Exercice

Soit $\mathbb{A} \in \mathcal{M}_N(\mathbb{R})$ avec $N = d \times n$ une matrice tridiagonale blocs

$$\mathbb{A} = \begin{pmatrix} \mathbb{D} & \mathbb{F} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ \mathbb{E} & \mathbb{D} & \mathbb{F} & \ddots & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{E} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \mathbb{0} & \ddots & \ddots & \mathbb{0} & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{F} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{E} & \mathbb{D} & \mathbb{F} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{E} & \mathbb{D} \end{pmatrix}, \text{ avec } \mathbb{D}, \mathbb{E} \text{ et } \mathbb{F} \text{ dans } \mathcal{M}_n(\mathbb{R})$$

Q. 1 Matrice pleine

On prend `n=2;d=7;N=d*n;` *avec* `E=eye(n);D=2*E;F=3*E;` *Comment calculer* $\mathbb{A}$? *(voir* `exemple03.m`*)*

Q. 2 Matrice creuse

On prend `n=2;d=7;N=d*n;` *avec* `E=speye(n);D=2*E;F=3*E;` *Comment calculer* $\mathbb{A}$? *(voir* `exemple03s.m`*)*

Q. 3

On se donne les 3 matrices creuses $\mathbb{D}$, $\mathbb{E}$ *et* $\mathbb{F}$ *dans* $\mathcal{M}_n(\mathbb{R})$

1. *Ecrire la fonction Assemble retournant la matrice creuse* $\mathbb{A} \in \mathcal{M}_N(\mathbb{R})$. *(voir* `Assemble.m`*)*
2. *Ecrire la fonction AssembleSpVec retournant la matrice creuse* $\mathbb{A} \in \mathcal{M}_N(\mathbb{R})$ *sans utiliser de boucle. (voir* `AssembleSpVec.m`*)*

Rappel Pour initialiser le bloc (1,1) : A(1:n, 1:n) = D;

Algo générique

```
I ← 1:n
pour  i ← 1 : d
    A(I,I) ← D
    I ← I+n
fin
I ← 1:n
pour  i ← 1 : d-1
    A(I, I+n) ← F
    A(I+n, I) ← E
    I ← I+n
fin
```

ou

```
I ← 1:N ; Ip ← I+n
A(I,I) ← D
A(I,Ip) ← F
Im ← I ; I ← Ip ; Ip ← I+n
pour  i ← 2 : d-1
    A(I,I) ← D
    A(I, Im) ← E
    A(I, Ip) ← F
    Im ← I ; I ← Ip ; Ip ← I+n
fin
A(I,I) ← D
A(I, Im) ← E
```

ou ...

Q1

```
clear all;close all
n=2;d=7;N=d*n;
E=eye(n);D=2*E;F=3*E;
A=zeros(N,N);
I=1:n; % ligne bloc courante
A(I,I)=D;A(I,I+n)=F;
I=I+n;
for i=2:d-1
  A(I,I)=D;
  A(I,I-n)=E;
  A(I,I+n)=F;
  I=I+n;
end
A(I,I)=D;A(I,I-n)=E;
```

Q2

```
clear all;close all
n=2;d=7;N=d*n;
E=speye(n);D=2*E;F=3*E;
A=sparse(N,N);
I=1:n; % ligne bloc courante
A(I,I)=D;A(I,I+n)=F;
I=I+n;
for i=2:d-1
  A(I,I)=D;
  A(I,I-n)=E;
  A(I,I+n)=F;
  I=I+n;
end
A(I,I)=D;A(I,I-n)=E;
```

