TP sur les volumes finis

Introduction:

La simulation des EDPs permet de modéliser et de résoudre les problèmes les plus complèxes, en impliquant des phénomènes de diffusion, de dispersion et de transport dans des systèmes réels. Dans ce compte rendu, nous nous concentrons sur le transport d'un scalaire passif θ à travers un écoulement bidimensionnel. Notre but est d'appliquer la méthode des Volumes Finis pour résoudre numériquement cette équation de diffusion-dispersion et de mettre une comparaison des trois schèmas numériques suivants : Linéaire-Linéaire, Linéaire-Upwind et Linéaire-Quick en terme de précision et de stabilité.

Objectif:

Dans ce TP nous allons implémenter le schéma numérique calculé dans le TD4 à l'aide de la méthode des volumes finis. Nous nous mettrons dans plusieurs cas:

- Cas conductif pur en mettant la condition initiale au centre (on vérifie le caractère isotrope de la diffusion pure).
- Schéma numérique conducto-convectif linéaire-linéaire.

Grandeurs physiques:

$a = \frac{\lambda}{\rho C_p}$ la diffusivité thermique.

Equation analytique:

$$\mathrm{div}\left(\lambda \overrightarrow{\mathrm{grad}}(\theta)\right) - \mathrm{div}\left(\rho C_p \theta \overrightarrow{V}\right) = \rho C_p \frac{\partial}{\partial t}\theta$$

Pas de convection $\longrightarrow \mathrm{div}\left(\lambda \overrightarrow{\mathrm{grad}}(\theta)\right) = 0$

Um l'amplitude du tourbillon = vitesse du fluide.

```
%routine MATLAB
clear all;
close all;
clc;

%données du problème
Lx = 1; %longueur selon x
Ly = 1; %longueur selon y
Nx = 25; %nombre de points de discrétisation selon x
Ny = 25; %nombre de points de discrétisation selon y
N = Nx + Ny; %total
a = 1;
```

```
Um = 10;
Nt = 1000; %nombre d'itérations temporelles
T = 0.1; %durée entre deux itérations
```

Numérisation de la solution (conduction pure):

L'objectif est de numériser l'équation aux dérivées partielles caractéristique du problème sous la forme d'un système linéaire:

$\theta^{(n+1)} = A\theta^{(n)}$

$\theta^{(0)} = \theta(t = 0)$

Avec:

- $\theta^{(n)}$ un vecteur colonne de taille N.
- A une matrice de taille NxN.

Le domaine sur lequel on résout l'EDP est $\Omega = [-L_x, L_x] \times [-L_y, L_y]$. D'où l'on pose:

- $h_x = \frac{2L_x}{N_x}$
- $h_y = \frac{2L_y}{N_y}$
- $\mathrm{dt} = \frac{T}{N_t}$

```
hx = 2*Lx/Nx;
hy = 2*Ly/Ny;
dt = T/Nt;
```

On a l'équation suivante en notant $p = i + (j-1)N_x$: ${\theta_p}^{(n)} = {\theta_{i,j}}^{(n)} = \theta\left(-L_x + \frac{h_x}{2} + ih_x, -L_y + \frac{h_y}{2} + jh_y\right)^{(n)}$

On peut traduire le schéma numérique suivant:

$${\theta_p}^{(n+1)} = {\theta_p}^{(n)} + \frac{\mathrm{adt}}{{h_x}^2}\left({\theta_W}^{(n)} - 2{\theta_p}^{(n)} + {\theta_E}^{(n)}\right) + \frac{\mathrm{adt}}{{h_y}^2}\left({\theta_S}^{(n)} - 2{\theta_p}^{(n)} + {\theta_N}^{(n)}\right)$$

En schéma matriciel, en posant:

$${\theta_j}^{(n)} = \begin{bmatrix} {\theta_{1,j}}^{(n)} \\ \vdots \\ {\theta_{N_x}}^{(n)} \end{bmatrix}$$

On a ainsi:

$$\theta^{(n+1)} = A\theta^{(n)}$$

Avec: $\theta^{(n)} = \begin{bmatrix} \theta_1{}^{(n)} \\ \vdots \\ \theta_{N_y}{}^{(n)} \end{bmatrix}$

et,

$$A = \begin{bmatrix} I - B & \frac{\text{adt}}{h_y{}^2} I & 0 & \cdots & \cdots & 0 \\ \frac{\text{adt}}{h_y{}^2} I & I - B & \frac{\text{adt}}{h_y{}^2} I & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & 0 & \frac{\text{adt}}{h_y{}^2} I & I - B & \frac{\text{adt}}{h_y{}^2} I \\ 0 & \cdots & \cdots & 0 & \frac{\text{adt}}{h_y{}^2} I & I - B \end{bmatrix}$$

Où $B = \begin{bmatrix} 2\text{adt}\left(\frac{1}{h_x{}^2} + \frac{1}{h_y{}^2}\right) & -\frac{\text{adt}}{h_x{}^2} & 0 & 0 \\ -\frac{\text{adt}}{h_x{}^2} & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -\frac{\text{adt}}{h_x{}^2} \\ 0 & 0 & -\frac{\text{adt}}{h_x{}^2} & 2\text{adt}\left(\frac{1}{h_x{}^2} + \frac{1}{h_y{}^2}\right) \end{bmatrix}$

Création de la matrice A:

```
A = zeros(Nx*Ny,Nx*Ny);

%création de la matrice B
temp = -a*dt/(hx^2)*ones(Nx-1,1);
B = 2*a*dt*(1/(hx^2) + 1/(hy^2))*eye(Nx) + diag(temp,1) + diag(temp,-1);

for j = 1:Ny

    p = (j-1)*Nx + 1; %indice p pour se simplifier la compréhension

    A(p:p+Nx-1,p:p+Nx-1) = eye(Nx) - B; %on remplit la diagonale
```

```
    if j > 1 %si on n'est pas au bord gauche de la matrice A
        A(p:p-1+Nx,p-Nx:p-1) = a*dt/(hy)^2*eye(Nx); %on remplit la diagonale supérieure
    end

    if j < Ny %si on n'est pas au bord droit de la matrice A
        A(p:Nx+p-1,p+Nx:p+2*Nx-1) =a*dt/(hy)^2*eye(Nx); %on remplit la diagonale inférieure
    end
end
Acond = A %on stocke la matrice diffusion
```

```
Acond = 625×625
    0.9375    0.0156         0         0         0         0         0         0 ···
    0.0156    0.9375    0.0156         0         0         0         0         0
         0    0.0156    0.9375    0.0156         0         0         0         0
         0         0    0.0156    0.9375    0.0156         0         0         0
         0         0         0    0.0156    0.9375    0.0156         0         0
         0         0         0         0    0.0156    0.9375    0.0156         0
         0         0         0         0         0    0.0156    0.9375    0.0156
         0         0         0         0         0         0    0.0156    0.9375
         0         0         0         0         0         0         0    0.0156
         0         0         0         0         0         0         0         0
    ⋮
```

Gestion des conditions aux limites:

On sait que θ est nul aux bordx du domaine, soit pour:

- $j = 1$ ou N_y
- $i = 1$ ou N_x

Ce qui nous donne au final la matrice A:

$$A_{\text{cond}} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \frac{\text{adt}}{h_y^2} I & I - B & \frac{\text{adt}}{h_y^2} I & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & 0 & \frac{\text{adt}}{h_y^2} I & I - B & \frac{\text{adt}}{h_y^2} I \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix}$$

En remplaçant en plus la première et dernière ligne de chaque sous-matrice par des lignes de 0.

