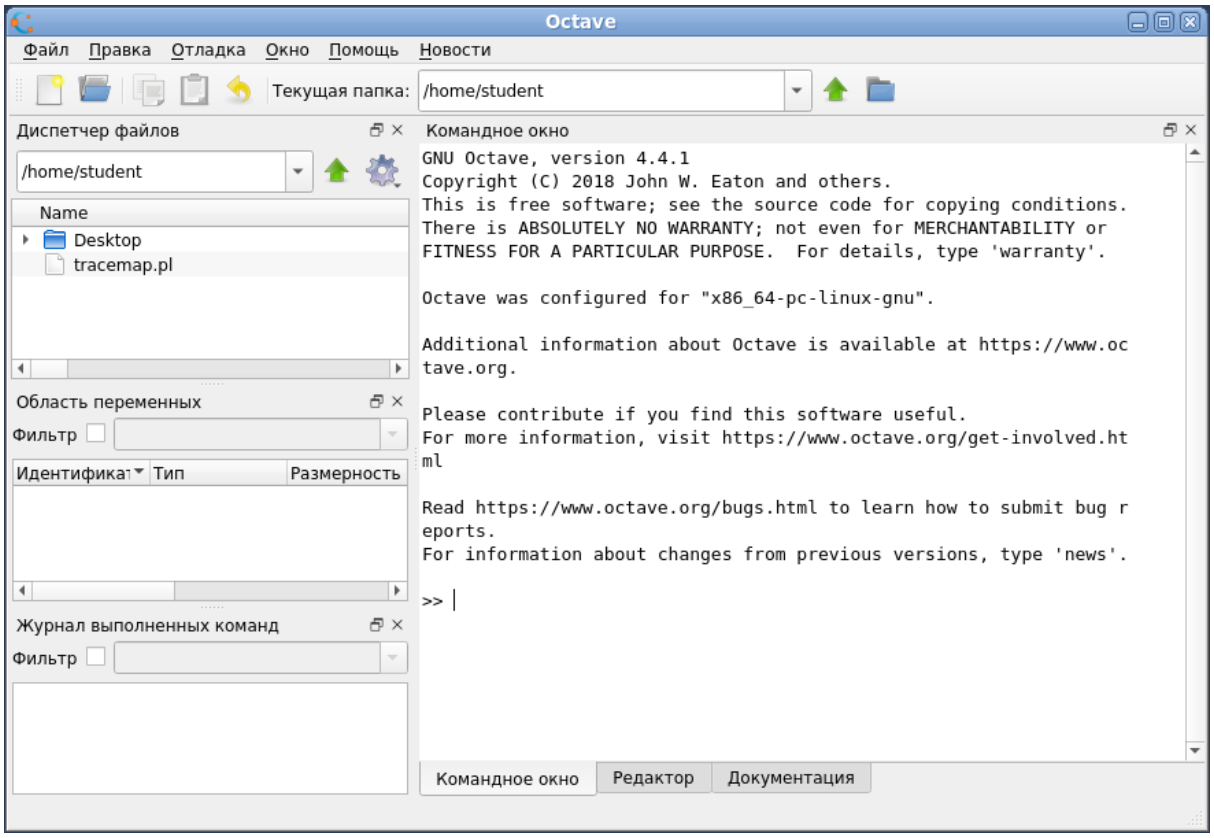

Рис. 1.1. Главное окно Octave

Основная рабочая область окна Octave имеет две основные вкладки. Первая вкладка *Командное окно* (рис. 1.1) предназначена для ввода отдельных команд и вывода результатов их выполнения. Также сюда выводится результат выполнения скриптов, написанных/открытых в *Редакторе*.

Вторая вкладка *Редактор* (рис. 1.2) предназначена для работы со скриптами. Написанный/открытый в *Редакторе* скрипт должен располагаться в рабочем каталоге Octave, который в стандартном интерфейсе указывается слева в *Диспетчере файлов*. Для запуска скрипта на выполнение используется кнопка меню *Редактора* «Сохранить файл и запустить» (желтая стрелка в шестеренке).

В зависимости от настроек могут присутствовать также вкладки *Документация* и *Редактор переменных*.

В левой части окна расположены служебные окна.

Диспетчер файлов отображает текущий рабочий каталог и его содержимое. Текстовые файлы и скрипты Octave, выбранные двойным кликом мыши в *Диспетчере*, будут открыты в *Редакторе*.

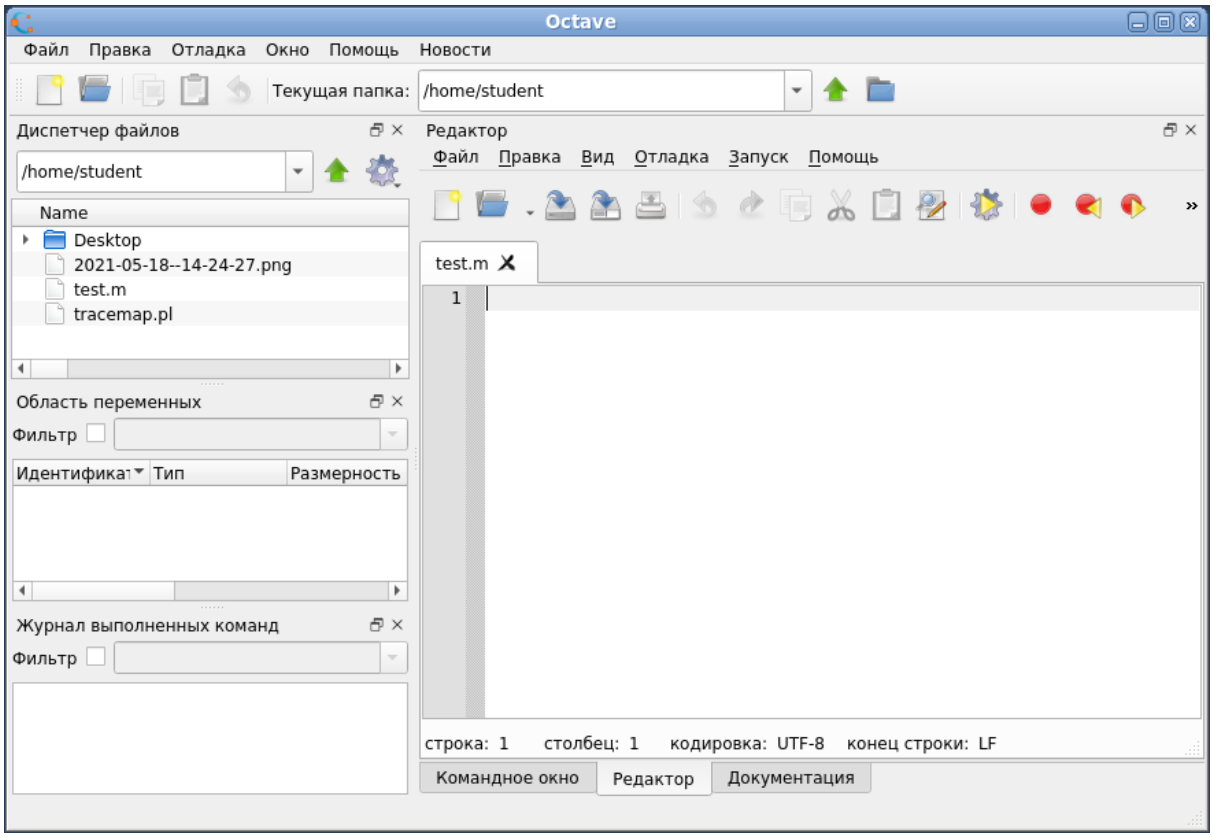

Рис. 1.2. Редактор скриптов Octave

В системе Linux рабочим каталогом Octave по умолчанию является домашний каталог пользователя (на лабораторных компьютерах это каталог `/home/student`). Сменить рабочий каталог можно в *Диспетчере файлов*. Также Octave можно настроить на запоминание последнего рабочего каталога.

Область переменных показывает записанные в память переменные, их тип и значение. При выборе переменной двойным кликом откроется вкладка *Редактор переменных*, в которой можно эту переменную отредактировать. *Область* и *Редактор переменных* удобно использовать для контроля правильности выполнения скриптов Octave.

Журнал выполненных команд содержит все команды, запущенные ранее в *Командном окне*. Двойной клик по команде запускает ее на выполнение в *Командном окне*.

1.1.2. Запуск системы Octave в терминале

Для запуска системы Octave в текстовом командном режиме используется терминал Linux.

Команда для вызова Octave:

```
user@host:[~]$ octave-cli -q
```

Флаг -q, указываемый после имени команды, говорит программе Octave не выводить приветствие и сразу переходить в командный режим.

В этом режиме пользователь должен последовательно вводить команды с клавиатуры, отправляя их на выполнение нажатием клавиши «Enter».

Рабочим каталогом Octave в текстовом режиме становится тот, из которого был вызван Octave. Обычно указывается в приглашении командной строки (терминала) Linux. На лабораторных ПК текущий каталог в окне терминала указан в квадратных скобках после имени компьютера.

Можно записать программу-скрипт на языке программирования Octave и передать файл с ней в качестве параметра при запуске программы. В этом случае Octave выполнит все команды из скрипта и завершит работу:

```
user@host:[~]$ octave-cli -q program.m
```

Для Octave, как и для Matlab, скрипт должен иметь расширение *.m.

1.2. Использование функций Octave

1.2.1. Функции Octave

Функции Octave имеют вид

```
>> function(par1, par2, ...);
```

Если функция завершается символом «;», то результат работы функции не будет выводиться на экран. В противном случае результат будет выведен.

Функции можно передавать как параметры:

```
>> func1(func2(par1))
```

В этом случае результат вычисления функции func2(par1) передается в функцию func1() в качестве параметра. Поскольку точки с запятой в конце нет, результат вычислений func1() будет выведен на экран.

Если функция возвращает несколько значений, то необходимо явно указать переменные, в которые они будут записаны, иначе будет выведено только первое возвращаемое значение, как показано в листинге 1.1.

Листинг 1.1

Пример вызова функции Octave, возвращающей два значения

```
>> deconv([2 3 5], [5 2])
ans =
   0.4000   0.4400

>> [B, R]=deconv([2 3 5], [5 2])
B =
   0.4000   0.4400

R =
         0         0   4.1200
```

Справку по функции Octave можно получить, введя команду:

```
>> help FunctionName
```

1.2.2. Служебные функции очистки

При написании скриптов Octave в ряде случаев удобно использовать функции очистки. К таким функциям относятся:

- `clc;` — очистка командного окна;
- `clear all;` — очистка области переменных и имен пользовательских функций;
- `close all;` — закрытие ранее открытых окон графика.

Эти три функции следует размещать в начале скрипта.

1.2.3. Загрузка библиотек функций

Для использования функций, прописанных в сторонних библиотеках, эти библиотеки необходимо подключить к Octave. Для этого используется встроенный пакетный менеджер `pkg`. Например, для загрузки пакета функций *communications* следует использовать команду:

```
>> pkg load communications;
```

При использовании скриптов эту команду следует либо вводить каждый раз при запуске Octave, либо указать в начале скрипта, либо прописать в автозапуске в конфигурационных файлах Octave.

При выполнении лабораторных работ по курсу из сторонних пакетов потребуется пакет *communications*, содержащий функции для работы с конечными полями, помехоустойчивыми кодами, методами модуляции и каналами передачи данных.

1.2.4. Создание пользовательской функции

Пользовательские функции Octave создаются в виде отдельных файлов с расширением «.m». Имя файла должно соответствовать имени функции.

Пример пользовательской функции для сложения двух чисел приведен в листинге 1.2.

Листинг 1.2

Пользовательская функция Octave `summa.m` для сложения двух чисел

```
function result = summa(A, B)
   result = A + B;
endfunction
```

Имя функции, указанное в файле, должно быть равно имени файла (без расширения «.m»).

В листинге 1.3 приведены примеры выполнения функции `summa` в Octave. Важно помнить, что файл с функцией должен находиться в рабочем каталоге Octave, который в стандартном интерфейсе указывается слева в *Диспетчере файлов.*

Листинг 1.3

Примеры выполнения функции `summa` в Octave

```
>> summa(2, 5)
ans = 7
>> K=summa(4, 9)
K = 13
```

Как и встроенные функции Octave, пользовательские функции могут возвращать несколько значений. Пример такой функции, рассчитывающей площадь круга, длину его окружности и диаметр по заданному радиусу, приведен в листинге 1.4.

Листинг 1.4

Пользовательская функция Octave `circle.m`
для вычисления параметров круга по заданному радиусу

```
function [area, circuit, diameter] = circle(R)
  area = pi * (R^2);
  circuit = 2 * pi * R;
  diameter = 2 * R;
endfunction
```

В листинге 1.5 приведен пример выполнения функции `circle` в Octave.

Листинг 1.5

Пример выполнения функции `circle` в Octave

```
>> [A, C, D] = circle(5)
A = 78.540
C = 31.416
D = 10
```

1.3. Системы счисления в Octave

В Octave по умолчанию используется десятичная система счисления. Также можно вводить данные в двоичной (через префикс `0b` или `0B`) и 16-ричной (через префикс `0x` или `0X`) системах:

```
>> a=23
a = 23
>> a=0b10111
a = 23
>> a=0x17
a = 23
```

1.3.1. Перевод из десятичной системы счисления в произвольную и обратно

Для перевода из одной системы счисления в другую используются функции `de2bi()` и `bi2de()` из библиотеки *communications*.

Функция `de2bi` для перевода из десятичной системы счисления в другую (по умолчанию в двоичную) имеет следующий формат:

```
B = de2bi(D, N, P, F)
```

Здесь:

`D` — десятичное число или матрица чисел, которые необходимо преобразовать. Это единственный обязательный параметр функции;

`N` — ширина выходного массива `B` (число колонок). По умолчанию равна `[]`, т. е. определяется результатом. Если задано `N` больше необходимого, то результат будет дополнен нулевыми старшими разрядами. Если задано `N` меньше необходимого, то результат будет обрезан до заданного числа младших разрядов;

`P` — основание целевой системы счисления. По умолчанию равно 2;

`F` — порядок записи результата. Значение по умолчанию равно `"right-msb"` (старшие разряды справа — обратная запись). Если задать значение `"left-msb"`, то результат будет выведен прямой записью (старшие разряды слева). Чтобы задать `F`, необходимо задать все предыдущие параметры.

Примеры использования функции `de2bi`:

```
>> de2bi(23)
ans =
   1   1   1   0   1

>> de2bi(23,4)
ans =
   1   1   1   0

>> de2bi(23,7)
ans =
   1   1   1   0   1   0   0
```

```
>> de2bi(23,[],2,"left-msb")
ans =
   1   0   1   1   1

>> de2bi(23,[],5)
ans =
   3   4

>> de2bi(23,4,5,"left-msb")
ans =
   0   0   4   3
```

Функция `bi2de` для перевода из произвольной системы счисления (по умолчанию двоичной) в десятичную имеет следующий формат:

```
D = bi2de(B, P, F)
```

Здесь:

`B` — число или матрица чисел, которые необходимо преобразовать в десятичную систему. Это единственный обязательный параметр функции;

P — основание исходной системы счисления. По умолчанию равно 2;

F — порядок записи вектора B. Значение по умолчанию равно "right-msb" (старшие разряды справа — обратная запись). Если задать значение "left-msb", то старшими будут считаться разряды слева.