Q3/ ① On peut écrire une même fonction pour les matrices creuses ou non.
Si $\mathbb{D}$, $\mathbb{F}$ et $\mathbb{E}$ sont creuses alors $\mathbb{A}$ le sera aussi
sinon on retournera $\mathbb{A}$ sous forme de matrice pleine.
La fonction `issparse(A)` permet de tester si une matrice est creuse.

```
function A=Assemble(D,E,F,d)
  n=size(D,1);isSp=(issparse(D) & issparse(E) & issparse(F));
  assert( (all(size(D)==[n,n])) & (all(size(E)==[n,n])) & (all(size(F)==[n,n])) )
  N=d*n;
  if isSp, A=sparse(N,N);else, A=zeros(N,N);end
  I=1:n; % ligne bloc courante
  A(I,I)=D;A(I,I+n)=F;
  I=I+n;
  for i=2:d-1
    A(I,I)=D;
    A(I,I-n)=E;
    A(I,I+n)=F;
    I=I+n;
  end
  A(I,I)=D;A(I,I-n)=E;
end
```

② Nous allons, bien entendu, utiliser la commande suivante pour générer la matrice A

```
A=sparse(Ig,Jg,Kg,N,N)
```

Il nous faut donc créer les tableaux Ig, Jg et Kg sans boucle !

La fonction `find` permet de récupérer les tableaux* I, J et K associés à une matrice creuse G

```
[I,J,K]=find(G);
H=sparse(I,J,K,size(G,1),size(G,2));
```
$\Rightarrow H = G$

Remarque : les tableaux I, J et K sont de dimension $(n_z, 1)$
n_z étant le nombre d'éléments non nuls de G

* voir description de sparse du TP 5

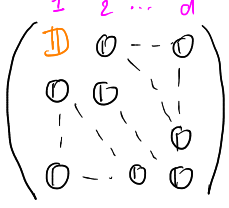
a) Pour mieux comprendre les mécanismes en jeu, on va tout d'abord assembler la matrice diagonale bloc

$$\mathbb{G} = \begin{pmatrix} \mathbb{D} & \mathbb{O} & \cdots & \mathbb{O} \\ \mathbb{O} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O} \\ \mathbb{O} & \cdots & \mathbb{O} & \mathbb{D} \end{pmatrix} \in \mathcal{M}_N(\mathbb{R}) \quad \text{avec } \mathbb{D} \in \mathcal{M}_n(\mathbb{R}) \text{ et } N = dn$$

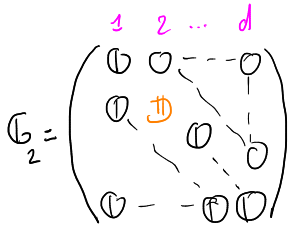
Mais tout d'abord que fait ce programme :

```
[I, J, K]=find(D);
G1=sparse(I, J, K, N, N);
G2=sparse(I+n, J+n, K, N, N);
G3=sparse(I+2*n, J+2*n, K, N, N);
```

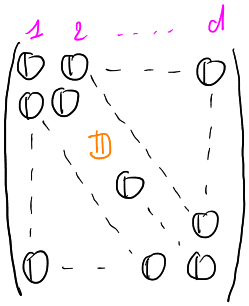
$\mathbb{G}_1$ =



$\mathbb{G}_2$ =



$\mathbb{G}_3$ =



On a d matrices $\mathbb{D}$ dans $\mathbb{G}$ et donc les tableaux I_g, J_g et K_g sont composés de d blocs, chaque bloc étant dans $\mathbb{R}^{n_2}$. On note $N_2 = n_2 d$

$$K_g = \begin{matrix}1\\2\\3\\\vdots\\d\end{matrix}\begin{pmatrix} K \\ K \\ K \\ \vdots \\ K \end{pmatrix} \in \mathbb{R}^{N_2}, \quad I_g = \begin{matrix}1\\2\\3\\\vdots\\d\end{matrix}\begin{pmatrix} I \\ I+n \\ I+2n \\ \vdots \\ I+(d-1)n \end{pmatrix} \in \mathbb{R}^{N_2} \text{ et } J_g = \begin{matrix}1\\2\\3\\\vdots\\d\end{matrix}\begin{pmatrix} J \\ J+n \\ J+2n \\ \vdots \\ J+(d-1)n \end{pmatrix} \in \mathbb{R}^{N_2}$$

Construction des tableaux I_g, J_g et K_g pour avoir

```
G=sparse(Ig, Jg, Kg, N,N);
```

