

## **MATRICES DISPERSAS**

ALAN REYES-FIGUEROA  
MÉTODOS NUMÉRICOS II

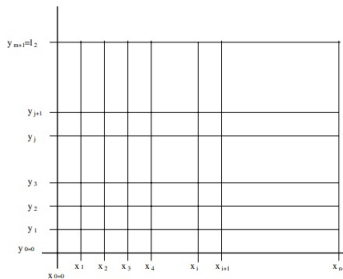
(AULA 13) 23.AGOSTO.2022

# Matrices Dispersas

Consideremos el siguiente problema. Se quiere resolver la ecuación de POISSON en una región rectangular  $D = [0, l_1] \times [0, l_2] \subset \mathbb{R}^2$ .

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f(x, y), \quad (x, y) \in [0, l_1] \times [0, l_2].$$

Para resolver el problema de forma numérica, el primer paso consiste en discretizar el dominio rectangular  $D$ , mediante una malla regular (nodos igualmente espaciados)



# Matrices Dispersas

$$x_i = i\Delta_x, \quad \text{para } i = 0, 1, 2, \dots, n+1,$$

$$y_j = j\Delta_y, \quad \text{para } j = 0, 1, 2, \dots, m+1,$$

Utilizando una aproximación (de diferencias finitas) para las segundas derivadas parciales, se tiene que

$$\frac{\partial^2 u}{\partial x^2}(u_i, u_j) \approx \frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{\Delta_x^2},$$

$$\frac{\partial^2 u}{\partial y^2}(u_i, u_j) \approx \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{\Delta_y^2},$$

donde  $u_{ij} = u(x_i, y_j)$ . Así, la ecuación de Poisson puede aproximarse en la forma

$$\frac{1}{\Delta_x^2}(u_{i-1,j} - 2u_{ij} + u_{i+1,j}) + \frac{1}{\Delta_y^2}(u_{i,j-1} - 2u_{ij} + u_{i,j+1}) = -f_{ij}.$$

Si el mallado tiene  $m \times n$  nodos, las ecuaciones anteriores definen un sistema lineal de tamaño  $mn \times mn$ .

# Matrices Dispersas

Para ello, se ordenan los nodos del mallado de algun modo. Una posibilidad es utilizar el orden dado por la filas (similar a cuando se vectoriza una matriz):

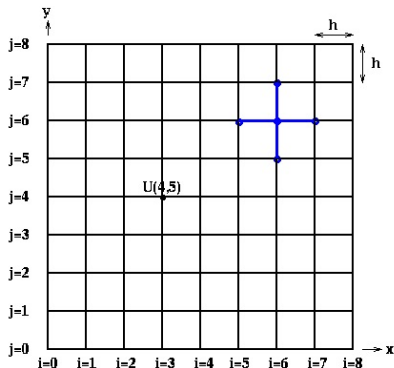
$$l = i + nj, \quad \text{para } i = 0, 1, \dots, n - 1; j = 0, 1, \dots, m - 1.$$

		13	14	15	16
4					
3		9	10	11	12
2		5	6	7	8
1		1	2	3	4
j=1					
	i=1	2	3	4	

Reordenamiento de los nodos  $(i, j)$ .

Así, se obtiene un sistema de ecuaciones de la forma  $A\mathbf{u} = \mathbf{f}$ , con  $A \in \mathbb{R}^{mn \times mn}$ , y  $\mathbf{u}, \mathbf{f} \in \mathbb{R}^{mn}$ . Por lo general,  $A$  es una matriz con poco contenido.