Примеры использования функции bi2de:

```
>> bi2de([1 1 1 0 1])
ans = 23
>> bi2de([3 4],5)
ans = 23
```

```
>> bi2de([1 0 1 1 1],"left-msb")
ans = 23
>> bi2de([0 0 4 3],5,"left-msb")
ans = 23
```

1.3.2. Перевод из десятичной системы счисления в двоичную (строковая переменная) и обратно

Для перевода десятичного числа в строковую переменную с двоичным числом и обратно используются функции dec2bin() и bin2dec(). Обе эти функции оперируют прямой записью двоичных чисел (старшие разряды располагаются слева).

Функция dec2bin для перевода из десятичной системы счисления в другую (по умолчанию в двоичную) и обратная ей функция bin2dec имеют следующий формат:

```
dec2bin(D, L)
```

```
bin2dec(S)
```

Здесь:

D — десятичное число или матрица чисел, которые необходимо преобразовать;

L — ширина выходного массива B (число колонок). По умолчанию равна [], т. е. определяется результатом. Если значение L задано больше, чем необходимое, то результат будет дополнен нулевыми старшими разрядами. Если L задано меньше, чем необходимое, то оно будет проигнорировано;

S — строка с двоичным числом или массив строк, которые необходимо преобразовать.

Примеры использования функций dec2bin и bin2dec:

```
>> dec2bin(23)
ans = 10111
>> dec2bin(23,7)
ans = 0010111
>> dec2bin([23 25],4)
ans =
      10111
      11001
```

```
>> bin2dec("10111")
ans = 23
>> bin2dec("0010111")
ans = 23
>> bin2dec(["10111"; "11001"])
ans =
      23
      25
```

1.3.3. Перевод из десятичной системы счисления в шестнадцатеричную (строковая переменная) и обратно

Для перевода десятичного числа в строковую переменную с шестнадцатеричным числом и обратно используются функции `dec2hex()` и `hex2dec()`. Они имеют формат, полностью аналогичный функциям для двоичной системы счисления:

```
dec2hex(D, L)
```

```
hex2dec(S)
```

Примеры использования функций `dec2hex` и `hex2dec`:

```
>> dec2hex(23)
ans = 17
>> dec2hex([23 28])
ans =
      17
      1C
```

```
>> hex2dec("17")
ans = 23
>> hex2dec(["17"; "1c"])
ans =
      23
      28
```

1.3.4. Перевод из десятичной системы счисления в произвольную (строковая переменная) и обратно

Для перевода десятичного числа в строковую переменную с числом в произвольной системе счисления и обратно используются функции `dec2base()` и `base2dec()`. Они имеют формат, подобный функциям для двоичной системы счисления:

```
dec2base(D, BASE, L)
```

```
base2dec(S, BASE)
```

`BASE`: Основание системы счисления.

Для систем счисления с основанием, большим 10, в качестве цифр используются буквы.

Примеры использования функций `dec2base` и `base2dec`:

```
>> dec2base(23, 5)
ans = 43
>> dec2base([23 28], 5)
ans =
      043
      103
```

```
>> base2dec("43", 5)
ans = 23
>> base2dec(["43"; "103"], 5)
ans =
      23
      28
```

1.4. Матрицы в Octave

1.4.1. Ввод матриц

Ввод матриц рассмотрим на примере.

Матрица

$$A = \begin{bmatrix} 1 & 3 & 5 & -3 \\ 3 & 5 & 12 & 6 \\ 7 & -4 & -8 & 2 \end{bmatrix}$$

вводится как

```
>> A=[1 3 5 -3 ; 3 5 12 6 ; 7 -4 -8 2]
A =
     1     3     5    -3
     3     5    12     6
     7    -4    -8     2
```

В *Редакторе* матрицу удобно вводить в несколько строк. Символ «;» в конце строки в этом случае необязателен:

```
A=[1 3 5 -3
   3 5 12 6
   7 -4 -8 2]
```

1.4.2. Обращение к элементам массива

Для обращения к элементам матрицы необходимо явно указать их позиции. Обратиться можно как к отдельному элементу, так и к целому диапазону элементов. Рассмотрим несколько примеров обращения к элементам поля.

Обращение к строке 7 массива M:

```
M(7,:)
```

Обращение к столбцу 5 массива M:

```
M(:,5)
```

Обращение к элементам 2–5 строки 6 массива M:

```
M(6,2:5)
```

Обращение к элементам 1–6 столбца 4 массива M:

```
M(1:6,4)
```

1.4.3. Получение размеров матриц

Для определения размеров матрицы используется функция

```
[a b c ...] = size()
```

Если переменные для записи размеров заданы, то соответствующие размеры будут записаны в них. Если переменные не заданы, или задана только одна переменная, то размеры будут выведены в виде одной вектор-строки: первый элемент — число строк, второй — число столбцов.

Например, для вышеприведенной матрицы A размер выводится как

```
>> s=size(A)
s =
   3   4
```

```
>> [a b]=size(A)
a = 3
b = 4
```

1.4.4. Автогенерация матриц

В Octave можно генерировать матрицы необходимого размера с предопределенными значениями.

Матрицы нулей и единиц:

```
>> zeros(2,3)
ans =
   0   0   0
   0   0   0
```

```
>> ones(2,4)
ans =
   1   1   1   1
   1   1   1   1
```

Матрица случайных целочисленных значений в заданном диапазоне (если диапазон не задан, то создается матрица из случайных 0 и 1):

```
>> randint(2,5,[4 13])
ans =
    5   12   13    7    6
   13   10   13   12   13
```

```
>> randint(2,5)
ans =
   0   1   1   1   1
   1   0   0   1   1
```

Альтернативный вариант матрицы случайных целочисленных значений в заданном диапазоне:

```
>> randi([4 13],2,5)
ans =
    4    9    6    9    6
   12   13    8   11    8
```

```
>> randi([0 1],2,5)
ans =
   0   1   0   1   0
   1   0   0   0   0
```

Для генерации случайного вещественного числа в диапазоне $(0,1)$ или массива таких чисел используется функция

```
>> rand
ans = 0.3338

>> rand(2, 3)
ans =
   1.6653e-01   4.0656e-01   6.7517e-01
   6.9799e-02   1.5014e-03   9.6623e-01
```

```
>> rand([2 3])
ans =
   0.1463   0.2291   0.1658
   0.8906   0.2893   0.1647
```

1.4.5. Создание матрицы по данным из файла

Для чтения данных из файла в матрицу удобно использовать функцию

```
>> dlmread(FILE,DELIM,RANGE)
```

Здесь:

`FILE` — имя файла с данными;

`DELIM` — символ, используемый как разделитель столбцов в файле;

`RANGE` — диапазон данных, которые необходимо считать в матрицу. Диапазон задается либо 4-элементным вектором `[R0,C0,R1,C1]` с координатами верхнего левого и нижнего правого углов (индексы берутся от нуля), либо в стиле табличных редакторов: колонки обозначаются буквами от 'A', а строки — числами от 1.

Для примера приведем считывание массива точек данных из файла «data.csv» в матрицу f. В качестве разделителя используется символ «;». Диапазон данных: A19500:B21100 — первые два столбца, строки от 19500 до 21100:

```
f = dlmread('data.csv',';',"A19500:B21100");
```

1.4.6. Операции с матрицами

Для матриц определены все стандартные операции. Сложение и вычитание производятся поэлементно. Матрицы-операнды должны иметь одинаковый размер. Перемножение матриц производится по стандартному правилу «строка на столбец». Таким образом, длина строки первого множителя должна быть равна высоте столбца второго множителя.

Дополнительно определены операции поэлементного умножения, деления и возведения в степень: «.*», «./», «.^». Матрицы-операнды должны иметь одинаковый размер.

Для операции транспонирования используется унарная операция «'».

```
>> A'
ans =
    1    3    7
    3    5   -4
    5   12   -8
   -3    6    2
```

Для вычисления обратной матрицы используется операция возведения в степень «-1» либо функция

```
inv(A)
```

Обратную матрицу можно вычислить только для квадратной матрицы:

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

```
>> A^-1
ans =
  -2.0000   1.0000
   1.5000  -0.5000
```

```
>> inv(A)
ans =
  -2.0000   1.0000
   1.5000  -0.5000
```

1.5. Полиномы (векторы) в Octave

1.5.1. Ввод полиномов

Полином

$$f(x) = 2 + 4x + 5x^2 + 3x^4 + 2x^6$$

записывается вектор-строкой коэффициентов, начиная со старшей степени:

```
>> f=[2 0 3 0 5 4 2]
f =

   2   0   3   0   5   4   2
```

Двоичный полином

$$f(x) = x^4 + x + 1$$

записывается как вектор-строка коэффициентов над простым конечным полем по основанию 2:

```
>> f=gf([1 0 0 1 1],1,3)
f =
GF(2) array.

Array elements =

   1   0   0   1   1
```

В данной записи функция `gf([A],1,3)` предназначена для перевода матрицы/вектора `A` в конечное поле степени 2^1, образованное полиномом $x+1$ (в системе Octave обозначается 3).

1.5.2. Операции с полиномами

Полиномы складываются как вектор-строки (матрицы) коэффициентов. Следовательно, для сложения их необходимо привести к одной длине (по размеру большей строки), добавив нули в начало меньшего вектора.

Для умножения полиномов используется операция *свертки* `conv`, а для деления — обратная ей операция `deconv`. В случае операций умножения и деления начальных нулей в векторе быть не должно. Приведем примеры для двоичных полиномов:

$$a(x) = x^5 + x^4 + x^3 + x + 1,$$
$$b(x) = x^2 + 1.$$

Их произведение:

$$a(x) \cdot b(x) = x^7 + x^6 + x^4 + x^2 + x + 1.$$

Их частное и остаток от деления:

$$\frac{x^5 + x^4 + x^3 + x + 1}{x^2 + 1} = (x^3 + x^2 + 1) + \frac{x}{x^2 + 1}.$$

В Octave:

```
>> a=gf([1 1 1 0 1 1],1,3);
>> b=gf([1 0 1],1,3);
>> conv(a,b)
ans =
GF(2) array.

Array elements =
   1   1   0   1   0   1   1   1

>> [c,r]=deconv(a,b)
c =
GF(2) array.

Array elements =
   1   1   0   1

r =
GF(2) array.

Array elements =
   0   0   0   0   1   0
```

В случае функции деления `deconv`, частное записывается в переменную `c`, а остаток — в переменную `r`. Если явно не указать переменные, то будет выведено только частное.

1.6. Построение графиков

1.6.1. Построение двумерного графика функции

Для построения двумерного графика функции $f(x)$ используется стандартная функция

```
plot(x,y)
```

Параметрами функции являются: x — массив координат по оси абсцисс, а y — массив значений функции (координаты по оси ординат). Эти массивы должны иметь одинаковый размер. Если не задать массив x, то в качестве координат по оси абсцисс будут использованы номера элементов массива y (от 1 до длины массива).

При использовании функции `plot` и по оси абсцисс (ось X) и по оси ординат (ось Y) будет использован линейный масштаб.

Для того чтобы использовать логарифмический масштаб, применяются функции:

`semilogx` — логарифмический масштаб по оси абсцисс;
`semilogy` — логарифмический масштаб по оси ординат;
`loglog` — логарифмический масштаб по обеим осям.

В остальном эти функции аналогичны функции `plot`.

При необходимости построения нескольких графиков на одной координатной сетке в функцию `plot` можно передать сразу несколько функций:

```
plot(x1,y1,x2,y2, ...)
```

Также в функцию `plot` можно передавать параметры, определяющие вид кривой графика функции. Например, для отрисовки первого графика красной линией, а второго графика синими точками необходимо использовать функцию

```
plot(x1,y1,"r",x2,y2,"b.", ...)
```

Полный список доступных цветов, стилей линий и точек приведен в справке:

```
>> help plot
```

1.6.2. Построение трехмерного графика поверхности функции

Для построения трехмерного графика поверхности функции $f(x,y)$ используется функция

```
surf(x,y,z)
```

Параметры x и y — массивы координат по осям абсцисс и ординат, а z — массив значений функции.

Перед построением графика поверхности необходимо задать прямоугольную сетку координат из связанных между собой массивов x и y. Для этого необходимо использовать функцию

```
meshgrid(X array, Y array)
```

Например, чтобы задать диапазон x от -3 до 3 с шагом 0,2, а y от 0 до 5 с шагом 0,2, необходимо использовать функцию

```
[x y]= meshgrid(-3:0.2:3, 0:0.2:5);
```

1.6.3. Способы задания массивов данных

Основной способ задания массива данных:

```
x=b:s:e
```

где b и e — начальное и конечное значения, а s — шаг изменения.

Например, для того чтобы задать массив значений x от 12 до 23 с шагом 0,25, необходимо использовать функцию

```
x=12:0.25:23
```

Также существует функция `linspace`, которая создает вектор равномерных интервалов (иногда также называемый вектором «линейно распределенных значений»).

Общий вид функции:

```
linspace(b,e,c)
```

где b и e — начальное и конечное значения, а c — количество точек между значениями b и e.

Например, для того чтобы задать массив значений x из 200 точек от 0 до 3π, необходимо использовать функцию

```
x=linspace(0,3*pi,200)
```

1.6.4. Подписи осей

Для создания подписей осей координат используются функции

```
xlabel("x");
ylabel("y");
zlabel("z");
```

Эти функции необходимо размещать после функции `plot`.

1.6.5. Название графика

Для вывода названия графика используется функция

```
title("Name of the plot");
```

Эту функцию необходимо размещать после функции `plot`.

1.6.6. Легенда графика

Для вывода легенды используется функция

```
legend("Legend 1","Legend 2", ..., "location", pos);
```

Параметр *pos* определяет месторасположение легенды в окне графика:

- `"best"` — наилучшее с точки зрения программы;
- `"east"`, `"north"`, `"south"`, `"west"` — справа, сверху, снизу, слева;
- `"northeast"`, `"northwest"`, `"southeast"`, `"southwest"` — по углам.

Функцию `legend` необходимо размещать после функции `plot`.

1.6.7. Размещение надписи (метки)

Для размещения на графике произвольной надписи в заданных координатах используется функция

```
text(x,y,"Text of the label");
```

где x и y — координаты по соответствующим осям, левее которых будет выведена надпись.

Эту функцию необходимо размещать после функции `plot`.

1.6.8. Размещение нескольких графиков в одном окне

Для размещения нескольких графиков в одном окне перед каждой функцией `plot` используется функция

```
subplot(rows,columns,position)
```

где `rows` и `columns` указывают число строк и столбцов на которые делится окно графика, а `position` — расположение текущего графика.

Например, для размещения в окне шести графиков — два по горизонтали, три по вертикали — используется функция

```
subplot(3,2,Position)
```

`Position` может принимать значения от 1 до 6. Отсчет идет с левого верхнего графика обычным способом — слева направо, сверху вниз.

