Errores de redondeo y aritmética del computador

Si p* es una aproximación de p, definimos dos tipos de errores:

- El error absoluto, que viene dado por $EA = | p^* - p |$
- El error relativo, que está dado por $ER = \frac{| p^* - p |}{| p |}$ siempre y cuando p sea distinto de cero.

Ejemplo:

$p = 0.3000 \times 10^{1}$

$p^* = 0.3100 \times 10^{1}$

$EA = 0.1$

$ER = 0.3333 \times 10^{-1}$

$p = 0.3000 \times 10^{-3}$

$p^* = 0.3100 \times 10^{-3}$

$EA = 0.1 \times 10^{-4}$

$ER = 0.3333 \times 10^{-1}$

Observaciones:

- el error relativo en ambos casos es el mismo, mientras los errores absolutos son diferentes
- como medida de precisión el error absoluto puede ser engañoso, en cambio el error relativo puede ser más significativo

Representación de números en el computador

Ejemplo:

Consideremos la siguiente suma:

$$0.99 + 0.0044 + 0.0042.$$

Con aritmética exacta, el resultado es 0.9986.

Sin embargo, si usamos una aritmética de tres dígitos, y las operaciones se realizan siguiendo el orden de izquierda a derecha, encontramos que

$$(0.99+0.0044)+0.0042 = 0.994+0.0042 = 0.998.$$

Por otra parte, si operamos primero los dos últimos números, tenemos que:

$$0.99+(0.0044+0.0042) = 0.99+0.0086 = 0.999,$$

lo cual demuestra el efecto del error por redondeo en un caso tan simple como éste.

(¿Por qué sucede esto?).

Desde el punto de vista numérico es importante el orden en que sumamos.

Representación de números en el computador

Operaciones que producen errores en Matlab:

- La suma de muchos números siempre genera errores de redondeo, lo importante es que estos errores no inutilicen el resultado.
- La suma de números de magnitudes distintas.
- la sustracción de números casi iguales.

Ejemplo:

10^20 - (10^20 +1) es igual a 0 en Matlab

y (10^20 - 10^20) -1 es igual a -1 en Matlab

aunque 10^20 - (10^20 +1) = (10^20 - 10^20) -1 = -1

Representación de números en el computador

Ejemplo:

Sabemos que 1.0 – 5 * 0.2 = 0, pero si lo calculamos en MATLAB

```
>> 1 -  0.2 - 0.2 - 0.2 - 0.2 - 0.2
ans =
5.5511e-017
```

Este resultado es bastante pequeño, pero no es cero.

La razón de esto es que el numero binario correspondiente a 0.2 es

0.00110011001100 …

Esta representación requiere un número infinito de dígitos. La consecuencia de esto es que el computador trabaja con una aproximación de 0.2, y restar a 1 una aproximación de 0.2 cinco veces no conduce a cero.

Nociones básicas

Para trabajar cómodamente, MATLAB permite guardar una secuencia de instrucciones en una archivo de texto cuya extensión debe ser .m , esto se puede hacer desde cualquier editor de texto.
MATLAB tiene su propio editor, el cual permite ver en distintos colores las diferentes instrucciones y comandos.

Ejemplo: archivo prueba.m

```
% Esto es un script
a=5;
b=6;
fprintf('El resultado es %d \n',a+b)
% fin archivo prueba.m
```

Para ejecutar este tipo de archivos, sólo es necesario hacer el llamado desde el espacio de trabajo.

Desde el espacio de trabajo hacer:

```
>> prueba
El resultado es 11
```

Estructuras de datos

Números

Los formatos de números admisibles en MATLAB son:

- Notación decimal estándar.

  2, +2, -105, 3.1416, -2.5

  Nota: MATLAB no hace una distinción entre enteros y reales (todos los números son reales de doble precisión). A pesar de ello, es buen estilo de programación distinguir entre 2 (entero) y 2.0 (real).

- Notación científica

  0.31416e+1, 0.000031416e5, 314.16e-2

- Números imaginarios

  3*i 3*j -3.1416*i 2.3e4*i

- Internamente, MATLAB almacena los números en formato de doble precisión (estándar IEEE):
  - Precisión: 16 dígitos significativos
  - Rango: $[10^{-308}, 10^{308}]$

Estructuras de datos

Variables

Una variable es una unidad de almacenamiento de datos, pueden ser: números reales, complejos, vectores, matrices, texto y estructuras.

- MATLAB no requiere la declaración del tipo de variable (integer, float, double, bool), ni especificar su dimensión.

  Nota: Esto es debido a que todas las variables son consideradas como matrices de tamaño variable.

- El nombre de una variable consiste en una letra, seguida por un número arbitrario de letras, dígitos o el carácter ”_”, de los cuales sólo se usan los 63 primeros caracteres (no se aceptan %, $, #, … etc.)
- MATLAB distingue mayúsculas y minúsculas para variables.
- Variables lógicas: Las variables de MATLAB pueden ser utilizadas como variables lógicas:
  - Un valor igual a 0 equivale a “False”.
  - Un valor distinto de 0 equivale a “True”.

