Andhra Pradesh State Skill Development Corporation

SciLab

Matrix Operations in Scilab

Matrix Operations in Scilab

INTRODUCTION

We have already mentioned that matrices are central to both Matlab and Scilab. In this chapter, we will attempt to cover a wide variety of Scilab features on matrix handling. A matrix, in layman’s language, is tabular data arranged in rows and columns. Let us see an example of a matrix.

$$A=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

This can be defined in Scilab in a single line, with rows separated by semicolons or in three lines, with a line break after each row as shown below.

- A= [11 12 13; 21 22 23; 31 32 33]

$$A=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

In previous tutorial we have seen that vector is a special case of a matrix arranged in a single row or single column for instance B= [23,34,56,78,90] represents a vector Matrix having a single element is treated as a scalar, A=39 and A=[39] are treated the same way by Scilab.

Any element of the matrix can be referred to by specifying its row and column indices. Note that indices start from 1 and not 0, as in many modern programming languages If we want to access the second element of the first row, we may refer to it as ‘A(1,2)’

Column 1 2 3 Row

$$A=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

The first number in parenthesis refers to the row to which the element belongs and the second number indicates the column containing the element.

- A(1,2)

ans=12

Similarly, we can display the third element of the second row using

- A(2,3)

ans=23

To overwrite any defined value, we can simply reassign

- A(2,3)=0

```
      11.   12.   13
A=    21    22    23
      31    32    33
```

In this chapter, we will discuss arithmetic and relational operators on matrices and then a set of basic matrix processing features. All mathematical functions introduced in the chapter on scalar are also applicable to the matrix. This is however left as an exercise to the reader.

ARITHMETIC OPERATORS FOR MATRICES

In matrix arithmetic we have basically three kinds of commands to be aware of:

- Basic arithmetic
- Element wise arithmetic and left and right division
- Relational and logical operations

Basic arithmetic

Addition, subtraction, and multiplication are done on matrices in Scilab in exactly the same way as they are traditionally defined in mathematics. These operations work only if the size of the matrices is compatible. In the case of addition and subtraction. The matrices have to be of exactly the same dimensions In the case of multiplication an (mxk) matrix is compatible with only a (kxn) matrix. 1.e, the second matrix must have the same number of rows as the number of columns of the matrix. The following examples illustrate addition, subtraction, and multiplications.

- a= [1 2; 3 4]
- b=[2 3; 4 5]
- c=a+b

```
c= 3.    5.
   7.    9.
```

- d=b-a

```
d=  1.     1.
    1.     1.
```

- e=a*b

```
e=  10.     13.
    22.     29.
```

Division of matrices is not straightforward and involves a concept of a matrix inverse. However, here it may be noted that the division operator has the same effect as multiplication by the inverse. By using the operator/ and \, we can choose which matrix to invert a\b is equivalent to inv(a)*b and a/b are equivalent to a*inv(b). We show below the equivalence using the inv( ) function which we will look at closely in later sections.

- a=[1 2;3 4]
- b=[2 3;4 5]
- a\b

```
ans= 0.    1.
     1.    2.
```

- inv(a)*b

```
ans=  0.     -1.
      1.      2.
```

- a/b

```
ans=  1.5    -0.5
      0.5     0.5
```

- a*inv(b)

```
ans=  1.5    -0.5
      0.5     0.5
```

Element wise arithmetic

Even though element-wise operations on matrices may not always be mathematically meaningful, in practice many uses of such operations may occur. Scilab differentiates elementwise matrix operations with a dot preceding the operator. It may be noted that the traditional addition and subtraction are element-wise operations.

See the examples below which demonstrate element-wise multiplication and division. Be sure to compare the results with those of traditional multiplications and division in above examples. Also note that the matrices only need to be of the same size to be compatible, unlike in the case of traditional multiplication where they need to have dimensions (mxk) and (kxn).

- a= [1 2; 3 4]
- b=[2 3; 4 5]
- a*b

```
ans= 2.   6.
     12.   20
```

Element wise division has left and right options indicated by / and\ operators. See examples below. Both metrics a and b must be of the same size. In the left division, the elements of b are divided by the element of a. In the right division, the elements of a are divided by the elements of b.

- a=[1 2; 3 4]
- b= [2 3; 4 5]
- a\b

```
ans=  2            1.5
      1.3333333   1.25
```

- a/b

```
ans=  0.5        0.666667
      0.75       0.8
```

Basic Matrix Operations

In this section, we shall see how to handle a matrix, including navigation through it in special ways and implementing common operations such as transpose, inverse, etc.

One of the basic information about matrices is their dimensions. For two dimensional matrices, it is of the form [mxn], where m is the number of rows and n is the number of columns. We can use size( ) function in Scilab to determine the dimension of a matrix, as demonstrated below

$$A=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

- size(A)

```
ans=   3.
```

The answer shows that there are three rows and three columns for the matrix A or A is a [3*3] matrix.

