# PRACTICO Nº 10

## Introducción

El objetivo de este práctico es que el alumno se familiarice con los temas representación vectorial de polinomios y matrices dispersas. Los ejercicios o ejemplos presentados deben ser ejecutados en el entorno de desarrollo de *Octave.*

# *Matrices dispersas*

## Ejercicio 1

**a)** Utilizando matrices completas, se pide:

- **i.** Implementar una función que reciba una matriz y devuelva otra matriz que sea su transpuesta. Verifique su funcionamiento utilizando el cálculo de transpuesta implementado en Octave.
- **ii.** Implementar una función que calcule la suma de matrices. Verificar las dimensiones de las matrices.
- **iii.** Implementar una función que calcule el producto de matrices. Verificar las dimensiones de las matrices.

Sean $A_{mxk}$ y $B_{kxn}$, entonces C = A * B donde $c_{ij} = \sum_{k=1}^{m} a_{ik} * b_{kj}$

**b)** Utilizando matrices dispersas (representadas por vectores $a_n$, $a_i$ y $a_j$)

- **i.** Implementar una función que reciba una matriz dispersa ($a_n$, $a_i$ y $a_j$) y devuelva otra matriz que sea su transpuesta.
- **ii.** Implementar una función que haga la suma de matrices dispersas.
- **iii.** (Opcional) Implementar una función que haga el producto de matrices dispersas.

**Nota:** Asumir que las matrices de entrada están ordenadas. La salida no tiene por qué estar ordenada.

## Ejercicio 2

Se define la **traza** de una matriz cuadrada *A* de *nxn* como la suma de los elementos de la diagonal principal de *A*.

**a)** Realizar una *función recursiva* **`TrDisp(As, Ai, Aj)`** en *Octave* que reciba como parámetros una matriz dispersa almacenada en formato elemental (representada mediante ***tres vectores***) y devuelva la traza de dicha matriz.

**b)** Realizar una *función iterativa* **`TrDispIter(As, Ai, Aj)`** en *Octave* que reciba como parámetros una matriz dispersa almacenada en formato elemental (representada mediante ***tres vectores***) y devuelva la traza de dicha matriz.

## Ejercicio 3

Vea el help de **`whos`**, y ejecute el siguiente script en Octave:

```
a = diag(-6:6) + diag(ones(2*6,1),1) + diag(ones(2*6,1),-1);
b = sparse(a);
c = full(b);
whos
```

**a)** Implemente un script que obtenga una matriz identidad de 20x20 (ver la función **`eye`** de Octave), la matriz **`[1,1,0;0,1,1;1,0,1]`** , y la representación dispersa de las mismas. Compare el espacio utilizado en las diferentes representaciones.

## Ejercicio 4

Para comprimir una matriz completa que contiene datos que son generalmente iguales (con poca o casi nula variación entre ellos) vamos a utilizar la siguiente metodología:

1. Determinar la frecuencia de ocurrencia de cada uno de los datos que aparece en la matriz.
2. Obtener el dato más frecuente.
3. Restar a la matriz original el dato hallado en el punto anterior.
4. Convertir la nueva matriz (resta) completa en la representación a 3 vectores para matices dispersas vista en el curso.

Se pide por lo tanto que escriba y ejecute las siguientes funciones en *Octave*:

a) una función *iterativa* llamada **DatoFrec** que dada una matriz completa devuelva una matriz de 2 columnas y N filas. Esta función debe calcular la frecuencia de aparición de los diferentes elementos (datos) que se encuentran en la matriz. En la primera columna debe aparecer el dato y en la segunda su frecuencia (o número de apariciones). En caso de invocarla con una matriz nula devolver vector vacío.

b) una función *iterativa* llamada **MasFrec** que dada la matriz calculada en la parte a), devuelva la fila correspondiente al elemento más frecuente. Si existe más de uno con igual frecuencia devolver cualquiera de ellos.

c) una función *iterativa* llamada **Comprimir** que dada una matriz completa realice el procedimiento de compresión descrito anteriormente y devuelva la matriz en su representación dispersa. Para eso debe invocar a las funciones implementadas en las partes *a* y *b* y realizar la descomposición en 3 vectores vista en el curso para representar matrices dispersas. En caso de invocarla con una matriz nula devolver los 3 vectores vacíos.

## Ejercicio 5

**a)** Escribir y ejecutar una función *recursiva* **TSacarFilCol(As, Ai, Aj, fil, col)** en Octave que reciba como parámetros una matriz dispersa almacenada en formato elemental (representada mediante *tres vectores*) y una fila y una columna y devuelva la matriz dispersa que se obtiene de eliminar todos los elementos de la fila *fil* y todos los elementos de la columna *col*. Además, deberán decrementar en uno las filas mayores que *fil* y las columnas mayores que *col*.

**Observación:** el efecto de esta función es equivalente a sacar la fila *fil* y la columna *col* de la matriz.

La función se podrá invocar de la siguiente forma:
*[As,Ai,Aj] = TSacarFilCol(As, Ai, Aj, fil, col)*

**b)** Escriba y ejecute una función *iterativa* **MaxFilCol(As, Ai, Aj)** en *Octave* que reciba como parámetros una matriz dispersa almacenada en formato elemental (representada mediante *tres vectores*) y devuelva la máxima fila y máxima columna de la misma.