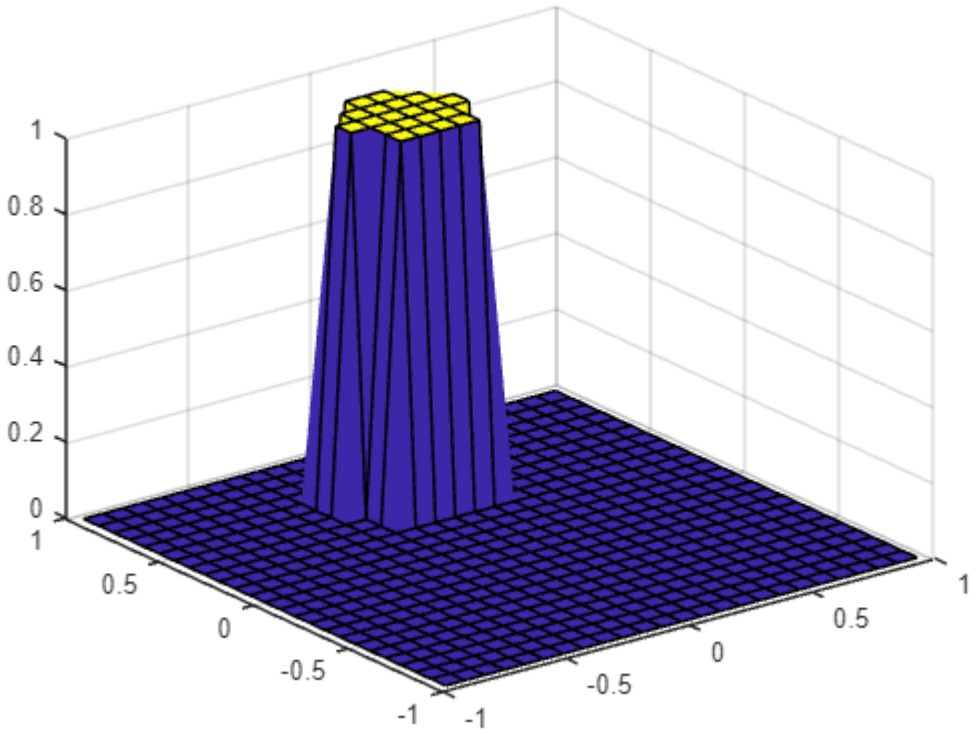
```
%Code MATLAB correspondant
A(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

for j = 2:Ny-1
```

```
    p = (j-1)*Nx + 1;
    A(p,:) = zeros(1,Nx*Ny);
    A(p+Nx-1,:) = zeros(1,Nx*Ny);
end
```

Création du vecteur théta et son initialisation:

Comme condition initiale, nous avons $\theta(x,y) = \theta^{(0)}(x,y) = 1$ si $(x,y)\,\epsilon\, D\left(\left(\frac{L_x}{2},0\right),\frac{L_x}{4}\right)$ et 0 sinon.

```
%code MATLAB correspondant
theta0 = zeros(Nx*Ny,1);
x_vec = -Lx:hx:Lx-hx;
y_vec = -Ly:hy:Ly-hy;
x_vec = x_vec + hx/2*ones(1,size(x_vec,2));
y_vec = y_vec + hy/2*ones(1,size(y_vec,2));

for i = 1:Nx
    for j = 1:Ny
        p = i + (j-1)*Nx;
        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        if (x-Lx/2)^2 + y^2 <= (Lx/4)^2
            theta0(p,1) = 1;
        end
    end
end

%affichage de theta0
    Thetam = reshape(theta0,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
```




```
    drawnow
    pause(0)
```

Résolution de l'EDP:

Ainsi, pour résoudre l'EDP il suffit de résoudre le système linéaire vu plus haut.

```
theta = theta0;
for n = 1:Nt/4
    theta = A*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```

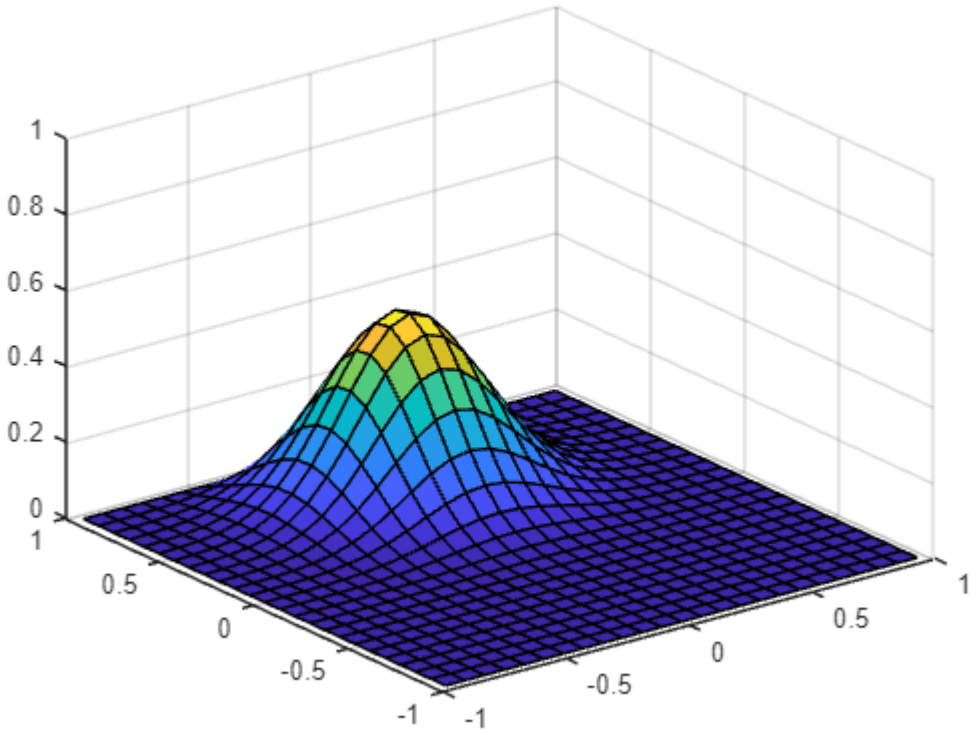


Schéma conducto-convectif linéaire-linéaire:

Désormais, intéressons-nous au schéma conducto-convectif linéaire-linéaire. On rajoute ainsi le terme convectif et l'équation devient:

$$\longrightarrow \mathrm{div}\left(\lambda \overrightarrow{\mathrm{grad}}(\theta)\right) - \mathrm{div}\left(\rho C_p \theta \overrightarrow{V}\right) = \rho C_p \frac{\partial}{\partial t}\theta$$

Ainsi, notre matrice A, par linéarité sera juste celle précédente + celle correspondant à la partie convection. De cette manière, penchons nous sur la discrétisation de la partie convective, soit: $-\mathrm{div}\left(\rho C_p \theta \overrightarrow{V}\right)$.

Pour ce faire, nous aurons besoin des fonctions:

- $u(x, y) = -u_m \frac{y}{x^2 + y^2}$
- $v(x, y) = u_m \frac{x}{x^2 + y^2}$

Qui correspondent au champ de vitesse "spirale aspirante":

$$\overrightarrow{V}(x, y) = \begin{bmatrix} u(x, y) \\ v(x, y) \end{bmatrix}$$

En moyennant sur un petit volume Ω_p, avec la formule de la divergence:

$$\frac{1}{\Omega_p}\int\int_{\Omega_p} \mathrm{div}\left(\rho C_p \theta \overrightarrow{V}\right) d\Omega_p = \frac{1}{\Omega_p}\int_{\partial\Omega_p} \rho C_p \theta \overrightarrow{V}.\overrightarrow{n}\, dS_p$$

$\partial\Omega_p$ pouvant être coupé en 4 surfaces $S_e, e \in \{N, E, S, W\}$, avec pour vecteurs normaux respectivement:

- $\overrightarrow{n_N} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- $\overrightarrow{n_E} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- $\overrightarrow{n_S} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$
- $\overrightarrow{n_W} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

Or: $\int_{S_e} f(x,y) dS_e \approx f_e S_e$.

On obtient finalement la discrétisation suivante pour $f = \rho C_p \theta \overrightarrow{V}.\overrightarrow{n}$:

$-\frac{\mathrm{dt}}{\Omega_p}\left(\frac{\theta_p^{(n)} + \theta_E^{(n)}}{2} u_E h_y + \frac{\theta_p^{(n)} + \theta_N^{(n)}}{2} v_N h_x - \frac{\theta_p^{(n)} + \theta_S^{(n)}}{2} v_S h_x - \frac{\theta_p^{(n)} + \theta_W^{(n)}}{2} u_W h_y\right)$ car on divise par ρC_p et multiplie par dt avec le terme de droite.

ou encore: $\frac{\theta_p^{(n)}}{2}\left[\frac{(u_W - u_E)}{h_x} + \frac{(v_S - v_N)}{h_y}\right]\mathrm{dt} - \frac{\theta_N^{(n)}}{2h_y} v_N \mathrm{dt} + \frac{\theta_S^{(n)}}{2h_y} v_S \mathrm{dt} + \frac{\theta_W^{(n)}}{2h_x} u_W \mathrm{dt} - \frac{\theta_E^{(n)}}{2h_x} u_E \mathrm{dt}$

Avec:

- $u_E = u\left(x_i + \frac{h_x}{2}, y_j\right)$
- $u_W = u\left(x_i - \frac{h_x}{2}, y_j\right)$
- $v_N = v\left(x_i, y_j + \frac{h_y}{2}\right)$
- $v_S = v\left(x_i, y_j - \frac{h_y}{2}\right)$
- $\Omega_p = h_x h_y$

