Métodos numéricos para sistemas de ecuaciones

(Prácticas)

Damián Ginestar Peiró

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Índice general

Práctica 2

Matrices dispersas

2.1. Introducción

Hay muchas aplicaciones que dan lugar a tener que manejar matrices que tienen gran cantidad de sus elementos nulos. Estas matrices se denominan *matrices dispersas*.

Se puede dar una definición informal de matriz dispersa, como *aquella matriz que tiene suficientes ceros de forma que vale la pena tener esto en cuenta*. De este modo, evitando operaciones sobre los elementos que son cero se ahorra tiempo de computación y no guardando los elementos nulos se ahorra en memoria.

Para manejar estas matrices se trata de aprovechar el hecho que tienen gran cantidad de elementos nulos para no almacenar dichos elementos, con lo que ello supone de ahorro en memoria y en tiempo de computación.

Para poder trabajar con las matrices sin almacenar los elementos nulos se utilizan distintos esquemas de almacenamiento.

2.2. Esquemas de almacenamiento

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

2.2.1. Esquema coordenado

Con este esquema, para representar la matrix A se utilizan tres vectores AA, IA, y JA de dimensión n_z. En el vector AA se almacenan los elementos no nulos de A, en IA, se almacenan los números de fila asociados a cada elemento de A y en JA el número de columna asociado a cada elemento de A. Por ejemplo, para la matriz

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

su almacenamiento en formato coordenado viene dado por los vectores

$$\begin{aligned} AA &= \begin{pmatrix} 12 & 9 & 7 & 5 & 1 & 2 & 11 & 3 & 6 & 4 & 8 & 10 \end{pmatrix} , \\ IA &= \begin{pmatrix} 5 & 3 & 3 & 2 & 1 & 1 & 4 & 2 & 3 & 2 & 3 & 4 \end{pmatrix} , \\ JA &= \begin{pmatrix} 5 & 5 & 3 & 4 & 1 & 4 & 4 & 1 & 1 & 2 & 4 & 3 \end{pmatrix} . \end{aligned}$$

Observad que con este formato no es necesario introducir los elementos de la matriz ordenados por filas o por columnas. Éste es esencialmente el formato que utiliza Matlab para el manejo de las matrices dispersas.

2.2.2. Formato CSR

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

Este formato hace uso de los vectores: AA, de dimensión n_z, que contiene los elementos no nulos de A ordenados por filas, JA, también de dimensión n_z, que contiene los números de las columnas de los elementos de A, e IA, de dimensión $m+1$ con la siguiente estructura:

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

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

$$\begin{aligned} AA &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix} , \\ JA &= \begin{pmatrix} 1 & 4 & 1 & 2 & 4 & 1 & 3 & 4 & 5 & 3 & 4 & 5 \end{pmatrix} , \\ IA &= \begin{pmatrix} 1 & 3 & 6 & 10 & 12 & 13 \end{pmatrix} . \end{aligned}$$

Análogamente, se puede definir un formato de almacenamiento por columnas que se llama CSC.

Hay que hacer notar que es necesario redefinir las operaciones usuales con matrices según el formato de almacenamiento que se utilice. Por ejemplo, la siguiente función implementa el producto matriz vector, cuando la matriz está almacenada en formato CSR.

```
function vec1=mymult(AA,IA,JA,N,vec)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   vec1=mymult(AA,IA,JA,N,vec)
% funcion para calcular el producto matriz vector
% Entrada:
%      AA, IA, JA, matriz A almacenada
%      en formato CSR.
%      vec, vector
% Salida:
%      vec1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:N
   k1=IA(i);
   k2=IA(i+1)-1;
   sum=0.;
   for j=k1:k2
      sum=sum+AA(j)*vec(JA(j));
   end
   vec1(i)=sum;
end
   vec1=vec1';
```

2.3. Matrices dispersas en Matlab

Para las matrices Matlab tiene dos tipos de almacenamiento, el denso (full) y el disperso (sparse). Dada una matriz A

```
A=[0 0 11; 22 0 0; 0 33 0];
```

la instrucción

```
S=sparse(A)
```

da como resultado

```
S=
   (2,1)    22
   (3,2)    33
   (1,3)    11
```

así, una matriz dispersa se muestra como una lista de los elementos no nulos indicando la fila y la columna donde se encuentran. Los elementos se muestran en orden de columnas creciente.

