****

Facultad de Ciencias Exactas y Naturales

Curso: Programación

Codigo: 0303158

Tema: Matrices

Semestre: 2020-1

Medellín, 2020

**MATRICES**

Un vector, o elemento , se puede ver como una formación de ***n*** números reales de tipo lineal, es decir, como formando una línea recta. Por eso, los programadores lo denominan un arreglo lineal, en una mala traducción de la palabra inglesa *array*.

Otro tipo de formación es la que agrupa elementos de forma rectangular, es decir, en filas y columnas, lo cual se conoce como un arreglo bidimensional o matriz. Se trata de agrupar ***m·n*** números reales en ***m*** filas de ***n*** elementos cada una.

En una agrupación de este tipo, se utiliza un índice para la fila y otro para la columna que ocupa el elemento, así:

En programación, una **matriz** es un conjunto de celdas de memoria con un nombre común y diferenciadas por **dos** índices.

En Matlab u Octave, similarmente a como se define un vector por extensión, es decir, encerrando todos sus elementos entre corchetes, se puede definir una matriz utilizando ";" para pasar a la siguiente fila.

**Ejemplo**:

|  |  |
| --- | --- |
| Instrucción | Resultado |
| **A = [3 1 5 2 4;-2 0 1 8 6;7 2 0 -5 4]** | **A =**  **3 1 5 2 4**  **-2 0 1 8 6**  **7 2 0 -5 4** |

Si no se escriben todos los elementos, se produce error:

|  |  |
| --- | --- |
| Instrucción | Resultado |
| **A = [3 1 5 2 4;-2 0 1 8; 7 2 0 -5 4]** | **error: number of columns must match (4 != 5)** |

La otra forma, asignando un valor a una de sus componentes, también funciona:

|  |  |
| --- | --- |
| Instrucción | Resultado |
| **B(4,3) = 7** | **B =**  **0 0 0**  **0 0 0**  **0 0 0**  **0 0 7** |
| **B(5,5) = 0** | **B =**  **0 0 0 0 0**  **0 0 0 0 0**  **0 0 0 0 0**  **0 0 7 0 0**  **0 0 0 0 0** |

**Ejemplos**:

|  |  |
| --- | --- |
| **function z = vector(n,a,b)**  **for k = 1:n**  **z(k) = floor(rand\*(b-a)+a);**  **end** | Función que produce un vector de n enteros aleatorios entre dos enteros a,b. |

|  |  |
| --- | --- |
| **function z = matriz(m,n,a,b)**  **for i = 1:m**  **for j = 1:n**  **z(i,j) = floor(rand\*(b-a)+a);**  **end**  **end** | Función que produce una matriz con enteros aleatorios entre dos enteros . |
| **function z = impares(m,n,p)**  **for i = 1:m**  **for j = 1:n**  **z(i,j) = p;**  **p = p+2;**  **end**  **end** | Función que produce una matriz con enteros impares, a partir del número impar . |
| **function z = impares2(m,n,p)**  **for j = 1:n**  **for i = 1:m**  **z(i,j) = p;**  **p = p+2;**  **end**  **end** | Igual que la anterior. |
| **function z = triangular(n,a,b)**  **for i = 1:m**  **for j = 1:i**  **z(i,j) = floor(rand\*(b-a)+a);**  **end**  **end** | Función que produce una matriz triangular inferior con enteros aleatorios entre dos enteros . |
| **function z = columna(n,a,b)**  **for i = 1:n**  **z(i,1) = floor(rand\*(b-a)+a);**  **end** | Función que produce un vector columna con enteros aleatorios entre dos enteros . |