1.6.9. Ограничение графика по осям

Для ограничения графика по осям абсцисс и ординат используются функции

```
xlim([X1, X2]);
ylim([Y1, Y2]);
```

где $X1$ и $X2$ — нижняя и верняя границы диапазона по оси абсцисс, а $Y1$ и $Y2$ — по оси ординат.

Эти функции необходимо размещать после функции `plot`.

1.6.10. Вывод сетки

Для вывода сетки с заданными диапазоном и шагом используется совокупность функций

```
set(gca,'XTick',X1:Xs:X2)
set(gca,'YTick',Y1:Ys:Y2)
grid
```

где $X1$ и $X2$ — нижняя и верняя границы диапазона по оси абсцисс, а Xs — шаг сетки. Для оси ординат аналогично.

Эти функции необходимо размещать после функции `plot`.

1.7. Условия в Octave

Условный оператор (if-else) реализуется функцией

```
if condition1
      operations1;
   elseif condition2
      operations2;
   else
      operations3;
endif
```

```
if a==5
      b=6;
   elseif a<5
      b=a;
   else
      b=8;
endif
```

Здесь `condition1` и `condition2` — условия соответствующих веток оператора, а `operations[1,2,3]` — функции, выполняющиеся в соответствующей ветке оператора.

Ветки `elseif` и `else` могут отсутствовать.

Условия `condition1` и `condition2` можно заключать в скобки. В случае формирования сложных составных условий использование скобок обязательно для указания порядка проверки условий.

1.8. Циклы в Octave

В Octave используются все три типа стандартных циклов: с предусловием, с постусловием, с заданным числом повторений. Необходимо помнить, что циклы в Octave выполняются достаточно медленно, и при написании программ, там, где это возможно, следует использовать матрицы — это позволяет значительно ускорить выполнение скриптов Octave.

Цикл `while` с предусловием записывается как

```
while (condition)
  body
endwhile
```

Цикл выполняется, пока условие `condition` истинно (не равно 0).

Цикл `do-until` с постусловием записывается как

```
do
  body
until (condition)
```

Цикл выполняется до тех пор, пока условие `condition` не станет истинным (т. е. не равным 0).

Цикл `for` с заданным числом повторений записывается как

```
for i = first:step:last
  body
endfor
```

Здесь `first` — начальное значение счетчика i; `step` — шаг изменения счетчика i; `last` — конечное значение счетчика i; `body` — те функции, которые будут выполняться в цикле.

1.9. Полезные математические функции

В этом разделе рассмотрим ряд математических функций, которые могут быть полезны при моделировании алгоритмов и систем помехоустойчивого кодирования.

1.9.1. Расчет среднего значения

Для расчета среднего значения (математического ожидания) одномерного вектора элементов следует использовать функцию

```
mean(x)
```

```
>> mean([1 3 5 6])
ans = 3.7500
```

1.9.2. Расчет стандартного отклонения

Стандартное отклонение (оценка среднеквадратического отклонения случайной величины относительно ее математического ожидания) считается по формуле

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

где $\bar{x}$ — среднее значение одномерного вектора элементов.

Для расчета стандартного отклонения вектора элементов следует использовать функцию

```
std(x)
```

```
>> std([1 3 5 6])
ans = 2.2174
```

1.9.3. Квадратный корень числа

Для вычисления квадратного корня числа используется функция

```
sqrt(x)
```

```
>> sqrt(16)
ans = 4
```

1.9.4. Кубический корень числа

Для вычисления кубического корня числа используется функция

```
cbrt(x)
```

```
>> cbrt(64)
ans = 4
```

1.9.5. Экспонента

Для вычисления экспоненты числа используется функция

```
exp(x)
```

```
>> exp(2.5)
ans = 12.182
```

1.9.6. Логарифм

Для вычисления натурального логарифма числа используется функция

```
log(x)
```

```
>> log(8)
ans = 2.0794
```

Десятичный логарифм вычисляется функцией

```
log10(x)
```

```
>> log10(8)
ans = 0.9031
```

Логарифм по основанию 2 вычисляется функцией

```
log2(x)
```

```
>> log2(8)
ans = 3
```

1.9.7. Сумма элементов массива

Для вычисления суммы элементов матрицы `A` используется функция

```
sum(A)
```

```
>> sum([0.1 2 5 4 2])
ans = 13.100
```

Если матрица `A` является двумерной, то сумма вычисляется по столбцам матрицы.

1.9.8. Произведение элементов массива

Произведение элементов матрицы A вычисляется функцией

```
prod(A)
```

```
>> prod([0.1 2 3 4])
ans = 2.4000
```

Если матрица A является двумерной, то произведение вычисляется по столбцам матрицы.

1.9.9. Округление дробных чисел

Для округления дробных чисел используется функция

```
round(x)
```

```
>> round([-3.7 3.7])
ans =
  -4   4
```

Округление дробного числа «вверх» (до ближайшего целого в большую сторону) выполняется функцией

```
ceil(x)
```

```
>> ceil([-3.7 3.7])
ans =
  -3   4
```

Округление дробного числа «вниз» (до ближайшего целого в меньшую сторону) выполняется функцией

```
floor(x)
```

```
>> floor([-3.7 3.7])
ans =
  -4   3
```

1.9.10. Определение минимального и максимального значений

Для определения минимального и максимального значений среди элементов вектора используются функции min и max соответственно. Обе функции возвращают значение минимального/максимального элемента и его индекс. Если в векторе несколько элементов с одинаковым минимальным/максимальным значением, то функции возвращают индекс первого такого элемента слева:

```
>> [m i]=min([6 5 2 9 2 3 9])
m = 2
i = 3
```

```
>> [m i]=max([6 5 2 9 2 3 9])
m = 9
i = 4
```

1.10. Моделирование каналов ПД в Octave

В системе Octave модели каналов определены в пакете функций *communications*. На данный момент доступны две модели каналов: модель двоичного симметричного канала (ДСК) и модель канала с абсолютно-белым гауссовским шумом (АБГШ). Прочие модели каналов необходимо реализовывать самостоятельно с использованием стандартных механизмов Octave.

1.10.1. Модель канала ДСК

В системе Octave канал ДСК реализуется функцией

```
bsc(B, P)
```

Здесь:

`B` — данные в двоичном виде (одиночная цифра или массив);

`P` — вероятность битовой ошибки в канале.

Например, для передачи последовательности бит $[1011101]$ через канал ДСК с вероятностью ошибки $p_0 = 0{,}1$ необходимо использовать функцию

```
>> bsc([1 0 1 1 1 0 1], 0.1)
ans =
   1   0   0   1   1   0   0
```

1.10.2. Модель канала АБГШ

В системе Octave канал АБГШ реализуется функцией

```
awgn(X, SNR, PWR)
```

Здесь:

`X` — массив значений сигнала в виде вещественных или комплексных чисел в зависимости от применяемого типа модуляции;

`SNR` — отношение сигнал/шум в дБ. Применяется с учетом уровня мощности сигнала `PWR`;

`PWR` — уровень мощности сигнала в дБВт. Опциональный параметр. Если `PWR` не задан, то он считается равным 0 дБВт.

1.10.3. Тестирование каналов в Octave

Для тестирования моделей каналов связи как правило используется *метод Монте-Карло*. Это численный метод, основанный на получении большого числа реализаций случайного процесса, который формируется так, чтобы вероятностные характеристики были равны величинам решаемой задачи.

1.10.4. Создание модели канала ДСК

В качестве простого примера создания собственной модели канала в Octave рассмотрим создание модели канала ДСК, граф которой представлен на рис. 1.3 [6, 7].

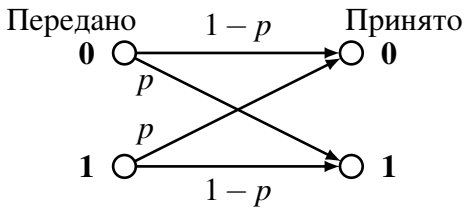

Рис. 1.3. Граф модели канала ДСК

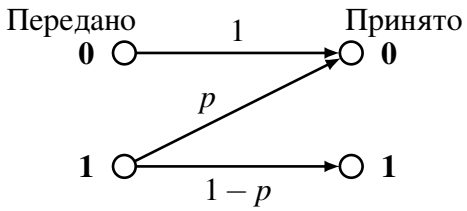

Рис. 1.4. Граф модели Z-канала

Фактически, работу канала ДСК можно представить следующим образом: при прохождении через канал бита данных срабатывает генератор случайного вещественного числа в диапазоне $(0, 1)$ и если результат меньше либо равен вероятности ошибки в канале p, то значение бита меняется на противоположное, а в противном случае остается неизменным.

Такой алгоритм может быть реализован представленной в листинге 1.6 функцией `MyBSC`.

Листинг 1.6

Функция `MyBSC`, моделирующая канал ДСК

```
1 function OUT = MyBSC(IN, P)
2   OUT = xor(IN, rand(size(IN)) <= P);
3 endfunction
```

Здесь:

`P` — вероятность битовой ошибки в канале;
`IN` — массив входных данных;
`OUT` — массив выходных данных.

Выражение `(rand(size(IN)) <= P)` генерирует массив случайных вещественных чисел того же размера, что и массив входных данных `IN`, сравнивает каждое число с вероятностью ошибки `P` и, если оно меньше либо равно `P`, то возвращает 1, а если больше `P`, то возвращает 0. В итоге получается массив двоичных ошибок, которые затем накладываются на входные данные функцией `xor`.

Следует отметить, что данная функция, в отличие от встроенной в Octave функции `bsc`, не осуществляет проверку входных данных. Это необходимо учитывать при написании программ.

1.10.5. Создание модели несимметричного Z-канала

В качестве другого примера создания собственной модели канала рассмотрим создание модели двоичного несимметричного Z-канала, граф которой представлен на рис. 1.4 [6, 7].

Модель Z-канала отличается тем, что при передаче через нее данных ошибка возможна только при передаче 1. Это необходимо учитывать при моделировании. Самый простой вариант — это использовать условный оператор `if` и цикл `for`, как показано в листинге 1.7. Пример реализован для двумерного массива входных данных.

Листинг 1.7

Функция `ZC` на основе цикла, моделирующая Z-канал

```
function OUT = ZC(IN, P)
  [M N] = size(IN);
  OUT = zeros(M, N);
  for i=1:1:M
    for j=1:1:N
      if IN(i,j)
        OUT(i,j) = xor(IN(i,j), rand <= P);
      endif
    endfor
  endfor
  OUT = xor(IN, rand(size(IN)) <= P);
endfunction
```

Такая функция выглядит достаточно наглядно, но будет работать заметно медленнее модели канала ДСК в листинге 1.6, которая оперирует целой матрицей.

Чтобы понять, как можно упростить реализацию модели Z-канала, напишем ее таблицу состояний (табл. 1.1).

Данная таблица не соответствует обычным логическим функциям. Однако если рассмотреть обратную проверку — проверку не ошибочной передачи, а правильной передачи через канал с условием `rand > P`, то полученная таблица (табл. 1.2) будет полностью соответствовать функции логического умножения «И», а значит, модель Z-канала можно свести к коду, представленному в листинге 1.8.

Листинг 1.8

Функция `ZC` на основе логического умножения «И», моделирующая Z-канал

```
function OUT = ZC(IN, P)
  OUT = and(IN, rand(size(IN)) > P);
endfunction
```

Следует отметить, что обе приведенные функции Z-канала не осуществляют проверку входных данных. Это необходимо учитывать при написании программ.

Таблица 1.1

Состояния Z-канала (проверка ошибочной передачи)

Вход	rand <= P	Выход
0	0	0
0	1	0
1	0	1
1	1	0

Таблица 1.2

Состояния Z-канала (проверка правильной передачи)

Вход	rand > P	Выход
0	0	0
0	1	0
1	0	0
1	1	1

1.11. Сравнение матриц данных (количество и доля ошибок)

Для сравнения матрицы с некоторой эталонной матрицей и определения количества и доли ошибок в ней можно использовать функции

```
[NUM, RATE] = biterr(A, B)
```

```
[NUM, RATE] = symerr(A, B)
```

Здесь:

A и B — сравниваемые матрицы (должны быть одного размера);
NUM — количество ошибок;
RATE — доля (вероятность) ошибок.

Отличие функций в том, что symerr сравнивает матрицы посимвольно, т. е. показывает количество символьных ошибок, а biterr предварительно переводит элементы матрицы в двоичный вид, т. е. показывает количество битовых ошибок.

В случае когда элементы матриц состоят из 0 и 1, обе функции покажут одинаковое количество ошибок. В ином случае результат может отличаться:

```
>> A=[1 1 1 1
      1 1 1 1];
>> B=[1 1 0 1
      0 1 1 1];

>> [num, rate] = biterr(A, B)
num = 2
rate = 0.2500

>> [num, rate] = symerr(A, B)
num = 2
rate = 0.2500
```

```
>> A=[1 1 2 1
      3 1 1 1];
>> B=[1 1 0 1
      0 1 1 1];

>> [num, rate] = biterr(A, B)
num = 3
rate = 0.1875

>> [num, rate] = symerr(A, B)
num = 2
rate = 0.2500
```

В правой части приведенного примера функция biterr перевела все элементы матриц A и B в двоичный вид. В результате каждый элемент матрицы был преобразован в двоичное число одинаковой длины 2. Таким образом, общий размер каждой матрицы составил $2 \cdot 8 = 16$ бит, а вероятность ошибки равна $3/16 = 0{,}1875$.

1.12. Работа с полями Галуа

1.12.1. Вывод образующих полиномов

Для вывода образующих полиномов на экран используется команда

```
primpoly(M, OPT)
```

Здесь:

M — степень полинома в диапазоне от 1 до 22.

OPT — опция вывода. Если ее задать равной ''all'', то будут выведены все возможные полиномы степени M.

Обратите внимание, что вместо «x» используется «D». То есть вместо полинома $x^4 + x + 1$ на экран выведено D^7+D+1. Ниже символьного представления полиномов указываются их десятичные представления в системе Octave, соответствующие их векторам коэффициентов от младшей степени к старшей. Если указана переменная, как в примере, то функция записывает в нее вектор десятичных представлений:

```
>> primpoly(4)

Primitive polynomial(s) =
D^4+D+1

ans = 19
```

```
>> pol=primpoly(4,"all")

Primitive polynomial(s) =
D^4+D+1
D^4+D^3+1

pol =  19   25
```

1.12.2. Задание чисел как элементов поля Галуа

Для того чтобы задать числа или инициализированные переменные как элементы поля Галуа, используется функция

```
gf(X, M, PRIM)
```

Здесь:

X — число или матрица чисел в диапазоне от 0 до $2^M - 1$;

M — степень образующего полинома в диапазоне от 1 до 16;

PRIM — образующий полином поля в виде десятичного числа.

```
>> a=gf([2 0 14 15 5 11],4,19)
a =
GF(2^4) array. Primitive Polynomial = D^4+D+1 (decimal 19)

Array elements =
    2    0   14   15    5   11
```

Для того чтобы использовать матрицу элементов поля Галуа как обычные десятичные числа, применяется модификатор-постфикс `.x`:

```
>> a.x
ans =
    2    0   14   15    5   11
```

1.12.3. Вычисление и вывод поля Галуа

Стандартной функции для вывода всего поля Галуа по порядку в Octave нет. Для этого можно использовать функцию `gf_field`, представленную в листинге 1.9.

