

Семинар №2 по курсу «Основы цифровой обработки сигналов»

Генерация звуковых сигналов на ПК

Кузнецов В.В., ассистент кафедры ЭИУ1-КФ

8 ноября 2013 г.

1 Цель работы

Целью семинара является генерация звуковых сигналов с использованием звуковой карты персонального компьютера и системы численной математики GNU/Octave.

Система GNU/Octave — это высокоуровневый язык программирования, предназначенный прежде всего для численных расчётов. Он предоставляет удобный интерфейс командной строки для численного решения линейных и нелинейных задач, а также для выполнения других численных экспериментов. С помощью GNU/Octave можно решать задачи в том числе генерации и обработки сигналов. Установить GNU/Octave для Linux можно в один клик через пакетный менеджер, а для Windows её можно бесплатно скачать с сайта разработчика <http://octave.sourceforge.net>.

Octave работает в режиме командной строки. Octave позволяет выполнять операции с действительными и комплексными числами, матрицами, решать системы линейных уравнений. Синтаксис команд Octave близок к языку C и повторяет среду Matlab.

2 Введение в Octave

В состав пакета входит интерактивный командный интерфейс (интерпретатор). Интерпретатор Octave запускается из терминала ОС Linux или из его порта в Windows. После запуска Octave пользователь видит окно интерпретатора (см. рис. 1). Чтобы запустить GNU/Octave нужно для Linux набрать в терминале `octave`, а для Windows щёлкнуть по ярлыку, создаваемому на рабочем столе после установки программы.

Чтобы выйти из программы в командной строке нужно набрать `quit`. После того как команда напечатана, нужно нажать Enter.

В окне интерпретатора пользователь может вводить как отдельные команды языка Octave, так и группы команд, объединяемые в программы. Если строка заканчивается символом `;` (точка с запятой) результаты на экран не выводятся. Если же в конце строки символ отсутствует, результаты работы выводятся на экран (см. рис. 1.2). Текст в строке после символа `%` (процент) или `#` (решётка) является комментарием и интерпретатором не обрабатывается. Комментарии можно вводить и в командном режиме. Рассмотрим несколько несложных примеров.

Рассчитаем резонансную частоту LC - контура по формуле:

$$f = \frac{1}{2\pi\sqrt{LC}} \quad (1)$$

```
GNU Octave, version 3.6.4
Copyright (C) 2013 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-suse-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type 'news'.

octave:1>
```

Рис. 1: Командная строка Octave

Где L — индуктивность катушки, Гн;

C — ёмкость конденсатора, Ф;

f — частота, Гц

Возможны два варианта решения любой задачи в Octave:

1. Терминальный режим. В этом режиме в окно интерпретатора последовательно вводятся отдельные команды.

2. Программный режим. В этом режиме создаётся текстовый файл с расширением `.m`, в котором хранятся последовательно выполняемые команды Octave. Затем этот текстовый файл (программа на языке Octave) запускается на выполнение в среде Octave.

Чтобы решить нашу задачу в терминальном режиме введём последовательно после того как Octave запустится и выдаст приглашение, подобное рис.1, команды, показанные на рис. 2. На рисунке приведён вывод для Linux, отличие для системы Windows заключается только в способе запуска программы.

```

vvk@linux-bmx0:~> octave
GNU Octave, version 3.6.4
Copyright (C) 2013 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-suse-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type 'news'.

octave:1> L=10e-6;
octave:2> C=47e-12;
octave:3> f=1/(2*pi*sqrt(L*C))
f = 7.3413e+06
octave:4> quit
vvk@linux-bmx0:~>

```

Рис. 2: Расчёт резонансной частоты LC-контура в командной строке Octave

В переменной `ans` хранится результат последней операции, если команда не содержит знака присваивания. Следует помнить, что значение переменной `ans` изменяется после каждого вызова команды без операции присваивания.

Теперь рассмотрим, как решить эту же задачу в программном режиме. Вызовем любой текстовый редактор, например `kwrite` в системе Linux, но подойдёт даже обычный Notepad для Windows, в окне которого последовательно введём следующие команды, приведённые в листинге 1.

Листинг 1: Файл LC-kontur.m

```

1 #!/usr/bin/octave -qf # Первая строка --- тип интерпретатора
2 L=10e-6; # Индуктивность катушки, Гн
3 C=47e-12; # Ёмкость конденсатора, Ф
4 f=1/(2*pi*sqrt(L*C)) # Считаем резонансную частоту и
5 # выводим её на экран
6 # pause # Раскомментировать строку для Windows. Задерживает выполнение
   скрипта
7 # до тех пор пока пользователь не нажмёт Enter.

