Лабораторная работа №3 по курсу «Радиотехнические устройства и системы» Статистические функции, программирование, файловый ввод/вывод в GNU/Octave

Кузнецов В.В., ассистент кафедры ЭИУ1-КФ

11 октября 2013 г.

1 Цель работы

Целью лабораторной работы является ознакомление с базовыми принципами статистического анализа, файловым вводом-выводом и программированием в системе численной математики GNU/Octave для применения с целью проведения расчётов различных радиотехнических устройств.

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. GNU/Octave позволяет выполнять операции с действительными и комплексными числами, матрицами, решать системы линейных уравнений, обрабатывать данные, строить графики и диаграммы. Синтаксис команд Octave близок к языку C и повторяет среду Matlab.

В ходе выполнения лабораторной работы необходимо ознакомится с принципами статистического анализа, файловым вводом-выводом и программированием в среде GNU/Octave.

При подготовке руководства использовались материалы с сайтов `http://mydebianblog.blogspot.com` (на русском языке) и `http://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Getting_started` (на английском языке).

2 Статистические функции

К статистическим функциям, относится вычисление математического ожидания m (среднее значение), математических моментов, среднеквадратического отклонения (СКО) σ.

Рассмотрим функции для вычисления этих параметров:

Сначала сгенерируем массив (вектор) x из случайных чисел, который будет служить нам объектом для вычислений. Для этого воспользуемся функцией `randn()`.

```
octave:1> x=randn(1,10000); # генерируем вектор из 10000 случайных чисел.
```

Теперь вычислим математическое ожидание (Арифметическое среднее). Арифметическое среднее значений для массива из N элементов вычисляется по формуле:

$$\overline{x} = \frac{1}{N}\sum_{i=0}^{N} x_i \tag{1}$$

В GNU/Octave для вычисления математического ожидания используется функция `mean()`. Параметром функции служит массив, среднее арифметическое значение элементов которого нужно вычислить: Например:

```
octave:2> mean(x)
ans =  0.0070703
```

СКО вычисляется по формуле:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(x_i - \overline{x})^2} \tag{2}$$

Для вычисления СКО в GNU/Octave служит функция `std()`. Например:

```
octave:3> std(x)
ans =  1.0037
```

Часто необходимо посчитать сумму всех элементов массива. Для этого служит функция `sum()`. Параметром функции служит массив, сумму элементов которого нужно вычислить. Например:

```
octave:4> sum(x)
ans =  70.703
```

Максимальный элемент массива возвращает функция `max()`, а минимальный — функция `min()`. Например:

```
octave:4> max(x) # считаем максимум
ans =  3.7975
octave:5> min(x) # считаем минимум
ans = -3.8206
```

Произведение всех элементов массива вычисляет функция `prod()`.

3 Ввод-вывод на терминал

3.1 Функции disp и input

Чтобы вывести на терминал значение переменной, служит функция `disp()`. Например:

```
octave:6> A = 1;
octave:7> disp(A);
 1
```

Чтобы ввести значение с переменной с клавиатуры, служит функция `input()`. Функция возвращает считанное числовое значение, а параметром функции служит приглашение при вводе (строка, заключённая в двойные кавычки). После вызова функция печатает приглашение, а затем ждёт ввода от пользователя.

```
octave:8> A=input("Enter number: ");
Enter number: 12
octave:9> disp(A); # Мы ввели число 12
 12
```

3.2 Функция printf

В GNU/Octave реализована функция `printf`. Синтаксис функции такой же как и у функции `printf` из стандартной библиотеки языка C. Формат вызова функции:

```
printf(fmt, var1, var2, ...)
```

Функция printf() печатает значения аргументов `var1`, `var2`, `...` из заданного списка аргументов в соответствии со строкой форматирования, адресуемой параметром `fmt`. Список переменных может быть пустым. В данном случае функция просто печатает строку `fmt`.

Строка форматирования состоит из элементов двух типов. К элементам первого типа относятся символы, которые выводятся на экран. Элементы второго типа содержат спецификации формата, определяющие способ отображения аргументов. Спецификация формата начинается символом процента, за которым следует код формата. Количество аргументов должно в точности совпадать с количеством спецификаций формата, причем соответствие устанавливается в порядке их следования. Например, при вызове следующей функции `printf()` на экране будет отображено

```
octave:10> printf("Hi %c %d %s \n", 'c', 10, "there!");
Hi c 10 there!
```

Спецификаторы формата перечислены в таблице 1.

Функция `printf()` возвращает число реально выведенных символов. Если функция возвратит отрицательное значение, то это будет свидетельствовать о наличии ошибки.

