Vectors and Matrices MATLAB (Part-3)

P. Sam Johnson

November 7, 2014

In MatLab, a matrix is a rectangular array of real or complex numbers. Matrices with only one row or with only one column are called row and column vectors, respectively. A matrix having only one element is called a scalar.

Although other high-level programming languages work with one number at a time, in MatLab it is possible to work with the complex matrix simultaneously. This feature is very important as it removes the unnecessary loops and repetition of same statements.

In MatLab, matrix is chosen as a basic data element. All variables when used as a single data element are treated as single element matrix, that is a matrix with one row and one column.

Scalars and Vectors

A scalar is a 1×1 matrix containing a single element only. A column vector is a $m \times 1$ matrix that has m number of rows but a single column only.

A row vector is a $1 \times n$ matrix which has n number of columns and has only one row.

Assigning data to elements of a vector / scalar. All the elements of a **row vector** are separated by blank spaces or commas and are enclosed in square brackets.

All the elements of a **column vector** are separated by semicolons and are enclosed in square brackets.

A **scalar** does not need any square brackets, for example, a scalar $p = 10$, can be entered just as

```
p = 10;
```

Vector Product.

A row vector and a column vector can be multiplied with each other. However, these vectors should be of the same length. If x is a column vector and y is a row vector, then the vector products $y * x$ and $x * y$ are different.

Vector Transpose.

Transpose operation turns a row vector into a column vector and vice versa. MatLab uses the apostrophe or single quote to denote transpose.

The command

$$xt = x'$$

will produce the transpose of the column vector x.

If x is a vector with complex entries, its conjugate can be obtained by using `conj(x)` command and the x' produces the **conjugate transpose** of the vector x.

Creation of Evenly Spaced Row Vectors.

A row vector with evenly spaced, real elements can be created by using the following command.

```
variable name = k :  1 :  m,
```

where k is the initial value of the row vector, m is the final value of the row vector and 1 is the step size or increment which comes in between initial and final values.

- **In MatLab, negative and non-integer step sizes are also allowed.**
- **If a step size is not specified, $+1$ is taken as default value of the step size.**

For example, the command

```
t = 1 :  2 :  11
```

gives

```
t = [1 3 5 7 9 11]
```

The command

```
t = 11 :  1 :  5
```

will produce **empty matrix**. Care should be taken that the initial and final values given must be appropriate.

A row vector can also be created by using `linspace` command. The syntax is given as follows:

`linspace`$(x1, x2, n)$

It generates a linearly spaced vector of length n, that is n equally spaced points between $x1$ and $x2$.

For example, the statement

`linspace`$(0, 10, 5)$

will produce

```
a = [0   2.5   5.0   7.5   10.0]
```

The command

```
logspace(a, b, n)
```

creates a logarithmically spaced vector of length n in the interval 10^a to 10^b.

For example, the statement

```
x = logspace(0, 4, 3)
```

will produce

```
x = [1    100    10000]
```

i.e. three points are generated in the interval 10^0 and 10^4.

Some useful commands.

Let x be a row or a column vector.

`sum(`x`)`	the sum of all elements in the row/column
`mean(`x`)`	the average of all elements in the row/column
`length(`x`)`	the number of elements in the row/column
`max(`x`)`	the maximum value in the row/column
`min(`x`)`	the minimum value in the row/column
`prod(`x`)`	the product of all elements in the row/column

The statement

`find(x)`

gives the locations of non-zero entries of the row/column vector.

For example, the statements

`x = [-1 3 0 11 0]` and `find(x)`

give the output

```
x = 1 2 4
```

because x has non-zero entries in fist, second and fourth positions.

The command

```
a=find(x > 3)
```

gives the output

```
a = 4
```

that is, fourth element of x has a value greater than 3.

For the vector

```
z = [1.4   10.7   -1.1   20.9],
```

command	description
`fix(z)`	rounds the elements of z
	to the nearest integers towards zero
`a= fix(z)`	`a = [1  10  -1  20]`
`floor(z)`	rounds the elements of z
	to the nearest integers towards $-\infty$
`a= floor(z)`	`a = [1  10  -2  20]`
`ceil(z)`	rounds the elements of z
	to the nearest integers towards $+\infty$
`a= ceil(z)`	`a = [2  11  -1  21]`
`round(z)`	rounds the elements of z
	to the nearest integers
`a= round(z)`	`a = [1  11  -1  21]`

The command

```
sort(z)
```

lists the elements of z in the ascending order.

The command

```
sort(z, 'descend')
```

lists the elements of z in the descending order.