**Ejemplos**:

|  |  |
| --- | --- |
| **function z = esprimo2(n)**  **z = 1;**  **for k = 2:sqrt(n)**  **if mod(n,k) == 0**  **z = 0;**  **break;**  **end**  **end** | Función que verifica si un entero positivo n es primo. |

|  |  |
| --- | --- |
| **function z = primos(m,n,a,b)**  **for j = 1:n**  **for i = 1:m**  **x = floor(rand\*(b-a)+a);**  **if esprimo2(x)**  **z(i,j) = x;**  **end**  **end**  **end** | Función **mal hecha** que produce una matriz con números primos entre dos enteros a,b. |
| **function z = primosbien(m,n,a,b)**  **for j = 1:n**  **for i = 1:m**  **do**  **x = floor(rand\*(b-a)+a);**  **until esprimo2(x);**  **z(i,j) = x;**  **end**  **end** | Función **bien hecha** que produce una matriz con números primos entre dos enteros a,b. |

**Tipos de problemas que se pueden resolver con matrices**

1. Generar matrices. Son funciones que crean matrices a partir de argumentos escalares.
2. Definir funciones booleanas con argumento matricial, para verificar propiedades de una matriz.
3. Definir funciones con argumentos matriciales, para operar con los elementos de una o más matrices o para generar matrices a partir de otras.

Cuando se recibe una matriz como argumento de una función, lo primero, casi siempre es averiguar el orden de la matriz, es decir, cuántas filas y columnas tiene, lo cual se hace con la función **size**, así: **[m,n] = size(A)**.

La función **length** aplicada a matrices no devuelve el número de elementos, sino el máximo entre el número de filas y el de columnas. El número de elementos se obtiene con la función **numel**.

Ejemplos:

|  |  |
| --- | --- |
| **function A = divmayor(A)**  **[m,n] = size(A);**  **for i = 1:m**  **may = A(i,1);**  **for j = 2:n**  **if A(i,j) > may;**  **may = A(i,j);**  **end**  **end**  **for j = 1:n**  **A(i,j) = A(i,j)/may;**  **end;**  **end** | Dividir cada elemento de una matriz por el mayor de su fila. |
| **function z = sumacol(A)**  **[m,n] = size(A);**  **for j = 1:n**  **S = 0;**  **for i = 1:m**  **S = S + A(i,j);**  **end**  **z(j) = S;**  **end** | Devolver un vector con las sumas de las columnas de una matriz. |
| **function z = insumacol(A)**  **[m,n] = size(A);**  **for j = 1:n**  **S = 0;**  **for i = 1:m**  **S = S + A(i,j);**  **end**  **A(m+1,j) = S;**  **end** | Variante del anterior: Dada una matriz, agregarle una fila con las sumas de las columnas. |
| **function C = producto(A,B)**  **% Producto de matrices**  **[m,n] = size(A); [s,r] = size(B);**  **if n == s**  **for i = 1:m**  **for j = 1:r**  **S = 0;**  **for k = 1:n**  **S = A(i,k)\*B(k,j);**  **end**  **C(i,j) = S;**  **end**  **end**  **else**  **disp('No conformables');**  **end** | Calcula  ,  dado que |
| **function z = tienecero(A)**  **[m,n] = size(A);**  **z = 0;**  **for i = 1:m**  **for j = 1:i**  **if A(i,j) == 0**  **z = 1;**  **break;**  **end;**  **end**  **end** | Función que verifica si una matriz **A** contiene un cero. |
| **function z = palindromo(A)**  **[m,n] = size(A);**  **z = 1;**  **for i = 1:m**  **for j = 1:i**  **if A(i,j) != A(m+1-i,n+1-j)**  **z = 0;**  **end;**  **end**  **end** | Función que verifica si una matriz **A** es un palídromo. |

**Funciones y operaciones especiales con matrices en Matlab y Octave**

Matlab trae muchas operaciones y funciones predefinidas para matrices, en particular, las utilizadas en Álgebra Lineal (determinante, inversa, transpuesta, etc.) y en Estadística (sum, prod, mean, rand, randn, reglin, etc.).