Avec `[I, J, K]=find(D);` et `nz=numel(I);`, on déduit que le nombre d'éléments des tableaux I_g, J_g et K_g est `nz*d`. On a alors l'algo. de construction de $\mathbb{G}$ avec boucles

```
[I, J, K]=find(D); nz=numel(I); Nz=nz*d;
Ig=zeros(Nz,1); Jg=zeros(Nz,1); Kg=zeros(Nz,1);
idx=1:nz;
for i=1:d
  Ig(idx)=I+(i-1)*n;
  Jg(idx)=J+(i-1)*n;
  Kg(idx)=Kg;
  idx=idx+nz;
end
G=sparse(Ig, Jg, Kg, N,N);
```

Pour éliminer les boucles, on va construire des matrices $\mathbb{I}_g$, $\mathbb{J}_g$ et $\mathbb{K}_g$ de $\mathcal{M}_{n_2,d}(\mathbb{R})$ tq

$$\mathbb{I}_g = \begin{matrix}1 & 2 & 3 & \cdots & d\end{matrix}\begin{pmatrix} I & I+n & I+2n & \cdots & I+(d-1)n \end{pmatrix}, \quad \mathbb{J}_g = \begin{matrix}1 & 2 & 3 & \cdots & d\end{matrix}\begin{pmatrix} J & J+n & J+2n & \cdots & J+(d-1)n \end{pmatrix}, \quad \mathbb{K}_g = \begin{matrix}1 & 2 & \cdots & d\end{matrix}\begin{pmatrix} K & K & \cdots & K \end{pmatrix}$$

Dans ce cas $I_g = \mathbb{I}_g(:)$, $J_g = \mathbb{J}_g(:)$ et $K_g = \mathbb{K}_g(:)$

K, n_2-by-1 array : `matKg = repmat(K, [1,d]);` ou `matKg = K*ones(1,d);`

I, n_2-by-1 array : `matIg = I + [0:d-1]*n;`

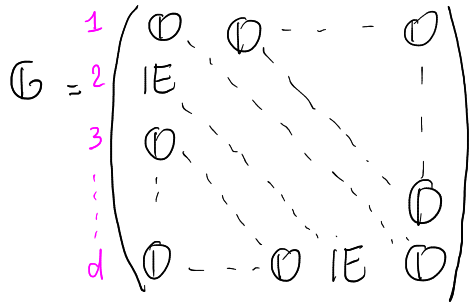
J, n_2-by-1 array : `matJg = J + [0:d-1]*n;`

⚠ ici le "+" n'est pas matriciel mathématiques au sens usuel. Avec Matlab / Octave

$$\underset{3\times 1}{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}} + \underset{1\times 2}{\begin{pmatrix} v_1 & v_2 \end{pmatrix}} = \underset{3\times 2}{\begin{pmatrix} v_1+v_1 & v_1+v_2 \\ v_2+v_1 & v_2+v_2 \\ v_3+v_1 & v_3+v_2 \end{pmatrix}}$$