```
%code MATLAB correspondant
u = @(x,y,Um) -Um*y/(x^2 + y^2);
v = @(x,y,Um) Um*x/(x^2 + y^2);
```

Création de la matrice de convection:

Du schéma numérique donnée plus haut, on peut établir la matrice de convection A_{conv} du problème:

$$A_{\text{conv}} = -\frac{\text{dt}}{2h_x h_y}\begin{bmatrix} C & v_N h_x I & 0 & \cdots & 0 \\ -v_S h_x I & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & v_N h_x I \\ 0 & \cdots & 0 & -v_S h_x I & C \end{bmatrix}$$

Avec:

$$C = \begin{bmatrix} (u_E - u_W)h_y + (v_N - v_S)h_x & u_E h_y & 0 & \cdots & 0 \\ -u_W h_y & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & u_E h_y \\ 0 & \cdots & 0 & -u_W h_y & (u_E - u_W)h_y + (v_N - v_S)h_x \end{bmatrix}.$$

Evidemment, les uE etc ne sont pas les mêmes à chaque coefficient. Il s'agit juste ici de montrer la forme de la matrice de manière syntaxique.

Par conséquent: $A = A_{\text{cond}} + A_{\text{conv}}$. A partir de là on sait résoudre: on fait comme avant!

```
%code MATLAB correspondant
Aconv = zeros(Nx*Ny,Nx*Ny);

for j = 2:Ny-1
    for i = 2:Nx-1
        p = i + (j-1)*Nx;
%         x = x_vec(i); %abscisse du maillon
%         y = y_vec(j); %ordonnée du maillon

        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        uE = u(x+hx/2,y,Um);
        uW = u(x-hx/2,y,Um);
        vN = v(x,y+hy/2,Um);
        vS = v(x,y-hy/2,Um);

        Aconv(p,p) = dt/2*((uW-uE)/hx + (vS-vN)/hy);
        Aconv(p,p+1) = -uE/(2*hx)*dt;
```

```
        Aconv(p,p-1) = uW/(2*hx)*dt;
        Aconv(p,p+Nx) = -vN/(2*hy)*dt;
        Aconv(p,p-Nx) = vS/(2*hy)*dt;