Estructuras de datos

Variables y funciones predefinidas en Matlab

ans	Posee el último valor asignado, no asignado a variable
i	Unidad imaginaria (valor: sqrt(-1))
j	Unidad imaginaria
pi	Número π (3.1416)
eps	Precisión relativa de real (doble precisión) (valor: 2^{-52})
realmax	Valor máximo de un real (doble precisión) (valor: $(2-eps)*2^{1023}$)
realmin	Valor absoluto mínimo de un número real
Inf	Infinito (ejemplo: 1/0. MATLAB opera con infinitos)
NaN	Not-a-Number (ejemplo: 0/0. Operaciones con NaN generan NaN)

Estructuras de datos

Variables reales

Se almacenan con una representación de punto flotante. Estas obtienen su valor mediante la asignación de un valor a la variable.

Ejemplo:
>> a = 5, masa = -2.3e-5

Operaciones entre variable reales

Operadores elementales o básicos

operación	símbolo	ejemplo
suma, a + b	+	6 + 9
Resta, a - b	-	5 - 8
Producto, a ∗ b	∗	-2.34 ∗ 4
división, a / b	/	0.98 / 4.56
Potencia a^b	^	2.34 ^ 9.5

Estructuras de datos

Operaciones entre variable reales

Regla de precedencia en una expresión:

Las operaciones se ejecutan respetando el orden de precedencia siguiente:

^ , * y / , + y -

mayor a menor

En una expresión con operadores de precedencia iguales, la evaluación se realiza de izquierda a derecha.

El uso de paréntesis puede alterar este orden.

Ejemplos:

$5+6/2=8 \quad \text{y} \quad (5+6)/2=5.5$

$4*25+6*52=412 \quad \text{y} \quad 4*(25+6*52)=1348$

$2/7\text{^}3=0.0058 \quad \text{y} \quad (2/7)\text{^}3=0.0233$

Estructuras de datos

Operaciones entre variable reales

Operadores relacionales

símbolo	operación
>	mayor que
>=	mayor o igual que
<	menor que
<=	menor o igual que
==	igual que
~=	distinto que

Operadores lógicos

operador	descripción
a & b - and(a,b)	retorna 1 si las expresiones a y b son ambas verdaderas
a | b - or(a,b)	retorna 1 si a o b es verdadera
xor(a,b)	Retorna 1 cuando a o b, pero no ambos, es verdadero
~ a - not(a)	Negado de a, retorna 1 si a es 0, 0 en caso contrario

Estructuras de datos

Operaciones entre variable reales

Tablas de comportamiento de los operadores lógicos

A	~ A
0	1
1	0

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	xor(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

Estructuras de datos

Operaciones entre variable reales

Regla de precedencia en una expresión para los operadores (mayor a menor)

1. paréntesis ( ), potencia ^
2. signo positivo +, signo negativo -, negación ~
3. multiplicación $*$, división /
4. suma +, resta –
5. Menor que <, menor o igual que <=, mayor que >,
   mayor o igual que >=, igual que ==, distinto que ~=
6. y &
7. o |

Ejemplos:

- 1 + 1 > 2, ans = 0
- -2 ~= 1, ans = 1
- 1 + (6.8 + 5 < 45) $*$ 3, ans = 4
- ~(1 + (6.8 + 5 < 45) $*$ 3), ans = 0
- ~(1 + (6.8 + 5 < 45)), ans = 0
- 36 < 4 | (1 > 2 ^ 2 == 2), ans = 0

Estructuras de datos

Funciones sobre variables reales

Sobre variables reales existen muchas funciones científicas en MATLAB

>> help elfun

Trigonometric.

sin	**- Sine.**
sinh	**- Hyperbolic sine.**
asin	**- Inverse sine.**
asinh	**- Inverse hyperbolic sine.**
cos	**- Cosine.**
cosh	**- Hyperbolic cosine.**
acos	**- Inverse cosine.**
acosh	**- Inverse hyperbolic cosine.**
tan	**- Tangent.**
tanh	**- Hyperbolic tangent.**
atan	**- Inverse tangent.**
atan2	**- Four quadrant inverse tangent.**
atanh	**- Inverse hyperbolic tangent.**
sec	**- Secant.**
sech	**- Hyperbolic secant.**
asec	**- Inverse secant.**
asech	**- Inverse hyperbolic secant.**
csc	**- Cosecant.**
csch	**- Hyperbolic cosecant.**
acsc	**- Inverse cosecant.**
acsch	**- Inverse hyperbolic cosecant.**
cot	**- Cotangent.**
coth	**- Hyperbolic cotangent.**
acot	**- Inverse cotangent.**
acoth	**- Inverse hyperbolic cotangent.**

Exponential.

