ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

С. С. Владимиров

МОДЕЛИ КАНАЛОВ ПЕРЕДАЧИ ДАННЫХ

Практикум

СПб ГУТ)))

Санкт-Петербург
2018

УДК XXXXX
ББК XXXXX
Х XX

Рецензенты
заведующий кафедрой ОПДС, профессор,
доктор технических наук *О. С. Когновицкий*

*Утверждено редакционно-издательским советом СПбГУТ
в качестве учебного пособия*

Владимиров, С. С.

Х XX Модели каналов передачи данных : практикум / С. С. Владимиров ; СПбГУТ. — СПб, 2018. — 31 с.

Учебное пособие призвано ознакомить студентов старших курсов с математическими основами теории помехоустойчивого кодирования. Представленный материал служит справочным и методическим пособием при выполнении курса практических работ по дисциплине «Модели каналов передачи данных».

Предназначено для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника».

УДК XXXXX
ББК XXXXX

Содержание

1. Ознакомление с системой численных вычислений Octave. Построение графиков в Octave (ЛР)

1.1. 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.

1.2. Рекомендуемая литература

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

2. Documentation // Octave-Forge.
URL: http://octave.sourceforge.net/docs.html

3. Е. Р. Алексеев, О. В. Чеснокова «Введение в Octave» // НОУ ИНТУИТ. URL: http://www.intuit.ru/studies/courses/3677/919/info

1.3. Теоретическая справка

Справка написана для ОС Debian Linux, использующейся в лабораториях кафедры, с учётом используемого интерфейса пользователя.

1.3.1. Запуск системы Octave в терминале

Программа Octave запускается в терминале. Для вызова терминала используется пункт главного меню «Терминал».

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».

Также 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 системы Matlab, скрипт должен иметь расширение `*.m`.

1.3.2. 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 &
```

Окно терминала после этого закрывать нельзя.

В системе Linux рабочим каталогом графической версии Octave по умолчанию является домашний каталог пользователя (на лабораторных компьютерах это каталог `/home/student`). Сменить рабочий каталог можно в «Диспетчере файлов» (обычно левый верхний угол окна программы).

Основная рабочая область окна Octave имеет три вкладки. Первая вкладка «Командное окно» предназначена для ввода команд и вывода результатов их выполнения. Также сюда выводится результат выполнения скриптов, написанных/открытых в «Редакторе».

Вторая вкладка «Редактор» предназначена для работы со скриптами. Написанный/открытый в редакторе скрипт должен располагаться в рабочем каталоге Octave (см. «Диспетчер файлов»). Для запуска скрипта на выполнение используется кнопка меню редактора «Сохранить файл и запустить» (желтая стрелка в шестеренке).

Третья вкладка предназначена для документации. На рабочих компьютерах лаборатории эта документация может отсутствовать.

1.3.3. Служебные функции очистки

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

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

1.3.4. 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)` передаётся в функцию `func1()` в качестве параметра. Поскольку точки-с-запятой в конце нет, результат вычислений будет выведен на экран.

Справку по функции Octave можно получить, введя команду

```
> help function-name
```

1.3.5. Перевод из двоичной системы в десятичную

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

Octave по умолчанию использует обратную запись двоичных чисел от старшей степени к младшей.

Пример ввода двоичного числа 110001_2.

```
> a=[1  0  0  0  1  1]
```

1.3.6. Ввод матриц и полиномов

Ввод матриц и полиномов 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
```

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).

1.3.7. Операции с матрицами и полиномами

Операции с матрицами указываются как обычно. 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. Обратную матрицу можно вычислить только для квадратной матрицы.

```
> A^(-1)
```

Поскольку полиномы складываются как вектор-строки коэффициентов, то 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. (Примеры приведены 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`.

1.3.8. Построение двумерного графика функции

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

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

где x — массив координат по оси абсцисс, а y — массив значений функции (координаты по оси ординат).

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 — синими точками, необходимо использовать функцию

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

1.3.9. Построение трехмерного графика поверхности функции

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

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

где x и y — массивы переменных, 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 использовтаь функцию

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

1.3.10. Способы задания массивов данных

Основной способ задания массива данных:

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

где b и e — начальной 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 — начальной и конечное значения, а c — количество точек между 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)
```

1.3.11. Подписи осей

Для создания подписей осей координат используются функции

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

Эти функции необходимо размещать после функции `plot`.

1.3.12. Название графика

Для вывода названия графика используется функция

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

Эту функцию необходимо размещать после функции `plot`.

1.3.13. Легенда графика

Для вывода легенды используется функция

```
legend("Legend 1","Legend 2", ..., m);
```

Параметр *m* определяет месторасположение легенды в графическом окне: 1 — в правом верхнем углу графика (значение по умолчанию); 2 — в левом верхнем углу графика; 3 — в левом нижнем углу графика; 4 — в правом нижнем углу графика.

Эту функцию необходимо размещать после функции `plot`.

1.3.14. 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`.

1.3.15. Размещение нескольких графиков в одном окне

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