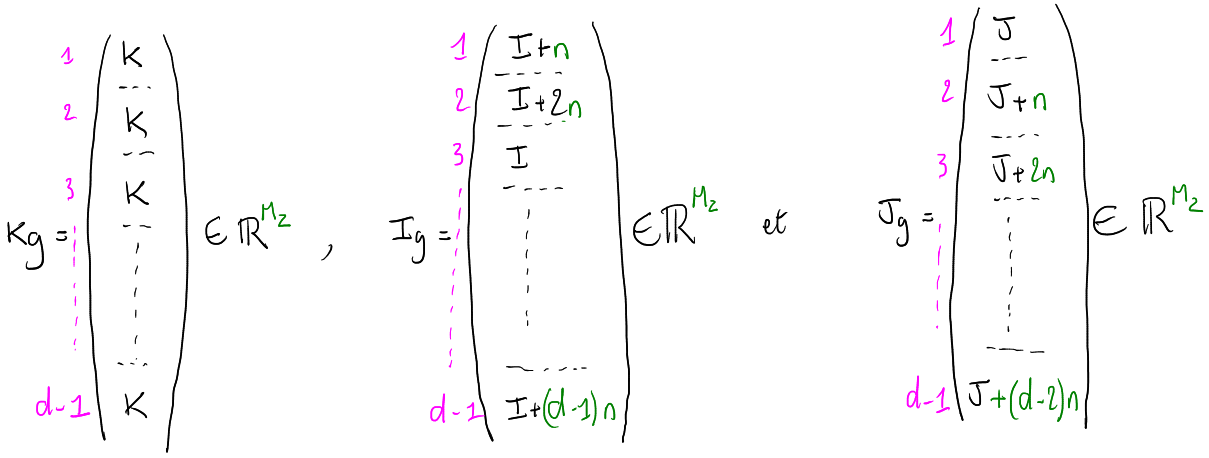
```
[I, J, K]=find(D); nz=numel(I); Nz=nz*d;
Kg = repmat(K, [1,d]);        % matrice
Ig = I + [0:d-1]*n;           % matrice
Jg = J + [0:d-1]*n            % matrice
G=sparse(Ig(:), Jg(:), Kg(:), N,N);
```

b) Pour mieux comprendre les mécanismes en jeu, on assemble la partie bloc strictement inférieure, c'est à dire



On a $(d-1)$ matrices $\mathbb{E}$ dans $\mathbb{G}$ et donc les tableaux I_g, J_g et K_g sont composés $(d-1)$ blocs, chaque bloc étant dans $\mathbb{R}^{n_2}$.

On note $M_2 = n_2(d-1)$. Avec `[I, J, K]=find(E);`, on a



On a alors l'algo. de construction de $\mathbb{G}$ avec boucles

```
[I, J, K]=find(E); nz=numel(I); Mz=nz*(d-1);
Ig=zeros(Mz,1); Jg=zeros(Mz,1); Kg=zeros(Mz,1);
idx=1:nz;
for i=1:d-1
  Ig(idx)=I+i*n;
  Jg(idx)=J+(i-1)*n;
  Kg(idx)=Kg;
  idx=idx+nz;
end
G=sparse(Ig, Jg, Kg, N,N);
```

Pour éliminer les boucles, on va construire des matrices $\mathbb{I}_g$, $\mathbb{J}_g$ et $\mathbb{K}_g$ de $\mathcal{M}_{n_2,d-1}(\mathbb{R})$ tq

$$\mathbb{I}_g = \begin{pmatrix} \overset{1}{I+n} & \overset{2}{I+2n} & \overset{\cdots}{\cdots} & \overset{d-1}{I+(d-1)n} \end{pmatrix}, \quad \mathbb{J}_g = \begin{pmatrix} \overset{1}{J} & \overset{2}{J+n} & \overset{3}{J+2n} & \overset{\cdots}{\cdots} & \overset{d-1}{J+(d-2)n} \end{pmatrix}, \quad \mathbb{K}_g = \begin{pmatrix} \overset{1}{K} & \overset{2}{K} & \overset{d-1}{\cdots} & K \end{pmatrix}$$

Dans ce cas $I_g = \mathbb{I}_g(:)$, $J_g = \mathbb{J}_g(:)$ et $K_g = \mathbb{K}_g(:)$

K, n_2-by-1 array : `matKg = repmat(K, [1,d-1]);` ou `matKg = K*ones(1,d-1);`

I, n_2-by-1 array : `matIg = I + [1:d-1]*n;`

J, n_2-by-1 array : `matJg = J + [0:d-2]*n;`

```
[I, J, K]=find(E); nz=numel(I); Nz=nz*(d-1);
Kg = repmat(K, [1,d-1]);      % matrice
Ig = I + [1:d-1]*n;           % matrice
Jg = J + [0:d-2]*n            % matrice
G=sparse(Ig(:), Jg(:), Kg(:), N,N);
```

c) A partir de ce que l'on vient de faire, la fonction

`function A=AssembleSpVec(D,E,F,d)`

devrait pouvoir s'écrire ...