exp	**- Exponential.**
log	**- Natural logarithm.**
log10	**- Common (base 10) logarithm.**
log2	**- Base 2 logarithm and dissect floating point number.**
pow2	**- Base 2 power and scale floating point number.**
sqrt	**- Square root.**
nextpow2	**- Next higher power of 2.**

Rounding and remainder.

fix	**- Round towards zero.**
floor	**- Round towards minus infinity.**
ceil	**- Round towards plus infinity.**
round	**- Round towards nearest integer.**
mod	**- Modulus (signed remainder after division).**
rem	**- Remainder after division.**
sign	**- Signum.**

Estructuras de datos

Funciones sobre variables reales

Ejemplo:

	fix	floor	round	ceil
-2.2	-2	-3	-2	-2
3.1	3	3	3	4
3.9	3	3	4	4
0.9	0	0	1	1
	redondeo hacia 0	redondeo hacia -∞	redondeo hacia entero más próximo	redondeo hacia +∞

Obs. round(-3.5) da en como resultado -4, mientras que round(3.5) da 4

Estructuras de datos

Funciones sobre variables reales

Ejemplo:

Definición:

rem(x,y) es x - n.*y donde n = fix(x/y)

mod(x,y) es x - n.*y donde n = floor(x/y)

Ejemplo:

mod(8,3) = 2, rem(8,3) = 2,

mod(-8,3) = 1, rem(-8,3) = -2

Estructuras de datos

Variables complejas

Están definidas por su parte real y su parte imaginaria

Ejemplo:
>> a = 1 - 2i, b = 3 ∗ (2 – sqrt(-1) ∗ 3)
a = 1 – 2i
b = 6 – 9i

Los operadores de suma, resta, multiplicación y división de números complejos usan los mismos símbolos que para variables reales.

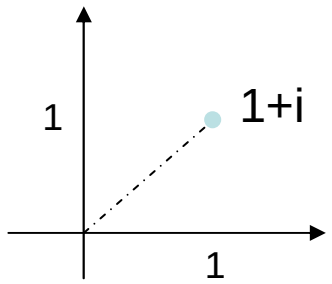
Funciones sobre variables complejas:

abs	- módulo.
angle	- ángulo.
complex	- construye un complejo a partir de (re, im).
conj	- conjugado complejo.
imag	- parte imaginaria.
real	- parte real.

Ejemplo:

>> angle(1 + i)
ans = 0.7854

>> abs(1 + i)
ans = 1.4142



Estructuras de datos

Matrices

MATLAB basa todos sus cálculos en la manipulación de matrices. Una matriz consiste en una colección ordenada de números:

- en 1 dimensión: vector (1 índice)
- en 2 dimensiones: matriz (2 índices)
- en n dimensiones: arreglo o hipermatriz (n índices)

$$a_{1\times4} = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$$

$$a_{4\times1} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

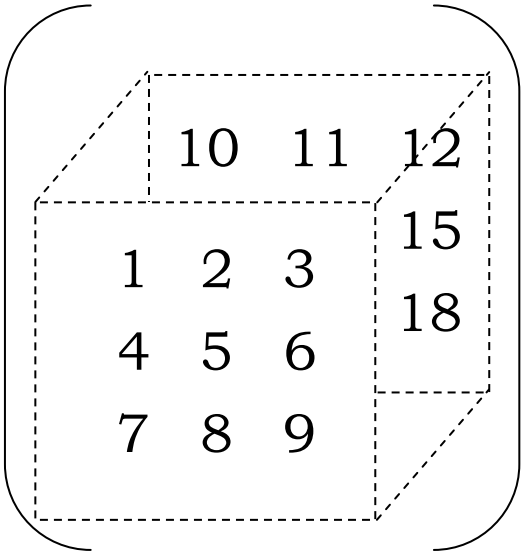
1 dimensión

$$a_{3\times3} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$a_{3\times4} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

2 dimensiones

$$a_{3\times3\times2} =$$



3 dimensiones

Estructuras de datos

Matrices

Para ingresar una matriz, solo hay que respetar algunas convenciones:

- Separar los elementos de una fila con espacios en blanco o comas.
- Usar el punto y coma, ; , o “enter” para indicar el fin de cada fila.
- Encerrar la lista de elementos con corchetes, [ ].

MATLAB posee varias formas de introducir un vector:

1) Enumerando las componentes

vectores fila

x = [1 2*pi sqrt(6) -4.67] 4 componentes

y = [1, 2, 3, 4, 5, 6] 6 componentes

vectores columna

x = [1 (enter) 2*pi (enter) sqrt(6) (enter) -4.67] 4 componentes

y = [1 ; 2 ; 3 ; 4 ; 5 ; 6] 6 componentes

Estructuras de datos

Vectores

2) Usando notación tipo subrango (operador :)

vectores fila

x = primero : último

y = primero : incremento : último

Ejemplos:

x = 1 : 6 → 1 2 3 4 5 6

y = 1 : 3 : 17 → 1 4 7 10 13 16

z = 10 : -2 : 1 → 10 8 6 4 2

3) Usando la función linspace

x = linspace(primero, último, n)

donde el primero y el último están incluidos en el vector y n es el número de elementos.