Листинг 1.9

Листинг программы для формирования поля Галуа

```
1 function gfFLD = gf_field(gfSTP,gfPOL)
2    gfLNG = 2^gfSTP-1;
3    gfFLD = gf(zeros(1,gfLNG),gfSTP,gfPOL);
4    gfFLD(1) = 1;
5    gfFLD(2) = 2;
6    for CNTR = 3:1:gfLNG
7       gfFLD(CNTR) = gfFLD(CNTR-1)*gfFLD(2);
8    endfor
9 endfunction
```

Функция получает на вход степень поля $\mathrm{GF}(2^m)$ и код образующего полинома, а возвращает массив (матрицу) элементов поля.

Представим пример для полинома x^3+x+1:

```
>> gf_field(3, 11)
ans =
GF(2^3) array. Primitive Polynomial = D^3+D+1 (decimal 11)

Array elements =
   1   2   4   3   6   7   5
```

1.12.4. Определение циклотомических классов

Для определения циклотомических классов, т. е. групп элементов поля Галуа, являющихся корнями одного минимального многочлена, используется функция

```
cosets(M, PRIM)
```

Здесь:

`M` — степень образующего полинома;

`PRIM` — образующий полином поля в виде десятичного числа.

Эта функция возвращает структуру, содержащую все циклотомические классы для указанного поля. Если образующий полином поля `PRIM` не задан,

то используется поле, у которого образующий полином имеет минимальное значение в десятичном виде:

```
>> cosets(3)
ans =
{
  [1,1] =
  GF(2^3) array. Primitive Polynomial = D^3+D+1 (decimal 11)

  Array elements =
     1

  [1,2] =
  GF(2^3) array. Primitive Polynomial = D^3+D+1 (decimal 11)

  Array elements =
     2   4   6

  [1,3] =
  GF(2^3) array. Primitive Polynomial = D^3+D+1 (decimal 11)

  Array elements =
     3   5   7
}
```

1.12.5. Определение минимального полинома

Для определения минимального полинома, соответствующего элементу поля Галуа, используется функция

```
minpol(EL)
```

где EL — элемент поля Галуа или вектор элементов поля.

Пример:

```
>> minpol(gf([1 5],3))
ans =
GF(2) array.

Array elements =
   0   0   1   1
   1   1   0   1
```

1.12.6. Вычисление функции-след

Функцией-след (следом) элемента ε поля $\mathrm{GF}(p^m)$ в простом поле $\mathrm{GF}(p)$ называют сумму следующих p-сопряженных элементов поля:

$$T(\varepsilon^i) = \sum_{k=0}^{m-1} \left(\varepsilon^i\right)^{p^k}.$$

Стандартной функции для вычисления функции-след в Octave нет. Для этого можно использовать функцию `gf_trace`, которая представлена в листинге 1.10.

Листинг 1.10

Листинг программы для вычисления функции-след

```
function gfTr=gf_trace(gfEL,gfSTP)
   gfTr=gfEL;
   for CNTR=1:1:gfSTP-1
      gfTr+=gfEL^(2^CNTR);
   endfor
endfunction
```

Функция получает на вход элемент поля, степень поля $\mathrm{GF}(2^m)$ и код образующего полинома, а возвращает значение функции-след.

Представим пример для полинома x^3+x+1:

```
>> gf_trace(gf(5,3,11), 3)
ans =
GF(2^3) array. Primitive Polynomial = D^3+D+1 (decimal 11)

Array elements =
   1
```

1.13. Моделирование помехоустойчивых кодов

1.13.1. Кодирование и декодирование двоичных кодов

Для кодирования двоичными помехоустойчивыми (n,k)-кодами используется функция

```
CODE = encode(MSG, N, K, TYPE, OPT)
```

Здесь:

`CODE` — переменная, в которую будет записано кодовое слово или массив кодовых слов;

`MSG` — информационное слово или массив информационных слов. Если вид кода задан без модификатора или с модификатором `binary`, то слово должно быть представлено вектором двоичных цифр длины K или массивом таких векторов. Кодовое слово в этом случае будет представлено вектором длины N или массивом таких векторов. Если использован модификатор `decimal`, то информационное слово должно представляться десятичным числом в диапазоне от 0 до 2^K-1 или массивом таких чисел. Кодовое слово будет представлено числом в диапазоне от 0 до 2^N-1 или массивом таких чисел;

`N` и `K` — длина кодового слова и длина информационного слова соответственно;

TYPE — вид двоичного помехоустойчивого кода:

– "linear", "linear/binary", "linear/decimal" — линейный блоковый код; в качестве параметра OPT может быть указана порождающая матрица кода;

– "cyclic", "cyclic/binary", "cyclic/decimal" — циклический код; в качестве параметра OPT может быть указан порождающий полином;

– "hamming", "hamming/binary", "hamming/decimal" — код Хэмминга; в качестве параметра OPT может быть указан порождающий полином;

– "bch", "bch/binary", "bch/decimal" — код БЧХ; в качестве параметра OPT может быть указан порождающий полином.

В следующем примере в реальном выводе функции encode двоичное сообщение выводится в виде вектор-столбца (для упрощения и наглядности в пособии он заменен на вектор-строку):

```
>> encode([0 1 0 1],7,4,"hamming")
ans =
   1   1   0   0   1   0   1

>> encode([10],7,4,"hamming/decimal")
ans = 83
```

```
>> de2bi(10)
ans =
   0   1   0   1
```

```
>> de2bi(83)
ans =
   1   1   0   0   1   0   1
```

Обратная операция декодирования реализуется функцией

```
[MSG, ERR, CCODE, CERR] = decode(CODE, N, K, TYP, OPT1, OPT2)
```

Здесь:

CCODE — в эту переменную функция возвращает декодированное кодовое слово полностью, в отличие от переменной MSG, в которую возвращается только информационная часть декодированного слова;

ERR — количество ошибок в слове. Для двоичного представления представляется вектором того же размера, что и MSG. Отрицательное значение указывает на неисправляемую ошибку;

CERR — то же самое, что и ERR, но для всего кодового слова;

OPT1 — опциональный параметр. Для линейных/циклических кодов и кодов Хэмминга содержит порождающую матрицу или порождающий полином соответственно. Для кодов БЧХ — исправляющую способность кода;

OPT2 — опциональный параметр. Для линейных и циклических кодов содержит таблицу синдромов. Для кодов БЧХ — образующий полином.

Одноименные с функцией encode параметры имеют то же самое значение и особенности применения.

В следующем примере в реальном выводе функции `decode` двоичные сообщения выводятся в виде вектор-столбцов (для упрощения и наглядности в пособии они заменены на вектор-строки):

```
>> [MSG, ERR, CCODE, CERR] = decode([1 1 0 0 1 0 1],7,4,"hamming")
MSG =
   0 1 0 1
ERR =
   0 0 0 0
CCODE =
   1 1 0 0 1 0 1
CERR =
   0 0 0 0 0 0 0

>> [MSG, ERR, CCODE, CERR] = decode([1 1 0 0 1 1 1],7,4,"hamming")
MSG =
   0 1 0 1
ERR =
   1 1 1 1
CCODE =
   1 1 0 0 1 0 1
CERR =
   1 1 1 1 1 1 1

>> [MSG, ERR, CCODE, CERR] = decode([83],7,4,"hamming/decimal")
MSG = 10
ERR = 0
CCODE = 83
CERR = 0

>> [MSG, ERR, CCODE, CERR] = decode([115],7,4,"hamming/decimal")
MSG = 10
ERR = 1
CCODE = 83
CERR = 1
```

1.13.2. Определение параметров кода Хэмминга

Для определения и вывода параметров (n, k)-кода Хэмминга по числу проверочных элементов $r = n - k$ используется функция

```
[H, G, N, K] = hammgen(R)
```

Здесь:

`R` — число проверочных элементов в коде Хэмминга. Может быть задано в диапазоне от 3 до 16;

`N` и `K` — длина кодового слова $n = 2^r - 1$ и длина информационного слова $k = n - r$ соответственно;

`H` — проверочная матрица кода Хэмминга;

`G` — порождающая матрица кода Хэмминга.

Пример:

```
>> [H, G, N, K] = hammgen(3)
H =
    1    0    0    1    0    1    1
    0    1    0    1    1    1    0
    0    0    1    0    1    1    1
G =
    1    1    0    1    0    0    0
    0    1    1    0    1    0    0
    1    1    1    0    0    1    0
    1    0    1    0    0    0    1
N = 7
K = 4
```

1.13.3. Определение параметров циклического кода

Для получения образующего полинома циклического (n, k)-кода необходимо использовать функцию

```
cyclpoly(N, K, OPT)
```

Здесь:

N и K — длина кодового слова и длина информационного слова соответственно;

OPT — опциональный параметр, который указывает, какие образующие полиномы выводить: "max" — с максимальным весом; "min" — с минимальным весом (по умолчанию); "all" — все полиномы.

Для вывода проверочной и порождающей матриц кода используется функция

```
[H, G] = cyclgen(N, P, TYP)
```

Здесь:

P — порождающий полином кода;

H и G — проверочная и порождающая матрицы кода соответственно;

TYP — опциональный параметр, указывающий тип кода: "system" — систематический код; "nosys" — несистематический код.

Пример:

```
>> p = cyclpoly(7, 4, "max")
p =
    1    0    1    1

>> [H, G] = cyclgen(7, p)
H =
    1    0    0    1    1    1    0
    0    1    0    0    1    1    1
    0    0    1    1    1    0    1
```

```
G =
     1     0     1     1     0     0     0
     1     1     1     0     1     0     0
     1     1     0     0     0     1     0
     0     1     1     0     0     0     1
```

1.13.4. Работа с кодами БЧХ

Параметры и порождающий полином кода Боуза–Чоудхури–Хоквингема (БЧХ) определяются функцией

```
[P, F, C, H, T] = bchpoly(N, K)
```

Здесь:

N и K — длина кодового слова и длина информационного слова соответственно;

P — порождающий полином кода;

F — минимальные полиномы, составляющие порождающий полином;

C — циклотомические классы поля Галуа, над которым построен код (т. е. корни минимальных полиномов);

H — проверочная матрица кода;

T — количество гарантированно исправляемых ошибок.

При запуске без параметров функция bchpoly возвращает полный список (n, k)-кодов БЧХ, которые могут быть промоделированы. Если задан только один параметр N, то возвращается список кодов длины N. В третьем столбце выводится число гарантированно исправляемых кодом ошибок:

```
>> bchpoly
ans =
     7     4     1
    15    11     1
    ...
   511    10   127
```

```
>> bchpoly(15)
ans =
    15    11     1
    15     7     2
    15     5     3
```

Если задать оба параметра N и K, то функция вернет все переменные P, F, C, H, T:

```
>> [p f c h t]=bchpoly(15,7)
p =
     1     0     0     0     1     0     1     1     1

f =
     1     1     0     0     1
     1     1     1     1     1
```

```
c =
{ ...
  [1,2] =
  GF(2^4) array. Primitive Polynomial = D^4+D+1 (decimal 19)

  Array elements =
     2    3    4    5
  ...
}

h =
    1   0   0   0   0   0   0   0   1   1   0   1   0   0   0
    0   1   0   0   0   0   0   0   0   1   1   0   1   0   0
    0   0   1   0   0   0   0   0   0   0   1   1   0   1   0
    0   0   0   1   0   0   0   0   0   0   0   1   1   0   1
    0   0   0   0   1   0   0   0   1   1   0   1   1   1   0
    0   0   0   0   0   1   0   0   0   1   1   0   1   1   1
    0   0   0   0   0   0   1   0   1   1   1   0   0   1   1
    0   0   0   0   0   0   0   1   1   0   1   0   0   0   1

t = 2
```

В примере приведена только часть структуры, в которую выводятся циклотомические классы.

Для кодирования и декодирования кодом БЧХ используются функции

```
CODE = bchenco(MSG, N, K)        MSG = bchdeco(CODE, K, T)
```

Здесь:

CODE — переменная, в которую будет записано кодовое слово или массив кодовых слов;

MSG — информационное слово или массив информационных слов;

N и K — длина кодового слова и длина информационного слова соответственно;

T — количество гарантированно исправляемых ошибок.

Пример:

```
>> c = bchenco([1 0 0 1 0 1 1], 15, 7)
c =
    0   0   0   0   1   0   1   0   1   0   0   1   0   1   1

>> m = bchdeco(c, 7, 2)
m =
    1   0   0   1   0   1   1
```

Учитывая, что эти функции используют разные параметры кода, удобнее использовать функции encode и decode (разд. 1.13.1).

1.13.5. Работа с недвоичными кодами Рида–Соломона

Порождающий полином недвоичного кода Рида–Соломона (РС) определяется функцией

```
[G, T] = rsgenpoly(N, K, P)
```

Здесь:

N и K — длина кодового слова и длина информационного слова соответственно;

P — образующий полином поля Галуа (можно не задавать);

G — порождающий полином кода;

T — количество гарантированно исправляемых ошибок.

Пример:

```
>> [g t] = rsgenpoly(15, 9)
g =
GF(2^4) array. Primitive Polynomial = D^4+D+1 (decimal 19)

Array elements =
    1    7    9    3   12   10   12

t = 3
```

Для кодирования и декодирования кодом РС используются функции

```
CODE = rsenc(MSG, N, K, G)
```

```
MSG = rsdec(CODE, N, K, G)
```

Здесь:

CODE — переменная, в которую будет записано кодовое слово или массив кодовых слов;

MSG — информационное слово или массив информационных слов;

N и K — длина кодового слова и длина информационного слова соответственно;

G — порождающий полином кода (можно не задавать).

Пример:

```
>> c = rsenc(gf([1 2 3 4 5 6 7 8 9],4,19), 15, 9)
c =
GF(2^4) array. Primitive Polynomial = D^4+D+1 (decimal 19)

Array elements =
   1   2   3   4   5   6   7   8   9   2   1   3  12  15  11

>> m = rsdec(c, 15, 9)
m =
GF(2^4) array. Primitive Polynomial = D^4+D+1 (decimal 19)

Array elements =
   1   2   3   4   5   6   7   8   9
```

Список использованных источников

1. Алексеев, Е. Р. Введение в Octave для инженеров и математиков / Е. Р. Алексеев, О. В. Чеснокова. — Москва : ALT Linux, 2012. — 368 с.

2. Алексеев, Е. Р. Введение в Octave / Е. Р. Алексеев, О. В. Чеснокова // НОУ ИНТУИТ : [сайт]. — 2021. — URL: http://www.intuit.ru/studies/courses/3677/919/info (дата обращения: 20.01.2021).

