MA261. Introduction au calcul scientifique
Corrigé 2 : Construction de matrices pour discrétisation par différences finies.

Exercice 5 : le Laplacien 1d (fil pesant, poutre en flexion).
Discrétisation du problème du fil pesant : *trouver u tel que*

$$-u'' = f \text{ sur }]0,1[, \quad u(0) = u(1) = 0.$$

1. Ecrire une fonction qui construise la matrice tridiagonale $\mathbb{A}_1$ d'ordre N obtenue par la discrétisation par différences finies du problème du fil pesant ou de la poutre en flexion (schéma à trois points pour le Laplacien 1d).
2. Comparer les temps de construction ainsi que la faisabilité, selon que la structure associée à $\mathbb{A}_1$ est pleine ou creuse, en fonction de N.

Corrigé 5 1. On rappelle que $N = n$, où n est égal au nombre de points de discrétisation intérieurs à l'intervalle de calcul, et que la matrice $\mathbb{A}_1 \in \mathbb{R}^{N\times N}$ est de la forme :

$$\mathbb{A}_1 = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \text{ avec } h = \frac{1}{(n+1)^2}.$$

On introduit à la fois N et n, bien qu'ils soient égaux ici en 1d (ce qui n'est plus le cas en 2d ou 3d !)
Dans le fichier `Laplacien1d.m`, créer la fonction :

```
function A1 = Laplacien1d(n)
N = n;
A1 = sparse(N,N);
for i = 1:n
  A1(i,i) = 2;
end
for i = 2:n
  A1(i,i-1) = -1;
end
for i = 1:n-1
  A1(i,i+1) = -1;
end
A1 = (n+1)^2*A1;
```

On peut examiner la structure de la matrice en tapant

```
A1 = Laplacien1d(100);
spy(A1)
```

2. On peut créer une fonction `Laplacien1dplein(n)` qui renvoie la même matrice, en *stockage plein*, en remplaçant `A1 = sparse(N,N)` par `A1 = zeros(N,N)`.

Pour mesurer le temps de construction, on peut par exemple taper en ligne

```
tic;A = Laplacien1d(1000);toc
tic;A = Laplacien1dplein(1000);toc
```

On constate que les temps d'exécution sont comparables. Par contre, lorsque N devient grand, des problèmes de capacité mémoire apparaissent pour le stockage plein !

```
tic;A = Laplacien1d_plein(10000);toc
??? Error using ==> zeros
Out of memory. Type HELP MEMORY for your options.
```

Il serait également possible de construire une fonction `TempsCalcul(n0,n1,ni)` qui construise les matrices $\mathbb{A}_1$ pour des valeurs de n variant de n_0 à n_1 par incrément de n_i, et qui mesure le temps de construction à l'aide de la commande `cputime`.

Exercice 6 : le Laplacien 2d (membrane élastique).

Discrétisation du problème de la membrane, posé dans $\Omega_2 =]0,1[^2$: *trouver u tel que*

$$-\Delta_2 u = f \text{ sur } \Omega_2, \quad u = 0 \text{ sur } \partial\Omega_2.$$

Le but de cet exercice est de construire *rapidement* la matrice $\mathbb{A}_2$ d'ordre N obtenue par la discrétisation par différences finies du problème de la membrane élastique (schéma à cinq points pour le Laplacien 2d).

Pour cela, on dispose d'une grille de $N = n \times n$ points, représentant les points de discrétisation *intérieurs* du domaine de calcul. D'un point de vue *algorithmique*, on veut utiliser des vecteurs de $\mathbb{R}^N$, dont les composantes correspondent à des valeurs aux points de la grille, ainsi que des matrices de $\mathbb{R}^{N\times N}$, qui représentent une action sur ces vecteurs.