    end
end
Aconv
```

```
Aconv = 625×625
     0     0     0     0     0     0     0     0     0     0     0     0     0 ···
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     ⋮
```

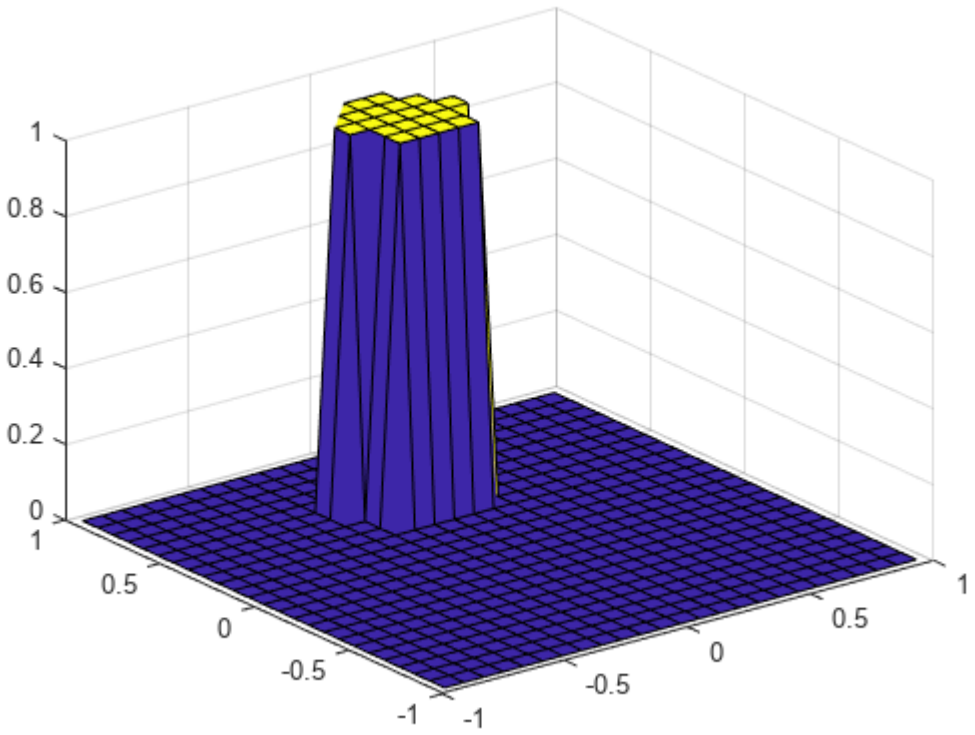
```
%d'où
A = Aconv + Acond;
```

Conditions aux limites:

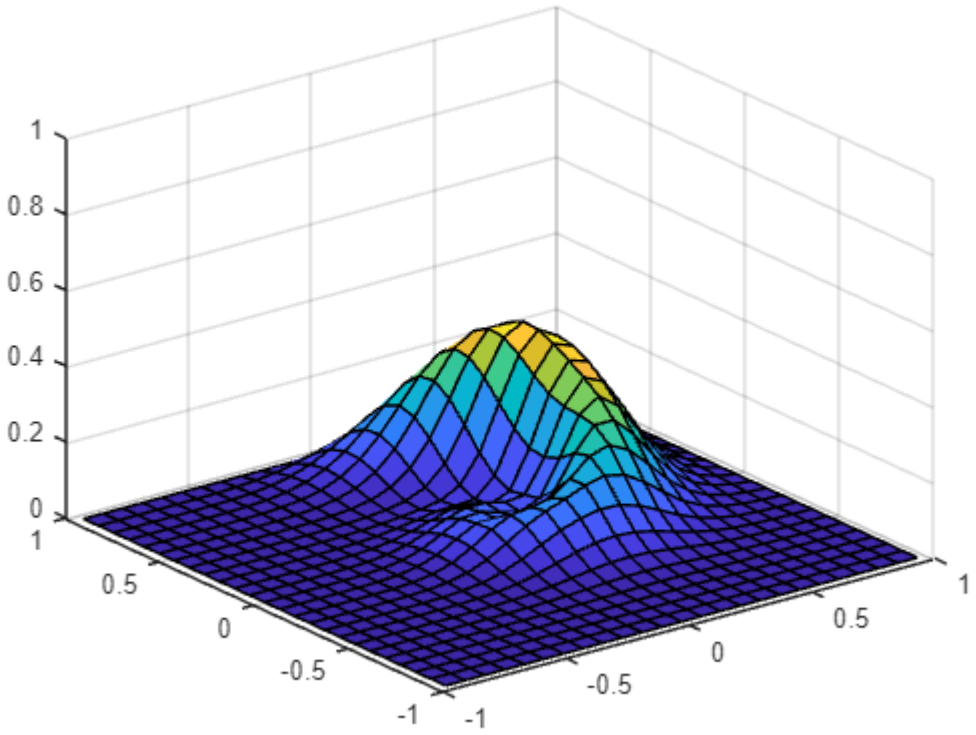
```
A(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

for j = 2:Ny-1
    p = (j-1)*Nx + 1;
    A(p,:) = zeros(1,Nx*Ny);
    A(p+Nx-1,:) = zeros(1,Nx*Ny);
end
```

Résolution:

```
theta = theta0;
Thetam = reshape(theta,Nx,Ny);
surf(x_vec,y_vec,Thetam);
```




```
pause(1)
for n = 1:Nt/4
    theta = A*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```



On observe la présence d'un tourbillon

Schéma Upwind

Désormais, nous allons nous pencher sur le schéma linéaire-Upwind (ordre 2 en espace et ordre 1 en temps), qui modifie le schéma convectif. On a comme approximation ***1) Surface Est:***

$$\frac{1}{\Omega_p}\int_{S_e} \rho C_p \theta \vec{V}.\vec{n}\, dS_p \approx \frac{\rho C_p}{h_x}\theta_e u_e$$

et on prend:

- $\theta_e = \theta_p$ si $u_e > 0$
- $\theta_e = \theta_E$ si $u_e < 0$

d'où $u_e\theta_e = \max(u_e, 0)\theta_p + \min(u_e, 0)\theta_E$

On retrouve des cas similaires pour les autres surfaces:

2) Surface nord:

$$\frac{1}{\Omega_p}\int_{S_n} \rho C_p \theta \vec{V}.\vec{n}\, dS_p \approx \frac{\rho C_p}{h_y}\theta_n v_n$$

et on prend:

- $\theta_n = \theta_p$ si $v_n > 0$
- $\theta_n = \theta_N$ si $v_n < 0$

d'où $v_n\theta_n = \max(v_n, 0)\theta_p + \min(v_n, 0)\theta_N$

3) Surface Ouest:

$$\frac{1}{\Omega_p}\int_{S_w} \rho C_p \theta \vec{V}.\vec{n}\, dS_p \approx -\frac{\rho C_p}{h_x}\theta_w u_w$$

et on prend:

- $\theta_w = \theta_p$ si $u_w > 0$
- $\theta_w = \theta_W$ si $u_w < 0$

d'où $u_w\theta_w = \max(u_w, 0)\theta_p + \min(u_w, 0)\theta_W$

4) Surface Sud:

$$\frac{1}{\Omega_p}\int_{S_s} \rho C_p \theta \vec{V}.\vec{n}\, dS_p \approx -\frac{\rho C_p}{h_y}\theta_s v_s$$

et on prend:

- $\theta_s = \theta_p$ si $v_s > 0$
- $\theta_s = \theta_S$ si $v_s < 0$

d'où $v_s\theta_s = \max(v_s, 0)\theta_p + \min(v_s, 0)\theta_S$

On obtient finalement pour schéma numérique de la partie convection:

$-\frac{\mathrm{dt}}{h_x}\theta_e u_e + \frac{\mathrm{dt}}{h_x}\theta_w u_w - \frac{\mathrm{dt}}{h_y}\theta_n v_n + \frac{\mathrm{dt}}{h_y}\theta_s v_s$ Dans la construction de la matrice de convection, il suffira d'étudier les différents cas cités plus haut. Réécrit de la manière suivante, voici le schéma du terme convectif:

$$-\mathrm{dt}\left(\frac{\max(0, u_e) - \max(0, u_w)}{h_x} + \frac{\max(0, v_n) - \max(0, v_s)}{h_y}\right)\theta_p - \frac{\mathrm{dt}}{h_x}\min(0, u_e)\theta_E + \frac{\mathrm{dt}}{h_x}\min(0, u_w)\theta_W - \frac{\mathrm{dt}}{h_y}\min(0, v_n)\theta_N + \frac{\mathrm{dt}}{h_y}\min(0, v_s)\theta_S$$

```
%code MATLAB correspondant
Aconv_up = zeros(Nx*Ny,Nx*Ny);

for j = 2:Ny-1
    for i = 2:Nx-1
        p = i + (j-1)*Nx;
%         x = x_vec(i); %abscisse du maillon
%         y = y_vec(j); %ordonnée du maillon
```

```
        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        uE = u(x+hx/2,y,Um);
        uW = u(x-hx/2,y,Um);
        vN = v(x,y+hy/2,Um);
        vS = v(x,y-hy/2,Um);

        Aconv_up(p,p) = -dt*((max(0,uE) - max(0,uW))/hx + (max(0,vN) - max(0,vS))/hy);
        Aconv_up(p,p+1) = -dt/hx*min(0,uE);
        Aconv_up(p,p-1) = dt/hx*min(0,uW);
        Aconv_up(p,p+Nx) = -dt/hy*min(0,vN);
        Aconv_up(p,p-Nx) = dt/hy*min(0,vS);

    end
end
Aconv_up
```

```
Aconv_up = 625×625
     0     0     0     0     0     0     0     0     0     0     0     0     0 ···
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     ⋮
```