На спецификации формата могут воздействовать модификаторы, задающие ширину поля, точность и признак выравнивания по левому краю. Целое значение, расположенное между знаком % и командой форматирования, играет роль спецификации минимальной ширины поля. Наличие этого спецификатора приводит к тому, что результат будет заполнен пробелами или нулями, чтобы выводимое значение занимало поле, ширина которого не меньше заданной минимальной ширины. Если длина выводимого значения (строки или числа) больше этого минимума, оно будет выведено полностью несмотря на превышение минимума. По умолчанию в качестве заполнителя используется пробел. Для заполнения нулями перед спецификацией ширины поля нужно поместить 0. Например, спецификация формата %05d дополнит нулями выводимое число, в котором менее пяти цифр, чтобы общая длина равнялась 5 символам.

Действие модификатора точности зависит от кода формата, к которому он применяется. Чтобы добавить модификатор точности, поставьте за спецификацией ширины поля десятичную точку, а после нее — требуемое значение точности. Для форматов a, A, e, E, f и F модификатор точности определяет число выводимых десятичных знаков. Например, спецификация формата %10.4f обеспечит вывод числа с четырьмя знаками после запятой в поле шириной не меньше десяти символов. Если модификатор точности применяется к коду формата g или G, то он определяет максимальное число выводимых значащих цифр. Применительно к целым, модификатор точности задает минимальное количество выводимых цифр. При необходимости перед числом будут добавлены нули.

Если модификатор точности применяется к строкам, число, следующее за точкой, задает максимальную длину поля. Например, спецификация формата %5.7s выведет строку

Таблица 1. Спецификаторы формата функции printf()

`%c` Символ
`%d` Десятичное целое число со знаком
`%i` Десятичное целое число со знаком
`%e` Экспоненциальное представление числа (в виде мантиссы и порядка) (e на нижнем регистре)
`%E` Экспоненциальное представление числа (в виде мантиссы и порядка) (E на верхнем регистре)
`%f` Десятичное число с плавающей точкой
`%g` Использует более короткий из форматов %e или %f
`%G` Использует более короткий из форматов %E или %F
`%o` Восьмеричное число без знака
`%s` Символьная строка
`%u` Десятичное целое число без знака
`%x` Шестнадцатеричное без знака (строчные буквы)
`%X` Шестнадцатеричное без знака (прописные буквы)
`%%` Выводит знак процента

длиной не менее пяти, но не более семи символов. Если выводимая строка окажется длиннее максимальной длины поля, конечные символы будут отсечены.

По умолчанию все выводимые значения выравниваются по правому краю: если ширина поля больше выводимого значения, оно будет выровнено по правому краю поля. Чтобы установить выравнивание по левому краю, нужно поставить знак "минус"сразу после знака %. Например, спецификация формата %-10.2f обеспечит выравнивание вещественного числа с двумя десятичными знаками в 10-символьном поле по левому краю.

Существуют два модификатора формата, позволяющие функции `printf()` отображать короткие и длинные целые. Эти модификаторы могут применяться к спецификаторам типа d, i, o, u, x и X.

Пример использования спецификаторов:

```
octave:11> printf("The color: %s\n", "black");
The color: black
octave:12> printf("First number: %d\n", 12345);
First number: 12345
octave:13> printf("Second number: %04d\n", 25);
Second number: 0025
octave:14> printf("Third number: %i\n", 1234);
Third number: 1234
octave:15> printf("Float number: %3.2f\n", 3.14159);
Float number: 3.14
octave:16> printf("Hexadecimal: %x\n", 255);
Hexadecimal: ff
octave:17> printf("Octal: %o\n", 255);
Octal: 377
octave:18> printf("Unsigned value: %u\n", 150);
Unsigned value: 150
octave:19> printf("Just print the percentage sign %%\n", 10);
Just print the percentage sign %
```

Чтобы обозначить, что соответствующий аргумент указывает на длинное целое, к спецификации n можно применить модификатор l. Для указания на короткое целое приме-

ните к спецификации n модификатор h.

Символ # при использовании с некоторыми кодами формата функции printf() приобретает специальное значение. Поставленный перед кодами a, A, g, G, f, e и E, он гарантирует наличие десятичной точки даже в случае отсутствия десятичных цифр. Если поставить символ # перед кодами формата x и X, то шестнадцатеричное число будет выведено с префиксом 0x. Если же его поставить перед кодами формата o и O, то восьмеричное число будет выведено с префиксом 0. Символ # нельзя применять ни к каким другим спецификациям формата.

Спецификации минимальной ширины поля и точности могут задаваться не константами, а аргументами функции printf(). Для этого в строке форматирования используется символ "звездочка"(*). При сканировании строки форматирования функции printf() каждый символ * будет сопоставляться с соответствующими аргументами в порядке их следования.