1. Pourquoi les points de discrétisation de la frontière du domaine ne sont-ils pas pris en compte ? Comment numéroter les points de la grille ? Quels sont les voisins des points du bord de la grille ?
2. Soit $\mathbb{A}_2$ la matrice de $\mathbb{R}^{N\times N}$ telle que :
   Pour $I \in \{1,\cdots,N\}$: $(\mathbb{A}_2)_{I,I} = 4(n+1)^2$.
   Pour $I, J \in \{1,\cdots,N\}$, $I \neq J$:
   $(\mathbb{A}_2)_{I,J} = -(n+1)^2$ si I et J sont deux points voisins sur la grille (voisins de gauche, droite, haut, bas).
   $(\mathbb{A}_2)_{I,J} = 0$ sinon.
   Pourquoi cette matrice $\mathbb{A}_2$ est-elle la matrice du Laplacien 2d ?
3. Proposer un algorithme *rapide* de construction de $\mathbb{A}_2$. Ecrire la fonction Matlab correspondante. Effectuer une étude du temps de construction en fonction de N.

Corrigé 6 1. Les conditions aux limites imposent la valeur à la frontière du domaine de calcul. Les inconnues restantes du problème sont donc les n^2 valeurs prises aux points intérieurs au domaine, ce qui correspond exactement à la grille.
Une numérotation conseillée de la grille est la suivante : de gauche à droite, et de bas en haut, avec
- **un** indice I, variant de 1 à N, ou
- **deux** indices (i,j) variant de 1 à n.

Correspondance :

$$I-1=(i-1)+(j-1)n \quad \text{ou} \quad \begin{cases} i=(I-1) \mod n+1 \\ j=\lfloor (I-1)/n \rfloor +1 \end{cases} . \tag{1}$$

Considèrons les voisins *sur la grille* du point $I = 18$ qui correspond au couple $(i, j) = (9, 2)$. Son voisin de dessous est le point numéroté $I = 9$ ($(i, j) = (9, 1)$), celui du dessus est le numéro $I = 27$ ($(i, j) = (9, 3)$), celui de gauche est le numéro $I = 17$ ($(i, j) = (8, 2)$) et le dernier, celui de droite est un point de la frontière du domaine dont la valeur est nulle et non pas le numéro $I = 19$ ($(i, j) = (3, 1)$). En d'autres termes, le point $I = 18$ n'admet pas de voisin de droite *sur la grille*.

Identification des points du bord de la grille.
- Avec le *double* indiçage (i, j), un point est sur le bord si, et seulement si, $i \in \{1, n\}$ ou $j \in \{1, n\}$.
- Pour l'indice I, ceci correspond à (cf. (1))

$$\begin{cases} (I \mod n) \in \{1, 0\} \text{ ou} \\ I < n+1,\ N-n < I. \end{cases}$$

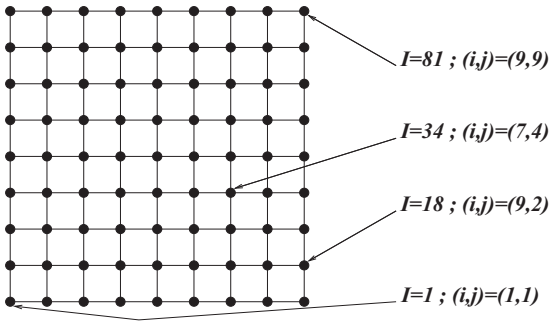

FIG. 1 – Avec $n = 9$, $N = 81$: $I \in \{1, \cdots, 81\}$ et $i, j \in \{1, \cdots, 9\}$

2. Par définition, on retrouve le schéma à cinq points, avec la pondération de $4/h^2$ pour le point considéré, et de $-1/h^2$ pour ses voisins immédiats, ce qui correspond bien à la discrétisation de $-\Delta_2 u = -\partial^2_{xx} u - \partial^2_{yy} u$ par différences finies...
3. Construction *rapide* de la matrice $\mathbb{A}_2$.
   Rapide = algorithme en $O(N)$ opérations, pour une matrice $N \times N$.