Como ya hemos comentado, un parámetro importante para describir una matriz dispersa, a parte del número de filas, n, y el número de columnas, m, es el número de elementos no nulos n_z. La función que calcula el número de elementos no nulos de A es `nnz(A)`, que en este caso es 3.

En ocasiones, la visualización gráfica de la estructura de una matriz dispersa es una herramienta útil.

Veamos algunos ejemplos. Consideramos una cierta malla triangular que se genera con las siguientes instrucciones:

```
nv = 12;
P = haltonset(2);
V = net(P,nv);
DT = DelaunayTri(V);
h = triplot(DT,'k-','Marker','.','MarkerSize',12);
axis square;
axis off
text(V(:,1),V(:,2),cellstr(num2str((1:nv)')),'FontSize',14,...
    'Color','r');
```

obteniendo el gráfico de la Figura 2.1.

Se puede representar la concetividad de la malla construyendo la matriz de adyacéncia A, de modo que a_{ij} es 1 si hay una línea del vértice i al vertice j. Si no hay una línea el elemento de matriz es cero. Esta matriz se puede construir con las instrucciones:

```
A = zeros(nv);
for j=1:size(DT,1)
A(DT(j,1),DT(j,:)) = 1;
A(DT(j,2),DT(j,:)) = 1;
A(DT(j,3),DT(j,:)) = 1;
end
Ad=sparse(A)      %Si se almacena en formato disperso
```

Podemos visualizar el patrón de dispersidad de la matriz de adyacéncia

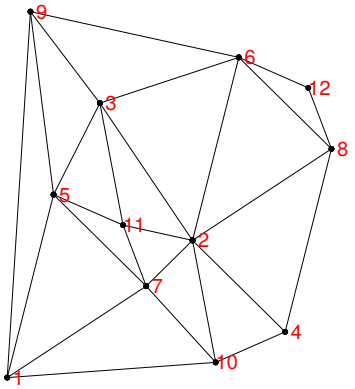


Figura 2.1: Malla triangular.

haciendo `spy(A)` obteniendo la Figura 2.2, donde los elementos no nulos de la matriz se representan mediante un símbolo.

Otro ejemplo, lo constituye la malla que define la cúpula geodésica de Fuller. Richard Buckminster Fuller es considerado el inventor de las cúpulas geodésicas, ya que es quien ostenta su patente en 1954. Fuller las desarrolló en la década de los 40, creando una de las cúpulas geodésicas más conocidas en 1967 en la Exposición Universal de Montreal, de 76 m de diámetro y 41,5 m de altura, que se muestra en la Figura 2.3.

Podemos visualizar el patrón de dispersidad de la matriz de adyacencia haciendo `spy(A)` obteniendo la Figura 2.2.

Se puede hacer una representación de una estructura geodésica conocida como la *Bucky ball*, mediante las siguientes instrucciones:

```
[B,V] = bucky;
clf;
[i,j] = find(B);
[~, p] = sort(max(i,j));
i = i(p);
j = j(p);
X = [V(i,1) V(j,1)]';
Y = [V(i,2) V(j,2)]';
Z = [V(i,3) V(j,3)]';
X = [X; NaN(size(i))'];
Y = [Y; NaN(size(i))'];
Z = [Z; NaN(size(i))'];
% Plot the vertices
plot3(X,Y,Z,'k-','LineWidth',2); hold on;
```