# Matrices Dispersas

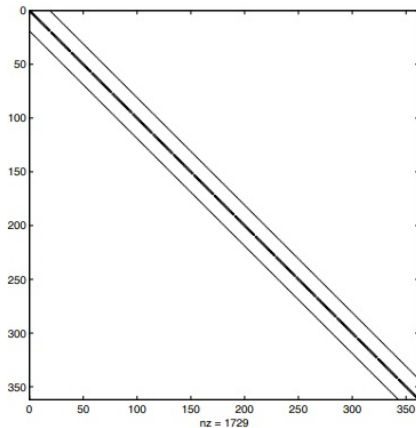


$$4 \cdot U(i,j) - U(i-1,j) - U(i+1,j) - U(i,j-1) - U(i,j+1) = b(i,j)$$

$$\begin{bmatrix}
 4 & -1 & & & & & & \\
 -1 & 4 & -1 & & & & & \\
 & -1 & 4 & -1 & & & & \\
 & & -1 & 4 & & & & \\
 -1 & & & & 4 & -1 & & \\
 & -1 & & -1 & 4 & -1 & & \\
 & & -1 & & -1 & 4 & & \\
 & & & -1 & & -1 & 4 & \\
 & & & & -1 & & -1 & 4
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 U(1,1) \\
 U(2,1) \\
 U(3,1) \\
 U(4,1) \\
 U(1,2) \\
 U(2,2) \\
 U(3,2) \\
 U(4,2) \\
 U(1,3) \\
 U(2,3) \\
 U(3,3) \\
 U(4,3) \\
 U(1,4) \\
 U(2,4) \\
 U(3,4) \\
 U(4,4)
 \end{bmatrix}
 =
 \begin{bmatrix}
 b(1,1) \\
 b(2,1) \\
 b(3,1) \\
 b(4,1) \\
 b(1,2) \\
 b(2,2) \\
 b(3,2) \\
 b(4,2) \\
 b(1,3) \\
 b(2,3) \\
 b(3,3) \\
 b(4,3) \\
 b(1,4) \\
 b(2,4) \\
 b(3,4) \\
 b(4,4)
 \end{bmatrix}$$

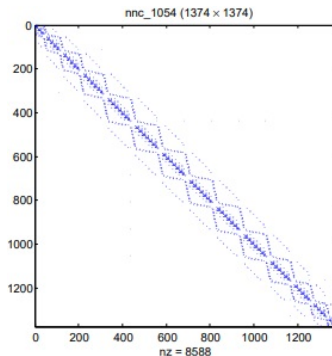
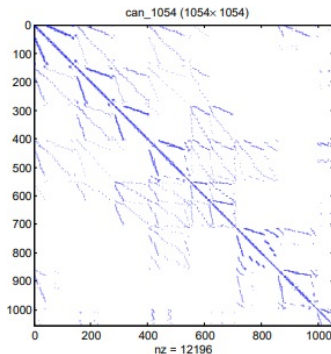
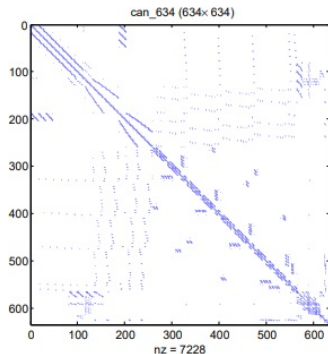
(a) Conectividad de cada nodo con sus 4 vecinos. (b) Matriz A para el mallado de  $4 \times 4$ .

# Matrices Dispersas



Matriz A para la ecuación de Poisson 2D.

# Matrices Dispersas



Otros ejemplos de matrices comunes en aplicaciones.

<https://math.nist.gov/MatrixMarket/>  
<https://sparse.tamu.edu/>

# Matrices Dispersas

Se llama **matriz dispersa** o **matriz rala** (*sparse matrix*) a una matriz cuyos elementos la mayoría son cero.

En el caso de matrices dispersas, se puede hacer uso de técnicas especiales para obtener ventaja del gran número de elementos ceros que posee.

Observaciones:

- Algunos autores definen una matriz  $n \times n$  como dispersa si el número de elementos no nulos se comporta como  $n^{\gamma+1}$ , con  $0 < \gamma < 1$ .
- Valores típicos de  $\gamma$  son:
  - ★  $\gamma = 0.2$  para problemas de análisis de sistemas eléctricos de generación, y transporte de energía;
  - ★  $\gamma = 0.5$  para matrices asociadas a problemas de análisis de estructuras.
- Estas matrices ralas aparecen en muchas aplicaciones: solución de EDPs, problemas de transporte, modelación de redes, estructuras, métodos de aprendizaje automático.

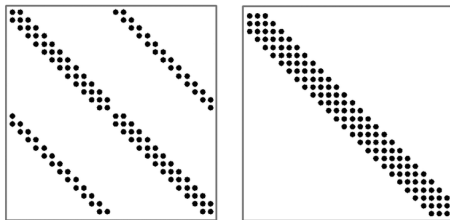


# Matrices Dispersas

Hay dos tipos de matrices dispersas:

- Estructuradas: Los elementos no cero forman un patrón regular, por ejemplo, se agrupan a lo largo de un número pequeño de diagonales.

Un caso especial de matrices estructuradas son las matrices banda



- No estructuradas: Los elementos no cero se distribuyen de forma irregular.

