TPs EDP

Travaux Pratiques N^o 4

Méthodes des différences finies
Matrices creuses, Laplacien 2D et plus si affinité

Version du 20 novembre 2025

1 Notations et approximation des dérivées secondes

Soient $\Omega =]a,b[\times]c,d[\subset \mathbb{R}^2$ et $\Gamma = \partial\Omega$ la frontière du domaine Ω. On note Γ_N, Γ_S, Γ_O et Γ_E respectivement les frontières nord, sud, ouest et est. on a

$$\Gamma = \Gamma_N \cup \Gamma_S \cup \Gamma_O \cup \Gamma_E.$$

Soit $v : \Omega \longrightarrow \mathbb{R}$ suffisament régulière.

Q. 1 [rapport] ☆☆☆

Montrer que l'on a

$$\frac{\partial^2 v}{\partial x^2}(x,y) = \frac{v(x+h,y) - 2v(x,y) + v(x-h,y)}{h^2} + \mathcal{O}(h^2) \tag{1.1}$$

$$\frac{\partial^2 v}{\partial y^2}(x,y) = \frac{v(x,y+h) - 2v(x,y) + v(x,y-h)}{h^2} + \mathcal{O}(h^2). \tag{1.2}$$

On note $(x_i)_{i=0}^{N_x}$ et $(y_j)_{j=0}^{N_y}$ les discrétisation régulières, respectivement, des intervalles $[a,b]$ et $[c,d]$ défines par

$$x_i = a + ih_x, \ \forall i \in [\![0, N_x]\!] \quad \text{et} \quad y_j = c + jh_y, \ \forall j \in [\![0, N_y]\!] \tag{1.3}$$

avec $h_x = (b-a)/N_x$ et $h_y = (d-c)/N_y$. On note aussi

$$n_x = N_x + 1, \quad n_y = N_y + 1 \quad \text{et} \quad N = n_x \times n_y \tag{1.4}$$

Q. 2 [rapport] ☆☆☆

Montrer que l'on a $\forall i \in]\!]0, N_x[\![, \ \forall j \in]\!]0, N_y[\![$

$$\begin{aligned}\Delta v(x_i,y_j) &\overset{\text{def}}{=} (\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2})(x_i,y_j) \\ &= \frac{v(x_{i+1},y_j) - 2v(x_i,y_j) + v(x_{i-1},y_j)}{h_x^2} \\ &\quad + \frac{v(x_i,y_{j+1}) - 2v(x_i,y_j) + v(x_i,y_{j-1})}{h_y^2} + \mathcal{O}(h_x^2) + \mathcal{O}(h_y^2)\end{aligned} \tag{1.5}$$

2 Equation de Poisson avec conditions de Dirichlet

Soient $f : \Omega \longrightarrow \mathbb{R}$ et, $\forall \alpha \in \{S,O,N,E\}$, $g_\alpha : \Gamma \longrightarrow \mathbb{R}$ des fonctions données. On veut résoudre le problème suivant

$$-\Delta u = f, \qquad \text{dans } \Omega \tag{2.1}$$

$$u = g_\alpha, \qquad \text{sur } \Gamma_\alpha, \ \forall \alpha \in \{S,O,N,E\} \tag{2.2}$$

On utilise par la suite les discrétisations $(x_i)_{i=0}^{N_x}$ et $(y_j)_{j=0}^{N_y}$ définies en section 1.

Q. 3 [rapport] ★★★

a. Montrer que ce problème peut s'écrire après discrétisation sous la forme

$$-\frac{U_{i+1,j}-2U_{i,j}+U_{i-1,j}}{h_x^2}-\frac{U_{i,j+1}-2U_{i,j}+U_{i,j-1}}{h_y^2}=f(x_i,y_j), \qquad \forall (i,j)\in]\!]0,N_x[\![\times]\!]0,N_y[\![, \tag{2.3}$$

$$U_{0,j}=g_O(a,y_j), \qquad \forall j\in[\![0,N_y]\!], \tag{2.4}$$

$$U_{N_x,j}=g_E(b,y_j), \qquad \forall j\in[\![0,N_y]\!], \tag{2.5}$$

$$U_{i,0}=g_S(x_i,c), \qquad \forall i\in[\![0,N_x]\!], \tag{2.6}$$

$$U_{i,N_y}=g_N(x_i,d), \qquad \forall i\in[\![0,N_x]\!]. \tag{2.7}$$

avec $U_{i,j}\approx u(x_i,y_j)$.

b. Les inconnues du problème discrétisé sont les $U_{i,j}$, *pour* $i\in[\![0,N_x]\!]$ *et* $j\in[\![0,N_y]\!]$. *Compter le nombre d'équations distinctes du problème discrétisé (bord compris). Que peut-on en conclure ?*