La distancia entre los elementos es (último-primero)/(n-1)

Ejemplo:

x = linspace(1, 10, 10) → 1 2 3 4 5 6 7 8 9 10

x = linspace(1, 2, 10) → 1. 1.1111 1.2222 1.3333 … 2.

distancia entre elementos = (2-1)/9 = 1/9 = 0.1111

Estructuras de datos

Vectores

4) Usando la función logspace

x = logspace(primero, último, n)

donde n es el número de elementos y las entradas están logaritmicamente espaciadas comenzando con $10^{primero}$ y terminando con $10^{último}$.

La distancia entre los exponentes de los elementos es

(último-primero)/(n-1)

Ejemplo:

x = logspace(0, 2, 11) $\rightarrow$ 1. 1.5899 2.5199 3.9811 … 100.0

distancia entre los exponentes = (2-0)/10 = 1/5 = 0.2

x = (10^0, $10^{0.2}$. $10^{0.4}$, … , 10^2)

5) Composición o concatenación

a = 1 : 5 , b = 1 : 2 : 9

c = [a b] $\rightarrow$ c = (1 ,2, 3, 4, 5, 1, 3, 5, 7, 9)

d = [a(1:2:5) 1 0 1] $\rightarrow$ d = (1, 3, 5, 1, 0, 1)

Estructuras de datos

Vectores

Accediendo un vector en distintas instancias

a = 2 : 6 $\rightarrow$ a = (2, 3, 4, 5, 6)

Acceso a entradas individuales de a

- a(1) corresponde a la entrada 1 del vector a $\rightarrow$ 2

Acceso por bloques de valores usando notación tipo subrango:

- a(1:3) corresponde al vector (2, 3, 4)
- a(1:2:5) corresponde al vector (2, 4, 6)

Acceso a conjunto de posiciones:

- a([1 2 5]) corresponde a las posiciones 1, 2 y 5 de a $\rightarrow$ (2, 3, 6)
- a([1 4 2]) corresponde a las posiciones 1, 4 y 2 de a $\rightarrow$ (2, 5, 3)

Estructuras de datos

Vectores

Operaciones sobre vectores

- Transposición: es una manera de obtener un vector columna a partir de uno fila:

  Si x = [1 2 3] dimensión 1×3

  entonces x' es el vector $\begin{matrix} 1 \\ 2 \\ 3 \end{matrix}$ de dimensión 3×1

- Suma y resta de vectores componente a componente:

  (los vectores deben tener la misma dimensión)

  Si a = 1:3, b = 3:2:7

  entonces a+b = [1 2 3] + [3 5 7] = [4 7 10]

Estructuras de datos

Vectores

Operaciones sobre vectores

- Multiplicación de vectores:

  Vector fila por vector columna con dimensiones compatibles genera un número

  Si a = [1; 2; 3] dimensión 3×1, b = [2, 1, 1] dimensión 1×3

  Podemos llevar a cabo el producto de b y a:

$$b * a = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = 2 + 2 + 3 = 7$$

  También podemos llevar a cabo el producto a*b como producto matricial, generando una matriz de dimensión 3×3:

$$a * b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 2 & 2 \\ 6 & 3 & 3 \end{pmatrix}$$

Estructuras de datos

Vectores

Operaciones sobre vectores

- Operaciones escalar - vector:

  Suma, resta, multiplicación y división de un vector por un escalar corresponde origina un nuevo vector donde sus entradas son la operación de las entradas del vector por el escalar.

  Si a = [1, 2, 3] dimensión 1×3: entonces

$$a - 2 = [1 \quad 2 \quad 3] - 2 = [-1 \quad 0 \quad 1]$$
$$2 * a - 1 = 2 * [1 \quad 2 \quad 3] - 1 = [2 \quad 4 \quad 6] - 1 = [1 \quad 3 \quad 5]$$

- Norma o módulo de un vector:

  Dada un vector a, existe la función norm(a) que la calcula.

  Si a = [1, 2, 3] dimensión 1×3: entonces norm(a) = 3.7417

Estructuras de datos

Vectores

Operaciones sobre vectores

- Longitud de un vector:

  Dada un vector a, existe la función length(a) que la calcula.

  Si a = [1, 2, 3] dimensión 1×3, entonces length(a) = 3.

- Producto escalar de 2 vectores:

  Dada vectores a y b de longitud n, existe la función dot(a,b) que lo calcula, siendo el resultado un número real.

  El producto escalar de los vectores a y b (notación matemática: a·b)

  si a = [1, 2, 3] y b = [3, 2, 1] son de dimensiones 1×3, entonces

$$dot(a,b) = 10$$

Estructuras de datos

Vectores

Operaciones sobre vectores

- Producto vectorial de 2 vectores en 3 dimensiones:

  Dada vectores a y b de longitud 3, existe la función cross(a,b) que la calcula, siendo este para MATLAB un vector columna.