On peut par exemple, dans le fichier `Laplacien2d.m` créer la fonction

```
function A2 = Laplacien2d(n)
N = n^2;
A2 = 4*speye(N,N);
for j = 1:n
  for i = 1:n
    I = i+(j-1)*n;
    % I n'est pas un point du bord gauche donc il a un voisin de gauche
    if i~=1
      A2(I,I-1) = -1;
    end
```

```
    % I n'est pas un point du bord droit donc il a un voisin de droite
    if i~=n
      A2(I,I+1) = -1;
    end
    % I n'est pas un point du bord bas donc il a un voisin de dessous
    if j~=1
      A2(I,I-n) = -1;
    end
    % I n'est pas un point du bord haut donc il a un voisin de dessus
    if j~=n
      A2(I,I+n) = -1;
    end
  end
end
% On multiplie par 1/h^2 = (n+1)^2
A2 = (n+1)^2*A2;
```

Un autre algorithme possible de construction de la matrice $\mathbb{A}_2$ est :

```
function A2 = Laplacien2d(n)
N = n^2;
% (i) Comme la matrice A2 est symetrique, on se concentre sur
% la partie triangulaire superieure, ie. A2(I,J) pour J > I...
% (ii) Pas besoin d'initialisation par A2 = sparse(N,N), car celle-ci
% est contenue dans la premiere affectation.

% 1. Les voisins de droite : A2(I,I+1) = -1 pour tout I < N,
tmp = [[sparse(N-1,1) (-1)*speye(N-1)]; sparse(1,N)];
% Sauf s'il n'y a pas de voisin de droite...
% On effectue donc la correction
for j = 1:n-1
  I = j*n;
  tmp(I,I+1) = 0;
end
A2 = tmp;

% 2. Les voisins du haut : A2(I,I+n) = -1 pour tout I < N-n+1,
tmp = [[sparse(N-n,n) (-1)*speye(N-n)]; sparse(n,N)];
A2 = A2 + tmp;

% La partie triangulaire superieure est construite !

% 3. On symetrise la matrice : A2(I,J) = A2(J,I) pour tout I different de J,
A2 = A2 + A2';

% 4. On met a jour la diagonale : A2(I,I) = 4 pour tout I,
A2 = A2 + 4*speye(N);

% 5. On multiplie par 1/h^2 = (n+1)^2
A2 = (n+1)^2*A2;
```

Exercice 7 : le Laplacien 3d (cavité électrostatique).
Discrétisation du problème de la cavité, posé dans $\Omega_3 =]0,1[^3$: *trouver u tel que*

$$-\Delta_3 u = f \text{ sur } \Omega_3, \quad u = 0 \text{ sur } \partial\Omega_3.$$

On dispose maintenant d'une grille de $N = n \times n \times n$ points, représentant les points de discrétisation *intérieurs* du domaine de calcul. Comme pour l'exercice 2, les vecteurs à manipuler appartiennent à $\mathbb{R}^N$, et les matrices qui représentent une action sur ces vecteurs sont dans $\mathbb{R}^{N\times N}$.

1. Comment numéroter les points de la grille ?
2. A quoi est égale la matrice $\mathbb{A}_3$ du Laplacien 3d ? Proposer un algorithme *rapide* de construction de $\mathbb{A}_3$. Ecrire la fonction Matlab correspondante. Effectuer une étude du temps de construction en fonction de N.

Corrigé 7 1. (i) Numérotation conseillée pour la grille : de gauche à droite, de bas en haut, et de l'avant vers l'arrière avec
- **un** indice I, variant de 1 à N, ou
- **trois** indices (i,j,k) variant de 1 à n.
Correspondance

$$(I-1) = (i-1) + (j-1)n + (k-1)n^2 \quad \text{ou} \quad \left\{ \begin{array}{l} i = (I-1) \mod n + 1 \\ j = \lfloor ((I-1) \bmod n^2)/n \rfloor + 1 \\ k = \lfloor (I-1)/n^2 \rfloor + 1 \end{array} \right. . \quad (2)$$

(ii) Identification des points du bord de la grille. Avec le *triple* indiçage (i,j,k), un point est sur le bord si, et seulement si, $i \in \{1,n\}$ ou $j \in \{1,n\}$ ou $k \in \{1,n\}$.