En notant $\beta_x=-\frac{1}{h_x^2}$, $\beta_y=-\frac{1}{h_y^2}$ et $\mu=2(\frac{1}{h_x^2}+\frac{1}{h_y^2})$, les équations (2.3) peuvent aussi s'écrire sous la forme

$$\beta_x(U_{i+1,j}+U_{i-1,j})+\mu U_{i,j}+\beta_y(U_{i,j+1}+U_{i,j-1})=f(x_i,y_j), \quad \forall (i,j)\in]\!]0,N_x[\![\times]\!]0,N_y[\![\tag{2.8}$$

Pour tout $j\in[\![0,N_y]\!]$, on note $U_{:,j}$ le vecteur de $\mathbb{R}^{n_x}$ définit par

$$U_{:,j}=\begin{pmatrix}U_{0,j}\\ \vdots \\ U_{N_x,j}\end{pmatrix}.$$

On note $\boldsymbol{V}\in\mathbb{R}^N$ le vecteur bloc

$$\boldsymbol{V}=\begin{pmatrix}U_{:,0}\\ \hdashline U_{:,1}\\ \hdashline \vdots \\ \hdashline U_{:,N_y}\end{pmatrix}$$

Q. 4 [rapport] ★★★

Explicitez la bijection $\mathcal{F}:[\![0,N_x]\!]\times[\![0,N_y]\!]\longrightarrow[\![1,N]\!]$ *telle que*

$$\forall (i,j)\in[\![0,N_x]\!]\times[\![0,N_y]\!], \quad V_k=U_{i,j}, \quad \textit{avec } k=\mathcal{F}(i,j).$$

Dans le cas de la numérotation en $(i,j)\in[\![0,N_x]\!]\times[\![0,N_y]\!]$ on parlera de **numérotation 2D** et pour la numérotation en $k\in[\![1,N]\!]$ on parlera de **numérotation globale**.

Q. 5 [code] ★★★

a. Ecrire la fonction k=bijF(i,j,nx) *correspondant à la bijection* $\mathcal{F}$ *(* ***numerotation 2D*** *vers* ***numerotation globale****).*

b. Ecrire la fonction réciproque [i,j]=bijRecF(k,nx) *correspondant à* $\mathcal{F}^{-1}$ *(* ***numerotation globale*** *vers* ***numerotation 2D****). On pourra utiliser la fonction* rem(x,y) *(reste de la division de* x *par* y*) et on vérifiera que cette fonction est naturellement vectorisée, c'est à dire que le paramètre* k *peut aussi être un tableau.*

2.1 Evaluation de fonctions sur la grille discrète

Soit $\boldsymbol{F}\in\mathbb{R}^N$ le vecteur (bloc) définit par

$$\boldsymbol{F}=\begin{pmatrix}F_{:,0}\\ \hdashline F_{:,1}\\ \hdashline \vdots \\ \hdashline F_{:,N_y}\end{pmatrix} \text{ avec } F_{:,j}=\begin{pmatrix}f(x_0,y_j)\\ \vdots \\ f(x_{N_x},y_j)\end{pmatrix}\in\mathbb{R}^{Nx+1}, \quad \forall j\in[\![0,N_y]\!].$$

En Algorithme 1, est présenté une version non vectorisée (deux boucles imbriquées) de la construction du vecteur $\boldsymbol{F}$. Dans cet algorithme l'ordre des boucles est primordial pour respecter le choix de la numérotation globale.

Algorithm 1 Algorithme non vectorisé de calcul du vecteur $\boldsymbol{F}$

```
k ← 1
for j ← 1 to n_y do
  for i ← 1 to n_x do
    F(k) ← f(x(i), y(j)))
    k ← k + 1
  end for
end for
```

Q. 6 [code] ★☆☆

Ecrire la fonction EvalFun2DSca *permettant à partir d'une fonction f donnée et des discrétisations $\boldsymbol{x}$ et $\boldsymbol{y}$ de retourner le vecteur $\boldsymbol{F}$ associé. On pourra utiliser des boucles* for .

Pour le cas où la fonction f est vectorisée sous Matlab/Octave, il est alors possible de calculer le vecteur $\boldsymbol{F}$ sans boucle. Pour celà, nous allons construire les deux vecteurs blocs $\boldsymbol{X}$ et $\boldsymbol{Y}$ de $\mathbb{R}^n$ définis par

$$\boldsymbol{X} = \begin{pmatrix} X_{:,0} \\ \hdashline X_{:,1} \\ \hdashline \vdots \\ \hdashline X_{:,N_y} \end{pmatrix} \text{ avec } X_{:,j} = \begin{pmatrix} x_0 \\ \vdots \\ x_{N_x} \end{pmatrix} \in \mathbb{R}^{Nx+1}, \quad \forall j \in [\![0, N_y]\!]$$

et

$$\boldsymbol{Y} = \begin{pmatrix} Y_{:,0} \\ \hdashline Y_{:,1} \\ \hdashline \vdots \\ \hdashline Y_{:,N_y} \end{pmatrix} \text{ avec } Y_{:,j} = \begin{pmatrix} y_j \\ \vdots \\ y_j \end{pmatrix} \in \mathbb{R}^{Nx+1}, \quad \forall j \in [\![0, N_y]\!].$$