```
subplot(Number of rows,Num of columns,Position)
```

где «Number of rows» и «Num of 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.

1.3.16. Ограничение графика по осям

Для ограничения графика по оси абсцисс используется функция

```
xlim([X1, X2]);
```

где $X1$ и $X2$ — нижняя и верняя границы диапазона.

Для ограничения графика по оси ординат используется функция

```
ylim([Y1, Y2]);
```

где $Y1$ и $Y2$ — нижняя и верняя границы диапазона.

Эти функции необходимо размещать после функции `plot`.

1.3.17. Вывод сетки

Для вывода сетки с заданными диапазоном и шагом используется функция

```
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.

Эту функцию необходимо размещать после функции `plot`.

1.4. Порядок выполнения задания

Задание выполняется бригадой не более чем из двух учащихся. Отчёт выполняется один на бригаду и сдается преподавателю в электронном виде в формате PDF по электронной почте.

1.4.1. Перевод чисел между системами счисления

Перевести десятичное число в двоичную систему счисления и совершить обратное преобразование. Число задано в табл. 1.1. Вариант выбирается по последним двум цифрам номера студенческого билета (зачетной книжки).

Таблица 1.1

Задание для перевода числа из десятичной системы счисления в двоичную

Предпосл. цифра номера	**Последняя цифра номера студ. билета**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**
1	2217	2944	2819	2289	2489	2843	2121	2851	2665	2983
2	2226	2525	2617	2713	2166	2675	2113	2138	2301	2318
3	2232	2262	2100	2619	2528	2572	2697	2672	2535	2724
4	2587	2883	2453	2305	2621	2525	2591	2158	2587	2657
5	2980	2987	2525	2180	2467	2415	2203	2866	2907	2617
6	2929	2240	2142	2809	2129	2412	2201	2853	2608	2542
7	2738	2380	2228	2570	2962	2489	2773	2526	2186	2345
8	2446	2464	2482	2826	2672	2930	2322	2611	2785	2101
9	2496	2676	2909	2687	2990	2474	2850	2312	2619	2999
0	2589	2541	2632	2857	2628	2652	2551	2937	2201	2550

1.4.2. Операции над матрицами

Для заданных матриц A и B осуществить следующие операции.

1. Поэлементное сложение матриц.
2. Транспонирование матрицы B.
3. Умножение матрицы A на транспонированную матрицу B. Результат должен быть записан в матрицу C.
4. Обратную матрицу для C.

Матрица A выбирается из табл. 1.2 по предпоследней цифре номера студенческого билета (зачетной книжки). Матрица B выбирается из табл. 1.3 по последней цифре номера студенческого билета (зачетной книжки).

Таблица 1.2

Матрица A. Выбирается по предпоследней цифре номера студ. билета

Цифра номера	**Матрица**	**Цифра номера**	**Матрица**
1	$\begin{bmatrix} 9 & 19 & 4 & 3 \\ 5 & 3 & 19 & 19 \\ 9 & 5 & 16 & 7 \end{bmatrix}$	**6**	$\begin{bmatrix} 10 & 6 & 19 & 16 \\ 7 & 8 & 6 & 5 \\ 1 & 8 & 9 & 9 \end{bmatrix}$
2	$\begin{bmatrix} 11 & 19 & 14 & 17 \\ 14 & 5 & 10 & 5 \\ 1 & 14 & 4 & 17 \end{bmatrix}$	**7**	$\begin{bmatrix} 4 & 1 & 12 & 5 \\ 9 & 19 & 4 & 8 \\ 12 & 4 & 20 & 4 \end{bmatrix}$
3	$\begin{bmatrix} 4 & 20 & 13 & 6 \\ 7 & 18 & 12 & 16 \\ 9 & 10 & 10 & 8 \end{bmatrix}$	**8**	$\begin{bmatrix} 17 & 13 & 20 & 14 \\ 5 & 8 & 10 & 6 \\ 7 & 4 & 13 & 14 \end{bmatrix}$
4	$\begin{bmatrix} 18 & 17 & 6 & 20 \\ 19 & 8 & 20 & 1 \\ 18 & 13 & 4 & 13 \end{bmatrix}$	**9**	$\begin{bmatrix} 6 & 12 & 13 & 15 \\ 11 & 7 & 9 & 9 \\ 9 & 16 & 11 & 9 \end{bmatrix}$
5	$\begin{bmatrix} 3 & 12 & 17 & 15 \\ 7 & 9 & 20 & 11 \\ 10 & 14 & 16 & 9 \end{bmatrix}$	**0**	$\begin{bmatrix} 15 & 6 & 16 & 11 \\ 8 & 4 & 9 & 5 \\ 8 & 9 & 12 & 17 \end{bmatrix}$