3. Eaton, J. W. GNU Octave. A high-level interactive language for numerical computations. Edition 6 for Octave version 6.2.0 / J. W. Eaton, D. Bateman, S. Hauberg, R. Wehbring // GNU Octave : [сайт]. — 2021. — URL: https://octave.org/doc/octave-6.2.0.pdf (дата обращения: 20.01.2021).

4. Eaton, J. W. GNU Octave (version 6.2.0) Documentation / J. W. Eaton // GNU Octave : [сайт]. — 2021. — URL: https://octave.org/doc/v6.2.0/ (дата обращения: 20.01.2021).

5. Documentation // Octave-Forge : [сайт]. — 2021. — URL: https://octave.sourceforge.io/docs.php (дата обращения: 20.01.2021).

6. Владимиров, С. С. Математические основы теории помехоустойчивого кодирования / С. С. Владимиров ; СПбГУТ. — Санкт-Петербург, 2016.

7. Когновицкий, О. С. Практика помехоустойчивого кодирования : в 2 ч. : учебное пособие. Часть 1. Системы с обнаружением ошибок и обратной связью : учебное пособие / О. С. Когновицкий, В. М. Охорзин, С. С. Владимиров ; СПбГУТ. — Санкт-Петербург, 2018.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Порядок выполнения лабораторных работ

Лабораторные работы выполняются бригадой учащихся. Количество студентов в бригаде зависит от количества студентов в группе и согласовывается с преподавателем на первом занятии. Вариант работы выбирается по данным одного из членов бригады.

По результатам каждой работы необходимо сформировать отчет. Отчет выполняется один на бригаду и отправляется преподавателю в электронном виде в формате PDF по электронной почте. Допускается сдавать отчет в распечатанном виде.

Содержание отчета

Отчет по лабораторной работе должен включать:

1) титульный лист с названием работы, номером варианта, номером группы и Ф. И. О. учащихся (образец приведен в приложении):
2) цель работы;
3) формулировку задания;
4) ход выполнения работы и результаты вывода команд;
5) результаты вычислений и полученные значения или параметры;
6) программный код решения задачи (при наличии);
7) графики и таблицы исходных данных и результатов (при наличии);
8) схемы и диаграммы исходных данных и результатов (при наличии);
9) анализ полученных результатов и вывод о проделанной работе.

Порядок представления данных и результатов пп. 4–8 определить самостоятельно, исходя из логики задания.

Порядок защиты лабораторной работы

Защита работы может осуществляться одним из следующих способов или их сочетанием на усмотрение преподавателя:

1) устный ответ по теме работы;
2) тестирование по теме работы;
3) задача по теме работы;
4) иные варианты на усмотрение преподавателя.

Лабораторная работа 1
Основы работы с системой компьютерной алгебры Octave

Цель работы

Ознакомиться с общими принципами работы в системе компьютерной алгебры Octave. Ознакомиться с общими принципами построения графиков в системе компьютерной алгебры Octave. Получить навыки по использованию сценариев в графическом интерфейсе Octave.

Порядок выполнения задания

1. Перевести десятичное число d в двоичную систему счисления и совершить обратное преобразование. Число задано в табл. 2.1.

2. Для заданных в табл. 2.1 матриц A и B осуществить следующие операции:

а) поэлементное сложение матриц;

б) транспонирование матрицы B;

в) умножение матрицы A на транспонированную матрицу B с записью результата в матрицу C;

г) вычисление обратной матрицы для C.

3. Для заданных в табл. 2.1 полиномов $a(x)$ и $b(x)$ осуществить следующие операции:

а) сложение полиномов;

б) умножение полиномов;

в) деление полинома $a(x)$ на полином $b(x)$;

г) преобразование полиномов $a(x)$ и $b(x)$ в двоичные полиномы над полем GF(2);

д) сложение двоичных полиномов;

е) умножение двоичных полиномов;

ж) деление двоичного полинома $a(x)$ на двоичный полином $b(x)$.

4. Построить график функции

$$f(x) = \sin(x) + a_1 \sin(\omega_1 x) + a_2 \sin(\omega_2 x)$$

для параметров, заданных в табл. 2.2, и диапазона x от -10 до $+10$ с шагом 0,1 (выполнить отдельным сценарием/скриптом):

а) создать и сохранить новый сценарий/скрипт;

б) график построить красной сплошной линией;

в) задать подписи осей абсцисс (x) и ординат ($f(x)$);

г) задать название графика — номер группы, Ф. И. О. студентов, вариант, номер задания;

д) разместить на графике надпись (метку) с формулой построенной функции;

е) изменить график так, чтобы на нем в дополнение к функции $f(x)$ отображалась функция

$$f_2(x) = \cos(x) + a_1 \cos(\omega_1 x) + a_2 \cos(\omega_2 x),$$

вычисленная для тех же исходных параметров и в том же диапазоне x; цвет нового графика — синий;

ж) добавить на график легенду.

5. Построить в одном окне два графика (один над другим) по данным из файла (выполнить отдельным сценарием):

а) создать и сохранить новый сценарий;

б) скачать с сайта файл «data.csv» с точками данных;

в) считать содержимое файла в массив: в качестве разделителя используется символ «;», считать необходимо первые два столбца, строки от 19500 до 21100 (разд. 1.4.5);

г) в качестве первого (верхнего) графика взять весь считанный из файла диапазон: первый столбец — координаты по оси абсцисс (x), второй столбец — координаты по оси ординат (y);

д) ниже изобразить второй график по тому же набору данных, но ограниченный диапазоном x_1–x_2 (табл. 2.2);

е) на нижнем графике вывести сетку с шагом 5 по оси абсцисс и шагом по оси ординат на выбор студента;

ж) для каждого графика задать подписи осей, название (номер группы, Ф. И. О. студентов, вариант, номер задания) и легенду.

6. Построить трехмерный график поверхности функции. Выполнить отдельным сценарием.

а) создать и сохранить новый сценарий;

б) построить график поверхности функции

$$f(x,y) = \sqrt{a_1(\sin(\omega_1 x))^2 + a_2(\cos(\omega_2 y))^2}$$

для параметров, заданных в табл. 2.2, и диапазона x от -2 до $+2$ с шагом 0,05, а y — от 0 до 4 с шагом 0,05;

в) задать подписи осей и название (номер группы, Ф. И. О. студентов, вариант, номер задания).

Варианты заданий

Таблица 2.1

Варианты заданий (указаны согласно номеру студента в журнале)

№	d	A	B	$a(x)$	$b(x)$
1	2068	$\begin{bmatrix} 7 & 5 & 15 & 19 \\ 8 & 17 & 14 & 9 \\ 15 & 4 & 17 & 18 \end{bmatrix}$	$\begin{bmatrix} 7 & 5 & 10 & 11 \\ 5 & 4 & 13 & 5 \\ 11 & 2 & 13 & 18 \end{bmatrix}$	$x^7+x^5+x^4+x^2$	x^5+x^2+1
2	2040	$\begin{bmatrix} 13 & 6 & 7 & 1 \\ 8 & 15 & 19 & 6 \\ 18 & 11 & 9 & 10 \end{bmatrix}$	$\begin{bmatrix} 1 & 6 & 11 & 5 \\ 20 & 11 & 11 & 7 \\ 17 & 18 & 6 & 4 \end{bmatrix}$	$x^7+x^6+x^3+x^2$	x^5+x^3+1
3	2546	$\begin{bmatrix} 4 & 8 & 2 & 1 \\ 16 & 4 & 8 & 16 \\ 5 & 3 & 5 & 12 \end{bmatrix}$	$\begin{bmatrix} 15 & 15 & 15 & 11 \\ 2 & 16 & 19 & 14 \\ 10 & 20 & 13 & 12 \end{bmatrix}$	$x^8+x^5+x^3+x^2$	x^4+x^2+1
4	2504	$\begin{bmatrix} 3 & 8 & 2 & 19 \\ 14 & 11 & 2 & 17 \\ 12 & 1 & 2 & 12 \end{bmatrix}$	$\begin{bmatrix} 15 & 12 & 12 & 6 \\ 9 & 15 & 20 & 5 \\ 8 & 7 & 3 & 19 \end{bmatrix}$	$x^8+x^5+x^4+x^1$	x^4+x^2+1
5	2546	$\begin{bmatrix} 6 & 13 & 15 & 10 \\ 12 & 14 & 15 & 10 \\ 11 & 9 & 2 & 9 \end{bmatrix}$	$\begin{bmatrix} 14 & 4 & 3 & 12 \\ 4 & 11 & 4 & 6 \\ 11 & 17 & 6 & 14 \end{bmatrix}$	$x^8+x^5+x^3+x^2$	x^5+x^2+1
6	2793	$\begin{bmatrix} 5 & 5 & 20 & 10 \\ 2 & 1 & 12 & 15 \\ 3 & 17 & 8 & 14 \end{bmatrix}$	$\begin{bmatrix} 20 & 17 & 16 & 2 \\ 2 & 2 & 6 & 1 \\ 5 & 1 & 3 & 15 \end{bmatrix}$	$x^7+x^6+x^4+x^3$	x^5+x^3+1
7	2261	$\begin{bmatrix} 13 & 5 & 8 & 18 \\ 4 & 8 & 15 & 19 \\ 15 & 18 & 17 & 1 \end{bmatrix}$	$\begin{bmatrix} 19 & 13 & 16 & 16 \\ 10 & 13 & 5 & 19 \\ 3 & 11 & 11 & 18 \end{bmatrix}$	$x^6+x^5+x^3+x^1$	x^5+x^3+1
8	2918	$\begin{bmatrix} 4 & 12 & 11 & 16 \\ 5 & 7 & 16 & 20 \\ 8 & 6 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 8 & 6 & 8 & 19 \\ 17 & 19 & 5 & 11 \\ 7 & 16 & 1 & 3 \end{bmatrix}$	$x^8+x^6+x^5+x^2$	x^4+x^2+1
9	2952	$\begin{bmatrix} 2 & 1 & 5 & 12 \\ 11 & 17 & 11 & 14 \\ 13 & 12 & 14 & 6 \end{bmatrix}$	$\begin{bmatrix} 19 & 11 & 11 & 18 \\ 17 & 18 & 13 & 14 \\ 13 & 19 & 11 & 4 \end{bmatrix}$	$x^7+x^6+x^4+x^2$	x^4+x^2+1
10	2124	$\begin{bmatrix} 12 & 3 & 6 & 8 \\ 4 & 17 & 7 & 6 \\ 9 & 13 & 12 & 10 \end{bmatrix}$	$\begin{bmatrix} 17 & 5 & 18 & 16 \\ 4 & 19 & 15 & 17 \\ 11 & 3 & 18 & 1 \end{bmatrix}$	$x^8+x^6+x^4+x^2$	x^4+x^2+1
11	2301	$\begin{bmatrix} 4 & 2 & 3 & 20 \\ 4 & 7 & 13 & 15 \\ 15 & 19 & 1 & 7 \end{bmatrix}$	$\begin{bmatrix} 10 & 4 & 7 & 8 \\ 10 & 13 & 4 & 2 \\ 9 & 20 & 18 & 4 \end{bmatrix}$	$x^8+x^5+x^3+x^2$	x^5+x^3+1
12	2033	$\begin{bmatrix} 5 & 6 & 12 & 13 \\ 20 & 4 & 17 & 20 \\ 5 & 12 & 14 & 17 \end{bmatrix}$	$\begin{bmatrix} 4 & 5 & 10 & 1 \\ 9 & 7 & 5 & 19 \\ 7 & 5 & 5 & 12 \end{bmatrix}$	$x^7+x^5+x^4+x^2$	x^4+x^3+1
13	2747	$\begin{bmatrix} 9 & 3 & 14 & 5 \\ 6 & 15 & 6 & 8 \\ 6 & 5 & 12 & 17 \end{bmatrix}$	$\begin{bmatrix} 16 & 9 & 13 & 12 \\ 8 & 3 & 19 & 4 \\ 7 & 7 & 1 & 19 \end{bmatrix}$	$x^7+x^6+x^3+x^1$	x^4+x^3+1

Продолжение табл. 2.1

Варианты заданий (указаны согласно номеру студента в журнале)

№	d	A	B	$a(x)$	$b(x)$
14	2780	$\begin{bmatrix} 4 & 20 & 3 & 12 \\ 13 & 12 & 1 & 19 \\ 2 & 13 & 19 & 8 \end{bmatrix}$	$\begin{bmatrix} 12 & 16 & 7 & 12 \\ 20 & 16 & 3 & 15 \\ 6 & 3 & 4 & 5 \end{bmatrix}$	$x^6+x^6+x^4+x^1$	x^4+x^3+1
15	2774	$\begin{bmatrix} 11 & 3 & 12 & 17 \\ 11 & 16 & 8 & 4 \\ 3 & 4 & 18 & 15 \end{bmatrix}$	$\begin{bmatrix} 9 & 17 & 10 & 17 \\ 16 & 11 & 5 & 5 \\ 3 & 19 & 8 & 16 \end{bmatrix}$	$x^8+x^5+x^3+x^1$	x^4+x^2+1
16	2764	$\begin{bmatrix} 1 & 13 & 2 & 16 \\ 6 & 11 & 3 & 6 \\ 3 & 5 & 16 & 14 \end{bmatrix}$	$\begin{bmatrix} 19 & 19 & 1 & 8 \\ 16 & 12 & 20 & 14 \\ 4 & 14 & 18 & 6 \end{bmatrix}$	$x^8+x^5+x^4+x^1$	x^4+x^3+1
17	2284	$\begin{bmatrix} 16 & 18 & 2 & 8 \\ 10 & 9 & 2 & 19 \\ 11 & 18 & 19 & 9 \end{bmatrix}$	$\begin{bmatrix} 9 & 10 & 16 & 5 \\ 19 & 3 & 10 & 14 \\ 20 & 20 & 4 & 3 \end{bmatrix}$	$x^7+x^6+x^4+x^2$	x^5+x^3+1
18	2596	$\begin{bmatrix} 5 & 12 & 14 & 17 \\ 8 & 15 & 5 & 19 \\ 15 & 12 & 4 & 10 \end{bmatrix}$	$\begin{bmatrix} 10 & 16 & 4 & 2 \\ 12 & 11 & 8 & 16 \\ 19 & 18 & 2 & 8 \end{bmatrix}$	$x^6+x^5+x^3+x^2$	x^5+x^2+1
19	2837	$\begin{bmatrix} 3 & 3 & 10 & 12 \\ 12 & 16 & 4 & 12 \\ 7 & 15 & 13 & 12 \end{bmatrix}$	$\begin{bmatrix} 19 & 1 & 4 & 4 \\ 13 & 5 & 18 & 12 \\ 6 & 3 & 19 & 11 \end{bmatrix}$	$x^6+x^5+x^3+x^2$	x^4+x^2+1
20	2452	$\begin{bmatrix} 2 & 5 & 14 & 7 \\ 3 & 7 & 2 & 1 \\ 6 & 20 & 7 & 12 \end{bmatrix}$	$\begin{bmatrix} 13 & 3 & 3 & 1 \\ 20 & 20 & 16 & 19 \\ 18 & 16 & 18 & 19 \end{bmatrix}$	$x^8+x^6+x^4+x^3$	x^4+x^2+1
21	2292	$\begin{bmatrix} 14 & 10 & 7 & 13 \\ 17 & 18 & 16 & 8 \\ 6 & 7 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} 10 & 5 & 14 & 18 \\ 3 & 9 & 5 & 13 \\ 17 & 10 & 3 & 9 \end{bmatrix}$	$x^7+x^6+x^5+x^2$	x^5+x^3+1
22	2766	$\begin{bmatrix} 8 & 10 & 11 & 16 \\ 1 & 9 & 18 & 15 \\ 16 & 9 & 18 & 10 \end{bmatrix}$	$\begin{bmatrix} 13 & 9 & 8 & 13 \\ 13 & 8 & 13 & 16 \\ 15 & 12 & 8 & 11 \end{bmatrix}$	$x^6+x^5+x^3+x^2$	x^5+x^3+1
23	2895	$\begin{bmatrix} 16 & 9 & 15 & 10 \\ 4 & 5 & 11 & 14 \\ 4 & 12 & 2 & 14 \end{bmatrix}$	$\begin{bmatrix} 19 & 13 & 20 & 13 \\ 13 & 7 & 17 & 4 \\ 5 & 15 & 12 & 17 \end{bmatrix}$	$x^7+x^6+x^5+x^1$	x^5+x^3+1
24	2224	$\begin{bmatrix} 11 & 14 & 4 & 17 \\ 18 & 8 & 2 & 14 \\ 19 & 10 & 15 & 11 \end{bmatrix}$	$\begin{bmatrix} 15 & 5 & 12 & 19 \\ 8 & 1 & 6 & 5 \\ 15 & 11 & 8 & 1 \end{bmatrix}$	$x^7+x^6+x^3+x^3$	x^4+x^3+1
25	2680	$\begin{bmatrix} 6 & 11 & 3 & 16 \\ 4 & 14 & 4 & 14 \\ 4 & 13 & 16 & 13 \end{bmatrix}$	$\begin{bmatrix} 7 & 12 & 16 & 8 \\ 11 & 15 & 16 & 12 \\ 5 & 10 & 2 & 20 \end{bmatrix}$	$x^7+x^5+x^3+x^2$	x^5+x^2+1
26	2328	$\begin{bmatrix} 20 & 20 & 18 & 18 \\ 14 & 11 & 9 & 6 \\ 2 & 8 & 19 & 20 \end{bmatrix}$	$\begin{bmatrix} 16 & 8 & 20 & 1 \\ 13 & 3 & 1 & 19 \\ 15 & 18 & 16 & 14 \end{bmatrix}$	$x^6+x^4+x^3+x^2$	x^4+x^3+1