Avec ces deux vecteurs, le calcul du vecteur $\boldsymbol{F}$ pourra être vectorisé (sous la condition que f le soit) :

```
F=f(X,Y);
```

Il nous faut donc écrire une version vectorisée du calcul des vecteurs $\boldsymbol{X}$ et $\boldsymbol{Y}$, mais auparavant on donne plusieurs manières d'écrire le calcul du vecteur $\boldsymbol{F}$

Algorithm 2 Calcul de $\boldsymbol{F}$ avec construction de $\boldsymbol{X}$ et $\boldsymbol{Y}$ (version 1)

```
k ← 1
for j ← 1 to n_y do
  for i ← 1 to n_x do
    X(k) ← x(i)
    Y(k) ← y(j)
    F(k) ← f(X(k), Y(k)))
    k ← k + 1
  end for
end for
```

Algorithm 3 Calcul de $\boldsymbol{F}$ avec construction de $\boldsymbol{X}$ et $\boldsymbol{Y}$ (version 2)

```
1: k ← 1
2: for j ← 1 to n_y do
3:   for i ← 1 to n_x do
4:     X(k) ← x(i)
5:     Y(k) ← y(j)
6:     k ← k + 1
7:   end for
8: end for
9: F ← f(X, Y)
```

On peut noter que l'ordre des boucles dans l'Algorithme 3 permet d'affirmer que la valeur de k en ligne 4 est donnée par $\mathcal{F}(i-1, j-1)$.

Algorithm 4 Calcul de $\boldsymbol{F}$ avec construction de $\boldsymbol{X}$ et $\boldsymbol{Y}$ (version 3)

```
for j ← 1 to n_y do
  for i ← 1 to n_x do
    k ← bijF(i − 1, j − 1, n_x)
    X(k) ← x(i)
    Y(k) ← y(j)
  end for
end for
F ← f(X, Y)
```

Grâce à l'application réciproque $\mathcal{F}^{-1}$ il est alors possible de transformer la double boucle en j et i en une seule boucle sur k : c'est l'objet de l'Algorithme 5. On en déduit alors l'Algorithme 6 totalement vectorisé.

Algorithm 5 Calcul de $\boldsymbol{F}$ avec construction de $\boldsymbol{X}$ et $\boldsymbol{Y}$ (version 4)

for $k \leftarrow 1$ **to** N **do**
  $[i, j] \leftarrow \text{bijRecF}(k, n_x)$
  $\boldsymbol{X}(k) \leftarrow \boldsymbol{x}(i+1)$
  $\boldsymbol{Y}(k) \leftarrow \boldsymbol{y}(j+1)$
end for
$\boldsymbol{F} \leftarrow f(\boldsymbol{X}, \boldsymbol{Y})$

Algorithm 6 Calcul de $\boldsymbol{F}$ avec construction de $\boldsymbol{X}$ et $\boldsymbol{Y}$ (version 5 : vectorisée)

$[I, J] \leftarrow \text{bijRecF}(1 : N, n_x)$ ▷ Deja vectorisee
$\boldsymbol{X} \leftarrow \boldsymbol{x}(I+1)$
$\boldsymbol{Y} \leftarrow \boldsymbol{y}(J+1)$
$\boldsymbol{F} \leftarrow f(\boldsymbol{X}, \boldsymbol{Y})$

Un programme Matlab/Octave complet basé sur ce dernier algorithme est proposé en Listing 2.

Listing 1 – Calcul et représentation d'une fonction sur la grille 2D

```
clear all
close all
a=-pi;b=pi;c=0;d=pi;
f=@(x,y)  4*cos(x + y).*sin(x - y);
Nx=133;Ny=222;
nx=Nx+1;ny=Ny+1;N=nx*ny;
x=linspace(a,b,nx); % 1-by-nx
y=linspace(c,d,ny); % 1-by-ny


[I,J]=bijRecF(1:N,nx); % I,J: 1-by-N, naturellement vectorisee!
X=x(I+1);Y=y(J+1);     % X,Y : 1-by-N
F=f(X,Y);              % F: 1-by-N

surf(x,y,reshape(F,nx,ny)') % 3eme param: ny-by-nx
shading interp
```

Il faut noter qu'il existe sous Matlab/Octave deux fonctions `meshgrid` et `ndgrid` qui peuvent être utilées pour le calcul des vecteurs $\boldsymbol{X}$ et $\boldsymbol{Y}$. Ces deux fonctions n'utilisent pas la fonction `bijRecF` .
D'autres techniques peuvent aussi être utilisées, par exemple avec la fonction `repmat` et l'opérateur `(:)` (ou la fonction `reshape` ou la commande `subsref(X,substruct('()',{':'}))`).

