Vectores

- Un vector es una matriz de 1 x N (vector fila) o de N x 1 (vector columna)

- Dar valores a un vector

  - `v = [ 1 2 3 4 5]`
  - `v = [ 1,2,3,4,5]`

  } vector fila

  - `v = [ 1 ; 2 ; 3; 4; 5]` vector columna

Vectores

- Lectura y asignación de posiciones específicas.

```
>> v = [0  2  4  6  8 10]
>> v(1)
ans =   0
>> v(5)
ans = 8
>> v(1) = -4
v =  -4     2     4     6     8    10
>> v(2) = v(1)
v =  -4    -4     4     6     8    10
>> v(5) = v(4) + v(3)
v =  -4    -4     4     6    10    10
```

Cada v(i) es una variable

Funciones exclusivas sobre vectores

Siendo `x,y` vectores:

- `length(x)`: dimensión del vector.
- `max(x)`: máximo elemento de un vector.
- `min(x)`: mínimo elemento de un vector.
- `sum(x)`: suma de los elementos de un vector
- `cumsum(x)`: devuelve un vector con la suma acumulativa de los elementos de x
- `mean(x)`: valor medio de los elementos de un vector
- `std(x)`: desviación típica
- `prod(x)`: producto de los elementos de un vector
- `cumprod(x)`: devuelve el vector producto acumulativo de los elementos de un vector
- `sort(x)`: ordena de menor a mayor de los elementos de un vector x.
- `dot(x,y)`, `cross(x,y)`: producto escalar y vectorial

Ejemplo

```
>> w = [1      2      3      4      5]

>> sum(w)
ans = 15

>> mean(w)
ans = 3

>> std(w)
ans = 1.5811

>> cumprod(w)
ans = 1     2     6     24    120

>> cumsum(w)
ans = 1     3     6     10     15
```

Operaciones posibles

Siendo `v = [1 2 3],  w = [4 5 6]`

- suma de vectores

```
>> v + w
ans = 5      7      9
```

- resta de vectores

```
>> v - w
ans = -3    -3    -3
```

- transposición (obtener un vector columna)

```
>> v'
ans =
      1
      2
      3
```

Operaciones posibles

Siendo `v = [1 2 3],  w = [4 5 6]`

- multiplicación

```
>> v * w
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> v * w'
ans = 32
```

- multiplicación por escalar

```
>> w * 3
ans =  12    15    18
```

- suma/resta de un escalar

```
>> v + 3
ans =  4    5    6
```

Operaciones posibles

Siendo `v = [1 2 3],  w = [4 5 6]`

- .^ elevar a una potencia elemento a elemento

```
>> v .^2
ans =  1     4     9
```

- .* producto elemento a elemento

```
>> v .* w
ans =  4    10    18
```

- ./ división elemento a elemento

```
>> w ./ v
ans = 4.0000    2.5000    2.0000
```

Vectores

- Generación de vectores a partir de rangos
  - `v = [1:4] -> [1 2 3 4]`
  - `v = [-2:2] -> [-2 -1 0 1 2]`

operador de "rango"

- Mas interesante:
  - `v = [0:2:10] -> [0 2 4 6 8 10]`

valor inicial

incre-mento

valor final

  - `v = [-pi:pi/4:pi] -> [-3.1416 -2.3562 -1.5708 -0.7854 0  0.7854 1.5708 2.3562 3.1416]`
  - `v = [10:-2:2] -> [10 8 6 4 2]`
  - `w = [0:0.2:1] -> [0 0.20 0.40 0.60 0.80  1.0]`

Vectores

Más funciones de creación de vectores

- rand(1,n): genera un vector fila de n elementos con números aleatorios entre 0 y 1

```
>> rand(1,5)
ans = 0.8147    0.9058    0.1270    0.9134    0.6324
>> min = -5; max = 5;
>> v = min + rand(1,5)*(max-min)
```

cuanto vale v ???