En el primer caso se pueden diseñar métodos basados en la estructura de las matrices, mientras que en el segundo caso sólo se puede hacer uso de la *raleza* de la matriz.

# Formatos de Almacenamiento

Dada una matriz  $A \in \mathbb{R}^{m \times n}$ , llamaremos  $n_z$  al número de elementos no nulos de  $A$ . Revisamos alguno de los esquemas de almacenamiento más comunes para matrices dispersas.

## Esquema coordinado: (COO)

Con este esquema, para representar la matriz  $A$  se utilizan tres vectores  $\mathbf{v}_A$ ,  $I_A$ , y  $J_A$  de dimensión  $n_z$ .

- En el vector  $\mathbf{v}_A$  se almacenan los elementos no nulos de  $A$  (los valores  $a_{ij}$ );
- en  $I_A$ , se almacenan los números de fila  $i$  asociados a cada entrada no nula  $a_{ij}$  de  $A$ ;
- y en  $J_A$  el número de columna  $j$  asociado a cada elemento no nulo de  $A$ .

# Formatos de Almacenamiento

**Ejemplo:** La matriz

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 0 & 7 & 8 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix}$$

se escribe en el formato coordenado, como los vectores

$$\mathbf{v}_A = (10 \ 8 \ 5 \ 1 \ 2 \ 3 \ 6 \ 4 \ 7 \ 9),$$

$$I_A = (5 \ 3 \ 2 \ 1 \ 1 \ 2 \ 3 \ 2 \ 3 \ 4),$$

$$J_A = (5 \ 5 \ 4 \ 1 \ 4 \ 1 \ 1 \ 2 \ 4 \ 3).$$

Cuidado! En el formato de indexado de Python, tendríamos

$$\mathbf{v}_A = (10 \ 8 \ 5 \ 1 \ 2 \ 3 \ 6 \ 4 \ 7 \ 9),$$

$$I_A = (4 \ 2 \ 1 \ 0 \ 0 \ 1 \ 2 \ 1 \ 2 \ 3),$$

$$J_A = (4 \ 4 \ 3 \ 0 \ 3 \ 0 \ 0 \ 1 \ 3 \ 2).$$

# Formatos de Almacenamiento

## Esquema CSR: (*Compressed sparse row*)

Si se introducen los elementos de la matriz ordenados por filas, es posible utilizar un formato más compacto denominado CSR (formato comprimido de filas).

En este caso

- $\mathbf{v}_A$ , de dimensión  $n_z$ , que contiene los elementos no nulos de  $A$ , ordenados por filas;
- $J_A$ , también de dimensión  $n_z$ , que contiene los números de las columnas  $j$  de los elementos no nulos  $a_{ij}$  de  $A$ ,
- $I_A$ , de dimensión  $m + 1$  con la siguiente estructura:

$$\begin{aligned} I_A(1) &= 1, \\ I_A(i+1) - I_A(i) &= \text{número de elementos no nulos en la fila } i. \end{aligned}$$

De esta última ecuación se tiene que  $I_A(i+1)$  corresponde a

$$I_A(i+1) = 1 + \#\{\text{elementos no nulos en las primeras } i \text{ filas}\}.$$

# Formatos de Almacenamiento

## Esquema CSC: (*Compressed sparse column*)

Análogamente, existe un formato similar, pero con los elementos de la matriz ordenados por columnas. Este formato se llama CSC (formato comprimido de columnas).

En este caso

- $\mathbf{v}_A$ , de dimensión  $n_z$ , que contiene los elementos no nulos de  $A$ , ordenados por filas;
- $I_A$ , también de dimensión  $n_z$ , que contiene los números de las filas  $i$  de los elementos no nulos  $a_{ij}$  de  $A$ ,
- $J_A$ , de dimensión  $n + 1$  con la siguiente estructura:

$$\begin{aligned} J_A(1) &= 1, \\ J_A(j+1) - I_A(j) &= \text{número de elementos no nulos en la columna } j. \end{aligned}$$

De esta última ecuación se tiene que  $J_A(j+1)$  corresponde a

$$J_A(j+1) = 1 + \#\{\text{elementos no nulos en las primeras } j \text{ columnas}\}.$$