```

Сохраним введённые команды в виде файла `LC-kontur.m`. Скрипты Octave обычно имеют расширение `*.m`. Теперь эту программу необходимо запустить на выполнение. В системе Linux нужно сначала сделать файл исполняемым любым способом, например из командной строки или файлового менеджера. Чтобы дать скрипту права на исполнение и затем запустить его нужно выполнить в каталоге, где находится файл скрипта следующие команды:

```
chmod a+x LC-kontur.m
./LC-kontur
```

В результате выполнения команд в окне терминала напечатается вычисленное значение резонансной частоты $f = 7.3413e+06$.

При обработке сигналов нам понадобится выполнять операции с одномерными массивами чисел с плавающей точкой (float). Одномерные массивы в Octave являются частным случаем матриц. Им соответствуют матрица-строка и матрица-столбец (вектор). Двухмерные матрицы — это двухмерные массивы. Индексация массивов начинается с единицы, в отличие от языка C.

Чтобы объявить массив нужно перечислить все его элементы через запятую и заключить их в квадратные скобки. Ниже объявлен массив X из трёх элементов и выполнено обращение к его второму элементу:

```
octave:1> X=[1,2,3]
X =
```

```
    1    2    3
```

```
octave:2> X(2)
ans = 2
```

Чтобы сгенерировать массив, содержащий последовательность чисел с некоторым шагом нужно воспользоваться операцией `:`. Например:

```
octave:3> X=0:1:10
X =
```

```
    0    1    2    3    4    5    6    7    8    9   10
```

```
octave:4> X(6)
ans = 5
```

Мы создали массив из целых чисел от 0 до 10 с шагом 1 и обратились к его шестому элементу.

Для построения графиков в Octave служит функция `plot(x,y)`. В качестве аргументов ей передаются два вектора (матрица-столбец, либо матрица-строка), содержащие точки по осям X и Y . Количество элементов в векторах должно быть одинаковым.

К матрицам и массивам в Octave можно применять все операции, известные из курса линейной алгебры: транспонирование, умножение на число, сложение, вычитание, умножение матриц, вычисление обратной матрицы.

Здесь нужно упомянуть про операцию транспонирования `'` (одинарная кавычка). Поместив знак `'` после матрицы мы заменяем строки на столбцы. Так можно получить из матрицы-строки матрицу-столбец.

Матрицы в Octave можно складывать так же как и обычные числа при помощи операции `+`. При этом первый элемент складывается с первым, второй со вторым и т.д. Этим пользуются для сложения сигналов.

Для примера построим график функции $\sin(t)$ на отрезке от 0 до 2π по 20 точкам. Команды, вводимые в окно Octave показаны на рис.3.

Если ввести в Octave указанные команды, то откроется графическое окно, в котором будет построена синусоида.

Как видно из простейших примеров, у Octave достаточно широкие возможности, а по синтаксису он близок к Matlab.

```

GNU Octave, version 3.6.4
octave:1> t=0:(2*pi/20):2*pi # генерируем вектор из 20 точек от 0 до 2*pi
t =

Columns 1 through 9:

    0.00000    0.31416    0.62832    0.94248    1.25664    1.57080    1.88496
2.19911    2.51327

Columns 10 through 18:

    2.82743    3.14159    3.45575    3.76991    4.08407    4.39823    4.71239
5.02655    5.34071

Columns 19 through 21:

    5.65487    5.96903    6.28319

octave:2> s=sin(t) # вычисляем таблицу функции sin()
s =

Columns 1 through 10:

0.00000    0.30902    0.58779    0.80902    0.95106    1.00000    0.95106    0.80902
0.58779    0.30902

Columns 11 through 20:

0.00000   -0.30902   -0.58779   -0.80902   -0.95106   -1.00000   -0.95106   -0.80902
0.58779   -0.30902

Column 21:

-0.00000

octave:3> plot(t,s); # строим график
octave:4>

```

Рис. 3: Пример построения графика функции в GNU/Octave

3 Способы работы с цифровыми сигналами в персональных компьютерах

Цифровые сигналы представляют собой поток двоичных чисел с некоторой разрядностью, поэтому в памяти ЭВМ сигналы хранятся в виде массивов. Отправив массив 16-битных чисел последовательно элемент за элементом, например на звуковую карту компьютера, мы получим на выходе звуковой карты звуковой сигнал. Такой массив чисел можно получить любым образом: вычислить поэлементно, загрузить из файла на диске и т.п. Так, например, wav-файлы состоят из заголовка, содержащего служебную

информацию, и следующей за ним последовательности чисел с разрядностью равной разрядности ЦАП звуковой карты.

