PART II : MATRICES

MATRICES

Create a matrix:

The entries are written, using brackets

- *The rows of a matrix are separated by a semicolon*
- *the entries of each row are separated by an empty space or a comma*

```
>> A = [1,2,3,4;5,6,7,8;9,10,11,12]
A =
        1    2    3    4
        5    6    7    8
        9   10   11   12
```

Transpose a matrix:

```
>> B = A'
B =
        1    5    9
        2    6   10
        3    7   11
        4    8   12
```

SPECIAL FUNCTIONS GENERATING MATRICES

- eye(n) : returns the identity matrix of size *n-by-n*
- zeros(n) : returns a square *n-by-n* matrix whose entries are zeros *(often used to preallocate a matrix)*
- zeros(m,n) : returns an *m-by-n* matrix whose elements are zeros *(often used to preallocate a matrix)*
- ones(n) : returns a square *n-by-n* matrix whose elements are ones
- ones(m,n) : returns an *m-by-n* matrix whose elements are all ones.

- rand(m,n) : generates an *m-by-n* matrix, where all entries are pseudo-random numbers in [0,1).

  rand(n) : generates an *n-by-n* matrix of pseudo-random numbers in [0,1).

  *fix(rand(n)*10) and fix(rand(m,n)*10) : generate matrices, where all entries are pseudo-random integers ranging between 0 and 9.*

- magic(n) : generates a square *n-by-n* matrix (n>2), where the *n* elements in all rows, all columns, and both diagonals sum to the same (constant sum = $n(n^2+1)/2$).

  N.B: magic matrices of order n, where n is odd, are invertible.

- hilb(n) : generates a square *n-by-n* Hilbert matrix, with entries : ***H(i,j) = 1/(i+j-1).***

- pascal(n) : generates a square n-by-n Pascal matrix with integer entries taken from Pascal’s triangle.

  (entries of the inverse matrix are also integers).

  N.B: Hilbert and Pascal matrices are symmetric and positive definite, therefore invertible.

- triu(A,k) : returns all elements on and above the kth diagonal of a square matrix A and assigns the value 0 to the remaining elements of A.

  triu(A) or triu(A,0) : returns the upper triangular part of a square matrix A and assigns the value 0 to the remaining elements of the A.

- tril(A,k) : returns all elements on and below the kth diagonal of a square matrix A and assigns the value 0 to the remaining elements of A.

  tril(A) or tril(A,0): returns the lower triangular part of a square matrix A and assigns the value 0 to the remaining elements of A.

- diag(v,k) : returns a square matrix with the elements of the vector v on its kth diagonal and assigns the value 0 to the remaining elements of A.*(k is a positive integer for diagonals above the main one, or < 0 below it)*

  diag(v) or diag(v,0) : returns a square diagonal matrix with the elements of the vector v on its main diagonal.

.

SPECIAL FUNCTIONS ON MATRICES

- **size(A)** : returns the sizes of each dimension of the matrix A, in a vector.

  [m,n] = size(A) returns the size of the matrix A in separate variables m and n.

- **sum(A)** : returns a row vector with the sum over each column of A.

  sum(A') : returns a row vector with the sum over each row of A.

  sum(sum(A)) returns the sum of all the elements of A.

- **max(A)** : returns a row vector with the maximum element from each column of A.

  max(A') returns a row vector with the maximum element from each row of A.

  max(A')' returns a column vector with the maximum element from each row of A.

  max(max(A)) returns the maximum element of A.

  [M,I] = max(A) returns the maximum element of each column of A, and its row index, in respectively 2 row vectors M and I.

  [m,ik]=max(A(:,j)) : returns the maximum value m and the row index ik, of the j^{th} column.

  [m,jk]= max(A(i,:)) : returns the maximum value m and the column index jk, of the i^{th} row.

- **diag(A,k)** : returns a column vector formed from the elements of the k^{th} diagonal of a square matrix A. *(k is a positive integer for diagonals located above the main one, or < 0 below it)*

  diag(A) or **diag(A,0)** : returns a column vector formed from the elements of the main diagonal of A.

HOW TO ACCESS SUBMATRICES

The colon notation is used to access submatrices of a matrix.
A colon by itself denotes an entire row or column.

Examples:

- **A(1:4,3)** is the column vector consisting of the first four entries of the third column of A.
- **v = A(: ,3)** returns a column vector equal to the third column of A.
- **v = A(3, :)** returns a row vector equal to the third row of A.
- **B = A(1:4, :)** returns a matrix made of the first four rows of A.
- **B = A (:, [2 4])** returns a matrix containing 2 columns: columns 2 and 4 of A.

Example: Row & Column Permutations

<u>Permute</u>: two rows or two columns

A =	>> A([1 2],:)=A([2 1],:)	>>A(:,[1 2])=A(:,[2 1])
	A =	A =
1 5 4	8 3 9	5 1 4
8 3 9	1 5 4	3 8 9
1 1 -5	1 1 -5	1 1 -5
2 -3 6	2 -3 6	-3 2 6

Let A be an *m-by-n* matrix

- If the index vector IV is a permutation of the integers {1,2,…,m}, then **A = A(IV,:)** <u>permutes the rows of A, in the order given by IV</u>
- If the index vector IV is a permutation of the integers {1,2,…,n}, then **A = A(:, IV)** <u>permutes the columns of A, in the order given by IV</u>

B =	>> C=B([2 1 3 4],:)	>> C=B(:,[2 1 3])
1 5 4	C =	C =
8 3 9	8 3 9	5 1 4
1 1 -5	1 5 4	3 8 9
2 -3 6	1 1 -5	1 1 -5
	2 -3 6	-3 2 6