Q. 7 [code] ★☆☆

Ecrire la fonction `EvalFun2DVec` *permettant à partir d'une fonction f et des discrétisations $\boldsymbol{x}$ et $\boldsymbol{y}$ de retourner le vecteur $\boldsymbol{F}$ associé sans utiliser de boucle* `for` .

2.2 A blocs

Chacune des équations du problème discret (2.3)-(2.7) correspond à une discrétisation en un point (x_i, y_j). Nous choisissons d'écrire ces équations en utilisant la même numérotation que lors de la construction du vecteur $\boldsymbol{V}$: l'équation *écrite* au point (x_i, y_j) sera écrite en ligne $k = \mathcal{F}(i, j)$ du système.

Q. 8 [rapport] ★☆☆

Etablir que le problème discret (2.3)-(2.7) *peut s'écrire sous la forme du système linéaire bloc*

$$\left(\begin{array}{c:c:c:c:c:c:c} \mathbb{E} & \mathbb{O} & \cdots & \cdots & \cdots & \mathbb{O} & \mathbb{O} \\ \hdashline \mathbb{M} & \mathbb{D} & \mathbb{M} & \mathbb{O} & \cdots & \mathbb{O} & \mathbb{O} \\ \mathbb{O} & \mathbb{M} & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \mathbb{O} & \ddots & \ddots & \ddots & \mathbb{O} & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \mathbb{M} & \mathbb{O} \\ \mathbb{O} & \mathbb{O} & \cdots & \mathbb{O} & \mathbb{M} & \mathbb{D} & \mathbb{M} \\ \hdashline \mathbb{O} & \mathbb{O} & \cdots & \cdots & \cdots & \mathbb{O} & \mathbb{E} \end{array}\right) \boldsymbol{V} = \left(\begin{array}{c} B_{:,0} \\ \hdashline B_{:,1} \\ \hdashline \vdots \\ \hdashline \vdots \\ \hdashline B_{:,N_y} \end{array}\right) \tag{2.9}$$

où chaque bloc de la matrice est une matrice $(N_x + 1)$ *par* $(Nx + 1)$*. La matrice* $\mathbb{O} \in \mathcal{M}_{N_x+1}(\mathbb{R})$ *est la matrice nulle. Les matrices creuses* $\mathbb{D}$, $\mathbb{M}$ *et* $\mathbb{E}$ *ainsi que les vecteurs* $B_{:,j} \in \mathbb{R}^{Nx+1}$*, pour tout* $j \in [\![0, N_y]\!]$*, devront être donnés explicitement. On notera* $\boldsymbol{B}$ *le vecteur second membre de* $\mathbb{R}^N$ *et* $\mathbb{A} \in \mathcal{M}_N(\mathbb{R})$ *la matrice du système.*

2.3 Assemblage de la matrice sans conditions aux limites

Nous avons vu que notre problème discret revient à résoudre le système linéaire

$$\mathbb{A}\boldsymbol{V} = \boldsymbol{B}. \tag{2.10}$$

Il nous faut donc construire/assembler la matrice $\mathbb{A} \in \mathcal{M}_N(\mathbb{R})$ et le second membre $\boldsymbol{B} \in \mathbb{R}^N$.
Nous allons, dans cette section, générer/assembler la matrice correspondant à la prise en compte de toutes les équations de (2.3) sans tenir compte, pour le moment, des conditions aux limites (i.e. les lignes du système correspondant à des points du bord sont nulles ... pour l'instant).
Nous noterons $-\mathbb{A}_{xy}$ cette matrice pour qu'elle corresponde à la matrice du Laplacien et non l'opposée du Laplacien.

2.3.1 Méthode 1 : utilisation de la bijection

Cette méthode est celle qui va, au final, se rapprocher le plus des techniques d'assemblages utilisées par les méthodes d'éléments finis et volumes finis sur des domaines $\Omega \subset \mathbb{R}^d$ quelconques.
Nous rappelons que nous avons choisi d'écrire l'équation (2.3) au point (x_i, y_j) en ligne $k = \mathcal{F}(i, j)$ du système.

Q. 9 [code] ★★☆

a. *Sans aucune vectorisation/optimisation de code, écrire la fonction* Lap2DBijection00 *retournant la matrice creuse* $\mathbb{A}_{xy}$ *en utilisant la fonction* bijF*. Il y aura ici deux boucles imbriquées, l'une en* i *et l'autre en* j *pour écrire chacune des équations en* (i, j)*.*

b. *Proposer au moins un programme permettant de tester/valider la matrice ainsi obtenue.*

Q. 10 [code] ★★★

Proposer plusieurs variantes de vectorisation/optimisation de la fonction Lap2DBijection00 *: il faudrait supprimer au moins une des boucles en* i *ou en* j*...*