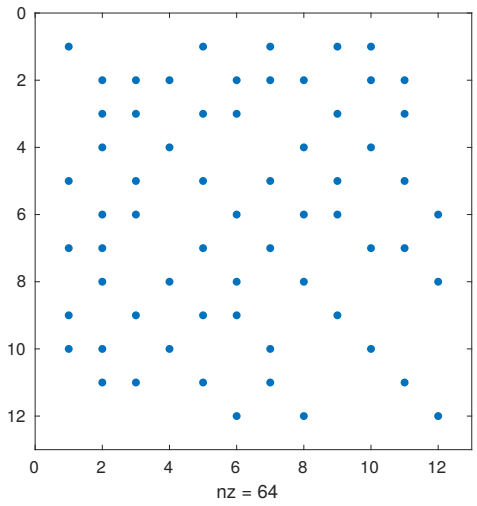


Figura 2.2: Patrón de dispersidad de la malla triangular.

```
plot3(V(:,1),V(:,2),V(:,3),'b.','MarkerSize',40);
% Add a sphere
sphere(100);
% Lighting and shading effects on the sphere
colormap([0.4 0.4 0.4])
alpha(0.5);
% transparency
shading interp;
lighting phong;
camlight left;
% Lighting
% Adjust aspect ratio, view angle, add title.
daspect([1 1 1])
view(42,10);
axis off;
hold off;
snapnow;
```

La estructura geodésica que se muestra en la Figura 2.4.

La matriz de adyacencia de esta estructura es una matriz dispersa. Si ejecutamos

```
A=bucky;
spy(A)
```

obtenemos la gráfica mostrada en la Figura 2.5,

Figura 2.3: Cúpula geodésica en la Exposición Universal de Montreal.

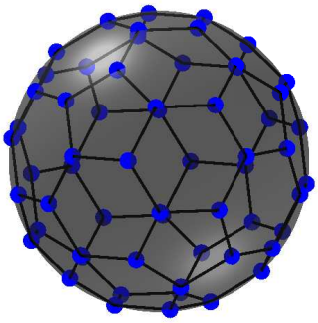


Figura 2.4: Cúpula geodésica de Fuller.

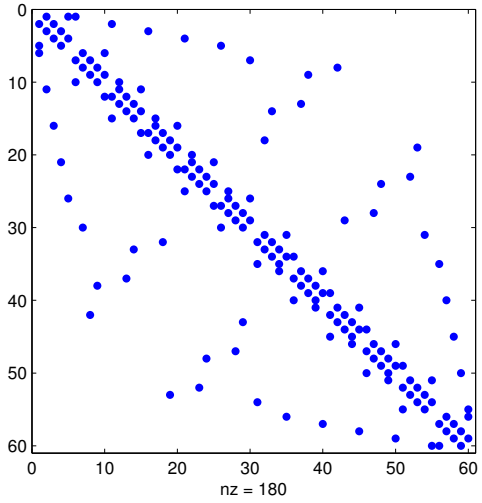


Figura 2.5: Visualización de una matriz dispersa.

Para crear una matriz dispersa, también es posible utilizar la instrucción

```
S=sparse(I,J,s,n,m,nzmax)
```

donde `I`, y `J` son vectores indicando las filas y las columnas de los elementos no nulos de la matriz. `s` es otro vector de reales o de complejos que contiene estos elementos. `m` es el número de filas, `n` el número de columnas y `nzmax` es el número máximo de elemenos no nulos que se reservan en la memoria. Si un par de índices fila y columna se repiten en los vectores `I` y `J` en el resultado los valores de `s` se suman.

Hay varias maneras de simplificar la llamada a la función `sparse( )`

`S=sparse(I,J,s,n,m)` utiliza `nzmax=length(s)`.

`S=sparse(I,J,s)` utiliza `n=max(I)`, `m=max(J)`.

Por otro lado, la función `find( )` nos permite crear los vectores con los índices de fila y de columna y el vector de elementos no nulos de una matriz. Para una matriz `A` en formato denso o disperso se puede ejecutar,

```
[I,J,s]=find(A)
```

Hay distintas funciones de Matlab que permiten construir matrices dispersas directamente así, por ejemplo, `speye( )` genera la matriz identidad en formato disperso. La función `sprandn( )` genera una matriz dispersa de números aleatorios que se distribuyen normalmente. La instrucción `sprand(m,n,density)`, genera una matriz $n \times m$ con aproximadamente `density`$\times$ `n` $\times$ `m` elementos no nulos.

