Practice 2

1. Working with Arrays and Matrices

Matrices are the basic type of data in Matlab.

1.1. Generating Vectors

Función	**Salida**
rand	uniformly distributed random numbers in the interval $(0,1)$
randn	Normally distributed random numbers
[a:step:b]	creates a regularly-spaced vector x using i as the increment between elements $(a, a+\text{ step}, a+2\text{ step}, \ldots, a+n\text{ step})$, where n is the greatest integer such as $a+n\text{ step} \in [a,b]$ if step > 0 and $a+n\text{ step} \in [b,a]$ if step < 0
linspace(a,b,n)	generates a row vector containing n evenly spaced points in $[a,b]$
linspace(a,b)	returns a row vector of 100 evenly spaced points between a and b

Generation of random arrays:

```
>> v=rand(1,10);
>> length(v);
>> v=[v rand(1)];
>> v(5)=0
```

Creating equally spaced arrays:

```
>> v=[0:2:50]
>> w=[1:25]'
>> u1=linspace(5,15,10)
>> u2=linspace(0,1)
```

We can do the same with `randn` (Normal ditribution).

Example 1 *Generate a population of* 10000 *individuals with a height corresponding to a normal distribution with mean* $= 175$*cm and standard deviation* 15*cm. Finding how many of them are higher than* 200*cm*

```
>> N=10000;
>> mu=175;
>> dst=15;
```

```
>> hlim=200;
>> heigth=mu+dst*randn(1,N);
>> altos=heigth>=hlim;
>> Naltos=sum(altos);
>> Paltos=Naltos/N*100;
```

1.2. Generating Matrices

Matrices are created by entering elements in each row as comma or space delimited numbers and using semicolons to separate the rows.

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

Another way to create a matrix is to use functions, such as the ones in the following table.

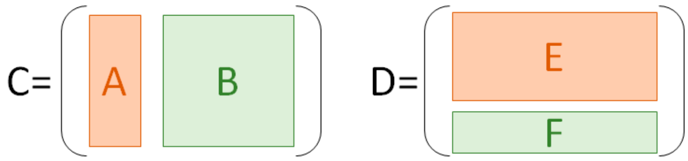
Functionn	**Output**
ones(n)	An n-by-n matrix of ones.
ones(m,n)	An m-by-n matrix of ones.
zeros(n)	An n-by-n matrix of zeros.
zeros(m,n)	An m-by-n matrix of zeros.
eye(n)	An n-by-n identity matrix.
eye(m,n)	An m-by-n matrix with ones on the main diagonal and zeros elsewhere.
diag(v)	A square diagonal matrix with the elements of vector v on the main diagonal.

Referencing and modifying the elements of a matrix:

Función	**Salida**
v(i)	Element i of vector v
A(i,j)	Element i,j of the matrix A
A(:,j)	Column j of matrix A
A(:,end)	Last column of A
A(i,:)	Row i of A
A(end,:)	last row of A
A(v,w)	The submatrix of A that contains the rows indicated by v and the columns indicated by w
A(:)	Transforms A into a column matrix by concatenating the columns of A
A(i,:)=[ ]	Deletes the row i of A
A(:,j)=[ ]	Delets the column j of A
	Nota: [ ] is the (*empty matrix*)

You can concatenate two matrices to create a larger matrix. The pair of square brackets ′[]′ is the concatenation operator. Two types of concatenations are possible:

- Horizontal concatenation
- Vertical concatenation



Example 2

```
>> A=ones(3), B=eye(3,2), C=[A B]
>> E=zeros(2,3), F=ones(3), D=[A;B]
```

Basic Operations:

Operation	Result
A+B	Adds A and B
A-B	Subtracts B from A
A*B	Matrix product of A and B
`A\B`	Compute $A^{-1}B$
λ*A	Multiply λ by all elements of A.
`A^n`	A to the nth power
A.'	Transpose of A
A'	Complex conjugate transpose of A