La función se podrá invocar de la siguiente forma:
*[maxFil,maxCol] = MaxFilCol(As, Ai, Aj)*

**c)** Escriba y ejecute una función *recursiva* **DarFila(As, Ai, Aj, fil)** en *Octave* que reciba como parámetros una matriz cuadrada dispersa almacenada en formato elemental (representada mediante *tres vectores*) y el número de fila y devuelva una matriz dispersa en formato elemental únicamente con los elementos de la fila *fil*.

La función se podrá invocar de la siguiente forma:
*[As,Ai,Aj] = DarFila(As, Ai, Aj, fil)*

**d)** Complete el código de la función **DetDisp(As, Ai, Aj)** en *Octave* que reciba como parámetros una matriz cuadrada dispersa almacenada en formato elemental (representada mediante ***tres vectores***) y devuelva el determinante de dicha matriz (asumiendo que el determinante es distinto de 0).

**Observación:** para resolver esta función necesita utilizar las funciones anteriores y mezclar iteración y recursión.

```
function det = DetDisp(As, Ai, Aj)
[maxFil,maxCol] = MaxFilCol(As, Ai, Aj);
if maxFil == 1 & maxCol == 1
     det = As(1);
else
     fil = 1; % podría ser cualquiera
     [Bs,Bi,Bj] = DarFila(As, Ai, Aj, fil);
     li = length(Bi);
     det = 0;
     for i=1:li
          ...
          Completar !!!!
          ...
     end
end
```

# *Polinomios*

## Ejercicio 6

**a)** Escribir y probar en Octave una función *recursiva* **`LimpiarCeros(p)`** que recibe un polinomio en formato Octave y elimina todos los coeficientes de más a la izquierda que valgan cero.

**b)** Escribir y probar en *Octave* una función *recursiva* **`SumaPol(v,p)`** que recibe dos polinomios en formato *Octave* y calcula la suma de los mismos. Tenga en cuenta que los coeficientes de los polinomios se pueden anular al realizar la suma.

**c)** Escribir y probar en Octave una función *iterativa* **`MultPol(v,p)`** que recibe dos polinomios en formato Octave y calcula la multiplicación de los mismos.

**Observación:** Las funciones tienen que poder operar con polinomios de distintos grados.

## Ejercicio 7

La regla de Ruffini se utiliza fundamentalmente cuando el polinomio dividendo tiene como única variable la *x* y el divisor es *(x - a)*. Utiliza los coeficientes del dividendo y el valor de *a*, obteniéndose los coeficientes del polinomio cociente y el valor del resto (obsérvese que el resto siempre será un número), disponiéndose en la forma que se muestra en el ejemplo siguiente:

Ejemplo: $(x^3 + x^2 - x - 1)/(x - 2)$

| | | | | |
|---|---|---|---|---|
| | 1 | 1 | -1 | -1 |
| 2 | | 2 | 6 | 10 |
| | 1 | 3 | 5 | 9 (resto) |

Metodología:

- Se deben colocar todos los coeficientes del dividendo ordenados de mayor a menor grado y, si falta el correspondiente a algún grado intermedio, colocar un 0.
- Se "baja" el primer coeficiente del dividendo.
- Se multiplica *a* por el coeficiente bajado y se coloca el resultado debajo del segundo coeficiente (el signo de *a* será positivo si el divisor es del tipo *(x-a)* y negativo si el divisor es del tipo *(x+a)*).
- Se suma el segundo coeficiente con el resultado anterior.
- Se continúa el proceso hasta terminar con los coeficientes.
- Los números de la fila inferior obtenida son los coeficientes del cociente (de un grado menor al dividendo) excepto el último número que es el valor del resto.

Como caso particular cuando *a* es raíz del polinomio dividendo, el resto es 0.

Escribir una función en Octave que implemente el método mencionado recibiendo como parámetro el polinomio $p_n$ y el valor de la raíz a, y devuelve el polinomio resultado y el resto.

## Ejercicio 8

Implementar en Octave una función que obtenga los coeficientes de la función derivada de un polinomio dado.

## Ejercicio 9

**a)** Ingrese un polinomio *p* en Octave y un valor en *x*, luego ejecute **`polyval(p,x)`**.

**b)** Ingrese dos polinomios *p*, *q* (con la misma cantidad de coeficientes) y calcule la suma de la evaluación de los dos en un valor *x*: **`polyval(p,x) + polyval(q,x).`** Verifique con el resultado de hacer **`t = p + q`** y luego ejecutar **`polyval(t,x)`**.

**c)** Vea en Octave **`help conv`** y **`help deconv`**. Multiplique dos polinomios *p* y *q* y luego divida el resultado por uno de los dos.

## Ejercicio 10

Los polinomios de Hermite se definen por las siguientes fórmulas:

*H0(x) = 1*
*H1(x) = 2x*
*Hn(x) = 2xHn-1(x) - 2nHn-2(x)* , para *n>1*

Escriba una función *recursiva* `h = hermite(n)` que calcule el polinomio de Hermite *Hn* en formato Octave.