Por ejemplo, se tiene que al hacer

```
AS=sprandn(50,50,0.1)
nnz(AS)
```

se obtiene un valor de `nnz`=244. Mientras que al hacer

```
AS=sprandn(50,50,0.5)
nnz(AS)
```

se obtiene un valor de `nnz`=974.

Algunos métodos de resolución de sistemas son sólo son válidos para matrices simétricas y definidas positivas. Se puede generar una matriz dispersa aleatoria con números distribuidos uniformemente en $[-1, 1]$ con la función `sprandsym(n,density)`.

La función `spdiags( )` es similar a la función `diag( )`, pero da como resultado una matriz dispersa.

Para determinar si una matriz puede considerarse como dispersa, podemos calcular la 'proporción de dispersidad', que es

```
sparsity_ratio = 1 - nnz(A)/numel(A)
```

consideraremos que una matrix es dispersa si esta proporción es cercana a uno.

El ancho de banda de una matriz se define como la máxima distancia de los elementos no nulos a la diagonal principal y se puede calcular como

```
[i,j] = find(A);
bandwidth = max(abs(i-j))
```

Así, una matriz es banda si su ancho de banda es 'pequeño'.

Podemos generar la matrix de un problema de Poisson 20×20 utilizando el comando

```
 A=gallery('poisson',20,20);
nnz(A)
nnz(A^2)
nnz(A^8)
```

obteniendo los valores

```
  1920
  4804
 42520
```

observamos que el número de elementos no nulos crece con cada multiplicación de la matriz. Este fenónmeno se denomina *relleno* y se da en muchas operaciones con matrices dispersas.

2.4. Cálculo del vector page-rank

Otra aplicación donde aparecen las matrices dispersas es en el cálculo del vector *page-rank* que propousieron los fundadores del buscador Google (Sergey Brin y Lawrence Page) para clasificar las páginas web según su importancia. La finalidad del método es construir un vector que proporcione la importancia relativa de cada hoja web.

Para ver brevemente cómo se construye dicho vector, llamaremos W al conjunto de páginas web que se pueden visitar siguiendo una cadena de hipervínculos comenzando en una cierta página raíz. Sea n el número de páginas web en W.

W se puede ver como un grafo dirigido donde las páginas web son los nodos y las aristas son los hipervínculos que hay entre las distintas hojas web. Un ejemplo de estos grafos se muestra en la Figura 2.6.

En `grafoupv.mat` se encuentra el grafo asociado a 100 hojas web conectadas con `https://www.upv.es`. Estos datos se generaron un rastreador de páginas automático (`surfer.m` de Cleve Moler). El rastreador de página comenzó en `https://www.upv.es` y siguió enlaces a páginas web subsiguientes hasta que la matriz de adyacencia contenía información sobre las conexiones de 100 páginas web únicas. Para visualizar el grafo dirigido asociado a esta web hacemos

```
load grafo_upv.mat
G = digraph(A,U)
```

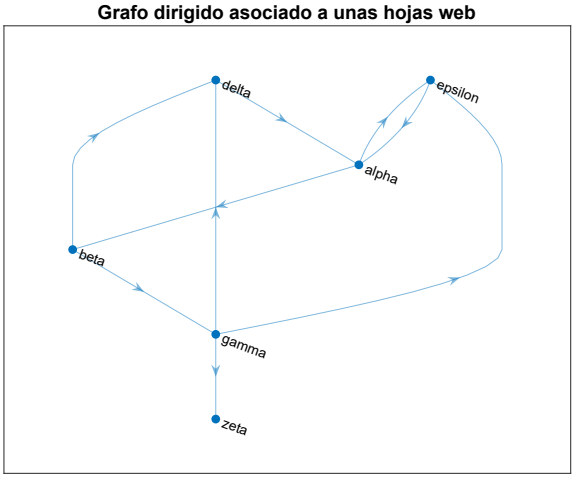


Figura 2.6: Visualización de un conjunto de hojas web como un grafo dirigido.

```
plot(G,'NodeLabel',{},'NodeColor',[0.93 0.78 0],'Layout','force');
title('Hojas web conectadas con https://www.upv.es.com')
```

La estructura del grafo obtenido se muestra en la Figura 2.7.