Таблица 1.3

Матрица B. Выбирается по последней цифре номера студ. билета

Цифра номера	Матрица	Цифра номера	Матрица
1	$\begin{bmatrix} 1 & 18 & 6 & 20 \\ 13 & 17 & 6 & 16 \\ 17 & 7 & 14 & 15 \end{bmatrix}$	**6**	$\begin{bmatrix} 7 & 5 & 10 & 16 \\ 16 & 9 & 1 & 16 \\ 5 & 12 & 15 & 12 \end{bmatrix}$
2	$\begin{bmatrix} 14 & 14 & 15 & 15 \\ 10 & 16 & 8 & 16 \\ 3 & 1 & 3 & 9 \end{bmatrix}$	**7**	$\begin{bmatrix} 10 & 15 & 17 & 7 \\ 2 & 19 & 4 & 20 \\ 15 & 11 & 16 & 11 \end{bmatrix}$
3	$\begin{bmatrix} 14 & 10 & 16 & 9 \\ 14 & 3 & 20 & 10 \\ 6 & 20 & 13 & 9 \end{bmatrix}$	**8**	$\begin{bmatrix} 7 & 12 & 1 & 4 \\ 2 & 12 & 18 & 5 \\ 2 & 18 & 15 & 6 \end{bmatrix}$
4	$\begin{bmatrix} 2 & 19 & 7 & 10 \\ 15 & 12 & 20 & 10 \\ 3 & 1 & 4 & 3 \end{bmatrix}$	**9**	$\begin{bmatrix} 20 & 15 & 3 & 7 \\ 19 & 12 & 7 & 16 \\ 13 & 16 & 6 & 11 \end{bmatrix}$
5	$\begin{bmatrix} 17 & 16 & 10 & 7 \\ 4 & 18 & 12 & 18 \\ 12 & 20 & 11 & 12 \end{bmatrix}$	**0**	$\begin{bmatrix} 19 & 2 & 7 & 13 \\ 12 & 2 & 18 & 8 \\ 4 & 4 & 3 & 1 \end{bmatrix}$

1.4.3. Операции над полиномами

Для заданных полиномов $a(x)$ и $b(x)$ осуществить следующие операции.

1. Сложить полиномы.
2. Перемножить полиномы.
3. Разделить полином $a(x)$ на полином $b(x)$.
4. Преобразовать полиномы $a(x)$ и $b(x)$ в двоичные полиномы.
5. Сложить двоичные полиномы.
6. Перемножить двоичные полиномы.
7. Разделить двоичный полином $a(x)$ на двоичный полином $b(x)$.

Полиномы $a(x)$ и $b(x)$ выбираются из табл. 1.4. Полином $a(x)$ по предпоследней цифре номера студенческого билета (зачетной книжки). Полином $b(x)$ по последней цифре номера студенческого билета (зачетной книжки).

Таблица 1.4

Полиномы $a(x)$ и $b(x)$

Предп. цифра	Полином $a(x)$	Посл. цифра	Полином $b(x)$
1	x^6+x^2+x+1	**1**	x^3+x^2+1
2	$x^7+x^4+x^2$	**2**	x^3+x+1
3	$x^6+x^5+x^3+1$	**3**	x^2+x+1
4	$x^7+x^3+x^2+1$	**4**	x^3+x^2
5	$x^6+x^4+x^2+x$	**5**	x^2+x

6	x^7+x^6+x+1	**6**	x^3+x
7	$x^6+x^3+x^2+1$	**7**	x^2+1
8	$x^7+x^5+x^4+x$	**8**	x^3+1
9	$x^6+x^4+x^3+1$	**9**	x^4+x^2+1
0	x^7+x^6+x+1	**0**	x^4+x+1

1.4.4. Построение графика функции

1. Запустить систему Octave в графическом режиме. Перейти на вкладку «Редактор».

2. Сохранить создаваемый сценарий в домашнем каталоге. Имя файла должно быть записано латиницей без пробелов.

3. Построить график функции

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

для параметров, заданных в табл. 1.5 и диапазона x от -10 до 10 с шагом 0,1. График построить красной сплошной линией.

4. Задать подписи осей абсцисс («x») и ординат («f(x)»). Задать название графика — номер группы, ФИО студентов, вариант, номер задания.

5. Разместить на графике надпись (метку) с формулой построенной функции.

6. 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.

7. Добавить на график легенду.

Таблица 1.5

Варианты задания (указаны согласно номеру студента в журнале)

№ вар.	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

Продолжение табл. 1.5

Варианты задания (указаны согласно номеру студента в журнале)

№ вар.	a_1	a_2	ω_1	ω_2	x_1	x_2
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

1.4.5. Построение нескольких графиков по данным из файла

1. Создать и сохранить новый сценарий.
2. Скачать с сайта файл «lb02ex.csv» с точками данных.
3. Считать содержимое файла в массив. Для этого необходимо использовать функцию

```
f = dlmread('lb02ex.csv',';',"A19500:B21100");
```

Здесь «lb02ex.csv» — имя файла с данными, «;» — разделитель колонок данных, «A19500:B21100» — диапазон данных, считываемых из файла, где буквами обозначаются столбцы, а цифрами — строки. При этом формируется массив данных f соответствующего размера.