Square matrices (matrices with an equal number of rows and columns) have their diagonal elements defined as indicated below.

$$A=\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}$$

If aij is the matrix element then in general all elements with i=j define the diagonal elements. In Scilab, we use the function diag( ) to find the diagonal elements.

- diag(A)

```
ans=    11.
        22.
        33.
```

We can find out the trace of a matrix, once the diagonal elements are found out. Trace is the sum of all diagonal elements

- trace(A)

```
ans= 66
```

The lower triangular matrix of A which are all elements in the diagonal and below it can be found with tril( ) function.

- tril(A)

```
ans= 11.   0.    0.
     21.   22.   0.
     31.   32.   33.
```

Similarly, the upper triangular matrix which is a square matrix whose elements below the diagonal elements are zeros is obtained using the triu( ) function.

- triu(a)

```
ans= 11.   12.   13.
     0.    22.   23.
     0.    0.    33.
```

Another basic operation is the transpose of a matrix which is obtained by interchanging rows and columns of a given matrix. The figure below illustrates a matrix A and its transpose AT

$$A=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

$$A^T=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

In Scilab, we use the (‘) operator to find the transpose

- A’

```
ans=   11.    21.     31.
       12.    22.     32.
       13.    23.     33.
```

Scilab uses the (:) operator to specify the range of rows and columns suppose we want to select rows 2 and 3 all columns i.e, 1-3, we proceed as follows:

- A(2:3, 1:3)

```
ans=  21.   22.    23.
      31.   32.    33.
```

If you want to navigate through the entire rows, but only columns 1 and 2 type colon followed by the columns (i.e; 1:2) to be displayed

- A(:, 1:2)

```
ans=  11.     12.
      21.     22.
      31.     32.
```

See an example of specifying entire columns and rows 1 and 2

- A(1:3, :)

```
ans=  11.    12.    13.
      21.    22.    23.
```

The entire matrix can be reshaped into a vector column wise

- b=a(:)

```
b=        11.
          12.
          13.
```

```
21.
22.
23.
31.
32.
33.
```

The ':' facility makes it possible to extract a submatrix of A. by specifying the required rows and columns of it.

$$A=\begin{bmatrix}11 & 12 & 13\\ 21 & 22 & 23\\ 31 & 32 & 33\end{bmatrix}$$

- C=A(2:3, 2:3)

  ```
  C=   22.   23.
       32.   33.
  ```

  We can delete a row or column of a matrix by setting it to a null vector [ ]. A null vector is a vector having no elements in it. Here is an example of deleting the second row.
- A(:,2)=[ ]

  ```
  A=   11.   12.    13.
       31.   32.    33.
  ```

  You can append a row or column to an existing matrix if the row or column has the same length as the existing matrix. See this example of appending a row to the matrix A.
- P=[41   42 43]
- A=[A;P]

  ```
  B=   11     12     13
       21     22     23
       31     32     33
       41     42     43
  ```

Likewise, we can see how a column can be appended

- Q= [14

  24

  34]

  - B= [A,Q]

  ```
  B=  11   12   13   14
      21   22   23   24
      31   32   33   34
  ```

- e=[ ]

  e=[e;1 2 3]

  e= 1. 2. 3.

A comparatively involved concept related to matrices is their eigenvalue and eigenvectors For every square matrix A, the equation A-1=0 is known as the characteristic equation is a vector whose values are known as eigenvalues and columns of the matrix with 's diagonals are known as eigenvectors. An insightful explanation of eigenvalues and vectors is beyond the scope of this book. We will only briefly note that every matrix can be considered as a transformation operator as in the example below. A point with coordinates x,y can be transformed into x',y' by the transformation

$$[x' \quad y'] = [A] \begin{bmatrix} x \\ y \end{bmatrix}$$

If A=1, then [x' y']. If A=[-1,0; 0,1] then [x' y'], resulting in a reflection of the point with respect to the y-axis. A general matrix A= [a,b; c,d] will result in x'=ax+by and y'=cx+dy. It is well known that such transformations have certain axes that are special as points on these axes are transformed as points along the axes only. These axes represent the eigenvectors of the wide applications in applied mathematics, computer graphics, soft computing, and control systems. We will limit our discussion to a simple example showing the evolution of eigenvalues and vectors. Consider A=[1 2; 3 4]. Its eigenvalues can be calculated with spec( ) function in Scilab

- A=[1 2; 3 4]
- spec(A)

  ```
  ans=  5.3722813
       -0.3722813
  ```

The eigenvectors corresponding to the eigenvalues can be computed by the same command by specifying variables for storing both.

- [x, y]=spec(A)

```
y=  5.3722813      0
         0          -0.3722813
x=   0.4159736     -0.8245648
         0.9093767     0.5657675
```

We find that eigenvalues appear as diagonal elements in the variable y. The eigenvectors appear as columns in the matrix x.