También trae funciones para generar matrices de muestra: zeros, ones, eye, diag, pascal, hilb, magic, etc.

|  |  |  |
| --- | --- | --- |
| **z3 = pascal(6)**  **z3 =**  **1 1 1 1 1 1**  **1 2 3 4 5 6**  **1 3 6 10 15 21**  **1 4 10 20 35 56**  **1 5 15 35 70 126**  **1 6 21 56 126 252** | | **z1 = zeros(3,5)**  **z1 =**  **0 0 0 0 0**  **0 0 0 0 0**  **0 0 0 0 0** |
| **z2 = ones(1,8)**  **z2 =**  **1 1 1 1 1 1 1 1** | **z4 = 7\*ones(1,8)**  **z4 =**  **7 7 7 7 7 7 7 7** | |

Las operaciones aritméticas elementales se pueden utilizar entre matrices y escalares. Si se trata de operaciones entre matrices, se debe tener en cuenta la conformabilidad de las operaciones y la definición de las operaciones inversas. Por ejemplo, se pueden sumar dos matrices si son del mismo orden, equivale a , es el producto matricial de por sí misma veces.

Si una operación entre matrices se debe realizar punto a punto, se utiliza un punto para eliminar la connotación matricial de la operación.

**Ejemplos**:

|  |  |  |
| --- | --- | --- |
| **b = floor(7\*rand(4,3))** | | **b =**  **1 4 1**  **5 0 5**  **3 1 4**  **3 1 1** |
| **b'** | | **ans =**  **1 5 3 3**  **4 0 1 1**  **1 5 4 1** |
| **b\*b** | **error: operator \*: nonconformant arguments (op1 is 4x3, op2 is 4x3)** | |
| **b\*b'** | | **ans =**  **18 10 11 8**  **10 50 35 20**  **11 35 26 14**  **8 20 14 11** |
| **b'\*b** | | **ans =**  **44 10 41**  **10 18 9**  **41 9 43** |
| **b.\*b** | | **ans =**  **1 16 1**  **25 0 25**  **9 1 16**  **9 1 1** |
| **det(b)** | | **error: det: argument must be a square matrix** |
| **det(b\*b')** | | **ans = -2.0428e-012** |
| **det(b'\*b)** | | **ans = 3314** |

Los corchetes sirven para pegar matrices de distintas maneras, siempre que se dé la conformabilidad para la operación.

|  |  |  |
| --- | --- | --- |
| **a = floor(12\*rand(5,3))** | | **a =**  **7 2 11 4 11**  **5 3 6 9 1**  **7 5 4 1 8** |
| **[a,b]** | **error: number of rows must match (4 != 3) near line 85, column 4** | |
| **[a,b']** | **ans =**  **7 2 11 4 11 1 5 3 3**  **5 3 6 9 1 4 0 1 1**  **7 5 4 1 8 1 5 4 1** | |

**Ejercicio**: Dado un conjunto de pares de puntos , encontrar la recta que mejor los aproxima, utilizando las fórmulas obtenidas en el proceso de "regresión lineal":

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

, el coeficiente de correlación, es una medida que indica qué tan bien aproximada queda la nube de puntos por medio de la recta obtenida, mientras más próximo a 1, en valor absoluto, mejor.

Una forma de hacerlo consiste en construir una matriz , a partir de los valores de , , así:

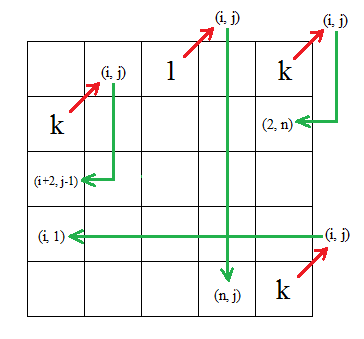
La suma de las columnas de produce las sumatorias requeridas para el cálculo de , y .

