Санкт-Петербургский политехнический университет Петра Великого Институт физики, нанотехнологий и телекоммуникаций

Высшая школа прикладной физики и космических технологий

ОТЧЕТ
по лабораторной работе

«Интерфейс MatLab, создание массивов в MatLab и операции с ними»

направление 11.03.01 – «Инфокоммуникационные технологии и системы связи»

Вариант № 16

Выполнил
студент гр. 23433/3 Е.Ю. Кулешов

Преподаватель
асс., к.т.н. Е.А. Щербинина

«__» _____ 2018 г

Санкт-Петербург
2018

Часть 1. Запись арифметических выражений

В результате выполнения части 1 была реализована первая секция скрипта (см. приложение 1), которая, в соответствии с вариантом, создаёт переменные, вычисляет выражение и выводит его результирующие значение двумя способами, а также сохраняет результаты вычислений.

1. В ходе работы в командном окне были заданы значения следующих переменных:

```
x = 0.5;

a = 2.71;

c = 3.25;

d = -3.53;

k = 5;
```

2. В ходе работы было записано следующие выражение на языке MATLAB:

$$y = \frac{\sin(ax^2 - c)}{0{,}25k^2xd} - \left|\sqrt[3]{x^2 + \ln 3} - \cos kx\right| + 10^4 x^5 cd$$

```
y = (sin(a*x^2-c)/(0.25*x*d*k^2))-abs((x^2+log(3))^(1/3)-
cos(k*x))+10^4*x^5*c*d;
```

На языке MatLab были использованы такие встроенные функции, как sin() – синус и `cos()` – косинус угла, заданного в радианах, `log()` – значение натурального логарифма, а также `abs()` – модуль.

3. В ходе работы были рассмотрены два варианта вывода результирующих значений с использованием следующих встроенных функций MATLAB:

`disp()` – вывод значения переменной в командное окно и `fprintf()` – вывод более одной переменной. В итоге в командном окне будет показано два варианта вывода:

```
y = -3587.0134
```

и

```
y = -3587.013375.
```

4. В ходе работы результаты вычислений и состояние командного окна были сохранены, соответственно, функциями `save()` и `diary`:

```
diary ON;

diary ('lab1_kuleshov.txt');

disp(['y = ' num2str(y)]);
fprintf('y = %f\n', y);

diary OFF;

save('lab1_kuleshov.mat', 'y');
```

Функция `dairy` создает текстовый документ с расширением .txt, в котором записывается все, что выводится в командном окне тогда, как функция `save()` создает документ с расширением .mat, в котором записываются все переменные, которые мы указываем в функции.

5. Последовательность выполненных команд была записана в первую секцию скрипта.

Часть 2. Скрипты

В результате выполнения части 2 была реализована вторая секция скрипта (см. приложение 1), которая, в соответствии с вариантом, выдаёт приглашение на ввод и принимает с клавиатуры все необходимые данные, а затем осуществляет расчёт и вывод результатов в командное окно в необходимом виде.

1. В ходе работы была реализована часть скрипта, которая выдаёт приглашение на ввод и принимает с клавиатуры следующие исходные данные:

```
a = input ('Enter the value of a = ');
h = input ('Enter the value of h = ');
P = a*6;
A = sqrt(h^2+(3/4)*a^2);
s = (3*3^(1/3)/2)*a^2;
x = ' ';
```

2. В ходе работы была реализована часть скрипта, которая осуществляет расчет следующего выражения:

```
S = s + (1/2)*P*A;

V = (1/3)*s*h;
```

3. В ходе работы была реализована часть скрипта, которая осуществляет вывод результатов расчета в следующем формате:

```
     Initial data:
Pyramid side a = 5
Pyramid height h = 8
-----------------------------------------

Apophem = 9
Perimeter = 30
                    Answer:
Veight pyramid = 144
Square pyramid = 191
```