Entering Data in Matrices

The following points are to be noted for entering an explicit list of matrix elements at the command window:

- Start with a left-handed square bracket '['.
- Type the matrix elements row wise. Each row vector should have the same length to create a valid matrix.
- Enter the elements of the first row with one or more blanks or a comma separating each element.
- Separate the different rows of the matrix by a semicolon (;) or a carriage return.
- Enclose the entire list of elements of the matrix with in square bracket, that is [ ].
- Elements of a matrix can be a real number, complex number or a valid MatLab expression (provided that the values of unknown variables in the expression are supplied before the execution of the expression).

```
A = [1 3 5 ; 7 9 11 ; 2 -9 0]
```

has three rows and three columns.

```
B = [1+2i 3i ; 4+4i 5]
```

has two rows and two columns and whose elements are complex numbers.

```
C = [1 -2 ; sqrt(3) exp(1)]
```

has two rows and two columns and whose elements are expressions

Line Continuation.

Sometimes, it is not possible to type the entire data input on the same line. Thus, to enter the data in continuity two methods are used:

1. Using `ellipsis`. In this method, three consecutive periods (. . .) are used to indicate continuation of the data to the next line. There periods are called `ellipsis`.
2. Using `carriage return`. A matrix can also be entered multiple lines using `carriage return` at the end of the each row. In this case, the semicolons and ellipses at the end of each row may be omitted.

Matrix Subscripts / Indices

The elements of a matrix can be specified / accessed by specifying their respective row and column numbers. The first index refers to the row number and second index refers to column number. The element in ith row and the jth column of a matrix A is denoted by $A(i, j)$.

If an attempt is made to access an element outside of the given dimension of matrix A, it will give an error.

For example, if matrix A has dimension 3×3 and an attempt is made to access fourth row, say, element $A(2, 3)$, then an error message:

```
'index exceeds matrix dimensions'
```

will be displayed, in the command window.

The command $A(i, j)$ is also used to expand the given matrix size.

Suppose

```
A=[6 7 ; 8 9]
```

and

$$A(2, 3) = 15$$

will result in the following change in the matrix A:

```
A=

    6     7     0

    8     9    10
```

The dimension of the matrix is extended to 2×3, element 15 is stored at the index $(2, 3)$ and remaining undefined elements are taken as zero.

Sub Matrices

Subsets of arrays can be selected by the following statements.

Statement `B(1:4; 2:8)` specifies rows 1 to 4 and columns 2 to 8 of the matrix B.

The colon when typed alone refers to all the elements in a row or column of a matrix.

Thus $A(:, 2:3)$ refers to elements of all rows and columns 2 to 3 of the matrix A.

Array as a single column vector.

An array can be regarded as one long column vector, which is formed from the columns of a given matrix.

The matrix element $A(i, j)$ can also be accessed by means of a single index $A(k)$, where $k = (j-1)m + i$ and m is the number of rows.

The keyword '`end`' refers to the last row or column. The command

```
A(:, end)
```

refers to all rows and last column of the matrix A.

Multi-dimensional matrices and arrays

To know the size of an existing matrix A, the command `size(A)`, can be used. The format is as follows :

```
size(A)
```

The command returns the number of elements of the matrix in each dimension.

The following command

```
[m,n]=size(A)
```

will assign the number of rows of the matrix A to variable 'm' and number of columns to variable 'n'.

Multi-dimensional arrays are an extension of two-dimensional matrices described by their row and column indices. Multi-dimensional arrays use additiional indices.

A three-dimensional array, for example, uses three indices; first index refers to rows, second refers to column and third refers to the third dimension.

For example, a book can be viewed as a collection of two-dimensional pages with the page number being the third dimension.

To access the element in the third row, fourth column of page 5, the indices will be $(3, 4, 5)$. As the dimensions of an array are increased, the number of indicies is also increased correspondingly. For example, a five-dimensional array will require five indices. The first two refer to a row and column, the next three for third, fourth and fifth dimension, respectively.

Matrix Manipulations

The shape and size of a matrix can be altered by using the following MatLab commands:

Reshaping Matrices

Matrices can be reshaped into a vector or into any other appropriately sized matrix as described below:

1. **Reshaping of a matrix as a vector.** All the elements of a matrix A can be grouped into a single column vector b by the command `b=A(:)`. Matrix A will be stored column wise in vector b, i.e. first column is stored first, then second column and so on.
2. **Reshaping of matrix as a differently sized matrix.** If a given matrix A is a $p \times q$ matrix, it can be reshaped into a new $m \times n$ matrix B, as long as total elements of the two matrices are same. i.e. $p * q = m * n$. The reshaping command is as follows:

   `B=reshape(A,m,n)`.

Expanding Matrix Size

When a single element or a few elements are entered in a matrix, MatLab creates the matrix of proper dimensions to accommodate the entered element(s). Remaining unspecified elements are assumed to be zero.

For example, suppose a matrix C does not exist and a statement

```
C(2, 2) = 10
```

is entered at the command window. In such a case, matrix C of size (2×2) is generated with element $C(2, 2) = 10$ and with all other elements equal to zero as shown below.

```
C =

     0     0

     0    10
```

Similarly, the command

```
D(2, 1:2) = [3  4]
```

produces

```
D

   0   0

   3   4
```

Appending a row / column to a matrix.

Sometimes, it is required that a column or a row to be added to a given matrix. A column can be appended by using following command :

```
A=[A  x]
```

It will append the column vector x to the columns of existing matrix A.