  El producto vectorial de los vectores a y b (notación matemática: $a \times b$) crea un tercer vector perpendicular ellos.

  Si a = [1, 2, 3] y b = [3, 2, 1] son de dimensiones 1×3, entonces

$$cross(a,b) = [-4 \quad 8 \quad -4]$$

  Obs:

  - En nuestro ejemplo: a*c = 0 y b*c = 0 (c es perpendicular a y b)
  - $a \times b = (a_2b_3 - a_3b_2,\ a_3b_1 - a_1b_3,\ a_1b_2 - a_2b_1)$

Estructuras de datos

Vectores

Operaciones sobre vectores

- Operaciones componente a componente:

  En MATLAB existe para las estructuras vectoriales las operaciones de “punto”. La sintaxis es colocar un punto antes de la operación (multiplicación, división y potencia) para hacer de la operación usual la operación equivalente pero elemento a elemento de la estructura. Sean

$$a = [a_1 \quad a_2 \quad \cdots \quad a_n], \quad b = [b_1 \quad b_2 \qquad b_n], \quad c \ \text{escalar}$$

multiplicación punto $$a.*b = [a_1 * b_1 \quad a_2 * b_2 \quad \cdots \quad a_n * b_n]$$

división punto $$a./b = [a_1 / b_1 \quad a_2 / b_2 \quad \cdots \quad a_n / b_n]$$

potenciación punto $$a.\wedge c = [a_1^c \quad a_2^c \quad \cdots \quad a_n^c]$$

$$c.\wedge a = [c^{a_1} \quad c^{a_2} \quad \cdots \quad c^{a_n}]$$

$$a.\wedge b = [a_1^{b_1} \quad a_2^{b_2} \quad \cdots \quad a_n^{b_n}]$$

Estructuras de datos

Vectores

Operaciones sobre vectores

- Aplicación de funciones a vectores:

  El siguiente ejemplo genera una tabla de funciones *seno* y *coseno*:

  >> x=[0.0: pi/50: 2*pi] ‘ ;

  >> y = sin(x); z = cos(x);

  >> c = [x y z] →

  c es una matriz de dimensión 101×3

c =

0	0	1.0000
0.0628	0.0628	0.9980
0.1257	0.1253	0.9921
0.1885	0.1874	0.9823
0.2513	0.2487	0.9686
0.3142	0.3090	0.9511
0.3770	0.3681	0.9298
0.4398	0.4258	0.9048
0.5027	0.4818	0.8763
…		
6.0947	-0.1874	0.9823
6.1575	-0.1253	0.9921
6.2204	-0.0628	0.9980
6.2832	-0.0000	1.0000

Estructuras de datos

Vectores

Operaciones sobre vectores

- Aplicación de funciones a vectores:

  Las siguientes funciones actúan sobre vectores:

[xm,im] = max(x)	máximo elemento de un vector. Devuelve el valor máximo y la posición que ocupa
[xm,im] = min(x)	mínimo elemento de un vector. Devuelve el valor mínimo y la posición que ocupa
sum(x)	suma de los elementos de un vector
cumsum(x)	devuelve el vector suma acumulativa de los elementos de un vector
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
[y,i] = sort(x)	ordena de menor a mayor de los elementos de un vector x. Devuelve el vector ordenado y, y un vector i con las posiciones iniciales en x de los elementos en el vector ordenado y

Estructuras de datos

Vectores

Aplicación de funciones a vectores. Ejemplos

```
>> b = [28    32    36    40]
>> sum(b)          →    ans = 136
>> cumsum(b)   →    ans = 28    60    96   136
>> prod(b)         →    ans = 1290240
>> cumprod(b)  →   ans = 28     896     32256    1290240
>> mean(b)         →   ans = 34
>> std(b)          →   ans = 5.1640
```

```
>> c = [3 4 1 6 9 10 2 3]
>> [xm,im] = max(c)    →    xm = 10,     im = 6
>> [y,i] = sort(c)  →
      y = 1     2     3     3     4     6     9    10
      i = 3     7     1     8     2     4     5     6
```

Estructuras de datos

Vectores

Funciones que actúan sobre vectores

- find: devuelve una lista de los índices o posiciones que cumplen cierta condición

  Ejemplo:

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

  >> find (A > 1) → 2 3 4 5 6 7 8 9 (vector con las posiciones en A)

  >> find (A <= 3) → 1 2 3

  >> find(A>=3 & A<8) → 3 4 5 6 7

Estructuras de datos

Vectores

Operaciones sobre vectores

- Operadores relacionales en vectores:

  Si una comparación se cumple el resultado es 1 (true), mientras que si no se cumple es 0 (false). Recíprocamente, cualquier valor distinto de cero es considerado como true y el cero equivale a false.