El patrón de dispersidad de la matriz de adyacencia del grafo se muestra en la Figura 2.8, con lo que tenemos que esta matriz es dispersa.

Llamaremos G a la matriz de conectividad (de adyacencia) del grafo que, en general, será una matriz grande y dispersa. El número de elementos no nulos de G es el número total de hipervínculos en W.

Denotamos por

$$r_i = \sum_j g_{ij}, \quad c_j = \sum_i g_{ij},$$

a los grados de entrada y salida de la página j. El proceso de ir visitando las distintas hojas web de W se puede ver como un paseo aleatorio (random walk). Si llamamos p a la probabilidad de visitar una cierta página web siguiendo un hipervínculo, la probabilidad de visitar una cierta página de forma aleatoria es

$$\delta = \frac{1}{n}(1-p).$$

El número de enlaces a otras páginas webs presentes en la página j, es c_j. Así, para otra página i, la probabilidad a_{ij} de que el *paseante aleatorio*

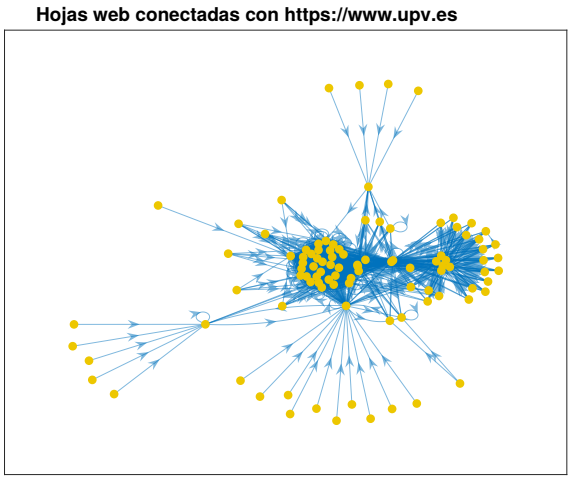


Figura 2.7: Visualización del grafo asociado con 100 hojas conectadas con `https://www.upv.es`.

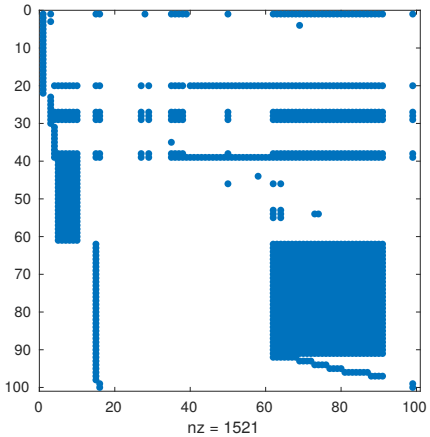


Figura 2.8: Matriz de ayaciencia del grafo asociado con 100 hojas conectadas con `https://www.upv.es`.

llegue a la hoja i cuando esté visitando la hoja j, viene dada por

$$a_{ij} = \begin{cases} pg_{ij}\frac{1}{c_j} + \delta & \text{si } \ c_j \neq 0 \\ \frac{1}{n} & \text{si } \ c_j = 0 \end{cases}.$$

Los elementos a_{ij} de la matriz A están en el intervalo $[0, 1]$ y los elementos de cada columna suman 1, por tanto, la matriz A es una matriz estocástica.