В ОС Linux звуковая карта представляется файлом устройства `/dev/dsp`. Обнаружив звуковую карту, ядро ОС Linux создаёт файл `/dev/dsp`, записав в который любым способом поток байт, мы получим воспроизведение звука. И на звуковую карту может быть выведен абсолютно любой двоичный или текстовый файл. Например, можно вывести с помощью команды

```
cat /dev/random>/dev/dsp
```

на звуковую карту поток псевдослучайных чисел. В результате в динамиках ПК будет слышен белый шум.

Поток чисел который будет отправлен на звуковую карту можно получить различными способами с использованием различного ПО.

Аналогично и сигнал, считанный с микрофонного входа звуковой карты представляется массивом двоичных чисел.

Для примера рассмотрим программную генерацию звуковых сигналов на персональном компьютере. Для решения поставленной задачи будем использовать или среду численных вычислений GNU/Octave (распространяемый бесплатно аналог MATLAB, работает в ОС Linux и Windows), либо программирование на языке C/C++. В последнем случае алгоритм будет иметь наибольшую производительность.

Сначала рассмотрим принцип построения звукового сигнала в системе GNU/Octave. Код скрипта для генерации синусоидального сигнала в системе Octave приведён в листинге 2.

В результате исполнения скрипта генерируется звуковой сигнал частотой 500 Гц и длительностью 1 секунда.

Листинг 2: Скрипт для генерации синусоидального сигнала в системе Octave

```
1 #!/usr/bin/octave -qf
2 pkg load audio # подгружаем библиотеку для работы со звуком
3 dur = 1.0; # длительность сигнала 1 секунда
4 fs = 16000; # частота дискретизации 16 кГц=16000 Гц
5 t = 0 : (1/fs) : dur; # генерируем вектор (матрицу-строку) содержащий
6 # отсчёты времени от нуля до 1 секунды через интервал дискретизации
7 # T=(1/fs)=(1/16000) секунды
8 t=t'; # трансформируем матрицу-строку в матрицу-столбец, т.к функции
9 # генерации звука работают только с матрицей-столбцом
10 s = sin( 2*pi*500*t ); # вычисляем 16000 отсчётов синусоидального
11 # сигнала с частотой 500 Гц
12 plot(t,s); # строим график синусоиды
13 axis([0,0.01]'); # на отрезке от 0 до 0.01 секунды
14 wavwrite(s,fs,"sinus.wav"); # записываем сгенерированный сигнал в
15 #wav-файл, который затем можно проиграть в любом аудиопроигрывателе.
16 sound(s,16000); # отправляем сгенерированный звук на звуковую карту.
17 # В динамиках компьютера слышен тон частотой 500 Гц
18 # и длительностью 1 секунда
19 pause;
```

Теперь рассмотрим способ генерации синусоидального сигнала с применением языка C. Здесь подход отличается только тем, что добавляется код для инициализации системы вывода звука. Так же, как в скрипте на Octave нужно сначала подготовить массив,

содержащий отсчёты синусоидального сигнала, а потом отправить его на звуковую карту. Обобщённый алгоритм программ для генерации звука можно представить так (см. листинг 3):

Листинг 3: Алгоритм для генерации звуковых сигналов для программ на языке С

```
1 unsigned signal[length]; // объявляем массив содержащий отсчёты
   сигнала
2 generate_signal(signal); // заполняем массив отсчётами сигнала,
   которые
3                               // генерирует функция generate_signal
4 init_audio_output_device(); // инициализируем звуковую карту
5 send_signal_to_soundcard(); // отправляем сигнал на звуковую карту
```

Данный алгоритм верен для всех ОС и сред разработки и будет отличаться только техническими подробностями реализации инициализации звуковой карты и отправки на неё отсчётов сигнала.

Ниже приведён пример (листинг 4) практической реализации программы для системы Linux, генерирующей синусоидальный звуковой сигнал частотой 1000 Гц.