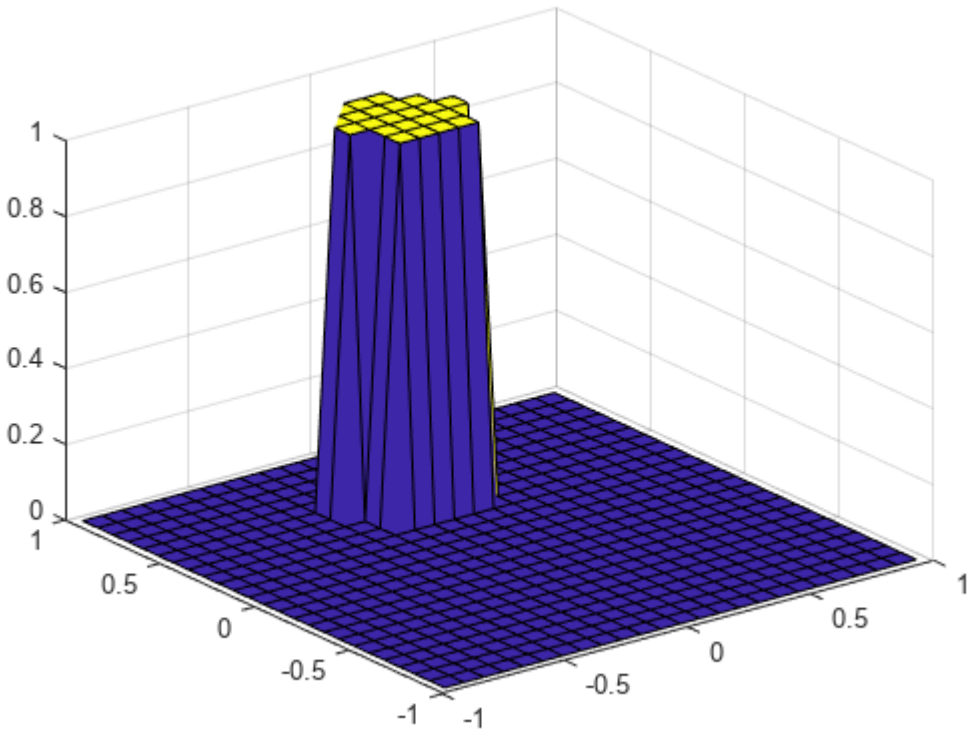
```
A_up = Aconv_up + Acond;
```

Conditions aux limites:

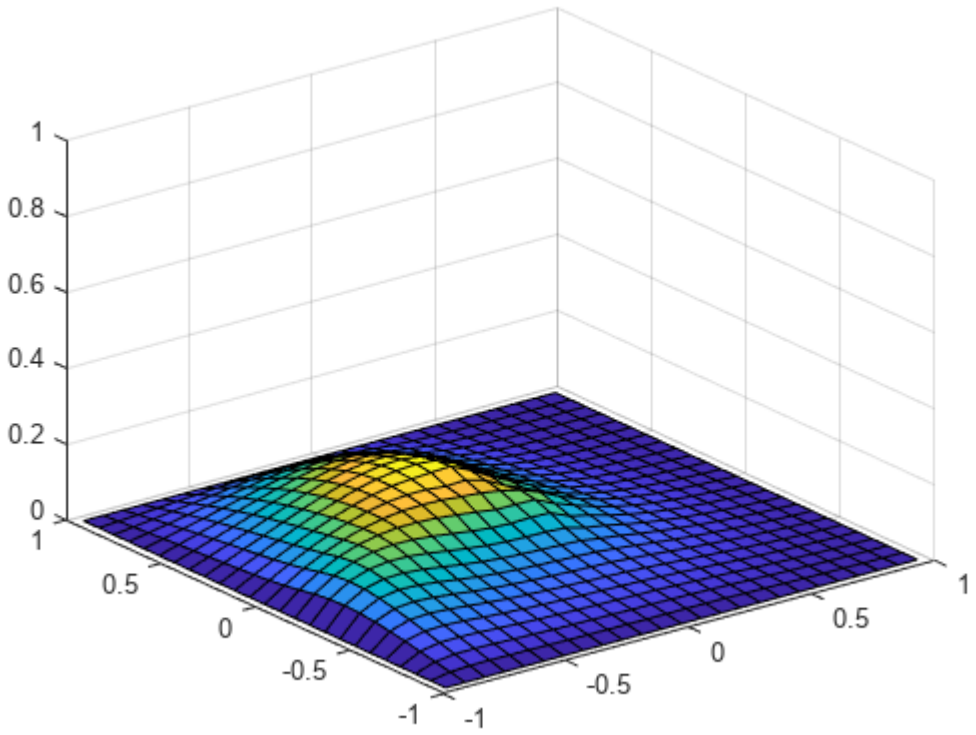
```
A_up(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A_up((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

for j = 2:Ny-1
    p = (j-1)*Nx + 1;
    A_up(p,:) = zeros(1,Nx*Ny);
    A_up(p+Nx-1,:) = zeros(1,Nx*Ny);
end
```

Résolution

```
theta = theta0;
Thetam = reshape(theta,Nx,Ny);
surf(x_vec,y_vec,Thetam);
```




```
pause(1)

for n = 1:Nt
    theta = A_up*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```



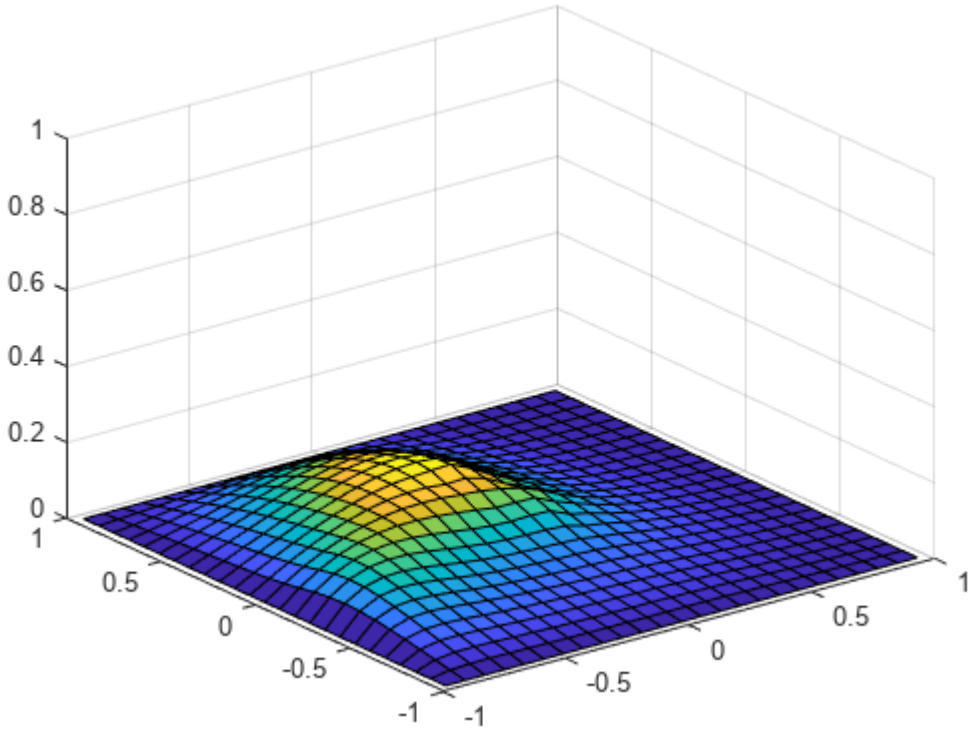
Etude de stabilité des schémas:

Désormais, penchons-nous sur la stabilité de ces deux schémas. Pour ce faire, nous avons fait des simulations en faisant varier les pas d'espace en x et en y, mais aussi la valeur de Um. En voici les résultats:

1. **Première simulation:Um = 1.5; Nx = 99; Ny = 99:**

