Tutorial 2 **Arrays in SCILAB**

Date:

Aim

To create simple matrix and do simple matrix operations using SCILAB.

Excercises

1. Array is collection of data objects. Its MATLAB equivalent is a vector of cells. Like MATLAB cell arrays these need not be of the same type. They can be scalars, matrices, character strings, string matrices, functions, as well as other lists.SCILAB treats a scalar number as a matrix of size 1*1 (and not as a simple number) because the basic data type in SCILAB is matrix. A range of number to form a vector.

   Example:

   ```
           -->a=[5]
                 a  =
                  5.
   ```

2. SCILAB treats a vector as matrix in the form of rows and columns which is given with the help of square brackets and semi-colons.

   Example: Row Vector

   ```
   -->b=[1 2 3]
         b  =
          1.    2.    3.
   ```

   Column Vector :

   ```
   -->c=[1;2;3]

          c  =

             1.
             2.
             3.
   ```

3. In SCILAB, matrix is a rectangular arrangement of numbers in rows and columns enclosed in brackets. It is given by either column vector or row vector with the help of semi-colons and square brackets.

   ```
   -->x=[1 2 3]
         x =
   ```

```
    1.    2.    3.
```

4.
```
-->x=[1 2 3]
   x  =

      1.    2.    3.
```

5.
```
-->y=[1;2;5]
          y =

             1.
             2.
             5.
```

6.
```
-->z=[2 1 3]
          z  =

             2.    1.    3.

 -->x+z

          ans =

              3.    3.    6.
```

7. Is it possible to add a row vector and a column vector.

No, it is not possible to add a row vector and a column vector because of the inconsistency of matrix

```
          -->b=x+y
                !--error 8
          Inconsistent addition.
```

8.
```
-->a=x.*z
          a  =

                2.    2.    9.

     -->b=x./z
      b  =

         0.5    2.    1.
```

9.
```
-->k=[2]
      k =

         2.

     -->a.*k
      ans =

       4.    4.    18.
```

10.
```
-->a=[1 2 3; 4 5 6; 7 8 9;]
```

```
a  =

    1.    2.    3.

    4.    5.    6.

    7.    8.    9.
```

To suppress the output, we use semicolon(;) operator.

```
-->a=[1 2 3; 4 5 6; 7 8 9];
```

11.

```
-->b=a'
 b  =

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

12.

```
-->x=[1 2 3]
 x  =

   1.    2.    3.

-->y=[4 5 6 7]

 y  =

   4.    5.    6.    7.
-->a=[x y]
 a  =

   1.    2.    3.    4.    5.    6.    7.
```

13.

```
-->x=[0:10]
 x =

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

14.

```
-->y= sin(x)
 y  =

    0.    0.8414710    0.9092974    0.1411200  - 0.7568025
- 0.9589243  - 0.2794155    0.6569866    0.9893582
0.4121185  - 0.5440211
```

15.

```
-->z=sqrt(x.*y)
```

```
z  =

    0    0.9173173    1.3485529    0.6506612    1.7398879i
2.1896624i    1.2947946i    2.144506    2.8133372
1.9258937    2.332426i
```

16.
```
-->x=[0 1.5 3 4 5 7 9 10]
 x  =

    0.    1.5    3.    4.    5.    7.    9.    10.

-->m=[0.5]

 m  =

  0.5

-->c=[2]
 c  =

 2.
-->y=[m.*x+c]
 y  =

    2.    2.75    3.5    4.    4.5    5.5    6.5    7.
```

17. creating list of row vectors and working with them

```
a. -->t=[1:10]

 t  =

    1.    2.    3.    4.    5.    6.    7.    8.    9.
    10.

  -->x=[t.* sin(t)]

 x  =

    0.8414710    1.8185949    0.4233600  - 3.02721  -
4.7946214  - 1.676493    4.5989062    7.914866
3.7090664  - 5.4402111

b. -->y=[(t-1)./(t+1)]

 y  =

    0.    0.3333333    0.5    0.6    0.6666667
0.7142857    0.75    0.7777778    0.8    0.8181818

c. -->z=sin(t.^2)/((t.^2))
z  =
    0.8414710  - 0.1892006    0.0457909  - 0.0179940  -
0.0052941  - 0.0275494  - 0.0194643    0.0143754  -
0.0077764  - 0.0050637
```

18.
```
-->x=1:10
 x  =

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

19.
```
-->t=0:2:20
 t  =

  0.    2.    4.    6.    8.    10.    12.    14.    16.
 18.    20.