Использовались все возможности функции `fprintf()`:

```
fprintf('%10sInitial data:\n%5sPyramid side a =
%.0f\n%5sPyramid height h = %.0f\n', x,x,a,x,h);
```

```
fprintf('%s\n\n','-----------------------------------
------');
fprintf('%10sApophem = %.0f\n%10sPerimeter = %.0f\n',
x,A,x,P);
fprintf('%30sAnswer:\n%20sVeight pyramid =
%.0f\n%20sSquare pyramid = %.0f\n\n', x,x,V,x,S);
```

Часть 3. Создание матриц и операции с ними

1. В ходе работы реализована 3 секция скрипта, которая на основе матрицы `M` произвольного размера создает модифицированную матрицу `Mmodified`, элементы которой являются элементами матрицы `M`, расположенными в четных строках и столбцах.

Матрица, полученная при запуске программы (`m` = 5, `n` = 8):

2	7	5	6	1	6	1	8
1	5	8	6	1	1	8	5
6	5	8	4	2	9	9	2
3	5	3	2	2	7	9	4
5	2	2	9	6	7	8	2

А ее модификация:

5	6	1	5
5	2	7	4

Ввод размера матриц и выведение данных:

```
m = input('Enter line = ');
n = input('Enter column = ');
M = randi(9, [m, n]);
Mmodified = M(2:2:end, 2:2:end);
disp(M);
disp(Mmodified);
```

С помощью встроенной функции `input()` вводится размер матрицы `M`.

При помощи функции `randi()` данная матрица произвольно заполняется числами от 1 до 9. Далее создается матрица `Mmodified`, которой поочередно присваиваются элементы четных строк и столбцов матрицы `M`. `(2:2:end, 2:2:end)` означает, что мы все строки и столбцы проходим с шагом 2 и записываем их в новую матрицу. С помощью функций `disp()` мы выводим в командное окно исходную матрицу `M` и `Mmodified`, чтобы можно было увидеть результат данной части скрипта.

2. В ходе работы реализована 4 секция скрипта, в которой создается вектор-строка `v`, на основании которого создается другой вектор-строка w, такой же длины как исходный вектор. Вектор `w` содержит все элементы исходного вектора, но в обратном порядке.

```
L = input('Enter lengh = ');
v = randi (9, 1, L);
w = v(end:-1:1);
disp(v);
disp(w);
```

Вначале вводится длина вектора `v` при помощи функции `input()`. Далее с помощью `randi()` создается случайный вектор длиной `x` из целых чисел от 1 до 9. Создаем новый вектор w, который заполняется элементами вектора `v` с конца (для этого мы прописываем `end` – последний элемент) с шагом -1 до первого элемента. С помощью функций `disp()` выводим два вектора в командную строку, чтобы можно было видеть результат данной части скрипта (`L` = 7):

```
4     1     5     2     2     2     2

2     2     2     2     5     1     4
```

3. В ходе работы реализована 5 секция скрипта, в которой с помощью функции `diag()` необходимо было сформировать матрицу 16x16:

$$D = \begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}.$$

С помощью `diag()` создается матрица, на главной диагонали которой стоит значение “-2” , затем матрицы сформированные из единиц , но смещенные от главной диагонали на “-1”, “+1”, “-15”, “+15”, после чего, матрицы складываются.

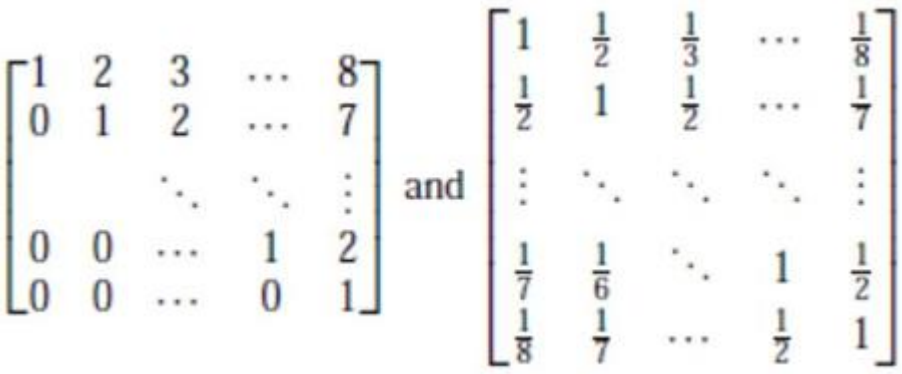
```
x1 = [-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2];
x2 = ones(16);
A = diag(x1)+diag(diag(x2,1),1)+diag(diag(x2,1),-
1)+diag(diag(x2,15),15)+diag(diag(x2,15),-15);
```