```
Um = 1.5;
Nx = 99;
Ny = 99;
hx = 2*Lx/Nx;
hy = 2*Ly/Ny;

%conditions initiales
%code MATLAB correspondant
theta0 = zeros(Nx*Ny,1);
x_vec = -Lx:hx:Lx-hx;
y_vec = -Ly:hy:Ly-hy;
x_vec = x_vec + hx/2*ones(1,size(x_vec,2));
y_vec = y_vec + hy/2*ones(1,size(y_vec,2));

for i = 1:Nx
```

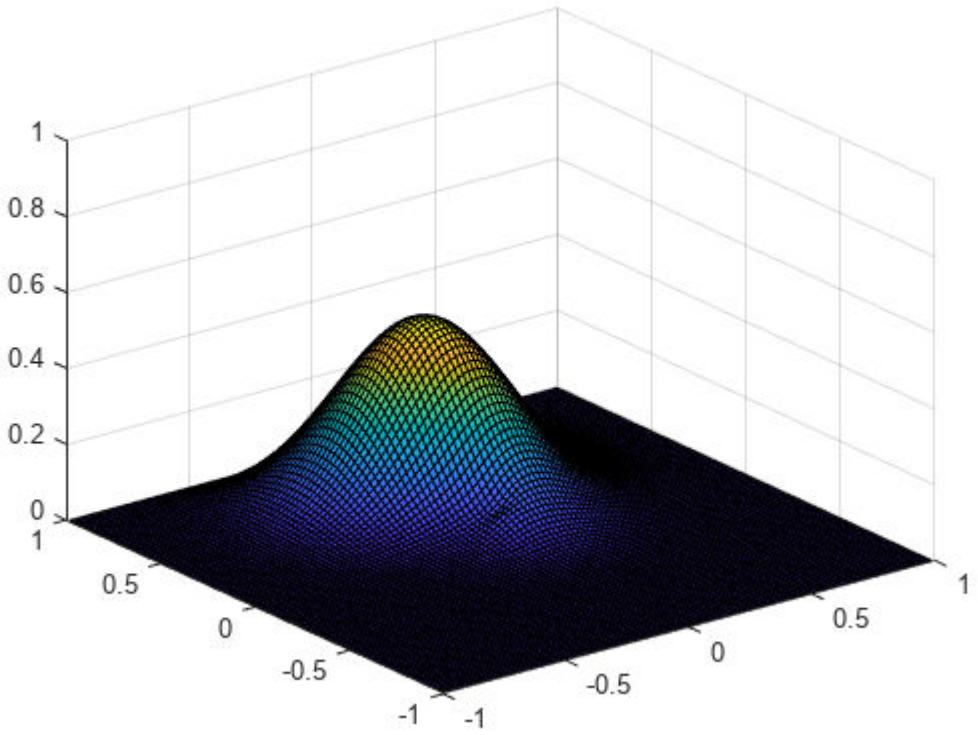
```
    for j = 1:Ny
        p = i + (j-1)*Nx;
        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        if (x-Lx/2)^2 + y^2 <= (Lx/4)^2
            theta0(p,1) = 1;
        end
    end
end
    axis([-Lx,Lx,-Ly,Ly,0,1])
```




```
%construction de la matrice de diffusion
Acond = zeros(Nx*Ny,Nx*Ny);

%création de la matrice B
temp = -a*dt/(hx^2)*ones(Nx-1,1);
B = 2*a*dt*(1/(hx^2) + 1/(hy^2))*eye(Nx) + diag(temp,1) + diag(temp,-1);

for j = 1:Ny

    p = (j-1)*Nx + 1; %indice p pour se simplifier la compréhension

    Acond(p:p+Nx-1,p:p+Nx-1) = eye(Nx) - B; %on remplit la diagonale

    if j > 1 %si on n'est pas au bord gauche de la matrice A
```

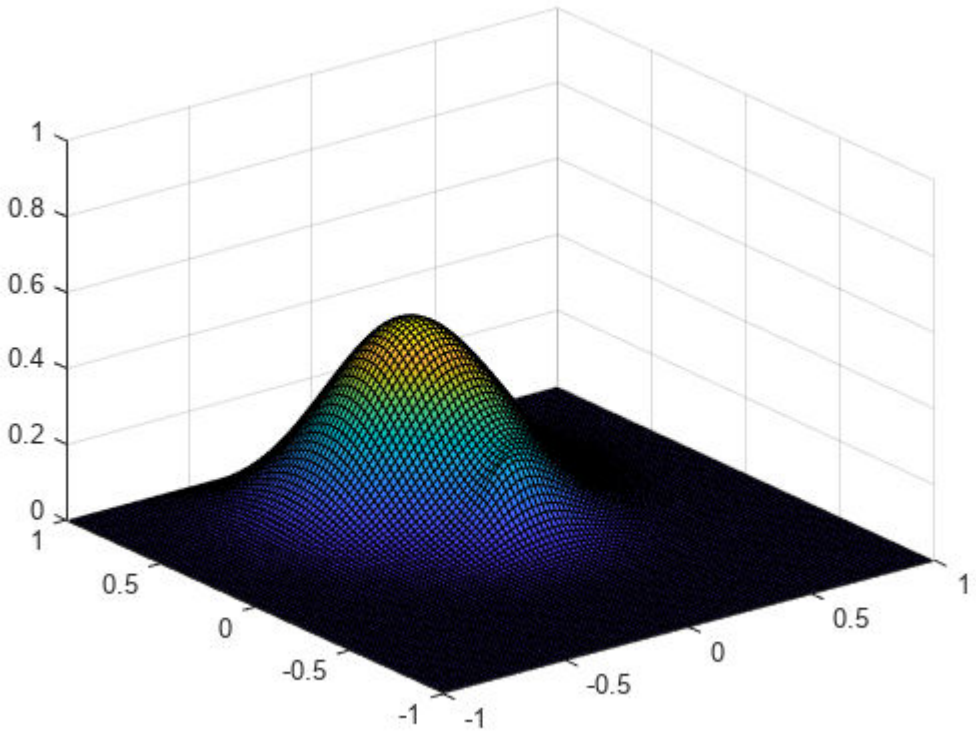
```
        Acond(p:p-1+Nx,p-Nx:p-1) = a*dt/(hy)^2*eye(Nx); %on remplit la diagonale supérieure
    end

    if j < Ny %si on n'est pas au bord droit de la matrice A
        Acond(p:Nx+p-1,p+Nx:p+2*Nx-1) =a*dt/(hy)^2*eye(Nx); %on remplit la diagonale inférieure
    end
end