  Los operadores relacionales de MATLAB se aplican a dos vectores del mismo tamaño, la comparación se realiza elemento a elemento, y el resultado es otro vector de unos y ceros del mismo tamaño, que recoge el resultado de cada comparación entre elementos.

  Ejemplo:

  >> a = [1 2 0 3]; b = [4 2 1 5];

  >> a == b $\rightarrow$ ans = [0 1 0 0]

  >> a ~= b $\rightarrow$ ans = [1 0 1 1]

  >> a < b $\rightarrow$ ans = [1 0 1 1]

Ejemplo: ¿Qué da a(a>1)? ¿qué da a([0 1 0 1])? ¿qué da a(logical([0 1 0 1]))?

Estructuras de datos

Vectores.

Ejercicios:

Dado un vector *v* de números enteros, calcule cuántos de ellos son positivos e impares a la vez. ¿Cuáles son?.

Dado un vector *v* de números enteros, genere otros tres vectores que contengan:

- los números impares,
- los números pares,
- las posiciones en v donde se encuentra el número 0.

Estructuras de datos

Ejemplos

- Nota/calificación en base 100 (1 – 100) a nota en base 5 (1 – 5), sabiendo 1 a 29 -> 1; 30 a 49 -> 2; 50 a 69 -> 3; 70 a 84 -> 4; 85 a 100 -> 5

  (nota >= 1) + (nota >= 30) + (nota >= 50) + (nota >= 70) + (nota >= 85)

- Factorial de un número natural mayor que 1

  prod(2:numero)

- Representación decimal (base 10) de un número natural x en base 2, donde el número natural x esta dado como el vector (d1, … , dn) con

  $$x = d1 * 2^{n-1} + \ldots + dn * 2^{0}$$

  sum(2.^(find(d(end : -1 : 1) == 1) -1))

  o

  sum(d.*(2.^(length(d) -1 : -1 : 0)))

Estructuras de datos

Ejemplos

- Ordenar un vector en consecuencia de otro
  (1 : ing, 2: lic)

personal = [1, 1, 2, 1, 2, 2, 1]; edad = [20, 10, 15, 50, 75, 40, 15];

ordenar el vector personal en función de ordenar edad de manera descendente

```
[eo, ind] = sort(edad, 'descend');
po = personal(ind);
```

calcular el número de licenciados

```
sum(personal == 2)
```

calcular número de licenciados mayores de 40 años

```
sum(personal == 2 & edad > 40)
```

Estructuras de datos

Matrices

- Los vectores son un caso particular de una matriz
- Estas son estructuras en 2 dimensiones que tienen muchos usos en cálculo científico
- Estas son las estructuras más idóneas para almacenar los datos de un sistema lineal de ecuaciones.

Declarar un matriz

Enumeramos los componentes de la matriz separando las distintas filas con punto y coma (;), y los valores en una misma fila con coma (,) o espacio en blanco.

Ejemplo:

>> A = [1 2 3 4; 8 9 10 11; 15 16 17 18]; $\rightarrow$ $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 8 & 9 & 10 & 11 \\ 15 & 16 & 17 & 18 \end{pmatrix}$

Es importante que cada fila de la matriz tenga el mismo número de entradas, de lo contrario MATLAB dará un error.

Estructuras de datos

Matrices

Las matrices pueden ser introducidas en MATLAB

- como lista explícita de los elementos
- cargadas de archivos externos
- generadas por comandos o funciones de MATLAB

Los siguientes comandos son equivalentes:

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

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

- filas separadas por comas o blancos
- columnas separadas por punto y coma o “return”

y generan la matriz la matriz siguiente

```
A =
     1  2  3
     4  5  6
     7  8  9
```

Estructuras de datos

Matrices

Generación de matrices usando comandos o funciones de MATLAB

zeros(m,n)	matriz m×n cuyas entradas son ceros (0)
ones(m,n)	matriz m×n con entradas unos (1)
eye(n)	Matriz identidad de dimensión n×n
rand(m,n)	matriz m×n de entradas aleatorias con distribución uniforme entre 0 y 1
randn(m,n)	matriz m×n de entradas aleatorias con distribución normal, de valor promedio 0 y varianza 1
magic(n)	matriz n×n con la propiedad de que la suma de filas y columnas es igual
hilb(n)	matriz de Hilbert de dimensión n×n La matriz de Hilbert es una matriz cuyos elementos (i,j) responden a la expresión (1/(i+j-1)). Esta es una matriz especialmente difícil de manejar por los grandes errores numéricos a los que conduce
diag(t) t un vector de longitud n	matriz n×n con t en su diagonal y las otras entradas ceros
compan(pol)	matriz cuyo polinomio característico tiene como coeficientes los elementos del vector pol (ordenados de mayor grado a menor)

Estructuras de datos

Matrices

Funciones sobre matrices. Ejemplos:

>> D = magic(3) → ans =

$$\begin{matrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{matrix}$$

>> sum(D,1) → ans = 15 15 15

>> sum(D,2) →

$$\text{ans} = \begin{matrix} 15 \\ 15 \\ 15 \end{matrix}$$

>> A = [3 -1; 1 0] → $A = \begin{pmatrix} 3 & -1 \\ 1 & 0 \end{pmatrix}$

el polinomio característico es

$$\det\left(A - xI\right) = \det\begin{pmatrix} 3-x & -1 \\ 1 & -x \end{pmatrix} = (3-x)(-x) + 1 = x^2 - 3x + 1$$

>> compan([1 -3 1]) → ans =

$$\begin{matrix} 3 & -1 \\ 1 & 0 \end{matrix}$$

Estructuras de datos

Matrices

Acceso a las entradas de una matriz

Ejemplos:

- A = [1 2 3; 4 5 6; 7 8 9] $\rightarrow$ $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$
- A(2,3) $\rightarrow$ corresponde a la 2da fila y 3era columna de A, es decir, 6
- A(3,3) = 8.1 $\rightarrow$ asignamos 8.1 a la posición (3,3) de la matriz A

  b = A(3, :) $\rightarrow$ asignamos a la variable b la 3era fila de A,

  es decir, b = [7 8 9];

  El operador 2 puntos (:) en este caso indica todo el rango en la dimensión correspondiente a las columnas

- C = A(:,3:-1:1) $\rightarrow$ $C = \begin{pmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{pmatrix}$

  La variable C se obtiene intercambiando las columnas de la matriz A comenzando por la tercera hasta la primera.

Estructuras de datos

Matrices

Acceso a las entradas de una matriz

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Ejemplos (cont.):

- C = A(3:-1:1; :) $\rightarrow$ $$C = \begin{pmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$$
- C = A(1:2, 2:3) $\rightarrow$ $$C = \begin{pmatrix} 2 & 3 \\ 5 & 6 \end{pmatrix}$$
- C = A([1 3], :) $\rightarrow$ $$C = \begin{pmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix}$$
- C = A(: ,[1 3]) $\rightarrow$ $$C = \begin{pmatrix} 1 & 3 \\ 4 & 6 \\ 7 & 9 \end{pmatrix}$$

Estructuras de datos

Matrices

Operaciones con matrices

- Para matrices de igual dimensión podemos sumarlas o restarlas entre si. El resultado es la suma o resta componente a componente de las 2 matrices.

  Ejemplo:

A = rand(3) y B = eye(3)

0.9501	0.4860	0.4565	1	0	0
0.2311	0.8913	0.0185	0	1	0
0.6068	0.7621	0.8214	0	0	1

suma = A + B y resta = A - B

1.9501	0.4860	0.4565	-0.0499	0.4860	0.4565
0.2311	1.8913	0.0185	0.2311	-0.1087	0.0185
0.6068	0.7621	1.8214	0.6068	0.7621	-0.1786

Estructuras de datos

Matrices

Operaciones con matrices

- Si A y B son matrices de dimensión m×n y n×p respectivamente, la multiplicación de A y B, es decir A*B, es una matriz de dimensión m×p

  Ejemplo:

  >> A = [1 2 3; 4 5 6], B = [[1 2 3 5] ; rand(2,4)]

  ```
  1    2    3
  4    5    6
  ```

  ```
  1.0000    2.0000    3.0000    5.0000
  0.4447    0.7919    0.7382    0.4057
  0.6154    0.9218    0.1763    0.9355
  ```

  >> producto = A * B

  ```
  3.7357    6.3493    5.0052    8.6178
  9.9161   17.4906   16.7486   27.6413
  ```

- Para matrices cuadradas se tiene la potenciación matricial.

  Así, para la matiz cuadrada A, A^2 es equivalente a A*A

Estructuras de datos

Matrices

Operaciones con matrices

- Operaciones componente a componente

  Ejemplo:

  >> A = round(rand(3) * 3), B = [3 1 2 ; 4 5 6 ; 1 7 2]

```
A =
     3     0     0
     1     1     0
     3     2     1

B =
     3     1     2
     4     5     6
     1     7     2
```

  >> resultado1 = A .* B

```
resultado1 =
     9     0     0
     4     5     0
     3    14     2
```

  >> resultado2 = B .^ A

```
resultado2 =
    27     1     1
     4     5     1
     1    49     2
```

  >> resultado3 = 2 .^ A

```
resultado3 =
     8     1     1
     2     2     1
     8     4     2
```

  >> resultado4 = A ./ B

```
resultado4 =
    1.0000         0         0
    0.2500    0.2000         0
    3.0000    0.2857    0.5000
```

Estructuras de datos

Matrices

Operaciones con matrices

- Operadores relacionales y matrices

  Sean A y B matrices de igual dimensión.