Окончание табл. 2.1

Варианты заданий (указаны согласно номеру студента в журнале)

№	d	A	B	$a(x)$	$b(x)$
27	2710	$\begin{bmatrix} 2 & 15 & 3 & 2 \\ 20 & 17 & 16 & 20 \\ 16 & 7 & 19 & 4 \end{bmatrix}$	$\begin{bmatrix} 9 & 20 & 11 & 20 \\ 14 & 5 & 14 & 6 \\ 16 & 20 & 1 & 3 \end{bmatrix}$	$x^6+x^5+x^4+x^2$	x^5+x^3+1
28	2677	$\begin{bmatrix} 4 & 15 & 13 & 19 \\ 10 & 14 & 17 & 14 \\ 18 & 14 & 2 & 11 \end{bmatrix}$	$\begin{bmatrix} 7 & 1 & 20 & 19 \\ 12 & 3 & 16 & 17 \\ 1 & 18 & 15 & 15 \end{bmatrix}$	$x^8+x^5+x^4+x^3$	x^4+x^3+1
29	2353	$\begin{bmatrix} 1 & 6 & 10 & 12 \\ 5 & 20 & 1 & 16 \\ 20 & 16 & 13 & 10 \end{bmatrix}$	$\begin{bmatrix} 15 & 5 & 7 & 12 \\ 18 & 2 & 5 & 8 \\ 4 & 17 & 13 & 11 \end{bmatrix}$	$x^7+x^5+x^3+x^3$	x^4+x^3+1
30	2336	$\begin{bmatrix} 3 & 4 & 20 & 19 \\ 6 & 9 & 19 & 10 \\ 7 & 2 & 8 & 8 \end{bmatrix}$	$\begin{bmatrix} 5 & 8 & 5 & 20 \\ 3 & 17 & 19 & 13 \\ 10 & 10 & 13 & 16 \end{bmatrix}$	$x^8+x^5+x^3+x^3$	x^5+x^2+1

Таблица 2.2

Варианты задания (указаны согласно номеру студента в журнале)

№	a_1	a_2	ω_1	ω_2	x_1	x_2	№	a_1	a_2	ω_1	ω_2	x_1	x_2
1	0,5	0,6	2	3	1530	1570	2	0,25	0,7	3	3	1500	1540
3	0,15	0,8	4	4	1510	1550	4	0,2	0,5	5	4	1520	1560
5	0,3	0,4	6	5	1530	1575	6	0,4	0,3	2	5	1540	1580
7	0,6	0,2	3	2	1550	1590	8	0,7	0,25	4	2	1560	1600
9	0,8	0,15	5	3	1570	1610	10	0,2	0,7	6	3	1490	1530
11	0,3	0,8	7	4	1505	1545	12	0,4	0,6	3	4	1515	1555
13	0,5	0,5	4	5	1525	1565	14	0,33	0,4	5	5	1535	1575
15	0,6	0,3	6	3	1545	1585	16	0,3	0,2	2	3	1555	1595
17	0,25	0,4	3	4	1565	1605	18	0,15	0,5	4	4	1575	1615
19	0,35	0,6	5	2	1485	1625	20	0,45	0,7	6	2	1495	1635
21	0,55	0,8	7	3	1500	1545	22	0,5	0,45	4	3	1510	1555
23	0,3	0,15	5	4	1520	1565	24	0,6	0,25	2	4	1530	1575
25	0,7	0,35	3	5	1540	1585	26	0,2	0,45	4	5	1550	1595
27	0,4	0,55	5	2	1560	1605	28	0,3	0,65	6	2	1570	1615
29	0,2	0,5	2	3	1535	1580	30	0,25	0,6	4	3	1565	1615

Контрольные вопросы

1. Что такое GNU/Octave?
2. Как задаются матрицы в Octave?
3. Как задаются полиномы в Octave?

Лабораторная работа 2
Исследование моделей каналов передачи данных

Цель работы

Ознакомиться с общими принципами проведения анализа по методу Монте-Карло на примере тестирования модели канала ДСК и модели несимметричного Z-канала.

Порядок выполнения задания

1. Протестировать встроенную в Octave модель канала ДСК (разд. 1.10.1) по методу Монте-Карло. Для этого построить графики зависимости экспериментально полученного значения вероятности битовой ошибки в канале ДСК от размера тестовой выборки (массива данных, передаваемого через канал). Выбрать вероятность ошибки p как $(40-N)/100$, где N — номер студента по журналу.

Базовая программа представлена в листинге 2.1. Программу дополнить так, чтобы на графике присутствовали название графика (включающее номер группы и Ф. И. О. студентов), подписи осей и легенда (на английском языке или транслитом). По полученным результатам сделать выводы.

Листинг 2.1

Программа тестирования модели канала ДСК для разного размера массива данных

```
N = 20; % Номер варианта
p = (40-N)/100; % Вероятность ошибки в канале
L = [100, 500, 1e3, 5e3, 1e4, 5e4, 1e5]; % Размер массива данных
% Статистика (5 строк экспериментальных значений, мат. ожидание, СКО)
stat = zeros(7, length(L));
% Основной цикл
for i=1:1:length(L)
  d = randint(1,L(i));
  % 5 экспериментов
  for j=1:1:5
    [a stat(j, i)] = biterr(d, bsc(d, p));
  endfor
  % Вычисление среднего
  stat(6,i) = mean(stat(1:5,i));
  stat(7,i) = std(stat(1:5,i));
endfor
% Построение графиков
semilogx(L,stat(1:5,:),"b",L,stat(6,:),"r",L,stat(7,:),"g");
grid;
xlim([L(1) L(length(L))]);
title("Octave BSC");
```

2. Протестировать и сравнить встроенную в Octave модель канала ДСК (разд. 1.10.1) и модель ДСК MyBSC на основе сравнения случайного вещественного числа (разд. 1.10.4), построив графики зависимости экспериментально полученного значения вероятности битовой ошибки в канале ДСК от заданной вероятности (для каждой из моделей). В качестве контрольных точек взять следующие значения вероятности p: `[1e-4 5e-4 1e-3 5e-3 1e-2 5e-2 1e-1]`. Для определения экспериментального значения использовать метод Монте-Карло с усреднением по пяти последовательным экспериментам. В каждом эксперименте передавать через канал 10^5 бит. Базовая программа представлена в листинге 2.2. Программу дополнить так, чтобы на всех графиках присутствовали название графика (включающее номер группы и Ф. И. О. студентов), подписи осей и легенда (на английском языке или транслитом). По полученным результатам сделать выводы.

Листинг 2.2

Программа тестирования моделей каналов ДСК для разных вероятностей ошибки

```
p = [1e-4 5e-4 1e-3 5e-3 1e-2 5e-2 1e-1]; % Вероятности ошибки
d = randint(1, 1e5); % Массив данных для передачи через каналы
% Массивы статистики (5 строк экспериментальных значений и среднее)
statBSC = zeros(6, length(p));
statMyBSC = zeros(6, length(p));
% Основной цикл
for i=1:1:length(p)
  % 5 экспериментов
  for j=1:1:5
    [a statBSC(j, i)] = biterr(d, bsc(d, p(i)));
    [a statMyBSC(j, i)] = biterr(d, MyBSC(d, p(i)));
  endfor
  % Вычисление среднего
  statBSC(6,i) = mean(statBSC(1:5,i));
  statMyBSC(6,i) = mean(statMyBSC(1:5,i));
endfor
% Построение графиков
subplot(1, 2, 1);
loglog(p(:),statBSC(1:5,:),"b",p,statBSC(6,:),"r");
grid;
title("Octave BSC");
subplot(1, 2, 2);
loglog(p(:),statMyBSC(1:5,:),"b",p,statMyBSC(6,:),"r");
grid;
title("My BSC");
```

3. По аналогии протестировать модель Z-канала (листинг 1.8). Необходимо построить график зависимости результирующей вероятности ошибки на выходе Z-канала от задаваемой вероятности ошибки в канале при подаче на его вход: 1) массива нулей; 2) массива единиц; 3) массива случайных двоичных чисел. Для получения экспериментальных значений результирую-

щей вероятности ошибки использовать метод Монте-Карло с усреднением по пяти последовательным экспериментам. Размер битового массива — 10^5 бит. Для наглядности графики построить на одной координатной плоскости. По оси абсцисс использовать логарифмический масштаб, по оси ординат — линейный масштаб.

Контрольные вопросы

1. Что такое двоичный симметричный канал?
2. Что такое Z-канал?
3. В чем заключается метод Монте-Карло?

Лабораторная работа 3
Вычисления в полях Галуа

Цель работы

Получить навыки в проведении вычислений с элементами полей Галуа в системе Octave.

Порядок выполнения задания

1. Вывести полный список образующих полиномов поля Галуа степени $m = 7$.

2. Для заданного в табл. 2.3 образующего полинома вывести на экран все элементы поля (можно использовать функцию `gf_field` из разд. 1.12.3), а также вывести на экран первые 15 элементов поля в двоичном (векторном) виде, используя функции перевода чисел из десятичной системы в двоичную.

3. Написать функцию, которая преобразует элемент поля из степенной формы ε^i в его численное значение, т. е. по заданной степени i возвращает значение элемента.

4. Используя написанную функцию, вычислить значение формулы из табл. 2.4 для заданных в табл. 2.5 значений переменных. Поле и полином использовать те же, что и в предыдущих пунктах.

Варианты заданий

Таблица 2.3

Образующий полином поля Галуа $p(x)$
(выбирается согласно номеру студента в журнале)

№	Полином
1, 11, 21	$x^7+x^3+x^2+x+1$
2, 12, 22	$x^7+x^5+x^2+x+1$
3, 13, 23	$x^7+x^5+x^4+x^3+1$
4, 14, 24	$x^7+x^6+x^4+x+1$
5, 15, 25	$x^7+x^6+x^5+x^2+1$
6, 16, 26	$x^7+x^4+x^3+x^2+1$
7, 17, 27	$x^7+x^5+x^3+x+1$
8, 18, 28	$x^7+x^6+x^3+x+1$
9, 19, 29	$x^7+x^6+x^4+x^2+1$
10, 20, 30	$x^7+x^6+x^5+x^4+1$

Таблица 2.4

Формула для расчета
(выбирается согласно номеру студента в журнале)

№	Формула	№	Формула
1, 11, 21	$\frac{a+b}{c}+ad^e$	6, 16, 26	$\frac{ab}{a+c}+d^e$
2, 12, 22	$ab+\frac{b+c}{d^e}$	7, 17, 27	$(a+c)b^e+\frac{d}{c}$
3, 13, 23	$\frac{ad}{b+c}+a^e$	8, 18, 28	$(a^e+b)c+\frac{d}{a}$
4, 14, 24	$(a+b)c+\frac{d^e}{a}$	9, 19, 29	$\frac{a}{c}+(b+c^e)d$
5, 15, 25	$\frac{a^e}{b+c}+cd$	10, 20, 30	$\frac{a+d^e}{bc}+c$

Таблица 2.5

Переменные для расчета
(выбираются согласно номеру студента в журнале)

№	a	b	c	d	e	№	a	b	c	d	e
1	ε^{4}	ε^{40}	ε^{34}	ε^{90}	5	16	ε^{4}	ε^{73}	ε^{90}	ε^{75}	2
2	ε^{9}	ε^{57}	ε^{4}	ε^{115}	5	17	ε^{77}	ε^{67}	ε^{76}	ε^{57}	5
3	ε^{87}	ε^{35}	ε^{65}	ε^{70}	3	18	ε^{117}	ε^{67}	ε^{10}	ε^{90}	4
4	ε^{103}	ε^{32}	ε^{14}	ε^{114}	5	19	ε^{66}	ε^{6}	ε^{62}	ε^{121}	4
5	ε^{123}	ε^{72}	ε^{39}	ε^{92}	3	20	ε^{82}	ε^{35}	ε^{15}	ε^{56}	2
6	ε^{14}	ε^{94}	ε^{58}	ε^{97}	4	21	ε^{115}	ε^{55}	ε^{14}	ε^{18}	2
7	ε^{3}	ε^{61}	ε^{20}	ε^{10}	4	22	ε^{34}	ε^{14}	ε^{76}	ε^{73}	5
8	ε^{47}	ε^{57}	ε^{109}	ε^{52}	4	23	ε^{39}	ε^{44}	ε^{36}	ε^{96}	5
9	ε^{59}	ε^{47}	ε^{9}	ε^{14}	3	24	ε^{93}	ε^{44}	ε^{59}	ε^{88}	4
10	ε^{55}	ε^{6}	ε^{105}	ε^{8}	4	25	ε^{11}	ε^{80}	ε^{6}	ε^{118}	5
11	ε^{25}	ε^{57}	ε^{44}	ε^{36}	5	26	ε^{62}	ε^{77}	ε^{93}	ε^{38}	5
12	ε^{14}	ε^{71}	ε^{54}	ε^{59}	3	27	ε^{107}	ε^{57}	ε^{2}	ε^{3}	5
13	ε^{121}	ε^{121}	ε^{91}	ε^{117}	3	28	ε^{32}	ε^{3}	ε^{88}	ε^{101}	5
14	ε^{88}	ε^{10}	ε^{79}	ε^{49}	3	29	ε^{54}	ε^{56}	ε^{13}	ε^{100}	2
15	ε^{67}	ε^{59}	ε^{42}	ε^{38}	2	30	ε^{73}	ε^{96}	ε^{43}	ε^{57}	4

Контрольные вопросы

1. Что такое поле Галуа?
2. Как рассчитать значение элемента поля по показателю его степени?
3. Что такое левый степенной базис поля?