%construction de la matrice de convection (linéaire)
Aconv = zeros(Nx*Ny,Nx*Ny);

for j = 2:Ny-1
    for i = 2:Nx-1
        p = i + (j-1)*Nx;
        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        uE = u(x+hx/2,y,Um);
        uW = u(x-hx/2,y,Um);
        vN = v(x,y+hy/2,Um);
        vS = v(x,y-hy/2,Um);

        Aconv(p,p) = dt/2*((uW-uE)/hx + (vS-vN)/hy);
        Aconv(p,p+1) = -uE/(2*hx)*dt;
        Aconv(p,p-1) = uW/(2*hx)*dt;
        Aconv(p,p+Nx) = -vN/(2*hy)*dt;
        Aconv(p,p-Nx) = vS/(2*hy)*dt;

    end
end

%construction de la matrice de convection (upwind)
Aconv_up = zeros(Nx*Ny,Nx*Ny);

for j = 2:Ny-1
    for i = 2:Nx-1
        p = i + (j-1)*Nx;
%         x = x_vec(i); %abscisse du maillon
%         y = y_vec(j); %ordonnée du maillon

        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        uE = u(x+hx/2,y,Um);
        uW = u(x-hx/2,y,Um);
        vN = v(x,y+hy/2,Um);
        vS = v(x,y-hy/2,Um);

        Aconv_up(p,p) = -dt*((max(0,uE) - max(0,uW))/hx + (max(0,vN) - max(0,vS))/hy);
        Aconv_up(p,p+1) = -dt/hx*min(0,uE);
```

```
        Aconv_up(p,p-1) = dt/hx*min(0,uW);
        Aconv_up(p,p+Nx) = -dt/hy*min(0,vN);
        Aconv_up(p,p-Nx) = dt/hy*min(0,vS);

    end
end

A_up = Acond + Aconv_up;

A = Acond + Aconv;

%conditions aux limites
A(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

A_up(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A_up((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

for j = 2:Ny-1
    p = (j-1)*Nx + 1;
    A(p,:) = zeros(1,Nx*Ny);
    A(p+Nx-1,:) = zeros(1,Nx*Ny);
    A_up(p,:) = zeros(1,Nx*Ny);
    A_up(p+Nx-1,:) = zeros(1,Nx*Ny);
end

theta = theta0;

%résolution (linéaire)
pause(1)

for n = 1:Nt/4
    theta = A*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```



On voit que le code est toujours efficace et qu'on conserve la stabilité du schéma. Un pas plus petit n'est pas possible puisqu'on dépasse le cas échéant la capacité de stockage de MATLAB. Regardons ce que nous donne le schéma upwind:

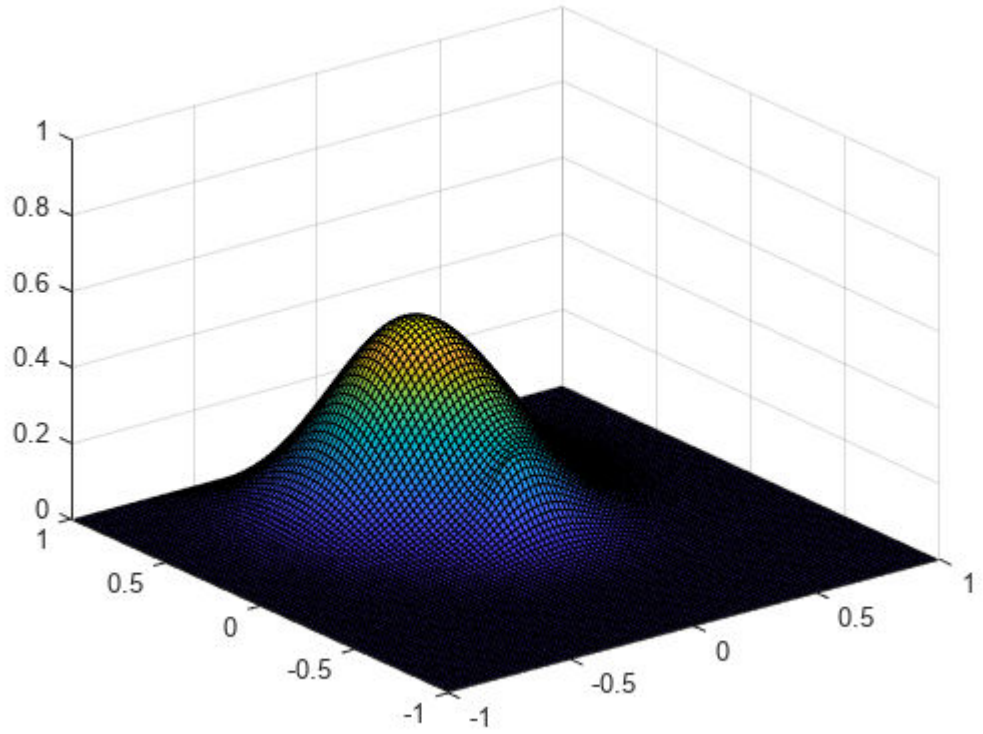
```
%résolution (upwind)
pause(1)
theta = theta0;

for n = 1:Nt/4
    theta = A_up*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```



2. **Deuxième simulation: Um = 10; Nx = 99; Ny = 99:**

```
Um = 10;
Nx = 99;
Ny = 99;
hx = 2*Lx/Nx;
hy = 2*Ly/Ny;