4. Построить два графика один над другим.
5. В качестве первого (верхнего) графика взять весь считанный из файла диапазон. Воспользоваться функцией

```
plot(f(:,1),f(:,2))
```

6. Ниже изобразить график, ограниченный диапазоном x_1–x_2 (табл. 1.5).
7. На графике 2 вывести сетку с шагом 5 по оси абсцисс и шагом по оси ординат на выбор студента.
8. Для каждого графика задать подписи осей, название и легенду.

1.4.6. Построение трехмерного графика поверхности функции

1. Создать и сохранить новый сценарий.
2. 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. 1.5 и диапазона x от -2 до 2 с шагом 0,05, а y от 0 до 4 с шагом 0,05. Для вычисления квадратного корня используется функция

```
sqrt(x);
```

Для возведения в степень необходимо использовать оператор поэлементного возведения в степень

```
.^
```

3. Задать подписи осей и название.

1.5. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных 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.

2. Исследование канала ДСК по методу Монте-Карло в системе Octave

2.1. Цель работы

Ознакомиться с общими принципами проведения анализа по методу Монте-Карло на примере проверки правильности модели канала ДСК, реализованной в системе Octave.

2.2. Рекомендуемая литература

1. Е. Р. Алексеев, О. В. Чеснокова «Введение в Octave для инженеров и математиков» М. : ALT Linux, 2012. — 368 с.
2. Documentation // Octave-Forge.
URL: http://octave.sourceforge.net/docs.html
3. Е. Р. Алексеев, О. В. Чеснокова «Введение в Octave» // НОУ ИНТУИТ. URL: http://www.intuit.ru/studies/courses/3677/919/info

2.3. Теоретическая справка

Справка написана для ОС Debian Linux, использующейся в лабораториях кафедры, с учётом используемого интерфейса пользователя.

2.3.1. Метод Монте-Карло

Численный метод, основанный на получении большого числа реализаций случайного процесса, который формируется так, чтобы вероятностные характеристики были равны величинам решаемой задачи.

2.3.2. Модель канала ДСК

В системе Octave канал ДСК реализуется функцией

```
bsc(data, p)
```

где *data* — данные в двоичном виде, а p — вероятность битовой ошибки в канале. Например, для передачи последовательности бит $[1011101]$ через канал ДСК с вероятностью ошибки $p_0 = 0{,}01$ необходимо использовать функцию

```
bsc([1 0 1 1 1 0 1], 0.01)
```

2.3.3. Цикл с заданным числом повторений

Цикл с заданным числом повторений (цикл for) реализуется функцией

```
for i=begin:step:end
   operations;
endfor
```

где *begin* — начальное значение счетчика *i*; *step* — шаг изменения счетчика *i*; *end* — конечное значение счетчика *i*; *operations* — те функции, которые будут выполняться в цикле.

Завершать цикл `for` можно как служебным словом `endfor`, так и служебным словом `end`.

2.3.4. Условный оператор

Условный оператор (if-else) реализуется функцией

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

где *condition*1 и *condition*2 — условия соответствующих веток, а *operations*1,2,3 — функции, выполняющиеся в соответствующей ветке.

Ветки *elseif* и *else* могут отсутствовать.

Условия *condition*1 и *condition*2 можно заключать в скобки. В случае формирования сложных составных условий использование скобок обязательно для указания порядка проверки условий.

Завершать условный оператор можно как служебным словом `endif`, так и служебным словом `end`.

2.3.5. Расчет среднего значения и стандартного отклонения

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

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

Для расчета среднего (мат. ожидания) можно использовать функцию `mean(x)`, где x — одномерный массив.

Для расчета стандартного отклонения можно использовать функцию `std(x)`, где x — одномерный массив.

2.3.6. Обращение к элементам массива

Обращение к 7 строке массива *M*:

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

Обращение к 5 столбцу массива M:

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

Обращение к 2–5 элементам 6 строки массива M:

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

Обращение к 1–6 элементам 4 столбца массива M:

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

2.4. Порядок выполнения задания

Задание выполняется каждым учащимся индивидуально. По результатам работы необходимо сформировать отчет, в котором отразить цель работы, последовательность выполненных действий, в качестве которых должен фигурировать написанный сценарий Octave с поясняющими комментариями, а также результат выполнения работы — график экспериментальной вероятности ошибки в канале ДСК и график стандартного отклонения.

Отчёт формируется в электронном виде в формате PDF и отправляется на электронную почту преподавателя.

1. Выбрать вероятность ошибки p_0 как $(40-N)/100$, где N — номер студента по журналу.