Также нужно было сформировать данную матрицу с использованием команды `toeplitz()`, которая создает квадратную матрицу, размером равным количеству элементов вектора - строки. Для этого создается вектор строка:

```
y = [-2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
D = toeplitz(y);
```

```
-2     1     0     0     0     0     0     0     0     0     0     0     0     0     0     1
 1    -2     1     0     0     0     0     0     0     0     0     0     0     0     0     0
 0     1    -2     1     0     0     0     0     0     0     0     0     0     0     0     0
 0     0     1    -2     1     0     0     0     0     0     0     0     0     0     0     0
 0     0     0     1    -2     1     0     0     0     0     0     0     0     0     0     0
 0     0     0     0     1    -2     1     0     0     0     0     0     0     0     0     0
 0     0     0     0     0     1    -2     1     0     0     0     0     0     0     0     0
 0     0     0     0     0     0     1    -2     1     0     0     0     0     0     0     0
 0     0     0     0     0     0     0     1    -2     1     0     0     0     0     0     0
 0     0     0     0     0     0     0     0     1    -2     1     0     0     0     0     0
 0     0     0     0     0     0     0     0     0     1    -2     1     0     0     0     0
 0     0     0     0     0     0     0     0     0     0     1    -2     1     0     0     0
 0     0     0     0     0     0     0     0     0     0     0     1    -2     1     0     0
 0     0     0     0     0     0     0     0     0     0     0     0     1    -2     1     0
 0     0     0     0     0     0     0     0     0     0     0     0     0     1    -2     1
 1     0     0     0     0     0     0     0     0     0     0     0     0     0     1    -2
```

4. В ходе работы реализована 6 секция скрипта, в которой необходимо сформировать следующие матрицы с использованием любой из известных команд (при выводе второй матрицы использовать команду `format rat`):



```
G = toeplitz(1:8);
O = triu(G);
U = 1./G;
disp(O);
format rat;
disp(U);
format short;
```

Для первой матрицы с помощью функции `toeplitz()` создается квадратная матрица 8*8, которая преобразуется в верхнетреугольную командой `triu()`. Для второй создается та же матрица, каждый последующий элемент которой преобразуется в обратный. `Format rat` позволяет вывести на экран элементы матрицы в виде правильных дробей, а `format short` возвращает в стандартную форму для последующей работы с выводом.

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

1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
1/2	1	1/2	1/3	1/4	1/5	1/6	1/7
1/3	1/2	1	1/2	1/3	1/4	1/5	1/6
1/4	1/3	1/2	1	1/2	1/3	1/4	1/5
1/5	1/4	1/3	1/2	1	1/2	1/3	1/4
1/6	1/5	1/4	1/3	1/2	1	1/2	1/3
1/7	1/6	1/5	1/4	1/3	1/2	1	1/2
1/8	1/7	1/6	1/5	1/4	1/3	1/2	1

5. В ходе работы реализована 7 секция скрипта, в которой необходимо подобрать значения для `X`, `m`, `n`, чтобы данный кусок кода успешно выполнялся в MatLab. Дать описание каждой из четырех строк кода:

```
Y = reshape(X,[m size(X,1)/m n size(X,2)/n]);

Y(:,:,:) = Y(:,:,end:-1:1);

Y = permute(Y,[1 4 3 2]);

Y = reshape(Y, [m*size(X,2)/n n*size(X,1)/m]) \
```

`X = 1:16;` — задаем вектор – строку `X` с элементами от 1 до 16;
`X = reshape(X, [4 4]);` — функция превращает вектор – строку в двумерную матрицу размером 4*4.
`Y = reshape(X, [m size(X,1)/m n size(X,2)/n]);` - функция превращает двумерную матрицу `X` в четырёхмерную с необходимым размером. При этом `size(X,1)` и `size(X,2)` кратны `m` и `n` соответственно.