Программа компилируется при помощи компилятора gcc следующей командой (пусть текст программы сохранён в файл `sinusoid.c`):

```
gcc sinusoid.c -Wall -lm -o sinusoid
```

На современных версиях Linux перед запуском программы необходимо от администратора (root) подгрузить модуль совместимости звуковой подсистемы следующей командой:

```
modprobe snd-pcm-oss
```

Листинг 4: Программа на С для генерации тонального сигнала частотой 1000 Гц.

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <fcntl.h>
5 #include <sys/ioctl.h>
6 #include <sys/soundcard.h>
7 #include <math.h>
8
9 #define SIZE (48)
10 #define PI (4*atan(1))
11
12 int fd_out;
13 const int freq = 1000; // частота сигнала, Гц
14 int sample_rate = 48000; // частота дискретизации 48 кГц
15 short buf[SIZE]; // буфер для хранения отсчётов сигнала - длина
   буфера - один
16 период синусоиды (48000 Гц)/(1000 Гц) = 48 отсчётов
17
18 static void generate_sinewave (void) // заполняем буфер отсчётами
   // синусоидального сигнала
19 {
20
21     int i;
22     for (i=0;i<SIZE;i++) buf[i] = (short
```

```

23 int)lround(32767*sin(2*PI*i*freq/sample_rate));
24 }
25
26 void write_sinewave(void) // посылаем сгенерированный звуковой сигнал
27 // на звуковую карту
28 {
29     if (write (fd_out, buf, sizeof (buf)) != sizeof (buf))
30     {
31         perror ("Audio write");
32         exit (-1);
33     }
34 }
35
36
37 static int open_audio_device (char *name, int mode) //
38     инициализировать
39 // звуковую карту
40 {
41     int tmp, fd;
42
43     if ((fd = open (name, mode, 0)) == -1)
44     {
45         perror (name);
46         exit (-1);
47     }
48
49     tmp = AFMT_S16_NE; // установить разрядность 16 бит
50     if (ioctl (fd, SNDCTL_DSP_SETFMT, &tmp) == -1)
51     {
52         perror ("SNDCTL_DSP_SETFMT");
53         exit (-1);
54     }
55
56     if (tmp != AFMT_S16_NE)
57     {
58         fprintf (stderr,
59             "The device doesn't support the 16 bit sample format.\n");
60         exit (-1);
61     }
62
63     tmp = 1; // установить моно режим
64     if (ioctl (fd, SNDCTL_DSP_CHANNELS, &tmp) == -1)
65     {
66         perror ("SNDCTL_DSP_CHANNELS");
67         exit (-1);
68     }
69
70     if (tmp != 1)
71     {
72         fprintf (stderr, "The device doesn't support mono mode.\n");
73         exit (-1);
74     }

```



```

75
76 // установить частоту дискретизации 48 кГц.
77 if (ioctl (fd, SNDCTL_DSP_SPEED, &sample_rate) == -1)
78 {
79     perror ("SNDCTL_DSP_SPEED");
80     exit (-1);
81 }
82
83 return fd;
84 }
85
86 int main (int argc, char *argv[])
87 {
88
89     char *name_out = "/dev/dsp"; // имя устройства звуковой карты
90
91     if (argc > 1) name_out = argv[1];
92
93     fd_out = open_audio_device (name_out, O_WRONLY);
94
95     generate_sinewave(); // готовим синусоидальный сигнал
96
97     while (1) write_sinewave (); // посылаем в бесконечном цикле период
        синусоиды
98 на звуковую карту
99
100     exit (0);
101 }

```

Программа отличается от скрипта для GNU/Octave главным образом тем, что здесь добавляются аппаратные подробности инициализации системы вывода звука. В остальном принцип формирования сигналов одинаков. Формируется массив, содержащий отсчёты синусоидального сигнала и затем он отправляется на звуковое устройство при помощи системного вызова `write()`.

4 Задание для самостоятельной работы

С помощью скрипта из листинга 2 нужно сгенерировать в GNU/Octave звуковой синусоидальный сигнал. Частота сигнала вычисляется по нижеприведённым формулам в зависимости от варианта.

Для группы РПД-91:

$$f = 100N, \quad \text{Гц} \quad (2)$$

Для группы РПД-92:

$$f = 120N, \quad \text{Гц} \quad (3)$$

Амплитуду сигнала везде принять равной 1 В.