Se pretende calcular un vector x cuya componente i-ésima es la probabilidad que el visitante elija como primera página para visitar la página i, partiendo de un vector con iguales probabilidades

$$x_0 = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right).$$

El cálculo de estas probabilidades se puede modelizar mediante un proceso de la forma

$$x_{k+1} = Ax_k, \quad k = 0, 1, 2, \ldots$$

que al ser A una matriz estocástica es un proceso de Markov. El vector *page-rank* es el límite del proceso de Markov,

$$Ax = x,$$

que no es más que el autovector asociado al autovalor 1 de A cuyas componentes suman 1. Este autovector se podría calcular como una solución del sistema homogéneo

$$(A - I)\, x = 0.$$

Por ejemplo, dado el grafo que se muestra en la Figura 2.9, Suponiendo que $\delta = 0$, la matriz de transición para este grafo es,

$$A = \begin{pmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

Se puede plantear el sistema homogéneo,

$$\begin{aligned} x_1 &= x_3 + \frac{1}{2}x_4, \\ x_2 &= \frac{1}{3}x_1, \\ x_3 &= \frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_4, \\ x_4 &= \frac{1}{3}x_1 + \frac{1}{2}x_2. \end{aligned}$$

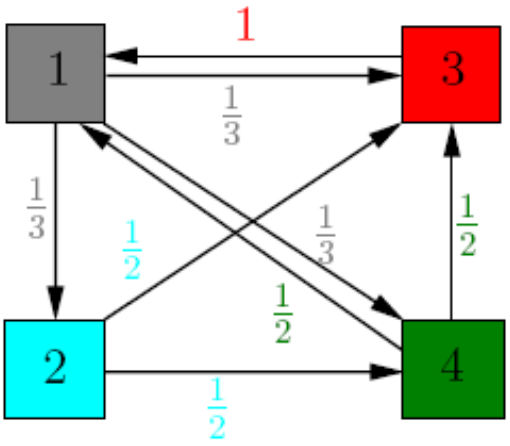


Figura 2.9: Grafo sencillo.

Imponiendo que $x_1 = 1$ se obtiene la solución

$$x_1 = 1,\ x2_{=}\frac{1}{3},\ x_3 = \frac{3}{4},\ x_4 = \frac{1}{4},$$

como las components del vector x han de sumar 1, tenemos que

$$x = \frac{12}{31}\left(1, \frac{1}{3}, \frac{3}{4}, \frac{1}{4}\right).$$

En general, este método está mal condicionado y se usan otras técnicas basadas en el cálculo del autovalor dominante de una matriz, como el método de la potencia.

2.5. Ejercicios

1. Los vectores almacenados en los ficheros `V.dat`, `I.dat` y `J.dat` definen una matriz dispersa, A en formato coordenando. Construye con ellos una matriz dispersa de Matlab y obtén el tamaño y los elementos no nulos de la matriz. Visualiza la estructura de la matriz. Obtén los elementos de la diagonal de la matriz. Calcula $I - A$ y almacena el resultado en formato coordenado.

2. Construye una función que pase una matriz almacenada en el formato coordenado al formato CSR. Obtén el producto de la matriz definida

por los vectores dados en los ficheros `V.dat`, `I.dat` y `J.dat` en formato coordenado por el vector que tiene la primera coordenada igual a 1 y las otras cero. (observa que así se obtiene la primera columna de la matriz).

3. Construye una función que pase del formato de almacenamiento denso al formato CSR. Utiliza esta función para generalizar la función `mymult( )`, que se ha presentado antes, para que calcule el producto de dos matrices en formato CSR y devuelva el resultado en este formato.

4. Construye una función que pase del formato CSR al formato coordenado. Utiliza esta función y la función que pasa del formato coordenado al CSR para construir una función que calcule la transpuesta de una matriz en formato CSR. Esta función es equivalente a una función que pasa del formato CSR al formato CSC.

5. Un vector v en formato disperso se puede almacenar con la ayuda de dos vectores Jv y vv. El primero nos da las posiciones con elementos no nulos y el segundo el valor de los elementos no nulos. En la última posición de Jv se almacena la dimension del vector. Construye una función que sume dos vectores en formato disperso y devuelva el resultado en formato disperso.

6. Construye la matriz del problema de Laplace si se usa el método de las diferencias finitas con un mallado de 100×100 nodos para un rectángulo de 4×5. Visualiza el patrón de elementos no nulos de la matriz. Construye tres vectores IA, JA y AA que contengan esta matriz en formato coordenado y guardalos en el disco. Comprueba que la matriz obtenida es simétrica.