  A == B retorna una matriz de igual dimensión cuyas entradas son 1 o 0 según sean las entradas de A y B iguales o no

  A ~= B retorna una matriz de igual dimensión cuyas entradas son 1 o 0 según sean las entradas de A y B diferentes o no

  Ejemplo:

  >> A = round(rand(3) * 3),

  ```
  3   0   0
  1   1   0
  3   2   1
  ```

  B = [3 1 2 ; 4 5 6 ; 1 7 2]

  ```
  3   1   2
  4   5   6
  1   7   2
  ```

  >> A == B,

  ```
  1   0   0
  0   0   0
  0   0   0
  ```

  A ~= B

  ```
  0   1   1
  1   1   1
  1   1   1
  ```

Estructuras de datos

Matrices

Operaciones con matrices

- Operadores lógicos y matrices

  Sean A y B matrices de igual dimensión.

  A op B retorna una matriz de igual dimensión cuyas entradas son 1 o 0 según las entradas de A y B satisfagan la condición del operador lógico “op”.

  Ejemplo:

  >> A = round(rand(3) * 3), B = [3 1 2 ; 4 5 6 ; 1 7 2]

3	0	0
1	1	0
3	2	1

3	1	2
4	5	6
1	7	2

>> A & B, A | B

1	0	0
1	1	0
1	1	1

1	1	1
1	1	1
1	1	1

Estructuras de datos

Matrices

Funciones que actúan sobre matrices: A una matriz de dimensión m×n

A ‘	transpuesta conjugada de A
A .’	Transpuesta sin conjugar de A
trace(A)	devuelve la traza de A (suma de los elementos de la diagonal de A)
[m,n] = size(A)	devuelve el número de filas m y de columnas n de A size(A,1): número de filas, size(A,2): número de columnas de A
diag(A)	diagonal de A
det(A)	determinante de la matriz A, para el caso m=n (A cuadrada)
sum(A), sum(A,1)	suma por columna de los elementos de A
sum(A,2)	suma por filas de los elementos de A
norm(A)	norma 2 de A (radio espectral de A^tA) norm(A,1): norma 1 de A (max(sum(abs(A)))) norm(A,inf): norma inf de A (max(sum(abs(A'))))
inv(A)	Matriz inversa de A, cuando esta exista

Estructuras de datos

Matrices

Funciones que actúan sobre matrices (cont.): A una matriz de dimensión m×n

cond(A)	condición numérica (basado en la norma 2) de A (norm(A)*norm(A^{-1}))
isequal(A,B)	retorna 1 si A es igual a B, y 0 en cualquier otro caso, para matrices de igual dimensión
and(A,B)	matriz cuyas entradas resultan de aplicar el operador lógico “y” a las entradas de A y B
or(A,B)	matriz cuyas entradas resultan de aplicar el operador lógico “o” a las entradas de A y B
not(A)	matriz cuyas entradas resultan de negar las entradas de A
all(A,1) o all(A) all(A,2)	operador lógico “y” múltiple por columna operador lógico “y” múltiple por fila
any(A,1) o any(A) any(A,2)	operador lógico “o” múltiple por columna operador lógico “o” múltiple por fila
[L,U] = lu(A)	Calcula las matrices triangular inferior L y triangular superior U resultantes de la descomposición LU de A
[X,D] = eig(A)	Calcula los valores propios (diagonal de D) y vectores propios (columnas de X) de una matriz cuadrada A

Estructuras de datos

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Matrices

Funciones que actúan sobre matrices

- find: devuelve una lista de los índices o posiciones que cumplen cierta condición

  Ejemplo:

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

  >> find (A > 1) $\rightarrow$ 2 3 4 5 6 7 8 9 (vector columna con las posiciones en A)

  >> find (A <= 3) $\rightarrow$ 1 4 7

  >> [i, j] = find (A <= 3) $\rightarrow$ 1 1 1 (índice de fila)

  1 2 3 (índice de columna)

  posiciones (1,1), (1,2) y (1,3) de A

  >> ind = find (A < 3), b = A(ind)

  se obtiene ind $\rightarrow$ 1 4 (los posiciones en A)

  se obtiene b $\rightarrow$ 1 2 (los elementos en A)

Estructuras de datos

Función “reshape”

Esta función permite cambiar la estructura de un arreglo dado.

reshape(s,m,n) retorna la matriz m×n cuyos elementos se toman recorriendo s por columnas. Se produce un error si s no tiene mn elementos.

>> s = [1 2 3 4; 5 6 7 8]; x = reshape(s, 1, 8)

→ produce el arreglo [1 5 2 6 3 7 4 8] de dimensión 1×8

$$s = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \quad \Longrightarrow \quad x = \begin{pmatrix} 1 & 5 & 2 & 6 & 3 & 7 & 4 & 8 \end{pmatrix}$$

>> s = [1 2 3 4; 5 6 7 8]; x = reshape(s, 4, 2)

→ produce el arreglo [1 3; 5 7; 2 4; 6 8] de dimensión 4×2

$$s = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \quad \Longrightarrow \quad x = \begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 2 & 4 \\ 6 & 8 \end{pmatrix}$$