2. Construction *rapide* de la matrice $\mathbb{A}_3$.
Rapide = algorithme en $O(N)$ opérations.

On peut par exemple, dans le fichier `Laplacien3d.m`, créer la fonction

```
function A3 = Laplacien3d(n)
N = n^3;
A3 = 6*speye(N,N);
for k = 1:n
  for j = 1:n
    for i = 1:n
      I = i+(j-1)*n+(k-1)*n*n;
      % I n'est pas un point du bord gauche donc il a un voisin de gauche
      if i~=1
        A3(I,I-1) = -1;
      end
      % I n'est pas un point du bord droit donc il a un voisin de droite
      if i~=n
        A3(I,I+1) = -1;
      end
      % I n'est pas un point du bord bas donc il a un voisin de dessous
      if j~=1
        A3(I,I-n) = -1;
      end
```

```
      % I n'est pas un point du bord haut donc il a un voisin de dessus
      if j~=n
        A3(I,I+n) = -1;
      end
      % I n'est pas un point du bord avant donc il a un voisin de devant
      if k~=1
        A3(I,I-n*n) = -1;
      end
      % I n'est pas un point du bord arriere donc il a un voisin de derriere
      if k~=n
        A3(I,I+n*n) = -1;
      end
    end
  end
end
% On multiplie par 1/h^2 = (n+1)^2
A3 = (n+1)^2*A3;
```

ou encore

```
function A3 = Laplacien3d(n)
N = n^3;

% (i) Comme la matrice A3 est symetrique, on se concentre sur
% la partie triangulaire superieure, ie. A3(I,J) pour J > I...
% (ii) Pas besoin d'initialisation par A3 = sparse(N,N), car celle-ci
% est contenue dans la premiere affectation.

% 1. Les voisins de droite : A3(I,I+1) = -1 pour tout I < N,
tmp = [[sparse(N-1,1) (-1)*speye(N-1)]; sparse(1,N)];
% Sauf s'il n'y a pas de voisin de droite...
% On effectue donc la correction
for k = 1:n
  for j = 1:n
    I = (k-1)*n^2+j*n;
    if (I+1 <= N)
      tmp(I,I+1) = 0;
    end
  end
end
A3 = tmp;

% 2. Les voisins du haut : A3(I,I+n) = -1 pour tout I < N-n+1,
tmp = [[sparse(N-n,n) (-1)*speye(N-n)]; sparse(n,N)];
% Sauf s'il n'y a pas de voisin du haut...
% On effectue donc la correction
for k = 1:n
  for i = 1:n
    I = (k-1)*n^2+(n-1)*n+i;
    if (I+n <= N)
      tmp(I,I+n) = 0;
```

```
    end
  end
end
A3 = A3 + tmp;

% 3. Les voisins arriere : A3(I,I+n^2) = -1 pour tout I < N-n^2+1,
tmp = [[sparse(N-(n^2),n^2) (-1)*speye(N-(n^2))]; sparse(n^2,N)];
A3 = A3 + tmp;

% La partie triangulaire superieure est construite !

% 4. On symetrise la matrice : A3(I,J) = A3(J,I) pour tout I different de J,
A3 = A3 + A3';

% 5. On met a jour la diagonale : A3(I,I) = 6 pour tout I,
A3 = A3 + 6*speye(N);

% 6. On multiplie par 1/h^2 = (n+1)^2
A3 = (n+1)^2*A3;
```

Mise en œuvre informatique à l'aide de Matlab

1. *Points à retenir* :
   - Notions et commandes Matlab du TP1.
   - *Evaluation des performances* d'un algorithme et de sa mise en œuvre.
   - *Différence* entre stockage *plein* et stockage *creux*.
2. *Fonctionnalités Matlab à maîtriser* :
   - Dimensions d'une matrice : `size(A)`.
   - Initialisation (structures *creuses*) pour les matrices : `A = sparse(N,N)`.
   - Affichage de la structure d'une matrice : `spy(A)`.
   - Matrice identité I_N : `A = eye(N)` (structure *pleine*), ou `A = speye(N)` (structure *creuse*).
   - Mesure du temps d'exécution d'une suite d'instructions : `tic ; ... ;toc` ou `cputime`.