7. Las instrucciones

```
A=gallery('poisson',50,50);
B=gallery('wathen',40,40);
```

generan dos matrices dispersas que provienen de la discretización de ecuaciones en derivadas parciales. Pasa estas matrices al formato denso y compara el tiempo que tarda Matlab en resolver un sistema de ecuaciones asociado a estas matrices, donde el término independiente es el vector de unos. Calcula el patrón de dispersidad y el ancho de banda de estas matrices. Visualiza el patrón de elementos no nulos de

las matrices $A, A^2, \cdots, A^8$ y calcula el patrón de disparsidad de cada una de las matrices. ¿Qué ocurre?.

8. Dada una matriz dispersa A en formato coordenado y en formato CSR, construye funciones de matlab que realicen las siguientes operaciones en los dos formatos:

   a) Cuente los elementos no nulos en la diagonal principal.

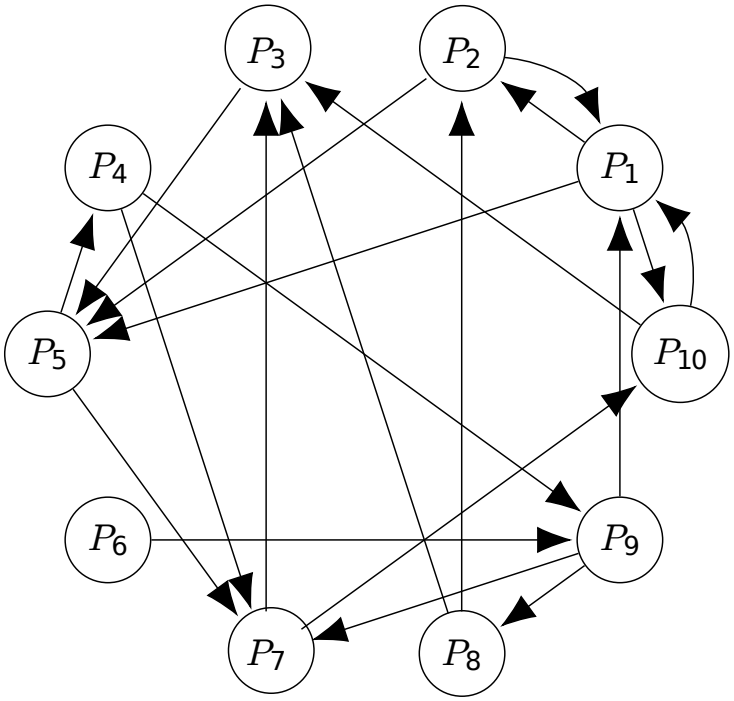
   b) Sume un elemento a, no nulo, en la posición (i, j) de la matriz A.

   c) Extraiga la diagonal con offset=k.

9. Construye una función de matlab que dada una matriz en formato CSR, le asigne en la posición (i, j) un elemento a, y devuelva la nueva matriz en el formato CSR. Comprueba el funcionamiento de la función con un ejemplo.

10. Sean $A_1, A_2 \in \mathbb{R}^{m \times n}$ matrices dispersas en formato CSR. Construye una función que sume las dos matrices en formato CSR.

11. Dada la siguiente estructura de páginas web



Suponiendo que la probabilidad de elegir un hipervículo es $p = 0.1$ y utilizando la siguiente función,

```
function A = createpgmatrix(links,pagecount,p)
% Uso:
% links = [1,2;1,3;2,3;3,4;4,3];
% pagecount = 4;
% p= 0.15 probabilidad aleatoria de elegir un link
%

part1 = ones(pagecount,pagecount);
part2 = zeros(pagecount,pagecount);
linksize = size(links);
numlinks = linksize(1);
for i = [1:numlinks]
 part2(links(i,2),links(i,1)) = 1;
end
for i = [1:pagecount]
if (sum(part2(:,i)) > 0)
part2(:,i) = part2(:,i)/sum(part2(:,i));
else
part2(:,i) = ones(pagecount,1)/pagecount;
end
end
A = p/pagecount*part1 + (1-p)*part2;
end
```

Obtén la matriz de transiciones y calcula el vector 'page rank'.