2. Написать скрипт Octave, 5 раз последовательно пересылающий через канал ДСК 50000 случайных двоичных цифр, определяющий вероятность ошибки в точках $10, 50, 100, 500, 1e3, 5e3, 1e4, 5e4$, вычисляющий среднее значение по 5 экспериментам в каждой из точек и строящий графики экспериментальной вероятности ошибки в канале (всего 5 графиков (одного цвета) на одной координатной проскости) и график среднего значения (другого цвета) на той же координатной проскости. Добавить название графика, подписи осей и легенду.

3. Посчитать в каждой точке стандартное отклонение. Построить соответствующий график на отдельной координатной плоскости. Добавить название графика, подписи осей и легенду.

4. Сделать вывод о правильности работы модели.

Алгоритм работы скрипта может выглядеть следующим образом:

1. Задать массив нулей для записи в них экспериментальных значений вероятности ошибки для всех экспериментов и среднего значения по всем экспериментам. Размер массива — 8 строк, 7 столбцов. команда `zeros(8,7)`. Задать массив граничных точек `x=[1 10 50 100 500 1e3 5e3 1e4 5e4]`.

2. Создать цикл на 5 повторений со счетчиком $i1$, в нем задать переменную (счетчик ошибок) с нулевым значением, в которую будет записываться число ошибок в текущем эксперименте.

3. Создать вложенный второй цикл на 8 повторений со счетчиком $i2$, в котором будут перебираться участки массива x. Для перебора участков создать третий вложенный цикл с границами изменения счетчика от $x(i2)$ до $x(i2+1)$.

4. Внутри третьего цикла генерировать случайное двоичное число $b = randint(1)$, передавать b на вход канала ДСК с вероятностью ошибки p_0, полученный результат сравнивать с исходным и при их несовпадении наращивать счетчик ошибок.

5. После третьего цикла значение счетчика ошибки делить на общее число переданных к этому моменту бит и записывать в соответствующее поле матрицы экспериментальных значений. Результат будет соответствовать экспериментальному значению вероятности битовой ошибки в данной точке.

6. Конец циклов.

7. Посчитать по каждой строке экспериментальных значений среднее значение (записать в 6 столбец) и стандартное отклонение (записать в 7 столбец).

8. Построить графики. Масштаб по оси абсцисс — логарифмический. Для этого используется функция `semilogx()`, использование которой аналогично функции `plot()`.

Отлаживать алгоритм рекомендуется при числе повторений внешнего цикла, равным 1 или 2, а второго цикла — 4 или 5.

2.5. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных 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.

3. Моделирование канала ДСК и Z-канала в системе Octave

3.1. Цель работы

Ознакомиться с принципами построения моделей каналов в системе Octave на примере канала Гилберта–Эллиотта и провести моделирование канала ДСК и Z-канала.

3.2. Рекомендуемая литература

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

2. Documentation // Octave-Forge.
URL: http://octave.sourceforge.net/docs.html

3. Е. Р. Алексеев, О. В. Чеснокова «Введение в Octave» // НОУ ИНТУИТ. URL: http://www.intuit.ru/studies/courses/3677/919/info

3.3. Теоретическая справка

Справка написана для ОС Debian Linux, использующейся в лабораториях кафедры, с учётом используемого интерфейса пользователя.

При построении моделей каналов в системе Octave можно идти двумя путями: моделировать собственно канал, который получает на вход исходный массив данных и возвращает массив данных с уже наложенной на него ошибкой, либо писать модель ошибок, которая возвращает массив ошибок, который требуется затем поэлементно сложить по модулю 2 с массивом данных. Ниже приведены оба варианта для модели канала Гилберта–Эллиотта, основанной на двух последовательных проверках генератора случайных чисел (двух «бросках кости»).

Модель канала Гилберта–Эллиотта

```
% Gilbert-Elliott Channel Model
function [outArr,lastState]=gec(dataArr,pBG,pGB,pG,pB,initState)
  State = initState;            % Get initial state
  [Height,Width]=size(dataArr); % Find size of data array
  outArr=zeros(Height,Width);   % Initialize output array
  for Cnt1=1:1:Height           % Start row searching cycle
    for Cnt2=1:1:Width          % Start inner searching cycle
      if State == 0             % Good state
        outArr(Cnt1,Cnt2) = xor(dataArr(Cnt1,Cnt2),rand(1)<=pG);
        State = rand(1)<=pGB;
      elseif State == 1         % Bad state
        outArr(Cnt1,Cnt2) = xor(dataArr(Cnt1,Cnt2),rand(1)<=pB);
        State = rand(1)>pBG;
```

```
      else
        printf('Error: Incorrect state\n');
        break;
      endif
    end
  end
  lastState = State;    % Return last state
end
```

В качестве параметров в приведенную функцию модели канала Гилберта–Эллиотта передаются массив исходных данных, состоящий из 0 и 1, вероятностные параметры модели канала и начальное состояние канала. Модель возвращает массив данных, прошедших через канал, — с наложенной ошибкой — и конечное состояние канала.