2.3.2 Méthode 2 : méthode blocs

Q. 11 [code] ★★☆

a. *Sans aucune vectorisation/optimisation de code, écrire la fonction* Lap2Dbloc00 *retournant la matrice creuse* $\mathbb{A}_{xy}$ *en utilisant la structure bloc de la matrice. Il faudra ici utiliser deux boucles : l'une sur les blocs lignes et l'autre sur les blocs colonnes.*

b. *Proposer au moins un programme permettant de tester/valider la matrice ainsi obtenue.*

Q. 12 [code] ★★★

Proposer plusieurs variantes de vectorisation/optimisation de la fonction Lap2Dbloc00 *: il faudrait supprimer au moins une des boucles : celle sur les blocs lignes et/ou celle sur les blocs colonnes.*

2.3.3 Méthode 3 : produits de Kronecker (facultatif)

On note $\mathbb{I}_n$ la matrice identité de $\mathcal{M}_n(\mathbb{R})$ et $\mathbb{J}_n$ la matrice de $\mathcal{M}_n(\mathbb{R})$ définie par

$$\mathbb{J}_n = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix} \in \mathcal{M}_n(\mathbb{R}).$$

Dans cette partie nous allons générer/assembler la matrice $\mathbb{A}_{xy}$, opposée de la matrice du système (2.9) **sans tenir compte des conditions aux limites** (i.e. les lignes du système correspondant à des points du bord sont nulles ... pour l'instant).
C'est une matrice bloc de $\mathcal{M}_N(\mathbb{R})$ avec n_y lignes bloc composées de blocs carrés de dimension n_x qui s'écrit sous la forme :

$$\mathbb{A}_{xy} = \left(\begin{array}{c|c|ccccc|c} \mathbb{0} & \mathbb{0} & \cdots & \cdots & \cdots & \mathbb{0} & \mathbb{0} \\ \hline \mathbb{0} & \mathbb{T}_x & \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{T}_x & \ddots & \ddots & \vdots & \vdots \\ \vdots & \mathbb{0} & \ddots & \ddots & \ddots & \mathbb{0} & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{T}_x & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} & \mathbb{T}_x & \mathbb{0} \\ \hline \mathbb{0} & \mathbb{0} & \cdots & \cdots & \cdots & \mathbb{0} & \mathbb{0} \end{array}\right) + \left(\begin{array}{c|c|ccccc|c} \mathbb{0} & \mathbb{0} & \cdots & \cdots & \cdots & \mathbb{0} & \mathbb{0} \\ \hline \mathbb{S}_y & \mathbb{T}_y & \mathbb{S}_y & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{S}_y & \mathbb{T}_y & \ddots & \ddots & \vdots & \vdots \\ \vdots & \mathbb{0} & \ddots & \ddots & \ddots & \mathbb{0} & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbb{T}_y & \mathbb{S}_y & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{S}_y & \mathbb{T}_y & \mathbb{S}_y \\ \hline \mathbb{0} & \mathbb{0} & \cdots & \cdots & \cdots & \mathbb{0} & \mathbb{0} \end{array}\right) \tag{2.11}$$

où

$$\mathbb{S}_y = \frac{1}{h_y^2}\mathbb{J}_{n_x}, \quad \mathbb{T}_y = -\frac{2}{h_y^2}\mathbb{J}_{n_x} \quad \text{et} \quad \mathbb{T}_x = \frac{1}{h_x^2}\begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 0 & 0 \end{pmatrix}$$

On peut noter que les matrices $\mathbb{T}_x$, $\mathbb{T}_y$ et $\mathbb{S}_y$ sont des matrices de $\mathcal{M}_{n_x}(\mathbb{R})$.
En notant $\mathbb{A}_x \in \mathcal{M}_{n_x}(\mathbb{R})$ et $\mathbb{A}_y \in \mathcal{M}_{n_y}(\mathbb{R})$ les matrices définies par

$$\mathbb{A}_x = \frac{1}{h_x^2}\begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 0 & 0 \end{pmatrix} \quad \text{et} \quad \mathbb{A}_y = \frac{1}{h_y^2}\begin{pmatrix} 0 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 0 & 0 \end{pmatrix}$$

on déduit de (2.11)

$$\mathbb{A}_{xy} = \mathbb{J}_{n_y} \otimes \mathbb{A}_x + \mathbb{A}_y \otimes \mathbb{J}_{n_x}. \tag{2.12}$$

Q. 13

a. Ecrire la fonction Lap2DKron *retournant la matrice bloc creuse* $\mathbb{A}$ *en utilisant* (2.12) *et les fonctions déjà écrites.*

b. Proposer au moins un programme permettant de tester/valider la matrice ainsi obtenue.