To append a row to a given matrix, the following command is used:

```
A=[A ; y]
```

This will append the row vector y to the rows of existing matrix A. Note that semicolon has been used to append the row vector.

Deleting a row / column of a matrix. Rows and columns of a matrix can be deleted by setting the corresponding row or column vector equal to a null vector. That is, a pair of empty square brackets [ ], without any element within the brackets.

To delete the first row from matrix F and first two columns from matrix G, the following commands are used.

```
F(1, :)=[ ]
```

```
G(:, 1:2)=[ ]
```

Error

To delete the particular element, the following command is given:

```
A(1,2)=[ ]
```

But this will result in an error because a single element. That is, $A(1,2)$ cannot be removed from a matrix and following message will be displayed:

```
??  Indexed empty matrix assignment is not allowed.
```

Concatenation of matrices

Concatenation is a method of combining or connecting or joining small matrices together to make larger or bigger matrices. The pair of square brackets [ ] is the concatenation operator.

Let matrix A be given by $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. We can commute the matrix

$$B = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$$

where the sub-matrices $A_1 = A, A_2 = A + 12, A_3 = A + 24, A_4 = A + 10$.

All this can be done using the following statements:

```
A = [1  2 ; 3  4];

B = [A  A+12 ; A+24  A+10]
```

Generation of special matrices

The following MatLab commands / functions, which generate special matrices, (like zero matrix, matrix with all entries are one and the identity matrix) are often used in engineering computations involving matrices:

```
A = zeros(2,3)
```

will generate a 2×3 matrix with all elements equal to zero.

```
A = zeros(10)
```

will generate a 10×10 square matrix with all elements equal to zero.

```
A = ones(2,3)
```

will generate a 2×3 matrix with all elements equal to one.

```
A = ones(10)
```

will generate a 10×10 square matrix with all elements equal to one.

```
A = eye(2,3)
```

will generate a 2×3 matrix with all main diagonal elements equal to one.

```
A = eye(10)
```

will generate a 10×10 square matrix with all main diagonal elements equal to zero.

Vandermonde Matrix. The command

```
V = vander(v)
```

returns Vandermonde matrix V whose columns are powers of the vector

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}.$$

Vandermonde matrix is an $m \times n$ matrix with monomial terms of a geometric progression in each row as shown below:

$$v = \begin{pmatrix} v_1^{n-1} & \cdots & v_1^2 & v_1 & 1 \\ v_2^{n-1} & \cdots & v_2^2 & v_2 & 1 \\ \vdots & \cdots & \vdots & \vdots & \vdots \\ v_m^{n-1} & \cdots & v_m^2 & v_m & 1 \end{pmatrix}.$$

Let $v = [1\ 2\ 3]$. To generate a Vandermonde matrix, the following command is used:

```
v = [1   2   3]

vander(v)
```

If v is a row or column vector with n elements, it returns a diagonal matrix, with vector v on the diagonal. The syntax for this command is

```
A = diag(v, k)
```

This will return a square matrix A, of order '$n + abs(k)$' with the elements of vector v on the $k$the diagonal.

When $k = 0$, the vector v will come in main diagonals of A. We may simply use

```
A = diag(v).
```

When $k = 1$, the vector v will come in 1st upper main diagonals of A.

When $k = -1$, the vector v will come in 1st lower main diagonals of A.

Let x be an $m \times n$ matrix, then the command

```
A = diag(x)
```

will return a column vector formed from the elements on the main diagonal of the matrix x.

```
A = diag(x, k)
```

will return a column vector formed from the elements of the kth diagonal of the matrix x.

Some useful commands related to matrices

Let A be a matrix.

`det(`A`)`	the determinant of a given square matrix A
`rank(`A`)`	the rank of a given rectangular matrix A
`trace(`A`)`	the sum of the diagonal elements of a rectangular matrix A
`inv(`A`)`	the inverse of non-singular square matrix A
`norm(`A`)`	the Euclidean norm of a given rectangular matrix A
A'`(`A `transpose)`	the transpose of a given rectangular matrix A

Some useful commands related to matrices

Let A be a matrix.

$x = A\backslash b$ (Left Division)	the solution vector x of a linear system $Ax = b$

`poly(`A`)`
`p = poly(`A`)`
gives `p = 1 -5 4`
	the coefficients of the characteristic polynomial of a given square matrix A
	The characteristic polynomial is $s^2 - 5s + 4$.

`eig(`A`)`	the eigenvalues of A
`[v, x] = eig(A)`	returns the eigenvectors v (columns of v); eigenvalues are main diagonal of matrix x
`eigs(`A`)`	returns six largest magnitude eigenvalues of A
`org(`A`)`	orthogonalizes the rectangular matrix A

References

- Misza Kalechman, "*Practical MATLAB Basics for Engineers*", CRC Press, London, 2009.
- Raj Kumar Bansal, Ashok Kumar Goel and Manoj Kumar Sharma, "*MATLAB and its applications in engineering*", Pearson, New Delhi, 2009.