4 Программирование

4.1 Ветвление

Ветвление организуется в GNU/Octave аналогично языку C при помощи оператора `if`.

Формат оператора `if` следующий:

```
if условие_1
  оператор_1
elseif условие_2
  оператор_2
 ...
elseif условие_N
  оператор_N
else
  оператор_3
end
```

Здесь реализована лестница if - else -if. Если условие 1 истинно, то выполняется оператор 1, иначе если условие 2 истинно, то выполняется оператор 2 и так далее. Если ни одно из условий не является истинным, то выполняется оператор 3.

Рассмотрим пример. Данный код печатает сообщение "Greater then two если переменная `a` больше двух, если переменная меньше двух печатается сообщение "Less then two"и если переменная равна двум, то печатается сообщение "Equals two".

```
octave:20> a=2; if a>2 printf("Greater then two\n"); elseif a<2 printf("Less
then two\n"); else printf("Equals two\n"); end
Equals two
```

4.2 Циклы

Аналогично языку C в GNU/Octave реализованы циклы с предусловием (while) с постусловием (do-while) и с параметром (for).

Рассмотрим наиболее часто применяемый цикл `for`. В общем случае цикл имеет следующий вид:

```
for переменная = вектор
   ... # тело цикла
end
```

Переменной поочерёдно присваивается значение каждого из элементов вектора (массив, матрица-строка, матрица-столбец) и каждый раз выполняется тело цикла.

Рассмотрим пример использования цикла `for`. Данный код печатает числа от 1 до 10 и разделяет их пробелом.

```
octave:21> for i=1:10 printf("%d ",i) end; printf("\n");
 1 2 3 4 5 6 7 8 9 10
```

5 Файловый ввод-вывод

Функции ввода - вывода предназначены для ввода данных из текстовых и двоичных файлов и для сохранения результатов вычислений в текстовые и двоичные файлы.

Для этой цели служат функции `save` и `load`.

Функция

`save("имя_файла","формат_файла","переменная1",...,"переменнаяN")`

Сохраняет в файл `"имя_файла"` переменные `"переменная1"` ... `"переменнаяN"`. Все параметры функции и даже имена переменных являются строками, заключёнными в двойные кавычки. Для сохранения данный обычно выбирают файлы с расширением dat. Чтобы сохранить переменные в текстовый файл, нужно формат_файла указать как `"-ascii"`. Файл располагается в текущем каталоге.

Функция `load("имя_файла")` загружает из текстового файла числа. Функция возвращает матрицу, которая содержит загруженные числа. Файл располагается в текущем каталоге.

Рассмотрим пример. Создадим матрицу M размерности 10x3 из случайных чисел и сохраним её в текстовый файл matrix.dat. Потом загрузим из этого файла данные в матрицу $M1$.

```
octave:1> M=randn(10,3)
M =

   0.509290   0.512708  -0.337731
   0.431837   0.142272  -1.682700
  -0.231123   0.248536   0.125299
   0.303610   0.719357   0.770016
  -1.511875  -0.018935  -0.024347
   1.101249   0.704358  -0.228250
  -0.316279   0.845822   0.321519
  -0.393148   0.565603   0.152713
  -0.625644  -0.377220   1.747263
   0.231609  -1.967496   1.930600

octave:2> save("matrix.dat","-ascii","M");
octave:3> M1=load("matrix.dat")
M1 =

   0.509290   0.512708  -0.337731
```

```
 0.431837   0.142272  -1.682700
-0.231123   0.248536   0.125299
 0.303610   0.719357   0.770016
-1.511875  -0.018935  -0.024347
 1.101249   0.704358  -0.228250
-0.316279   0.845822   0.321519
-0.393148   0.565603   0.152713
-0.625644  -0.377220   1.747263
 0.231609  -1.967496   1.930600
```

6 Контрольное задание

6.1 Цель контрольного задания

Используя материал лабораторных работ №1, 2, 3 сгенерировать строку, содержащую коэффициенты заполнения ШИМ для генерации синусоидального сигнала при помощи микроконтроллера.

6.2 Теоретическая часть

Любой современный микроконтроллер (МК) имеет аппаратный ШИМ. Если периодически изменять длительность импульса ШИМ, то на выходе можно получить синусоидальный сигнал (см. рис. 1). Если пропустить сформированнный как на рис.1 ШИМ-сигнал через фильтр низких частот (ФНЧ), то на выходе ФНЧ получим синусоиду.