Sur le même principe, il est possible de généraliser en dimension quelconque la construction de la matrice du Laplacien obtenue par différences finies sans prise en compte des conditions aux limites. Par exemple en dimension 3, cette matrice s'écrit

$$\begin{aligned} \mathbb{A}_{xyz} &= \mathbb{J}_{n_z} \otimes \mathbb{A}_{xy} + \left(\mathbb{A}_z \otimes (\mathbb{J}_{n_y} \otimes \mathbb{J}_{n_x})\right) \\ &= \mathbb{J}_{n_z} \otimes \mathbb{J}_{n_y} \otimes \mathbb{A}_x + \mathbb{J}_{n_z} \otimes \mathbb{A}_y \otimes \mathbb{J}_{n_x} + \mathbb{A}_z \otimes \mathbb{J}_{n_y} \otimes \mathbb{J}_{n_x} \end{aligned}$$

2.4 Quelques indices

2.4.1 Indices des bords

La frontière Γ du domaine est décomposée en 4 sous-ensembles :

$$\Gamma = \Gamma_N \cup \Gamma_S \cup \Gamma_E \cup \Gamma_O.$$

Q. 14 [rapport] ★☆☆

a. Pour Γ_S, donner le sous-ensemble $\mathcal{D}_S$ d'indices (i,j) dans $[\![0,N_x]\!] \times [\![0,N_y]\!]$ tel que $(x(i),y(j)) \in \Gamma_S$.

b. En déduire le sous-ensemble $\mathcal{I}_S$ (ordonné suivant $\mathcal{D}_S$) correspondant dans la numérotation globale.

c. Même chose pour Γ_N, Γ_E et Γ_O.

Q. 15 [code] ★☆☆

Ecrire la fonction BordDomaine *retournant la structure de données (*help struct*) contenant les champs suivants :*

- Sud *ayant pour valeur $\mathcal{I}_S$,*
- Nord *ayant pour valeur $\mathcal{I}_N$,*
- Est *ayant pour valeur $\mathcal{I}_E$,*
- Ouest *ayant pour valeur $\mathcal{I}_O$.*

Chacun des tableaux $\mathcal{I}_\alpha$, $\alpha \in \{S,O,N,E\}$, sera stocké sous forme ligne.

Une fois cette fonction écrite, il est aisé en numérotation globale de récupérer l'ensemble idxBD des indices des points du bords ainsi que l'ensemble idxBDc des indices des points strictement intérieurs (complémentaire du précédant) :

```
BD=BordDomaine(...);   % ... -> a remplacer
idxBD=unique([BD.Sud,BD.Nord,BD.Est,BD.Ouest]);
idxBDc=setdiff(1:N,idxBD);   % N=nx*ny
```

2.4.2 Evaluation de fonctions sur une partie de la grille

Soit $g : \Gamma_N \longrightarrow \mathbb{R}$. On suppose cette fonction vectorisée sous Matlab/Octave. Pour évaluer cette fonction aux points de la grille 2D appartenant à Γ_N, on utilise la structure de données retournée par la fonction BordDomaine et plus particulièrement le champ Nord de cette structure. Voici une manière de créer un vecteur G valant G(k)=g(x(i),y(j)) sur un point du bord Nord avec $k = \mathcal{F}(i,j)$ et 0 sinon.

Listing 2 – Calcul d'une fonction sur le bord Nord de la grille 2D

```
BD=BordDomaine(...);   % ... -> a remplacer
[I,J]=bijRecF(1:N,nx);
X=x(I+1);Y=y(J+1);% coordonnees des points de la grille
                  % en numerotation globale
g=@(x,y) sin(x+y).*x+y.^2;
Gloc=g(X(BD.Nord),Y(BD.Nord));
G=zeros(N,1);
G(BD.Nord)=Gloc;
```

Pour évaluer une fonction sur une autre partie de la grille, il suffit de récupérer l'ensemble des indices dans la numérotation globale des points de appartenant à cette partie et de remplacer BD.Nord par cet ensemble.

2.5 Assemblage du second membre sans conditions aux limites

Nous allons générer/assembler le vecteur second membre du système (2.9) **sans tenir compte des conditions aux limites**. Avec le choix de la numérotation globale, ce vecteur $\underline{\mathbf{B}}$ de $\mathbb{R}^N$ est défini par

$$\forall k \in [\![1,N]\!], \quad \underline{\mathbf{B}}_k = \begin{cases} f(x(i),y(j)) & \text{si } (x(i),y(j)) \notin \Gamma \text{ et } k = \mathcal{F}(i,j) \\ 0 & \text{sinon} \end{cases}$$

En utilisant les codes donnés (et expliqués) en section 2.4, on a immédiatement un code vectorisé permettant d'initialiser ce vecteur : il est donné en Listing 3.