В приведенной модели рекомендуется обратить внимание на строку

```
[Height,Width]=size(dataArr);
```

Эта функция определяет размеры исходного массива данных для последующего поэлементного перебора.

Также обратите внимание на *реализацию проверки вероятности на основе генератора случайных чисел*. Функция `rand(1)` возвращает случайное дробное число в промежутке 0–1. Соответственно, если выпадает число меньшее либо равное заданной вероятности, то проверка считается успешной. Например, если вероятность ошибки в канале ДСК равна `p0`, то проверку можно сделать выражением `rand(1)<=p0`. Это выражение сразу вернет правильный бит ошибки.

Стоит отметить, что моделирование канала ПД в виде модели собственно канала удобно для последующего его применения в имитационном моделировании СПД.

Для анализа модели канала, работа которого не зависит от потока входных данных, удобно пропускать через построенную модель массив нулевых исходных данных. В этом случае на выходе канала сразу будет получен будет массив ошибочных бит, который удобно анализировать. Сравнение с исходным массивом при этом не требуется.

Если же работа модели канала зависит от входящих данных, как например в Z-канале, то необходимо передавать через канал различные массивы исходных данных, чтобы проверить работу модели в разных условиях. В частности, при оценке канала удобно использовать в качестве исходных данных случайно сгенерированный битовый массив.

```
b = randint(n, m);
```

Эта функция возвращает битовый массив из n строк и m столбцов.

Для создания массива нулей используется функция

```
b = zeros(n, m);
```

Для создания массива единиц используется функция

```
b = ones(n, m);
```

Функцию модели канала можно реализовать в отдельном файле. Такой файл должен иметь то же имя, что и имя функции. Расширение файла — «.m». Чтобы использовать такую функцию, необходимо запустить систему Octave в том каталоге, в котором лежит эта функция.

Модель ошибок Гилберта–Эллиотта

```
% Gilbert-Elliott Error Model
function [erVek,lastState]=gem(Height,Width,pBG,pGB,pG,pB,initState)
  State = initState;            % Get initial state
  erVek=zeros([Height,Width]);  % Initialize error array
  for Cnt1=1:1:Height           % Start row searching cycle
    for Cnt2=1:1:Width          % Start inner (column) searching cycle
      if State == 0             % Good state
        erVek(Cnt1,Cnt2) = rand(1)<=pG;
        State = rand(1)<=pGB;
      elseif State == 1         % Bad state
        erVek(Cnt1,Cnt2) = rand(1)<=pB;
        State = rand(1)>pBG;
      else
        printf('Error: Incorrect state\n');
        break;
     endif
   end
  end
  lastState = State;    % Return last state
end
```

Использование модели ошибок возможно в том случае, когда работа канала не зависит от входных данных. Фактически, модель ошибок представляет собой модель канала при нулевых входных данных. Таким образом, если для таких каналов как канал ДСК или канал Гилберта–Эллиотта использование модели ошибок возможно, то для Z-канала этот способ моделирования неприменим.

3.4. Порядок выполнения задания

Задание выполняется каждым учащимся индивидуально. По результатам работы необходимо сформировать отчет, в котором отразить цель работы, последовательность выполненных действий, в качестве которых должен

фигурировать написанный сценарий Octave с поясняющими комментариями, а также результат выполнения работы — график экспериментальной вероятности ошибки в канале ДСК и график стандартного отклонения.

Отчёт формируется в электронном виде в формате PDF и отправляется на электронную почту преподавателя.

1. Использовав в качестве примера модель канала Гилберта–Эллиотта написать модель канала ДСК.

2. Сравнить результат действия модели с результатом встроенной в систему Octave модели канала ДСК, построив графики зависимости экспериментально полученного значения вероятности ошибки в канале ДСК от заданной вероятности (для каждой из моделей). В качестве контрольных точек взять следующие значения вероятности p_0: `[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 послевовательным экспериментам. В каждом эксперименте передавать через канал 10^5 бит. Упрощенно алгоритм эксперимента можно представить в виде последовательности действий

а) сформировать массив нулей размером 1 на 10^5;

б) передать его через встроенную в Octave модель канала ДСК;

в) посчитать сумму единиц в массиве, которая будет равна общему количеству ошибок; команда `sum(A)`;

г) поделить число ошибок на общее число переданных бит, получив экспериментальное значение вероятности ошибки в канале;

д) повторить эксперимент по пять раз для каждой вероятности ошибки p_0;

е) рассчитать для каждой точки среднее значение и построить графики (пять экспериментальных и среднее на одной плоскости);

ж) повторить весь эксперимент для модели, написанной самостоятельно; построить графики (пять экспериментальных и среднее на одной плоскости);

з) построить график с усредненными по пяти экспериментам значениями для обеих моделей на одной координатной плоскости (для сравнения).

На всех графиках должны присутствовать название графика, подписи осей и легенда (на английском или транслитом).