|  |
| --- |
| **function A = regresion(x,y)**  **n = length(x);**  **for i = 1:n**  **A(i,1) = x(i);**  **A(i,2) = y(i);**  **A(i,3) = x(i)^2;**  **A(i,4) = x(i)\*y(i);**  **A(i,5) = y(i)^2;**  **end**  **S = sumacol(A);**  **m = (n\*S(4) - S(1)\*S(2))/(n\*S(3) - S(1)^2);**  **b = (S(2) - m\*S(1))/n;**  **for i = 1:n**  **A(i,6) = m\*A(i,1) + b;**  **end**  **fprintf('Ecuacion: y = %fx +(%f)\n', m, b);**  **fprintf('Error de correlacion: \n');** |

**Ejercicio**: Un cuadrado mágico de lado es una matriz , formada por los enteros , cuyas filas, columnas y diagonales principales suman lo mismo. Cuando es **impar**, una forma de ir llenando una matriz con los números , de manera que resulte en un cuadrado mágico es la siguiente:

1. Coloque el número 1 en el centro de la primera fila.
2. La celda donde va el siguiente número es la que queda una columna adelante y una fila atrás. Si esta celda queda en la fila 0 o en la columna o ambas, se corrigen así:
   1. Si y , se hace .
   2. Si y , se hace .
   3. Si y , se hacen , .
   4. Si la celda ya ha sido asignada, se hacen , .

Gráficamente:



Con estas indicaciones, el programa queda así:

|  |  |
| --- | --- |
| **function A = magico(n)**  **A = zeros(n);**  **i = 1; j = (n+1)/2;**  **for k = 1:n^2**  **A(i,j) = k;**  **i = 1-1; j = j+1;**  **if i == 0 && j > n**  **i = 2; j = n;**  **elseif i == 0**  **i = n;**  **elseif**  **j = 1;**  **elseif A(i,j) > 0**  **i = i+2; j = j-1;**  **end**  **end** | **magico(5)**  **ans =**  **17 24 1 8 15**  **23 5 7 14 16**  **4 6 13 20 22**  **10 12 19 21 3**  **11 18 25 2 9** |
| **magic(5)**  **ans =**  **17 24 1 8 15**  **23 5 7 14 16**  **4 6 13 20 22**  **10 12 19 21 3**  **11 18 25 2 9** |

**Ejercicio**: Convertir matriz en vector y viceversa. Cuando hay una operación que podemos hacer en un vector, y se pide algo similar para una matriz, se podrían pasar los elementos de la matriz a un vector, hacer la operación en el vector y, luego, convertir el vector en matriz nuevamente.

|  |  |
| --- | --- |
| **function z = mat2vec(A)**  **% Convierte a vector la matriz A,**  **% por filas**  **[m,n] = size(A);**  **z = [];**  **for i = 1:m**  **for j = 1:n**  **z = [z, A(i,j)];**  **end**  **end** | **function A = vec2mat(x,m,n)**  **% Convierte el vector x a**  **% una matriz mxn**  **r = length(x);**  **if r < m\*n**  **x(m\*n) = 0;**  **end;**  **k = 1;**  **for i = 1:m**  **for j = 1:n**  **A(i,j) = x(k);**  **k = k+1;**  **end;**  **end** |
| **function z = noceros(x)**  **% Elimina ceros de vector x**  **z = [];**  **for k = 1:length(x)**  **if x(k) != 0**  **z = [z, x(k)];**  **end**  **end** | **function B = correrceros(A)**  **% Desplaza al final**  **% los ceros de una matriz**  **[m,n] = size(A);**  **x = mat2vec(A);**  **y = noceros(x);**  **B = vec2mat(y,m,n);** |

Prueba:

**w = floor(rand(5)\*10-5)**

**w =**

**3 1 0 -4 -3**

**4 2 -2 -3 -1**

**-2 0 0 3 3**

**4 -1 -4 0 -1**

**-1 -3 -1 1 -4**

**correrceros(w)**

**ans =**

**3 1 -4 -3 4**

**2 -2 -3 -1 -2**

**3 3 4 -1 -4**

**-1 -1 -3 -1 1**

**-4 0 0 0 0**