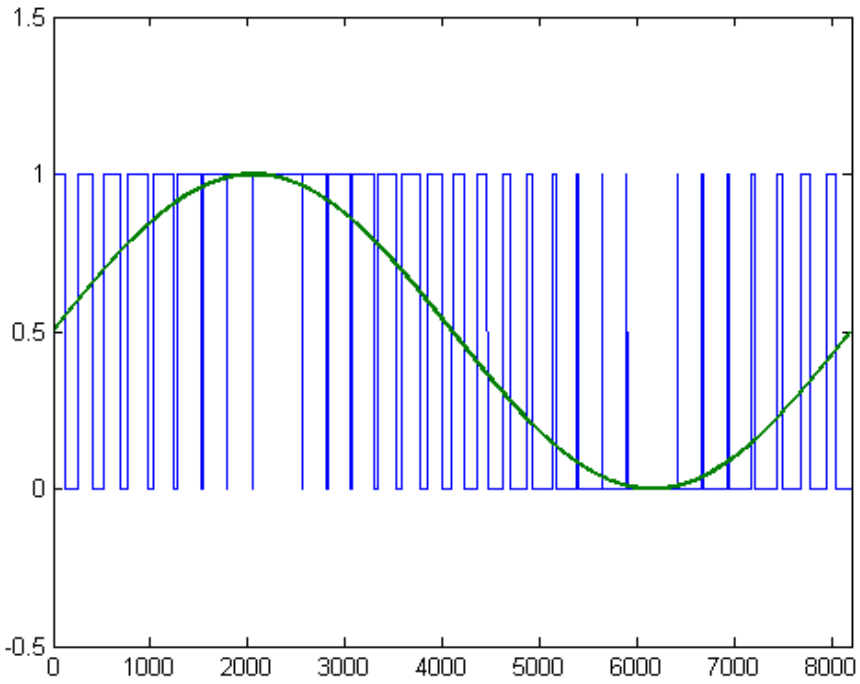

Рис. 1. Генерация синусоидального сигнала при помощи ШИМ

Временная диаграмма ШИМ-импульса показана на рис.2.

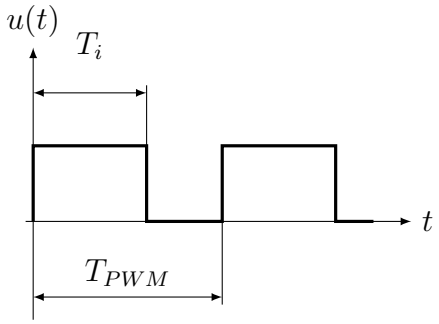

Рис. 2. Временная диаграмма ШИМ-импульсов

Итак, чтобы сформировать синусоидальный сигнал частотой F_{sin}, нужно периодически менять длительность ШИМ-импульсов. Для этого нужно рассчитать массив длиной N, содержащий длительности T_i ШИМ-импульсов, которые измеряется в тактах процессора. Должна быть известна тактовая частота МК $F_{\text{т}}$ и частота ШИМ F_{PWM}. Частота ШИМ связана с периодом T_{PWM} ШИМ-импульсов:

$$F_{PWM} = \frac{1}{T_{PWM}} \tag{3}$$

Длина массива N длительностей ШИМ-импульсов:

$$N = \frac{F_{PWM}}{F_{\text{sin}}} \tag{4}$$

Период следования ШИМ-импульсов в тактах процессора.

$$T_{PWM(cyc)} = \frac{F_{\text{т}}}{F_{PWM}} \tag{5}$$

Массив должен содержать N значений, каждое из которых (с номером i) вычисляется по формуле:

$$T_i = \frac{T_{PWM}}{2} + \frac{T_{PWM}}{2} \cdot \sin\left(\frac{2\pi i}{N}\right) \tag{6}$$

6.3 Задание для самостоятельного выполнения

Выполняется с использованием материала лабораторных работ №1, 2, 3.

Рассчитать массив по формуле (6) для следующих исходных данных: тактовая частота МК $F_{\text{т}} = 1\,\text{МГц}$; частота ШИМ $F_{PWM} = 5\,\text{кГц}$; частота синусоидального сигнала $F_{\text{sin}} = 50\text{Гц}$. Сначала рассчитать период ШИМ по формуле (5), затем длину N массива по формуле (4), затем сгенерировать массив целых чисел от 0 до $N-1$ с шагом 1 при помощи функции `linspace()`, затем рассчитать таблицу функции sin по формуле (6). После того, как получен массив, нужно вывести его элемнты, округлённые до целого при помощи функции `printf()`.

7 Заключение

В результате выполнения лабораторной работы произведено ознакомление с принципами статистического анализа данных, программирования и файлового ввода-вывода в системе GNU/Octave.