3. Сделать вывод о правильности работы модели.

4. По аналогии написать модель Z-канала.

5. Построить график зависимости результирующей вероятности ошибки на выходе Z-канала при подаче на его вход: 1) массива нулей; 2) массива единиц; 3) массива случайных двоичных чисел. Для получения экспериментальных значений результирующей вероятности ошибки использовать метод Монте-Карло с усреднением по пяти последовательным экспериментам. Размер битового массива — 10^5 бит.

6. Сделать выводы по результатам.

3.5. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных 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.

4. Работа с кодом Хэмминга в системе Octave (ЛР)

4.1. Цель работы

Рассмотреть на примере и получить навыки в исследовании кодов Хэмминга с использованием системы компьютерной алгебры Octave.

4.2. Рекомендуемая литература

1.
2.
3.

4.3. Порядок выполнения задания

Задание выполняется каждым учащимся индивидуально. Результаты заданий лабораторного практикума и соответствующих заданий практикума должны совпадать.

Отчёт формируется в электронном виде в формате PDF и отправляется на электронную почту преподавателя.

4.3.1.

Для (n,k) кода Хэмминга $(15,11)$ получить проверочную матрицу и порождающую матрицу. В системе Octave для этого используется функция `hammgen`, которая получает на вход число проверочных бит $r = n - k$, и вычисляет проверочную и порождающую матрицы, а также выводит n и k.

```
> [H,G,n,k]=hammgen(r)
```

4.3.2.

Закодировать заданный информационный вектор вначале встроенной функцией Octave, затем при помощи умножения на порождающую матрицу. Сравнить результаты. Информационный вектор берется из табл. 4.1 по предпоследней цифре зачетной книжки.

Таблица 4.1

Информационный вектор. По предпоследней цифре номера зачетной книжки

Цифра	Вектор	Цифра	Вектор
1	1 0 0 1 1 1 0 0 0 1 0	6	1 1 1 0 1 1 0 1 0 0 1
2	0 1 1 1 0 1 0 0 0 1 0	7	0 1 0 0 0 1 0 0 0 0 1
3	1 0 0 1 0 0 1 1 1 0 0	8	1 0 1 1 0 0 0 0 0 1 1
4	1 0 0 1 0 0 0 0 0 1 0	9	0 1 0 1 1 0 0 1 0 0 0
5	1 0 0 0 1 1 1 0 1 0 0	0	0 1 0 1 0 0 1 1 1 0 0

Для кодирования используется функция `encode`. В качестве параметров задаются исходное сообщение в двоичном виде, параметры кода n и k и указание использовать код Хэмминга.

```
> Menc=encode(Msg,n,k,"hamming")'
```

Оператор транспонирования «'» указывается, чтобы выводить сообщение строкой, а не столбцом.

Для умножения на порождающую матрицу предварительно необходимо задать информационный вектор и саму матрицу как структуры над простым полем Галуа GF(2).

```
> G2=gf(G,1,3);
> Msg2=gf(Msg,1,3);
```

Далее можно умножать обычным способом.

4.3.3.

Последовательно наложить заданные векторы ошибки на кодовый вектор и декодировать полученные векторы с ошибкой вначале при помощи встроенной функции Octave, затем посредством проверочной матрицы H по стандартному алгоритму для кодов Хэмминга. Сравнить результаты. Векторы ошибки берутся из табл. 4.2 по последней цифре зачетной книжки. Заданы векторы с одной, двумя и тремя ошибками.

Таблица 4.2

Вектор ошибки. По последней цифре номера зачетной книжки

Цифра	Вектор	Цифра	Вектор
1	0100000000000000	6	0000001000000000
	0100100000000000		1000001000000000
	0100100100000000		1000001000100000
2	0010000000000000	7	0000000100000000
	0010100000000000		0100000100000000
	0010100100000000		0100000100100000
3	0001000000000000	8	0000000001000000
	0001010000000000		0010000001000000
	0001010100000000		0010000010100000
4	0000100000000000	9	0000000000100000
	0000101000000000		1000000000100000
	0000101010000000		1000010001000000
5	0000010000000000	0	0000000000010000
	0000010100000000		0001000000010000
	0000010101000000		0001001000010000

Для наложения ошибки используется функция `xor`.

```
> Merr1=xor(Menc,Err1)
```

Для декодирования используется функция `decode`. В качестве параметров задаются исходное сообщение в двоичном виде, параметры кода n и k и указание использовать код Хэмминга. На выходе функция возвращает декодированное сообщение `Mdec` и вектор ошибок `err`.

```
> Mdec=decode(Merr1,n,k,"hamming")'
```

Для декодирования по стандартному алгоритму для кодов Хэмминга необходимо произвести умножение на транспонированную проверочную матрицу H. Предварительно необходимо задать вектор с ошибкой и саму матрицу как структуры над простым полем Галуа GF(2).

```
> H2=gf(H,1,3);
> Merr21=gf(Merr1,1,3);
```

Далее можно умножать обычным способом.