`Y(:,:,:) = Y(:,:,end:-1:1);` - функция показывает последовательно все матрицы, являющиеся элементами двух больших массивов с этими матрицами.

`Y = permute(Y,[1 4 3 2]);` — функция, меняющая порядок размерностей местами в указанном порядке.

`Y = reshape(Y, [m*size(X,2)/n n*size(X,1)/m]);` — данная функция возвращает матрицу из четыреххмерного вида в двумерный и матрицы соединяются в изначальном порядке.

При `m` и `n` = 2:

```
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16

    13     9    15    11
    14    10    16    12
     5     1     7     3
     6     2     8     4
```

Вывод:

В ходе данной лабораторной работы были изучены: использование элементарных арифметических операций и функций в MatLab, создание скриптов, разделение их на секции, основы работы с матрицами, была осуществлена работа с вводом и выводом данных и, как итог, были изучены основы работы с m – файлами (или файлами – сценариями)

Приложение:

```
%% FirstPart

x = 0.5;
a = 2.71;
c = 3.25;
d = -3.53;
k = 5;

y = (sin(a*x^2-c)/(0.25*x*d*k^2))-
abs((x^2+log(3))^(1/3)-cos(k*x))+10^4*x^5*c*d;

diary ON;
```

```
diary ('lab1_kuleshov.txt');

disp(['y = ' num2str(y)]);
fprintf('y = %f\n', y);

diary OFF;

save('lab1_kuleshov.mat', 'y');

%% SecondPart

a = input ('Enter the value of a = ');
h = input ('Enter the value of h = ');
P = a*6;
A = sqrt(h^2+(3/4)*a^2);
s = (3*3^(1/3)/2)*a^2;
S = s + (1/2)*P*A;
V = (1/3)*s*h;
x = ' ';
fprintf('%10sInitial data:\n%5sPyramid side a =
%.0f\n%5sPyramid height h = %.0f\n', x,x,a,x,h);
fprintf('%s\n\n','-----------------------------------
-------');
fprintf('%10sApophem = %.0f\n%10sPerimeter = %.0f\n',
x,A,x,P);
fprintf('%30sAnswer:\n%20sVeight pyramid =
%.0f\n%20sSquare pyramid = %.0f\n\n', x,x,V,x,S);

%% ThirdPart_1

m = input('Enter line = ');
n = input('Enter column = ');
M = randi(9, [m, n]);
Mmodifide = M(2:2:end, 2:2:end);
disp(M);
disp(Mmodifide);

%%ThirdPart_2

L = input('Enter lengh = ');
v = randi (9, 1, L);
w = v(end:-1:1);
disp(v);
disp(w);

%%ThirdPart_3

x1 = [-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2];
```

```
x2 = ones(16);
A = diag(x1)+diag(diag(x2,1),1)+diag(diag(x2,1),-
1)+diag(diag(x2,15),15)+diag(diag(x2,15),-15);
y = [-2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
D = toeplitz(y);
disp(A);
disp(D);

%%ThirdPart_4

G = toeplitz(1:8);
O = triu(G);
U = 1./G;
disp(O);
format rat;
disp(U);
format short;

%%ThirdPart_5

X = 1:16;
X = reshape(X, [4 4]);
m = 2;
n = 2;
Y = reshape(X,[m size(X,1)/m n size(X,2)/n]);
Y(:,:,:) = Y(:,:,end:-1:1);
Y = permute(Y,[1 4 3 2]);
Y = reshape(Y,[m*size(X,2)/n n*size(X,1)/m]);
disp(X);
disp(Y);
```