%conditions initiales
%code MATLAB correspondant
theta0 = zeros(Nx*Ny,1);
x_vec = -Lx:hx:Lx-hx;
y_vec = -Ly:hy:Ly-hy;
x_vec = x_vec + hx/2*ones(1,size(x_vec,2));
y_vec = y_vec + hy/2*ones(1,size(y_vec,2));

for i = 1:Nx
    for j = 1:Ny
        p = i + (j-1)*Nx;
        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        if (x-Lx/2)^2 + y^2 <= (Lx/4)^2
            theta0(p,1) = 1;
```

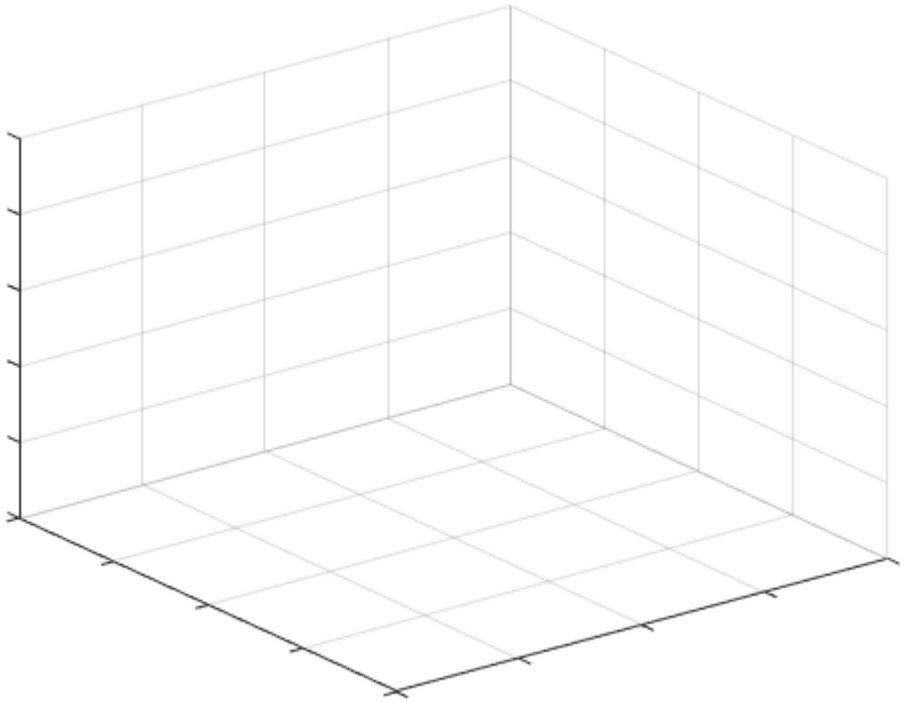
```
        end
    end
end
    axis([-Lx,Lx,-Ly,Ly,0,1])
```




```
%construction de la matrice de diffusion
Acond = zeros(Nx*Ny,Nx*Ny);

%création de la matrice B
temp = -a*dt/(hx^2)*ones(Nx-1,1);
B = 2*a*dt*(1/(hx^2) + 1/(hy^2))*eye(Nx) + diag(temp,1) + diag(temp,-1);

for j = 1:Ny

    p = (j-1)*Nx + 1; %indice p pour se simplifier la compréhension

    Acond(p:p+Nx-1,p:p+Nx-1) = eye(Nx) - B; %on remplit la diagonale

    if j > 1 %si on n'est pas au bord gauche de la matrice A
        Acond(p:p-1+Nx,p-Nx:p-1) = a*dt/(hy)^2*eye(Nx); %on remplit la diagonale supérieure
    end

    if j < Ny %si on n'est pas au bord droit de la matrice A
        Acond(p:Nx+p-1,p+Nx:p+2*Nx-1) =a*dt/(hy)^2*eye(Nx); %on remplit la diagonale inférieure
    end
end
```

```
%construction de la matrice de convection (linéaire)
Aconv = zeros(Nx*Ny,Nx*Ny);

for j = 2:Ny-1
    for i = 2:Nx-1
        p = i + (j-1)*Nx;
        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        uE = u(x+hx/2,y,Um);
        uW = u(x-hx/2,y,Um);
        vN = v(x,y+hy/2,Um);
        vS = v(x,y-hy/2,Um);

        Aconv(p,p) = dt/2*((uW-uE)/hx + (vS-vN)/hy);
        Aconv(p,p+1) = -uE/(2*hx)*dt;
        Aconv(p,p-1) = uW/(2*hx)*dt;
        Aconv(p,p+Nx) = -vN/(2*hy)*dt;
        Aconv(p,p-Nx) = vS/(2*hy)*dt;

    end
end

A = Acond + Aconv;

%création de la matrice de convection (upwind)
Aconv_up = zeros(Nx*Ny,Nx*Ny);

for j = 2:Ny-1
    for i = 2:Nx-1
        p = i + (j-1)*Nx;
%         x = x_vec(i); %abscisse du maillon
%         y = y_vec(j); %ordonnée du maillon

        y = -Ly + hy/2 + (j-1)*hy; %ordonnee du maillon
        x = -Lx + hx/2 + (i-1)*hx; %abscisse du maillon

        uE = u(x+hx/2,y,Um);
        uW = u(x-hx/2,y,Um);
        vN = v(x,y+hy/2,Um);
        vS = v(x,y-hy/2,Um);

        Aconv_up(p,p) = -dt*((max(0,uE) - max(0,uW))/hx + (max(0,vN) - max(0,vS))/hy);
        Aconv_up(p,p+1) = -dt/hx*min(0,uE);
        Aconv_up(p,p-1) = dt/hx*min(0,uW);
        Aconv_up(p,p+Nx) = -dt/hy*min(0,vN);
        Aconv_up(p,p-Nx) = dt/hy*min(0,vS);

    end
```

```
end

A_up = Aconv_up + Acond;

%conditions aux limites
A(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

A_up(1:Nx,:) = zeros(Nx,Nx*Ny); %premier bloc de ligne à 0
A_up((Nx-1)*Ny+1:Ny*Nx,:) = zeros(Nx,Nx*Ny); %dernier bloc de ligne à 0

for j = 2:Ny-1
    p = (j-1)*Nx + 1;
    A(p,:) = zeros(1,Nx*Ny);
    A(p+Nx-1,:) = zeros(1,Nx*Ny);
    A_up(p,:) = zeros(1,Nx*Ny);
    A_up(p+Nx-1,:) = zeros(1,Nx*Ny);
end


theta = theta0;

%résolution (linéaire)
pause(1)

for n = 1:Nt/4
    theta = A*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```



On **observe** la divergence du système avec le schéma linéaire. Le système est donc beaucoup moins stable et sensible aux pas hx et hy. Regardons désormais le schéma Upwind:

```
%résolution (upwind)
pause(1)
theta = theta0;

for n = 1:Nt/4
    theta = A_up*theta;

    %affichage de theta
    Thetam = reshape(theta,Nx,Ny);
    surf(x_vec,y_vec,Thetam);
    axis([-Lx,Lx,-Ly,Ly,0,1])
    drawnow
    pause(0)
end
```

On atteste également d'une divergence du schéma. On peut également en remarquer une sur ce dernier pour Nx = Ny = 41 (alors que le linéaire-linéaire lui ne diverge pas dans ce cas). Ce schéma a l'air plus précis mais est beaucoup moins stable.

Conclusion :

En conclusion, nous avons réussi à étudié le transport d'un scalaire passif dans un domaine bidimensionnel en utilisant la méthode des Volumes Finis. Nous avons mis en place quatre schémas numériques, tels que le schéma de la diffusion pure, le schéma linéaire-linéaire, le schéma linéaire-Upwind et le schéma Quick d'ordre 2 dans le but de traiter le terme de transport. Nous avons effectué une analyse de stabilité et de précision de ces schémas. Enfin, nous avons éffectué une comparaison entre les trois derniers schémas. L'étude de ce problème nous a permis de mieux comprendre les principes et les limites des différents schémas numériques et aussi leur impact sur la précision et la stabilité des résultats. La simulation des équations aux dérivées partielles reste un domaine de recherche fascinant et essentiel pour modéliser et résoudre une variété de problèmes physiques et scientifiques.