-->t'

  ans  =

           0.

           2.

           4.

           6.

           8.

          10.

          12.

          14.

          16.

          18.

          20.

-->length (t)

 ans  =

       11.

-->size (t)

 ans  =

 1.    11.
```

19 a. Linearly spaced vector. `linspace(x1, x2)` generates a row vector of n (default value=100) linearly equally spaced points between `x1` and `x2`.

```
-->linspace(1,2,10)

 ans  =

         column 1 to 7

     1.    1.11111    1.22222    1.33333    1.44444    1.55556
    1.66667

         column  8 to 10

     1.77778    1.88889    2.
```

```
20. -->A=[1 5 6;7 4 2; -3 6 7]

    A     =


           1.    5.    6.
           7.    4.    2.
         - 3.    6.    7.
      -->size(A)
       ans  =

           1.    3.
```

```
21.
      a. -->x= zeros(6,6)
      x  =

           0.    0.    0.    0.    0.    0.
           0.    0.    0.    0.    0.    0.
           0.    0.    0.    0.    0.    0.
           0.    0.    0.    0.    0.    0.
           0.    0.    0.    0.    0.    0.
           0.    0.    0.    0.    0.    0.


      b. -->y=ones(3,3)
      y  =

           1.    1.    1.
           1.    1.    1.
           1.    1.    1.

      c. -->z=eye(3,3)*10
      z  =

           10.    0.     0.
           0.     10.    0.
           0.     0.     10.
           d. -->x=rand(3, 3) x =

                0.2113249    0.3303271    0.8497452

                0.7560439    0.6653811    0.6857310

                0.0002211    0.6283918    0.8782165

          e. -->x=int(rand(3, 3)*100)

               x     =

                  48.     50.    63.

                 33.     43.    40.

                 59.     26.    91.


          f.  -->y=int(rand(3, 3)*100)

               y     =

                     6.   72.    23.

                    56.  19.    23.

                    66.  54.    21.


22. -->A = [9 -36 30; -36 192 -180; 30 -180 180]
     A   =

         9.    - 36.      30.

      - 36.     192.   - 180.

        30.  - 180.     180.

    -->b = [3 -24 30]
     b   =

        3.  - 24.     30.

    -->x=A/b
      x   =

         1.2060606

       - 6.8121212

         6.6060606
```

```
23. -->A = [9 -36 30; -36 192 -180; 30 -180 180]

     A  =

          9.    - 36.      30.

       - 36.     192. - 180.
          30.    - 180.  180.

 -->det(A)

     ans  =

          2160.
```

```
24. -->A = [9 -36 30; -36 192 -180; 30 -180 180]

     A  =

          9.    - 36.      30.

       - 36.     192.  - 180.

         30.  - 180.    180.

   -->inv(A)

     ans  =

          1.             0.5           0.3333333

          0.5            0.3333333     0.25

         0.3333333     0.25          0.2
```

```
25. -->A=[1  2  3;  3   3  4;2  3  3]

     A  =

            1.     2.     3.

            3.     3.     4.

            2.     3.     3.

   -->B=[1;1;2]

     B  =

           1.

           1.

           2.
```

```
-->X=A/b

   X  =

        0.0303030
        0.0383838
        0.0161616



   -->inv(A)

         ans  =



        - 0.75    0.75  - 0.25

        - 0.25  - 0.75    1.25

          0.75    0.25  - 0.75

   -->inv(A)*b

      !--error 10

     Inconsistent multiplication.
```

26. SCILAB Operators and their significance

S. No.	SCILAB Operator	Significance
1	**[ ]**	Matrix definition
2	;	Statement separator
3	**( )**	Extraction
4	`	Transpose
5	+	Addition
6	−	Subtraction
7	*	Multiplication
8	/	Right Division
9	\	Left Division
10	^	Exponent
11	.*	Element Wise Multiplication
12	./	Element Wise Right Division
13	.\	Element Wise Left Division

27. SCILAB built-in functions and their significance

S. No.	Built-in functions	Significance
1	`Size`	Size of Matrix
2	`Zeros`	Zero Matrix
3	`Eye`	Identity Matrix
4	`Length`	Length of the Matrix
5	`Rand`	Random number generation
6	`Det`	Determinant
7	`Inv`	Inverse of a Matrix

Result

Thus we used the SCILAB to understand creating the vectors and the matrix computations.