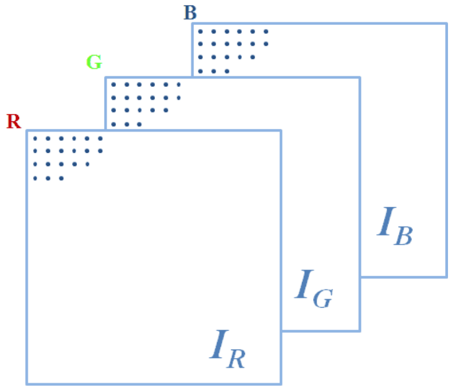
```
>> m=10;
>> n=5;
>> A=rand(m,n);
>> size(A)
% adding one row
>> A=[A;rand(1,size(A,2))];
% adding one column
>> A=[A, rand(size(A,1),1)];
% plot A
>> imagesc(A)
% Put to zero the elements lower than 0.5
>> A(A<0.5)=0;
>> imagesc(A)
% transforming A into a column vector
>> Ac=A(:);
```

1.3. Example of matrices: Images in Matlab

Exercise 1 *Download two images from the web, a grey scale one and a color one, and:*

a) Read the grey-scale image using the order `imread` *and save it into the variable* $I1$. *Visualize* $I1$ *in Matlab with imshow.*

b) Read the color image $'playa.jpg'$ *and save it into the variable* $I2$. *What type of variable is it? Visualize* $I2$. *Save the red channel into a variableIR and compare the three images.*



1.4. Matlab Control Structures *for*

for executes a group of statements in a loop for a specified number of times.

```
%% For loop
for index=value1:value2
    statement1
    statement2
    ...
end
```

Example 3 *Read and visualize the image 'moon.tif' in Matlab. Create a new image by repeating the initial image* 10 *times, horizontally. Visualize it a each step.*

```
>  I=imread('moon.png')
>  size(I)
>  imshow(I)
>  imshow(I')% shows the traspose image
>  ik=I; % the image to repeat at each step
>  for k=1:10
>  ik=[ik I];
```

```
>>     imshow(ik)
>>     pause(1)
>> end
```

2. Vectorial Spaces

Example 4 *Consider 3 row vectors in $\mathbb{R}^4$. Create a matrix R with the vectors as rows and a matriz C with the vectors as columns. Compute the rank of R and C. Compute the reduced echelon form of C.*

```
>> v1=[1,0,1,2];
>> v2=[2,1,1,1];
>> v3=[1,1,0,-1];
>> R=[v1; v2; v3];
>> C=[v1(:) v2(:) v3(:)]; % BTW C is R'
>> rank(R)
>> rank(C)
% Row Reduction Echelon Form:
>> RC=rref(C); % we see that the rank is 2
```

Example 5 *Checking the independence of 3 matrices and finding the coordinates of D in the span of A,B,C.*

```
>> A=rand(3,2);
B=rand(3,2);
C=rand(3,2);
P=[A(:) B(:) C(:)];
rank(P);
% Coordinates
D=3*A-B+C;
% let us find the coordinates of D in basis {A,B,C}
% c1*A(:)+c2*B(:)+c3*C(:)=D(:)
% P*coor=D(:)
Pa=[P,D(:)];
R=rref(Pa);
coor=R(1:size(P,2),end);
```

Example 6 *Check the independence of the polynomials $p(x) = 3x^2 + 2x + 1$ and $q(x) = x^2 - 2x + 1$ in $\mathbb{R}_2[X]$ two ways:*

- *Checking the independence of their coordinates*

- *Using the definition of independence for functions*

```
>> p=[3,2,1]; % the coefficients of the pol. in decreasing order - coordina
>> q=[1,-2,1];
>> A=[p;q];
>> rank(A)% the rank(A) tells us if p and q are independent
% a*p(x)+b*q(x)=0(x)
>> r1=[polyval(p,0), polyval(q,0)];
>> r2=[polyval(p,1), polyval(q,1)];
>> r3=[polyval(p,-1), polyval(q,-1)];
>> B=[r1;r2;r3]
>> rank(B)
```