Лабораторная работа 4
Моделирование кода Хэмминга

Цель работы

Получить навыки в исследовании кодов Хэмминга с использованием системы компьютерной алгебры Octave.

Порядок выполнения задания

1. Вывести на экран проверочную и порождающую матрицы для кода Хэмминга $(15, 11)$ (разд. 1.13.2). В дальнейшем в лабораторной работе используется этот же код.

2. Закодировать с помощью функции `encode` (разд. 1.13.1) информационный вектор, заданный в табл. 2.6.

3. Перевести информационный вектор в десятичный вид (разд. 1.3.1) и закодировать его в десятичном режиме функции `encode`.

4. Внести в полученный двоичный кодовый вектор одиночную ошибку Err согласно табл. 2.6 и декодировать получившийся вектор функцией `decode`. Параметр Err показывает позицию ошибки в кодовом слове.

5. Сравнить по методу Монте-Карло вероятностные характеристики двух кодов Хэмминга согласно варианту (табл. 2.7). Для этого воспользоваться написанной в листинге 2.3 программой, подставив в нее необходимые параметры кодов. В результате выполнения будет получен график, который необходимо проанализировать. График и выводы должны быть представлены в отчете. Также необходимо проанализировать текст самой программы и разобраться в ее работе.

Листинг 2.3

Программа сравнения двух кодов Хэмминга
по вероятности битовой ошибки в канале ДСК

```
r1 = 3;
r2 = 4;

[H1,G1,n1,k1] = hammgen(r1);
[H2,G2,n2,k2] = hammgen(r2);
s1 = sprintf("Hamming code (%d,%d)",n1,k1);
s2 = sprintf("Hamming code (%d,%d)",n2,k2);
s3 = sprintf("Without coding");

p = [5e-4 1e-3 5e-3 1e-2 5e-2 1e-1];
stat = zeros(3,6);

msg1 = randi([0 1],1e5,k1);
msg2 = randi([0 1],1e5,k2);

menc1 = encode(msg1,n1,k1,"hamming");
menc2 = encode(msg2,n2,k2,"hamming");

for i = 1:1:6
  mrec1 = bsc(menc1,p(i));
  mdec1 = decode(mrec1,n1,k1,"hamming");
  [num stat(1,i)] = biterr(msg1,mdec1);
  mrec2 = bsc(menc2,p(i));
  mdec2 = decode(mrec2,n2,k2,"hamming");
  [num stat(2,i)] = biterr(msg2,mdec2);
  mrec3 = bsc(msg1,p(i));
  [num stat(3,i)] = biterr(msg1,mrec3);
end

mfig = figure;
loglog(p,stat(1,:),"r","LineWidth",2);
hold on;
loglog(p,stat(2,:),"b","LineWidth",2);
hold on;
loglog(p,stat(3,:),"k","LineWidth",2);
hold on;
title(sprintf("Hamming codes (%d,%d) and (%d,%d) in BSC
   channel",n1,k1,n2,k2));
xlabel("BER in BSC channel");
ylabel("Error rate after decoding");
legend(s1,s2,s3,"location","northwest");
grid on;
print(mfig,'-dpng',sprintf("ham-%d-%d_ham-%d-%d_bsc_err-rate"
   ,n1,k1,n2,k2));
```

Варианты заданий

Таблица 2.6

Информационный вектор и позиция ошибки
(выбираются согласно номеру студента в журнале)

№	Информационный вектор	*Err*	№	Информационный вектор	*Err*
1	[0 1 0 1 1 1 0 1 0 0 0]	2	16	[1 1 1 1 0 0 0 0 0 1 0]	14
2	[1 0 1 0 0 0 0 0 0 0 1]	7	17	[1 1 0 1 1 1 1 1 0 0 0]	11
3	[1 0 0 0 0 1 1 0 1 1 0]	3	18	[1 1 0 0 1 1 1 0 0 1 0]	2
4	[1 1 0 1 0 1 1 0 1 0 1]	3	19	[1 1 1 1 0 0 1 0 1 1 0]	4
5	[1 0 1 1 1 1 0 1 0 1 1]	10	20	[1 0 0 1 1 0 0 0 0 1 1]	12
6	[1 0 0 1 1 1 1 1 0 1 1]	2	21	[0 0 1 0 0 0 0 0 1 1 1]	8
7	[1 0 0 0 1 0 0 1 1 0 0]	10	22	[1 0 1 0 0 1 1 0 1 0 1]	11
8	[0 0 0 0 0 1 1 0 0 1 0]	12	23	[1 0 1 1 0 1 0 1 1 1 1]	5
9	[1 0 0 0 1 0 0 0 1 1 1]	7	24	[0 0 0 0 0 0 0 1 1 0 1]	2
10	[1 0 1 1 0 1 1 0 0 1 1]	3	25	[1 0 1 1 0 0 0 0 0 1 0]	2
11	[0 1 0 0 1 0 0 1 0 1 1]	7	26	[0 0 0 1 1 0 1 1 1 0 1]	6
12	[0 0 0 1 1 0 0 1 1 0 1]	2	27	[1 0 0 0 1 0 0 0 1 1 1]	5
13	[0 0 0 0 1 1 1 1 1 1 0]	12	28	[0 0 0 1 1 1 1 1 0 1 0]	2
14	[0 1 0 1 0 1 0 0 0 0 1]	6	29	[1 1 0 1 0 0 0 1 1 0 0]	7
15	[0 1 1 0 0 1 1 1 1 1 1]	4	30	[0 1 1 1 0 1 0 1 0 1 1]	2

Таблица 2.7

(n, k, r)-коды Хэмминга для сравнения
(выбираются согласно номеру студента в журнале)

№	Код 1	Код 2	№	Код 1	Код 2
1, 11, 21	(7, 4, 3)	(31, 26, 5)	6, 16, 26	(15, 11, 4)	(127, 120, 7)
2, 12, 22	(7, 4, 3)	(63, 57, 6)	7, 17, 27	(15, 11, 4)	(255, 247, 8)
3, 13, 23	(7, 4, 3)	(127, 120, 7)	8, 18, 28	(31, 26, 5)	(127, 120, 7)
4, 14, 24	(7, 4, 3)	(255, 247, 8)	9, 19, 29	(31, 26, 5)	(255, 247, 8)
5, 15, 25	(15, 11, 4)	(63, 57, 6)	10, 20, 30	(63, 57, 6)	(255, 247, 8)

Контрольные вопросы

1. Что такое код Хэмминга?
2. Как производится кодирование кодом Хэмминга?
3. Как производится декодирование кодом Хэмминга?

Лабораторная работа 5
Моделирование циклических кодов

Цель работы

Получить навыки в исследовании циклических кодов с использованием системы компьютерной алгебры Octave.

Порядок выполнения задания

1. Для заданного в табл. 2.8 систематического циклического (n, k)-кода определить образующий полином, а затем порождающую и проверочную матрицы (разд. 1.13.3).

2. Закодировать с помощью функции `encode` (разд. 1.13.1) информационный вектор, заданный в табл. 2.8.

3. Перевести информационный вектор в десятичный вид (разд. 1.3.1) и закодировать его в десятичном режиме функции `encode`.

4. Внести в полученный двоичный кодовый вектор одиночную ошибку Err согласно табл. 2.8 и декодировать получившийся вектор функцией `decode`. Параметр Err показывает позицию ошибки в кодовом слове.

5. Сравнить по методу Монте-Карло вероятностные характеристики двух циклических кодов согласно варианту (табл. 2.9). Для этого воспользоваться написанной в листинге 2.3 программой, откорректировав ее для работы с циклическими кодами и подставив в нее необходимые параметры кодов. В результате выполнения должен быть получен график, который необходимо проанализировать. График и выводы должны быть представлены в отчете.

Варианты заданий

Таблица 2.8

Параметры кода, информационный вектор и позиция ошибки (выбираются согласно номеру студента в журнале)

№	(n, k)-код	Информационный вектор	Err
1	(12, 7)	`[ 0 1 0 0 1 1 1 ]`	6
2	(15, 7)	`[ 0 1 0 1 1 0 1 ]`	2
3	(15, 9)	`[ 1 1 1 1 0 1 1 1 1 ]`	5
4	(15, 11)	`[ 1 1 1 0 1 0 1 1 1 1 0 ]`	12
5	(16, 11)	`[ 0 1 1 0 0 1 0 0 1 0 0 ]`	4
6	(12, 7)	`[ 0 1 0 1 0 1 1 ]`	12
7	(15, 7)	`[ 1 0 1 0 1 1 0 ]`	2
8	(15, 9)	`[ 0 0 1 0 1 1 1 0 1 ]`	9
9	(15, 11)	`[ 0 0 0 1 0 0 0 0 1 0 0 ]`	6
10	(16, 11)	`[ 1 1 1 1 1 1 0 1 1 0 0 ]`	6

Окончание табл. 2.8

Параметры кода, информационный вектор и позиция ошибки
(выбираются согласно номеру студента в журнале)

№	(n, k)-код	Информационный вектор	*Err*
11	(12, 7)	[0 1 1 1 0 0 1]	2
12	(15, 7)	[0 1 0 1 0 1 1]	10
13	(15, 9)	[1 1 0 0 0 0 0 1 1]	6
14	(15, 11)	[1 1 1 0 1 1 0 1 0 1 0]	10
15	(16, 11)	[1 0 1 0 0 0 1 1 1 1 0]	12
16	(12, 7)	[0 1 0 1 1 0 0]	11
17	(15, 7)	[1 1 1 0 0 1 0]	2
18	(15, 9)	[0 1 1 0 1 1 0 0 1]	2
19	(15, 11)	[1 1 0 1 0 1 0 0 0 0 1]	7
20	(16, 11)	[0 0 0 0 1 1 0 0 1 1 0]	6
21	(12, 7)	[1 1 1 0 1 0 1]	3
22	(15, 7)	[0 0 0 1 1 1 0]	6
23	(15, 9)	[0 0 0 1 0 1 0 1 0]	2
24	(15, 11)	[0 0 1 1 1 0 0 1 1 0 1]	5
25	(16, 11)	[1 1 0 0 1 0 0 0 1 1 1]	7
26	(12, 7)	[0 1 0 1 1 1 0]	7
27	(15, 7)	[1 1 1 1 1 1 1]	8
28	(15, 9)	[0 1 0 0 1 1 1 1 0]	8
29	(15, 11)	[1 0 0 1 1 1 1 0 0 0 1]	12
30	(16, 11)	[0 1 0 1 1 0 1 0 1 0 0]	4

Таблица 2.9

Циклические коды для сравнения
(выбираются согласно номеру студента в журнале)

№	Код 1	Код 2	№	Код 1	Код 2
1, 11, 21	(12, 7)	(15, 7)	6, 16, 26	(12, 7)	(18, 7)
2, 12, 22	(15, 7)	(18, 7)	7, 17, 27	(15, 7)	(22, 7)
3, 13, 23	(15, 9)	(17, 9)	8, 18, 28	(15, 9)	(21, 9)
4, 14, 24	(15, 11)	(21, 11)	9, 19, 29	(15, 11)	(23, 11)
5, 15, 25	(16, 11)	(21, 11)	10, 20, 30	(16, 11)	(23, 11)

Контрольные вопросы

1. Что такое циклический код?
2. Как производится кодирование циклическим кодом?
3. Как производится декодирование циклическим кодом?

Лабораторная работа 6
Моделирование кодов БЧХ

Цель работы

Получить навыки в исследовании двоичных циклических кодов Боуза–Чоудхури–Хоквингема (БЧХ) с использованием системы компьютерной алгебры Octave.

Порядок выполнения задания

1. Для заданного в табл. 2.10 систематического (n, k)-кода БЧХ определить порождающий полином, составляющие его минимальные полиномы, циклотомические классы поля Галуа, проверочную матрицу и количество гарантированно исправляемых ошибок (разд. 1.13.4).

2. Закодировать с помощью функции `encode` (разд. 1.13.1) информационный вектор, заданный в табл. 2.10. Сокращенная запись `0...0` обозначает последовательность нулей. Длина нулевой последовательности зависит от k.

3. Внести в полученный двоичный кодовый вектор двукратную ошибку Err согласно табл. 2.10 и декодировать получившийся вектор функцией `decode`. Параметр Err показывает позиции ошибочных элементов в кодовом слове.

4. Сравнить по методу Монте-Карло вероятностные характеристики двух кодов БЧХ согласно варианту (табл. 2.11). Для этого воспользоваться написанной в листинге 2.3 программой, откорректировав ее для работы с кодами БЧХ и подставив в нее необходимые параметры кодов. В результате выполнения должен быть получен график, который необходимо проанализировать. График и выводы должны быть представлены в отчете.

5. Используя программу из предыдущего пункта, подобрать два кода БЧХ, которые обеспечат исправляющую способность не хуже заданной координатами (x, y) в табл. 2.10. Выбрать из них код с наименьшей избыточностью. В качестве критерия исправляющей способности использовать вероятность ошибки декодирования на выходе декодера (координата y). Для выбираемого кода она не должна превышать значения, заданного координатой y, при определенной вероятности ошибки в канале ДСК (координата x), т. е. линия графика должна быть ниже заданной точки (x, y) на графике. Таблицу кодов БЧХ можно посмотреть командой `bchpoly` (разд. 1.13.4).