Listing 3 – Calcul du vecteur second membre sans conditions aux limites

```
BD=BordDomaine(...);   % ... -> a remplacer
[I,J]=bijRecF(1:N,nx);
X=x(I+1);Y=y(J+1);% coordonnees des points de la grille
                  % en numerotation globale
f=@(x,y) sin(x+y).*cos(x-y);

idxBD=unique([BD.Sud,BD.Nord,BD.Est,BD.Ouest]);
idxBDc=setdiff(1:N,idxBD);   % N=nx*ny

B=zeros(N,1);
B(idxBDc)=f(X(idxBDc),Y(idxBDc));
```

2.6 Prise en compte des conditions aux limites

Sans tenir compte des conditions aux limites, nous avons décrit le calcul de la matrice $\mathbb{A}_{xy} \in \mathcal{M}_N(\mathbb{R})$ (section 2.3) et du vecteur $\underline{\mathbf{B}} \in \mathbb{R}^N$ (section 2.5).
En reprenant les résultats et notations de **Q.8** (page 5), on en déduit que

$$\mathbb{A} = -\mathbb{A}_{xy} + \mathbb{A}_\Gamma \quad \text{et} \quad \boldsymbol{B} = \underline{\mathbf{B}} + \boldsymbol{B}_\Gamma$$

où $\mathbb{A}_\Gamma$ et $\boldsymbol{B}_\Gamma$ sont les *contributions* des conditions aux limites, c'est à dire que le système

$$\mathbb{A}_\Gamma \boldsymbol{U} = \boldsymbol{B}_\Gamma \tag{2.13}$$

contient uniquement les équations (2.4) à (2.7). Pour $k = \mathcal{F}(i,j) \in [\![1, N]\!]$, si $(x(i), y(j)) \in \Gamma_\alpha$ avec $\alpha \in \{S, O, N, E\}$ alors la ligne k du système (2.13) correspond à

$$\boldsymbol{U}(k) = g_\alpha(x(i), y(j)).$$

Toutes les autres lignes du système (2.13) sont nulles. On a donc pour tout $k \in [\![1, N]\!]$

$$\mathbb{A}_\Gamma(k,:) = \begin{cases} \boldsymbol{e}_k^{\mathsf{t}} & \text{si } (x(i), y(j)) \in \Gamma \text{ avec } (i,j) = \mathcal{F}^{-1}(k) \\ 0 & \text{sinon} \end{cases}$$

où $\boldsymbol{e}_k^{\mathsf{t}}$ est le $k^{\text{ième}}$ vecteur de la base canonique de $\mathbb{R}^N$, et

$$\boldsymbol{B}_\Gamma(k) = \begin{cases} g_\alpha(x(i), y(j)) & \text{si } (x(i), y(j)) \in \Gamma_\alpha \text{ avec } (i,j) = \mathcal{F}^{-1}(k) \text{ et } \alpha \in \{S, O, N, E\} \\ 0 & \text{sinon} \end{cases}$$

En utilisant les codes donnés (et expliqués) en section 2.4, on calcule dans le Listing 4, le vecteur `Bcl` correspondant à $\boldsymbol{B}_\Gamma$ et la matrice `Acl` correspondant à $\mathbb{A}_\Gamma$.

Listing 4 – Calcul des contributions du bord dnas le cas $g_\alpha = g$, $\forall \alpha \in \{S, O, N, E\}$.

```
BD=BordDomaine(...);   % ... -> a remplacer
[I,J]=bijRecF(1:N,nx);
X=x(I+1);Y=y(J+1);% coordonnees des points de la grille
                  % en numerotation globale
g=@(x,y) sin(x+y).*cos(x-y);

idxBD=unique([BD.Sud,BD.Nord,BD.Est,BD.Ouest]);
idxBDc=setdiff(1:N,idxBD);   % N=nx*ny

Bcl=zeros(N,1);
Bcl(idxBD)=g(X(idxBD),Y(idxBD));
Acl=sparse(idxBD,idxBD,ones(1,length(idxBD)),N,N);
```

2.7 Résolution du système

On doit résoudre le système $\mathbb{A}\boldsymbol{U} = \boldsymbol{B}$ définit en **Q.8** (page 5). Or on a construit les matrices $\mathbb{A}_{xy}$, $\mathbb{A}_\Gamma$ et les vecteurs $\underline{\mathbf{B}}$, $\boldsymbol{B}_\Gamma$ de telle sorte que

$$\mathbb{A} = -\mathbb{A}_{xy} + \mathbb{A}_\Gamma \quad \text{et} \quad \boldsymbol{B} = \underline{\mathbf{B}} + \boldsymbol{B}_\Gamma$$

2.7.1 Méthode 1

Dans cette section nous allons directement résoudre le système $\mathbb{A}\boldsymbol{U} = \boldsymbol{B}$.