# Formatos de Almacenamiento

Así, la matriz  $A$  en el formato CSR se representa por

$$\mathbf{v}_A = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10),$$

$$J_A = (1 \ 4 \ 1 \ 2 \ 4 \ 1 \ 4 \ 5 \ 3 \ 5),$$

$$I_A = (1 \ 3 \ 6 \ 9 \ 10 \ 11).$$

Mientras que  $A$ , en el formato CSC se representa por

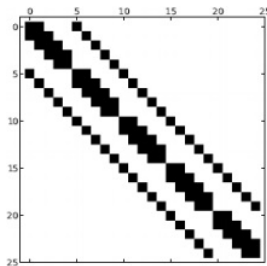
$$\mathbf{v}_A = (1 \ 3 \ 6 \ 4 \ 9 \ 2 \ 5 \ 7 \ 8 \ 10),$$

$$I_A = (1 \ 2 \ 3 \ 2 \ 4 \ 1 \ 2 \ 3 \ 3 \ 5),$$

$$J_A = (1 \ 4 \ 5 \ 6 \ 9 \ 11).$$

# Formatos de Almacenamiento

**Esquema DIA: (*/Diagonal*)** El esquema diagonal se usa cuando los elementos no nulos se restringen a un reducido número de diagonales.



El formato diagonal está formado por una matriz de datos (que almacena los valores no nulos  $a_{ij}$ ) y un vector con los *offsets* (que almacena el *offset* de cada diagonal con respecto a la diagonal principal).

# Formatos de Almacenamiento

Por convención, a la diagonal principal le corresponde el *offset* 0. Los valores de  $i$  positivos corresponden a diagonales superiores, mientras que los negativos a las inferiores.

Dada la matriz

$$A = \begin{pmatrix} 1 & 7 & 0 & 0 \\ 0 & 2 & 8 & 0 \\ 5 & 0 & 3 & 9 \\ 0 & 6 & 0 & 4 \end{pmatrix}.$$

El formato diagonal viene dado por

$$dat = \begin{pmatrix} * & 1 & 7 \\ * & 2 & 8 \\ 5 & 3 & 9 \\ 6 & 4 & * \end{pmatrix}, \quad off = (-2 \quad 0 \quad 1).$$



# Formatos de Almacenamiento

Una de las librerías más usadas para el manejo de matrices ralas es SPARSKIT. En Python, por ejemplo dentro de Scipy, tenemos el módulo SPARSE (`scipy.sparse` y `scipy.sparse.linalg`).

Esta librería incluyen funciones para la conversión entre formatos ralos:

- DNS Formato denso
- BND Linpack Banded format
- BSR Block Sparse Row format
- CSR Compressed Sparse Row format
- CSC Compressed Sparse Column format
- COO Coordinate format
- DIA Diagonal format
- LIL Row-based list of lists format
- ...

# Operaciones con Matrices Ralas

¿Qué ocurre con las operaciones relacionadas con matrices ralas?

¿Funcionan los mismos métodos que ya sabemos de álgebra lineal?

**Respuesta: No.** Hay que adaptar toda el álgebra lineal a estos formatos.

## Ejemplo:

Una de las operaciones más importantes en los métodos iterativos para la solución de sistemas de ecuaciones lineales es el producto matriz-vector.

En el esquema CSR, resulta

**Algoritmo:** (Producto matriz-vector, esquema CSR)

*Inputs:*  $A \in \mathbb{R}^{n \times n}$  en formato CSR,  $\mathbf{x} \in \mathbb{R}^n$ .

*Outputs:*  $\mathbf{y} = \mathbf{Ax} \in \mathbb{R}^n$ .

For  $i = 0, 2, \dots, n - 1$ :

$$k_1 = I_A[i].$$

$$k_2 = I_A[i + 1]:$$

$$\mathbf{y}[i] = \mathbf{v}_A[k_1 : k_2]^T \mathbf{x}[J_A[k_1 : k_2]].$$

Return  $\mathbf{y}$ .

# Operaciones con Matrices Ralas

**Algoritmo:** (Producto matriz-vector, esquema CSC)

*Inputs:*  $A \in \mathbb{R}^{n \times n}$  en formato CSC,  $\mathbf{x} \in \mathbb{R}^n$ .

*Outputs:*  $\mathbf{y} = A\mathbf{x} \in \mathbb{R}^n$ .

For  $j = 0, 2, \dots, n - 1$ :

$$k_1 = I_A[j].$$

$$k_2 = I_A[j + 1]:$$

$$\mathbf{y}[J_A[k_1 : k_2]] = \mathbf{y}[J_A[k_1 : k_2]]^T \mathbf{x}_j \mathbf{v}_A[k_1 : k_2].$$

Return  $\mathbf{v}$ .