- randperm(n): genera un vector fila de n elementos que son una permutación aleatoria de los enteros entre 1 y n

```
>> randperm(5)
ans = 1     4     5     3     2
>> randperm(7)
ans = 7     1     2     6     4     3     5
```

Vectores

Función logspace

- logspace(a, b, n) genera un vector fila de n puntos con espaciado logarítmico entre 10^a y 10^b

```
>> logspace(0,1,5)
  1.000000000000000   1.584893192461114
  2.511886431509580   3.981071705534972
  6.309573444801933  10.000000000000000
```

- Si se omite n, se genera un vector fila de 50 *puntos*

Vectores

Es posible operar sobre subvectores

```
>> v = [0:2:10]
v =  0     2     4     6     8    10
```

Asigno un valor igual a un rango del vector

```
>> v(1:3) = 1
v =  1     1     1     6    8    10
```

Asigno valores distintos

```
>> v(1:3) = [5 4 4*5]
v =  5     4    20     6    8    10
```

Extraigo un subvector y lo guardo en una variable

```
>> vec = v(1:3)
vec =  5     4    20
```

Ejercicios

- Crear un vector con los primeros 10 numeros naturales
- Obtener los elementos que se encuentran en las posiciones impares
- Sumar los elementos que se encuentran en las posiciones pares
- Mostrar el vector en orden inverso
- Poner a cero los elementos de las posiciones pares
- Multiplicar por 3 los elementos de las posiciones pares
- Dados dos vectores X, Y como haría para calcular

$$\sum_{i=1}^{n} (x_i - y_i)^2 \qquad \sum_{i=1}^{n} (x_i * y_i)$$

Funciones sobre vectores

Un aspecto muy potente de MATLAB es que permite hacer cosas como las siguientes:

```
>> x = [-pi:pi/3:pi];
>> sin(x)
   -0.000 -0.866 -0.866 0 0.866  0.866   0.000
```

Hemos pasado un vector como parámetro de una función.

```
>>  v =[ 8.0607 3.7785 5.1798  0.9460  9.0910]
>> ceil(v)
ans =  9     4     6     1    10

>> floor(v)
ans =  8     3     5     0     9
```

Funciones sobre vectores

```
>> v1 = [8.0607   3.7785   5.1798   0.9460   9.0910]
>> v2 = [2.0763   3.8206   6.6028   7.5837   1.7307]
>> sqrt(abs(v2-v1))
ans = 2.4463   0.2053   1.1929   2.5764   2.7130

>> A  = [2     4     6     8    10]
>> B = [ 12    14    16    18    20]
>> mod(B,A)
ans =   0     2     4     2     0

>> a = 2;b = -1; c = 0.25;
>> x = [1:1:10];
>> y = a * x.^2 + b * x + c

y = 1.25 6.25  15.25 28.25 45.25 66.25  91.25 120.25
  153.25  190.25
```

Matrices

- Lo que hemos visto para vectores, también vale para matrices.
- Inicialización

```
>> A = [1 2 3; 4 5 6; 7 8 9]  (cada fila es un vector)
A =  1     2     3
     4     5     6
     7     8     9

>> B = [ 1:1:4 ; -1:2]
B =  1     2     3     4
    -1     0     1     2

>> C = [ x ; sin(x)]
```

```
C =
  -3.1416  -2.1416 -1.1416 -0.1416  0.8584  1.8584   2.8584
  -0.0000  -0.8415 -0.9093 -0.1411  0.7568  0.9589   0.2794
```

Acceder a una matriz

- **`A(i,j)`** se refiere al elemento (i, j) de la matriz A, donde i es la fila, j la columna (i, j >= 1)

  ```
  A(1, 2) = 7 % guarda un 7 en la posición (1, 2) de la matriz A
  ```

- **`A(i,:)`** la fila i-ésima de la matriz A

  ```
  A(2, :) = v %sustituye la 2da fila de A por  v
  ```

- **`A(:,j)`** la columna j-ésima de la matriz A

  ```
  A(:, 3) = w  % sustituye la 3ra columna de A por w
  ```

Acceder a una matriz

- **`A([I1 I2 I3 …],:)`** se corresponde con las filas I_1 I_2 I_3,… de la matriz A

```
A([2 3], :)% submatriz con las filas 2 y 3 de
  la matriz A
```

- **`A(:,[j1 j2 j3 …])`** se corresponde con las columnas j_1 j_2 j_3 de la matriz A

```
A(: , [1 2 5])  % submatriz con las columnas
  1, 2 y 5 de la matriz A
```

- **`A([Ia:Ib],[jc:jd ])`** se corresponde con la submatriz que contiene desde la fila I_a hasta la I_b, y desde la columna $\mathbf{j_c}$ a la columna $\mathbf{j_d}$ de la matriz A

```
A(2:4, 3:6)   % submatriz 3x4  con las filas
  2, 3 y 4, y columnas 3, 4, 5 y 6 de A
```

Ejercicios

¿Como haría para obtener la siguiente matriz?