Q. 16 [code] ☆☆☆

a. Ecrire le programme `EDP2D01` *permettant de résoudre numériquement l'EDP* (2.1)-(2.2) *à l'aide du schéma discrétisé* (2.3) *à* (2.7). *On choisira judicieusement les données permettant sur une même figure de représenter de la solution numérique et de la solution exacte, et sur une autre figure de représenter l'erreur entre les deux solutions.*

b. Ecrire le programme `ordreEDP2D01` *permettant de retrouver numériquement l'ordre du schéma utilisé.*

2.7.2 Méthode 2

Dans cette section nous proposons une autre méthode permettant de résoudre le système $\mathbb{A}\boldsymbol{U} = \boldsymbol{B}$ en *éliminant* les équations portants sur les points Dirichlet.
Pour celà, on note

$$\mathcal{I}_D = \left\{k \in [\![1, N]\!];\ (x(i), y(j)) \in \Gamma \text{ avec } (i,j) = \mathcal{F}^{-1}(k)\right\}$$

et son complémentaire

$$\mathcal{I}_D^c = [\![1, N]\!] \backslash \mathcal{I}_D.$$

Le système linéaire à résoudre $\mathbb{A}\boldsymbol{U} = \boldsymbol{B}$ peut se décomposer de manière équivalente en

$$\mathbb{A}\boldsymbol{U} = \boldsymbol{B} \Leftrightarrow \begin{cases} \mathbb{A}(\mathcal{I}_D, :)\boldsymbol{U} = \boldsymbol{B}(\mathcal{I}_D) & \text{(2.14a)} \\ \mathbb{A}(\mathcal{I}_D^c, :)\boldsymbol{U} = \boldsymbol{B}(\mathcal{I}_D^c) & \text{(2.14b)} \end{cases}$$

Nous allons réécrire les deux systèmes (2.14a) et (2.14b).

- Pour (2.14a), nous avons, par construction, $\mathbb{A}(\mathcal{I}_D, :) = \mathbb{A}_\Gamma(\mathcal{I}_D, :)$ et donc

$$\begin{aligned} \mathbb{A}(\mathcal{I}_D, :)\boldsymbol{U} &= \mathbb{A}_\Gamma(\mathcal{I}_D, :)\boldsymbol{U} \\ &= \mathbb{A}_\Gamma(\mathcal{I}_D, \mathcal{I}_D)\boldsymbol{U}(\mathcal{I}_D) + \mathbb{A}_\Gamma(\mathcal{I}_D, \mathcal{I}_D^c)\boldsymbol{U}(\mathcal{I}_D^c) \end{aligned}$$

  Comme $\mathbb{A}_\Gamma(\mathcal{I}_D, \mathcal{I}_D^c) = \mathbb{0}$ et $\mathbb{A}_\Gamma(\mathcal{I}_D, \mathcal{I}_D) = \mathbb{I}$ on obtient

$$\mathbb{A}(\mathcal{I}_D, :)\boldsymbol{U} = \boldsymbol{U}(\mathcal{I}_D).$$

  Donc (2.14a) est équivalent à

$$\boldsymbol{U}(\mathcal{I}_D) = \boldsymbol{B}(\mathcal{I}_D). \tag{2.15}$$

- Pour (2.14b), nous avons par construction $\mathbb{A}(\mathcal{I}_D^c, :) = -\mathbb{A}_{x,y}(\mathcal{I}_D^c, :)$ et donc

$$\begin{aligned} \mathbb{A}(\mathcal{I}_D^c, :)\boldsymbol{U} &= -\mathbb{A}_{x,y}(\mathcal{I}_D^c, :)\boldsymbol{U} \\ &= -\mathbb{A}_{x,y}(\mathcal{I}_D^c, \mathcal{I}_D)\boldsymbol{U}(\mathcal{I}_D) - \mathbb{A}_{x,y}(\mathcal{I}_D^c, \mathcal{I}_D^c)\boldsymbol{U}(\mathcal{I}_D^c) \end{aligned}$$

  En utilisant (2.15), le système (2.14b) s'écrit

$$-\mathbb{A}_{x,y}(\mathcal{I}_D^c, \mathcal{I}_D^c)\boldsymbol{U}(\mathcal{I}_D^c) = \boldsymbol{B}(\mathcal{I}_D^c) + \mathbb{A}_{x,y}(\mathcal{I}_D^c, \mathcal{I}_D)\boldsymbol{B}(\mathcal{I}_D). \tag{2.16}$$

Résoudre le système linéaire $\mathbb{A}\boldsymbol{U} = \boldsymbol{B}$ est alors équivalent à calculer $\boldsymbol{U}(\mathcal{I}_D)$ par (2.15) et déterminer $\boldsymbol{U}(\mathcal{I}_D^c)$ en résolvant le système linéaire (2.16).

Q. 17 [code] ☆☆☆

a. Ecrire le programme `EDP2D02` *permettant de résoudre numériquement l'EDP* (2.1)-(2.2) *à l'aide du schéma discrétisé* (2.3) *à* (2.7) *que l'on résoudra par* (2.15) *et* (2.16). *On choisira judicieusement les données permettant sur une même figure de représenter de la solution numérique et de la solution exacte, et sur une autre figure de représenter l'erreur entre les deux solutions.*

b. Ecrire le programme `ordreEDP2D02` *permettant de retrouver numériquement l'ordre du schéma utilisé.*