4.3.4.

Сравнить по методу Монте-Карло вероятностные характеристики двух кодов Хэмминга согласно варианту. Для этого воспользоваться написанной в листинге 4.1 программой, изменив ее для своих нужд, подставив 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 4.1

Листинг программы для сравнения двух кодов Хэмминга по вероятности битовой ошибки в канале ДСК

```
1  r1=3;
2  r2=4;
3  %
4  [H1,G1,n1,k1]=hammgen(r1);
5  [H2,G2,n2,k2]=hammgen(r2);
6  s1=sprintf("Hamming code (%d,%d)",n1,k1);
7  s2=sprintf("Hamming code (%d,%d)",n2,k2);
8  %
9  p0=[5e-4 1e-3 5e-3 1e-2 5e-2 1e-1];
10 stat=zeros(2,6);
11 %
12 msg1=randi([0 1],1e5,k1);
13 msg2=randi([0 1],1e5,k2);
14 %
15 menc1=encode(msg1,n1,k1,"hamming");
16 menc2=encode(msg2,n2,k2,"hamming");
17 %
```

```
for i=1:1:6
  mrec1=bsc(menc1,p0(i));
  mdec1=decode(mrec1,n1,k1,"hamming");
  [num,rate]=biterr(msg1,mdec1);
  stat(1,i)=rate;
  mrec2=bsc(menc2,p0(i));
  mdec2=decode(mrec2,n2,k2,"hamming");
  [num,rate]=biterr(msg2,mdec2);
  stat(2,i)=rate;
end
%
format long;
%
stat
%
mfig=figure;
L1=loglog(p0,stat(1,:));
set(L1,"LineWidth",3,"Color","k");
hold on;
L2=loglog(p0,stat(2,:));
set(L2,"LineWidth",3,"Color","b");
hold on;
title(sprintf("Hamming codes (%d,%d) and (%d,%d) in BSC
   channel",n1,k1,n2,k2));
xlabel("BER in BSC channel, p0");
ylabel("Error rate after decoding");
legend(s1,s2,3);
legend("show");
grid on;
print(mfig,'-dpng',sprintf("ham-%d-%d_ham-%d-%d_bsc_err-rate"
   ,n1,k1,n2,k2));
```

Программу необходимо сохранить в виде файла с расширением `*.m` и запускать из командной строки (из каталога, в котором лежит программа) так, как указанно ниже.

```
user@name:[~]$ octave -q hamming_compar.m
```

Параметры кодов указаны в табл. 4.3. Выбор производится по последней цифре номера зачетной книжки.

Таблица 4.3

Параметры кодов Хэмминга для сравнения.
(По последней цифре номера зачетной книжки)

Цифра	**Код 1** (n,k,r)	**Код 2** (n,k,r)	**Цифра**	**Код 1** (n,k,r)	**Код 2** (n,k,r)
1	(7,4,3)	(31,26,5)	6	(15,11,4)	(127,120,7)
2	(7,4,3)	(63,57,6)	7	(15,11,4)	(255,247,8)
3	(7,4,3)	(127,120,7)	8	(31,26,5)	(127,120,7)

Продолжение табл. 4.3

Параметры кодов Хэмминга для сравнения. По последней цифре номера зачетной книжки

Цифра	**Код 1** (n,k,r)	**Код 2** (n,k,r)	**Цифра**	**Код 1** (n,k,r)	**Код 2** (n,k,r)
4	(7,4,3)	(255,247,8)	9	(31,26,5)	(255,247,8)
5	(15,11,4)	(63,57,6)	0	(63,57,6)	(255,247,8)

4.4. Пример выполнения работы для кода (7,4) (только основные команды)

```
> [H,G,n,k]= hammgen(3)
> Msg=[1 0 1 0]
> Menc=encode(Msg,n,k,"hamming")'
> G2=gf(G,1,3);
> Msg2=gf(Msg,1,3);
> Menc2=Msg2*G2
> Err1=[0 1 0 0 0 0 0]
> Merr1=xor(Menc,Err1)
> Mdec1=decode(Merr1,n,k,"hamming")'
> H2=gf(H,1,3);
> Merr21=gf(Merr1,1,3);
> S1=Merr21*H2'
> Err2=[0 1 0 1 0 0 0]
...
> Err3=[0 1 0 1 0 1 0]
...
> exit
```

4.5. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных 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.

Владимиров Сергей Сергеевич

МОДЕЛИ КАНАЛОВ ПЕРЕДАЧИ ДАННЫХ

Практикум

Издано в авторской редакции

План изданий 201X–201X гг., доп. п. XXX

Подписано к печати XX.XX.XXXX
Объем X,XX усл.-печ. л. Тираж XXX экз. Заказ XXX

Редакционно-издательский отдел СПбГУТ
191186 СПб., наб. р. Мойки, 61
Отпечатано в СПбГУТ