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16
5	10	15	20
6	12	18	24
7	14	21	28
8	16	24	32
9	18	27	36

Indique que obtiene con los siguientes comandos

```
>> M([1:4],[2,3])
>> M([4,7,8],:)
>> v = [2,4,6,8];
>> M(v,:)
```

```
>> v = [1,3,5,7];
>> M(:,v)
>> M([1:2],:)'
>> M([4,7,8],:) = -1
>> M(1,:) = M(9,:)
>> temp = M(2,:); M(2,:) =
  M(3,:); M(3,:) = temp;
>> M([2:4],[2:4]) = 1
>> M(10,:) = M(10,:) * -1
>> M(:,1) + M(:,2)
>> sum(M(:,1) + M(:,2))
```

Matrices

Matrices especiales (I)

- **zeros(n)** genera una matriz *nxn* con todos los valores iguales a 0
  - A1 = zeros(2)
- **zeros(m, n)** genera una matriz *mxn* con todos los valores iguales a 0
  - A2 = zeros(3, 2)
- **ones(n)** genera una matriz *nxn* con todos los valores iguales a 1
  - B1 = ones(3)
- **ones(m, n)** genera una matriz *mxn* con todos los valores iguales a 1
  - B2 = ones(1, 3)
- **eye(n)** genera una matriz identidad *nxn*
  - C1 = eye(2)
- **eye(m, n)** genera una matriz identidad *mxn*
  - C2 = eye(2, 1)

Matrices

```
>> A1 = zeros(2)
A1 =
     0     0
     0     0

>> A2 = zeros(3,2)
A2 =
     0     0
     0     0
     0     0

>> B1 = ones(3)
B1 =
     1     1     1
     1     1     1
     1     1     1
```

```
>> B2 = ones(1,3)
B2 =
     1     1     1

>> C1 = eye(2)
C1 =
     1     0
     0     1

>> C2 = eye(2,1)
C2 =
     1
     0
```

Matrices

Matrices especiales (II)

- **rand(n)** genera una matriz *nxn* con valores aleatorios entre 0 y 1, siguiendo una distribución uniforme
  - A1 = rand(2)
- **rand(m, n)** genera una matriz *mxn* con valores aleatorios entre 0 y 1, siguiendo una distribución uniforme
  - A2 = rand(2, 4)
- **randn(n)** genera una matriz *nxn* con valores aleatorios entre 0 y 1, siguiendo una distribución normal (de media 0 y varianza 1)
  - B1 = randn(3)
- **randn(m, n)** genera una matriz *mxn* con valores aleatorios entre 0 y 1, siguiendo una distribución normal (de media 0 y varianza 1)
  - B2 = randn(3, 1)
- **diag(v)** genera una matriz diagonal con el vector v como diagonal
  - v = [1 2 3 4]
  - C = diag(v)

Matrices

```
>> A1 = rand(2)
A1 =
    0.1946    0.2929
    0.4175    0.7021

>> A2 = rand(2,4)
A2 =
  0.2397  0.3055  0.5555  0.4439
  0.9595  0.1549  0.7905  0.9958

>> B1 = randn(3)
B1 =
   -0.4326    0.2877    1.1892
   -1.6656   -1.1465   -0.0376
    0.1253    1.1909    0.3273
```

```
>> B2 = randn(3,1)
B2 =
    0.1746
   -0.1867
    0.7258

>> V = [1 2 3 4];
>> C = diag(V)
C =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

Mas posibilidades