Варианты заданий

Таблица 2.10

Параметры кода, информационный вектор, позиция ошибок и требуемая исправляющая способность (выбираются согласно номеру студента в журнале)

№	(n, k)-код	Информационный вектор	Err	(x, y)
1	(63, 45)	[0...0 0 1 1 1 0 0]	24 ; 10	$(3\cdot10^{-2}, 1\cdot10^{-3})$
2	(63, 39)	[0...0 1 1 0 0 0 0]	41 ; 16	$(2\cdot10^{-2}, 2\cdot10^{-3})$
3	(63, 36)	[0...0 0 0 0 1 1 1]	22 ; 33	$(1\cdot10^{-2}, 9\cdot10^{-4})$
4	(63, 30)	[0...0 0 0 1 1 0 1]	11 ; 56	$(9\cdot10^{-3}, 1\cdot10^{-4})$
5	(63, 24)	[0...0 1 1 0 0 0 1]	7 ; 18	$(8\cdot10^{-3}, 1\cdot10^{-4})$
6	(63, 18)	[0...0 0 1 1 1 0 1]	38 ; 21	$(7\cdot10^{-3}, 7\cdot10^{-5})$
7	(63, 16)	[0...0 1 1 0 0 1 1]	19 ; 30	$(6\cdot10^{-3}, 1\cdot10^{-5})$
8	(63, 45)	[0...0 0 0 1 0 0 1]	10 ; 33	$(5\cdot10^{-3}, 1\cdot10^{-5})$
9	(63, 39)	[0...0 1 1 0 1 0 0]	9 ; 54	$(5\cdot10^{-3}, 1\cdot10^{-4})$
10	(63, 36)	[0...0 0 1 0 1 1 1]	18 ; 5	$(5\cdot10^{-3}, 5\cdot10^{-5})$
11	(63, 30)	[0...0 0 0 0 0 1 1]	20 ; 27	$(3\cdot10^{-2}, 1\cdot10^{-3})$
12	(63, 24)	[0...0 1 1 0 0 0 1]	2 ; 5	$(2\cdot10^{-2}, 2\cdot10^{-3})$
13	(63, 18)	[0...0 0 0 1 1 1 0]	40 ; 41	$(1\cdot10^{-2}, 9\cdot10^{-4})$
14	(63, 16)	[0...0 1 1 1 0 0 0]	43 ; 29	$(9\cdot10^{-3}, 1\cdot10^{-4})$
15	(63, 45)	[0...0 1 0 0 0 1 1]	42 ; 27	$(8\cdot10^{-3}, 1\cdot10^{-4})$
16	(63, 39)	[0...0 1 1 1 1 0 0]	19 ; 8	$(7\cdot10^{-3}, 7\cdot10^{-5})$
17	(63, 36)	[0...0 0 0 1 0 1 0]	15 ; 22	$(6\cdot10^{-3}, 1\cdot10^{-5})$
18	(63, 30)	[0...0 0 1 1 0 1 1]	44 ; 50	$(5\cdot10^{-3}, 1\cdot10^{-5})$
19	(63, 24)	[0...0 1 1 1 0 1 1]	26 ; 7	$(5\cdot10^{-3}, 1\cdot10^{-4})$
20	(63, 18)	[0...0 1 0 0 1 0 1]	34 ; 9	$(5\cdot10^{-3}, 5\cdot10^{-5})$
21	(63, 16)	[0...0 1 0 0 0 1 1]	49 ; 11	$(3\cdot10^{-2}, 1\cdot10^{-3})$
22	(63, 45)	[0...0 1 0 0 0 0 0]	54 ; 11	$(2\cdot10^{-2}, 2\cdot10^{-3})$
23	(63, 39)	[0...0 0 1 0 1 0 1]	46 ; 57	$(1\cdot10^{-2}, 9\cdot10^{-4})$
24	(63, 36)	[0...0 1 1 0 1 0 0]	42 ; 41	$(9\cdot10^{-3}, 1\cdot10^{-4})$
25	(63, 30)	[0...0 0 1 1 0 0 1]	19 ; 57	$(8\cdot10^{-3}, 1\cdot10^{-4})$
26	(63, 24)	[0...0 0 1 0 1 1 1]	17 ; 34	$(7\cdot10^{-3}, 7\cdot10^{-5})$
27	(63, 18)	[0...0 0 0 1 0 1 1]	6 ; 7	$(6\cdot10^{-3}, 1\cdot10^{-5})$
28	(63, 16)	[0...0 1 1 0 0 0 0]	24 ; 29	$(5\cdot10^{-3}, 1\cdot10^{-5})$
29	(63, 45)	[0...0 1 0 1 0 1 1]	42 ; 54	$(5\cdot10^{-3}, 1\cdot10^{-4})$
30	(63, 39)	[0...0 1 1 0 1 0 0]	49 ; 9	$(5\cdot10^{-3}, 5\cdot10^{-5})$

Таблица 2.11

Коды БЧХ для сравнения
(выбираются согласно номеру студента в журнале)

№	Код 1	Код 2	№	Код 1	Код 2
1, 11, 21	(31, 21)	(15, 7)	6, 16, 26	(31, 31)	(31, 16)
2, 12, 22	(15, 7)	(31, 11)	7, 17, 27	(15, 7)	(31, 16)
3, 13, 23	(15, 5)	(31, 6)	8, 18, 28	(31, 16)	(31, 6)
4, 14, 24	(15, 7)	(31, 11)	9, 19, 29	(15, 11)	(31, 21)
5, 15, 25	(31, 11)	(31, 16)	10, 20, 30	(15, 11)	(31, 11)

Контрольные вопросы

1. Что такое код БЧХ?
2. Как производится кодирование кодом БЧХ?
3. Как производится декодирование кодом БЧХ?

Лабораторная работа 7
Моделирование кодов Рида–Соломона

Цель работы

Получить навыки в исследовании недвоичных циклических кодов Рида–Соломона с использованием системы компьютерной алгебры GNU/Octave.

Порядок выполнения задания

1. Для (n, k)-кода РС $(15, 9)$ определить порождающий полином и количество исправляемых кодом ошибок (разд. 1.13.5).

2. Закодировать с помощью функции `rsenc` (разд. 1.13.5) информационный вектор, заданный в табл. 2.12.

3. Внести в полученный двоичный кодовый вектор двукратную ошибку Err согласно табл. 2.12 и декодировать получившийся вектор функцией `rsdec`. Параметр Err показывает позиции ошибочных элементов в кодовом слове. Значения ошибки принять равными 2 и 7.

4. Сравнить по методу Монте-Карло вероятностные характеристики двух кодов РС согласно варианту (табл. 2.12). Для этого воспользоваться написанной в листинге 2.4 программой, подставив в нее необходимые параметры кодов. В результате выполнения будет получен график, который необходимо проанализировать. График и выводы должны быть представлены в отчете. Также необходимо проанализировать текст самой программы и разобраться в ее работе.

Листинг 2.4

Программа сравнения двух кодов РС

```
n1 = 15;
k1 = 13;
n2 = 15;
k2 = 11;

s1 = sprintf("RS code (%d,%d)",n1,k1);
s2 = sprintf("RS code (%d,%d)",n2,k2);
s3 = sprintf("Without coding");

p = [1e-3 5e-3 1e-2 5e-2 1e-1];
stat = zeros(3,length(p));
N = 1e5;

m1 = log2(n1+1);
m2 = log2(n2+1);
msg1 = gf(randi([0 n1],N,k1),m1);
msg2 = gf(randi([0 n2],N,k2),m2);

menc1 = rsenc(msg1,n1,k1);
menc2 = rsenc(msg2,n2,k2);

for i = 1:1:length(p)
  mrec1 = reshape(bi2de(bsc(de2bi(menc1.x),p(i))),N,n1);
  mdec1 = rsdec(gf(mrec1,m1),n1,k1);
  [num stat(1,i)] = symerr(msg1.x,mdec1.x);
  mrec2 = reshape(bi2de(bsc(de2bi(menc2.x),p(i))),N,n2);
  mdec2 = rsdec(gf(mrec2,m2),n2,k2);
  [num stat(2,i)] = symerr(msg2.x,mdec2.x);
  mrec3 = reshape(bi2de(bsc(de2bi(msg1.x),p(i))),N,k1);
  [num stat(3,i)] = symerr(msg1.x,mrec3);
endfor

mfig = figure;
loglog(p,stat(1,:),"r","LineWidth",2);
hold on;
loglog(p,stat(2,:),"b","LineWidth",2);
hold on;
loglog(p,stat(3,:),"k","LineWidth",2);
hold on;
title(sprintf("RS codes (%d,%d) and (%d,%d) in BSC channel",
   n1,k1,n2,k2));
xlabel("BER in BSC channel");
ylabel("Symbol error rate after decoding");
legend(s1,s2,s3,"location","northwest");
grid on;
print(mfig,'-dpng',sprintf("rs-%d-%d_rs-%d-%d_bsc_err-rate",
   n1,k1,n2,k2));
```

Варианты заданий

Таблица 2.12

Информационный вектор, позиции ошибок и коды для сравнения (выбираются согласно номеру студента в журнале)

№	Информационный вектор	*Err*	Код 1	Код 2
1	[12 2 6 14 10 1 4 15 5]	7 ; 13	(15, 11)	(15, 9)
2	[10 3 1 4 15 13 6 13 9]	7 ; 13	(15, 11)	(15, 7)
3	[14 2 5 7 4 4 10 3 5]	6 ; 13	(15, 11)	(31, 29)
4	[3 13 5 1 7 14 4 13 6]	6 ; 9	(15, 11)	(31, 27)
5	[13 1 5 6 6 5 6 5 4]	6 ; 8	(15, 11)	(31, 25)
6	[6 14 5 6 10 1 14 14 3]	7 ; 12	(15, 11)	(31, 23)
7	[14 11 3 12 3 10 14 3 13]	3 ; 14	(15, 11)	(31, 21)
8	[5 8 6 12 14 7 7 7 2]	3 ; 9	(15, 9)	(15, 7)
9	[10 2 8 8 6 12 11 9 15]	4 ; 13	(15, 9)	(31, 29)
10	[12 8 12 4 13 14 4 11 13]	7 ; 8	(15, 9)	(31, 27)
11	[8 1 12 3 2 9 14 15 7]	5 ; 13	(15, 9)	(31, 25)
12	[6 14 12 9 14 11 4 15 2]	7 ; 13	(15, 9)	(31, 23)
13	[1 4 4 14 3 13 4 2 8]	6 ; 12	(15, 9)	(31,21)
14	[11 9 13 14 1 10 11 2 13]	2 ; 9	(15, 7)	(31, 29)
15	[2 6 12 7 7 3 2 9 13]	4 ; 8	(15, 7)	(31, 27)
16	[8 14 13 9 3 7 13 7 3]	2 ; 11	(15, 7)	(31, 25)
17	[11 9 13 11 11 7 2 9 15]	4 ; 9	(15, 7)	(31, 23)
18	[12 4 14 6 15 6 12 4 3]	5 ; 8	(15, 7)	(31, 21)
19	[13 8 11 14 10 14 13 11 9]	4 ; 14	(31, 29)	(31, 27)
20	[6 13 3 3 10 14 1 12 7]	3 ; 12	(31, 29)	(31, 25)
21	[15 11 8 14 6 2 9 6 11]	4 ; 11	(31, 29)	(31, 23)
22	[13 8 15 11 12 13 11 3 9]	6 ; 12	(31, 29)	(31, 21)
23	[12 10 3 14 13 12 11 4 2]	7 ; 10	(31, 27)	(31, 25)
24	[12 5 12 10 14 8 6 9 4]	3 ; 9	(31, 27)	(31, 23)
25	[8 4 5 15 3 3 3 3 1]	5 ; 12	(31, 27)	(31, 21)
26	[6 7 4 8 3 14 12 3 8]	5 ; 10	(31, 25)	(31, 23)
27	[7 12 14 3 15 15 13 3 11]	5 ; 11	(31, 25)	(31, 21)
28	[7 2 6 2 3 2 2 6 9]	3 ; 13	(31, 23)	(31, 21)
29	[4 2 13 15 12 6 5 12 15]	2 ; 12	(15, 9)	(15, 7)
30	[9 12 14 1 7 9 7 11 14]	3 ; 9	(15, 11)	(15, 7)

Контрольные вопросы

1. Что такое код РС?
2. Как производится кодирование кодом РС?
3. Как производится декодирование кодом РС?

Список рекомендуемых источников

1. Eaton, J. W. GNU Octave (version 6.2.0) Documentation / J. W. Eaton // GNU Octave : [сайт]. — 2021. — URL: https://octave.org/doc/v6.2.0/.

2. Documentation // Octave-Forge : [сайт]. — 2021. — URL: https://octave.sourceforge.io/docs.php.

3. Алексеев, Е. Р. Введение в Octave для инженеров и математиков / Е. Р. Алексеев, О. В. Чеснокова. — Москва : ALT Linux, 2012. — 368 с.

4. Вернер, М. Основы кодирования : учебник для вузов / М. Вернер. — Москва : Техносфера, 2006. — (Мир программирования).

5. Владимиров, С. С. Математические основы теории помехоустойчивого кодирования / С. С. Владимиров ; СПбГУТ. — Санкт-Петербург, 2016.

6. Кларк, Д. К. Кодирование и исправление ошибок в системах цифровой связи / Д. К. Кларк, Д. Б. Кейн. — Москва : Радио и связь, 1987.

7. Когновицкий, О. С. Двойственный базис и его применение в телекоммуникациях / О. С. Когновицкий. — Санкт-Петербург : Линк, 2009.

8. Когновицкий, О. С. Практика помехоустойчивого кодирования : в 2 ч. : учебное пособие. Часть 1. Системы с обнаружением ошибок и обратной связью : учебное пособие / О. С. Когновицкий, В. М. Охорзин, С. С. Владимиров ; СПбГУТ. — Санкт-Петербург, 2018.

9. Ланкастер, П. Теория матриц : пер. с англ. / П. Ланкастер. — 2-е изд. — Москва : Наука, 1982.

10. Лидл, Р. Конечные поля : в 2-х т. : пер. с англ. / Р. Лидл, Г. Нидеррайтер ; под ред. В. И. Нечаева. — Москва : Мир, 1988.

11. Морелос-Сарагоса, Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса. — Москва : Техносфера, 2005.

12. Прокис, Дж. Цифровая связь / Дж. Прокис ; под ред. Д. Д. Кловского. — Москва : Радио и связь, 2000.

13. Скляр, Б. Цифровая связь. Теоретические основы и практическое применение / Б. Скляр ; под ред. А.В. Назаренко. — Москва : Издательский дом «Вильямс», 2003.

14. Фомин, С. В. Системы счисления / С. В. Фомин. — 5-е изд. — Москва : Наука, 1987.

ПРИЛОЖЕНИЕ

Образец титульного листа

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

Кафедра сетей связи и передачи данных

Теория и практика помехоустойчивого кодирования

Лабораторная работа 1

Вариант 1

Название лабораторной работы

Группа: ИКХХ-00 «___» _____ 20___ г.

Студент(ы): Петров Василий Иванович _____

Иванов Иван Петрович _____

Преподаватель: доц. Владимиров С.С. _____
(Подпись)

Санкт-Петербург
2021

Владимиров Сергей Сергеевич

ТЕОРИЯ И ПРАКТИКА ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ. МОДЕЛИРОВАНИЕ В СИСТЕМЕ GNU/OCTAVE

Учебно-методическое пособие
по выполнению лабораторных работ

Редактор *И. И. Щенсняк*

План изданий 2021 г., п. 42

Подписано к печати 28.09.2021
Объем 4,0 печ. л. Тираж 10 экз. Заказ 1268

Редакционно-издательский отдел СПбГУТ
193232 СПб., пр. Большевиков, 22
Отпечатано в СПбГУТ