```
>> M = [eye(2) ones(2); ones(2) eye(2)]
M =  1     0     1     1
     0     1     1     1
     1     1     1     0
     1     1     0     1
```

Operaciones con matrices (I)

- **A + B** es la suma de las matrices A y B, ambas de dimensión *mxn*
  - `A = [2 1 0; -4 1 1]`
  - `B = [3 2 -2; 1 2 0]`
  - `C = A + B    % C = [5 3 -2; -3 3 1]`
- **A - B** es la resta de las matrices A y B, ambas de dimensión *mxn*
  - `D = A – B    % D = [-1 -1 -2; -5 -1 1]`
- **k * A** es el producto de un escalar k por la matriz A
  - `E = 2 * A     % E = [4 2 0; -8 2 2]`
- **A / k** es la división de matriz A entre el escalar k
  - `F = A / 2    % F = [1  0.5  0; -2  0.5  0.5]`

Operaciones con matrices (II)

- **A * B** es el producto matricial A·B, siendo A una matriz de dimensión *mxn* y B una matriz de dimensión *nxp*
  - `A = [2 1 0; -4 1 1]`
  - `B = [3; 0; 1]`
  - `C = A * B                % C = [6 -11]`
- **A .* B** es el producto elemento a elemento de las matrices A y B de dimensión *mxn*
  - `D = [3 2 -2; 1 2 0]`
  - `E = A .* B          % D = [6 2 0; -4 2 0]`
- **A / B** es la división entre las matrices A y B, que equivale a $A \cdot B^{-1}$
- **A ./ B** es la división elemento a elemento de las matrices A y B de dimensión *mxn*

Operaciones con matrices (III)

- **A ^ n** es la potencia matricial n-ésima de A, es decir A*A*A*... *A (con n-1 operaciones *)
- **A .^ n** es la potencia n-ésima elemento a elemento de A
  - Si B = A.^3, entonces $B(i, j) = (A(i, j))^3$

Operaciones con matrices (IV)

- **size(A)** es la dimensión de la matriz A
  - `A = [2 3 4; 1 -1 0]`
  - `[m n] = size(A)          % m = 2, n = 3`
- **rank(A)** es el rango de la matriz A
  - `x = rank(A)              % x = 2`
- **det(A)** es el determinante de la matriz A
  - `B = [8 1 6; 3 5 7; 4 9 2]`
  - `y = det(B)               % y = -360`
- **trace(A)** es la traza de la matriz A, es decir, la suma de los elementos de su diagonal
  - `z = trace(B)                  % z = 15`

Operaciones con matrices (V)

- **A'** es la matriz transpuesta de la matriz A
  - `A = [2 1; 3 2]`
  - `B = A'       % B = [2 3; 1 2]`
- **inv(A)** es la matriz inversa de la matriz A
  - `C = [8 1 6; 3 5 7; 4 9 2]`
  - `D = inv(C)`
  - ```
    D = [0.1472  -0.1444  0.0639;
         -0.0611  0.0222  0.1056;
         -0.0194  0.1889  -0.1028]
    ```
    ```

Mas funciones

- Las funciones que se aplicaban sobre vectores, tambien se pueden aplicar sobre matrices. Como resultado se obtiene un vector con los resultados parciales referidos a cada columna.

```
M = 10     6     9
    10     1     5
     6     6     4

>> sum(M)
ans = 26    13    18
>> max(M)
ans = 10     6     9
>> min(M)
ans =  6     1     4
>> mean(M)
ans = 8.6667 4.3333 6.0000
```

```
>> sort(M)
ans =
      6     1     4
     10     6     5
     10     6     9
```

```
>> min(M')
>> max(M')
>> max(mean(M))
>> max(max(M))
>> max(min(M))
```

?

Ejercicios

Dada una matriz NOTAS de 100 x 10, donde NOTAS(i,j) contiene la calificación del alumno i en la asignatura j, se pide calcular:

- la nota media de cada alumno
- la media de todas las notas
- la nota mas alta obtenida en la asignatura 5
- idem la más baja
- la diferencia entre la nota media del alumno 23 y el 48
- elimine los alumnos desde la posicion 20 a la 40.

Ejercicios

- Calcular las funciones $f = 4 \sin(3x)$ $g = 3 \sin(4x)$ para

  $x = 0, 0.1, 0.2, \ldots, 0.5$
- La velocidad *v* y la distancia *d* en función del tiempo *t*, de un coche que posee una aceleración constante *a* desde un estado de reposo, se calculan como:

  $$v(t) = at \qquad d(t) = \tfrac{1}{2} at^2$$

  Determina la velocidad y la distancia recorrida por un coche para cada uno de los primeros 10 segundos, que tiene una aceleración $a = 1.55\ m/s^2$. Muestra los resultados en una tabla de 3 columnas donde la primera es el tiempo, la segunda la distancia y la tercera, la velocidad.
- ¿Como haría para determinar si dos matrices *m1, m2,* ambas de $N \times N$, son iguales (tienen los mismos valores)?

Ejercicios

El coeficiente de fricción μ, puede determinarse en un experimento midiendo la fuerza *F* requerida para mover una masa *m*. Cuando *F* se mide y m es conocida, entonces el coeficiente μ se calcula como

$$\mu = \frac{F}{mg}$$

siendo $g = 9.81\ m/s^2$

La siguiente tabla muestra resultados de medición de F en 6 tests.

Test	1	2	3	4	5	6
Masa m	2	4	5	10	20	50
Fuerza F	12.5	23.5	30	61	117	294

Calcule el coeficiente de fricción en cada test y luego el promedio sobre todos los test.