SAMMAN Mohamad   **Projet NS 2D**   1

L'écoulement d'un fluide incompressible est régi par les EDP suivantes :

$$E1 \begin{cases} \partial_t u + u\partial_x u + v\partial_y u = -\partial_x p + \nu(\partial_x^2 u + \partial_y^2 u) \\ \partial_t v + u\partial_x v + v\partial_y v = -\partial_y p + \nu(\partial_x^2 v + \partial_y^2 v) \\ \partial_x u + \partial_y v = 0 \end{cases}$$

inconnues $u(x,y,t)$, $v(x,y,t)$ et $p(x,y,t)$. On considère un domaine $\Omega = [0, L_x] \times [0, L_y]$ et un intervalle de temps $[0, T]$ avec des conditions aux limites periodiques. On choisit une condition initiale.

1/ Discrétisation temporelle

On considere une discretisation explicite. Dans un premier temps on utilise le shéma "Euler explicite" d'ordre 1.

$$E2 \begin{cases} \frac{1}{\delta t}(u^{n+1} - u^n) + NLu^n = -\partial_x p^{n+1} + Lu^n & (1) \\ \frac{1}{\delta t}(v^{n+1} - v^n) + NLv^n = -\partial_y p^{n+1} + Lv^n & (2) \\ \partial_x u^{n+1} + \partial_y v^{n+1} = 0 & (3) \end{cases}$$

avec $E_2' \begin{cases} NLu^n = u^n\partial_x u^n + v^n\partial_y u^n \\ NLv^n = u^n\partial_x v^n + v^n\partial_y v^n \end{cases}$

$E_2' \begin{cases} Lu^n = \nu(\partial_x^2 u^n + \partial_y^2 u^n) \\ Lv^n = \nu(\partial_x^2 v^n + \partial_y^2 v^n) \end{cases}$

Afin d'imposer la contrainte d'incompressibilité on utilise la méthode de projection.

Un champ de divergence non nulle $\vec{u}^*$ se décompose en un champ à divergence nulle $\vec{u}$ et $\vec{\nabla}\phi$: $\vec{u}^* = \vec{u} + \vec{\nabla}\phi$

$\Rightarrow \vec{\nabla}\cdot\vec{u}^* = \vec{\nabla}\vec{u} + \vec{\nabla}\cdot\vec{\nabla}\phi$ et on impose $\vec{\nabla}\vec{u} = 0$

$\Rightarrow \vec{\nabla}\vec{\nabla}\phi = \vec{\nabla}\vec{u}^*$

$\Rightarrow \vec{u} = \vec{u}^* - \vec{\nabla}\phi$

2/ Algorithme de projection

étape 1
prédiction
$$\begin{cases} u^* = u^n + \Delta t\,(-NLu^n - \partial_x p^n + Lu^n) \\ v^* = v^n + \Delta t\,(-NLv^n - \partial_y p^n + Lv^n) \end{cases}$$

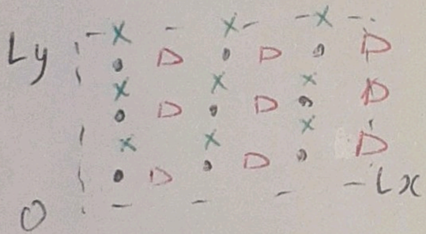
étape 2
correction
$$\vec{\nabla}\cdot\vec{\nabla}\phi = \frac{1}{\Delta t}\vec{\nabla}\cdot\vec{u}^*$$
$$\partial_x\partial_x\phi + \partial_y\partial_y\phi = \frac{1}{\Delta t}(\partial_x u^* + \partial_y v^*)$$

étape 3
projection
$$\begin{cases} u^{n+1} = u^* - \Delta t\,\partial_x\phi \\ v^{n+1} = v^* - \Delta t\,\partial_y\phi \end{cases}$$

ainsi on passe du temps t^n au temps $t^{n+1} = t^n + \Delta t$

3/ discrétisation spatiale

On utilise la méthode des différences finies (DF ordre 2 centrées)



∘ p × v ▷ u

$u^n_{ij} = u(t^n, x_i^f, y_j^c)$ x^f face x^c centre

$$\vec{\nabla}p = \begin{pmatrix}\partial_x p \\ \partial_y p\end{pmatrix}$$ on l'approche (discrétise) : $$\begin{pmatrix}(p_{i+1,j} - p_{i,j})/h_x \\ (p_{i,j+1} - p_{i,j})/h_y\end{pmatrix}$$

$$\vec{\nabla}\cdot\vec{u} \approx \frac{1}{h_x}(u_{i,j} - u_{i-1,j}) + \frac{1}{h_y}(v_{i,j} - v_{i,j-1})$$

$$Lu = \frac{1}{h_x^2}(u_{i+1,j} - 2u_{ij} + u_{i-1,j}) + \frac{1}{h_y^2}(u_{i,j+1} - 2u_{ij} + u_{i,j-1})$$

de même pour Lv

Pour les conditions périodiques on crée IM et IP (i moins et i plus)

$$IM(i) = \begin{cases} i-1 & \text{si } i \neq 0 \\ n_x - 1 & \text{si } i = 0 \end{cases} \qquad IP(i) = \begin{cases} i+1 & \text{si } i \neq n_x - 1 \\ 0 & \text{si } i = n_x - 1 \end{cases}$$

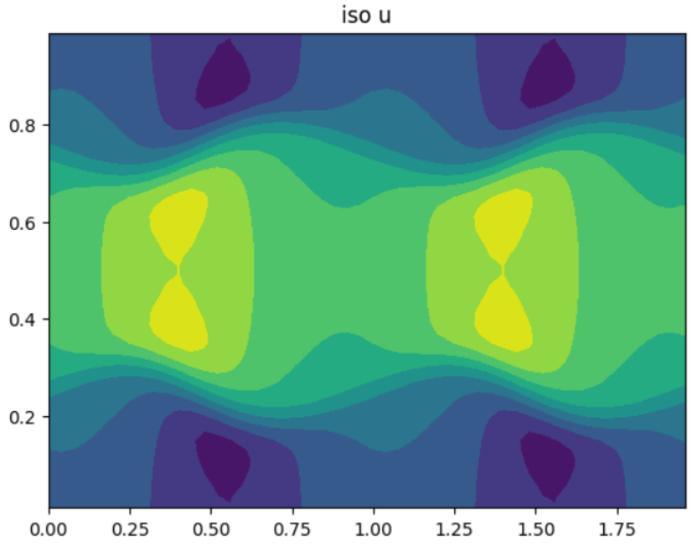
$NLu = u_p(u_r - u_l)/h_x + v_p(u_t - u_b)/h_y$ avec $u_p = u_{ij}$

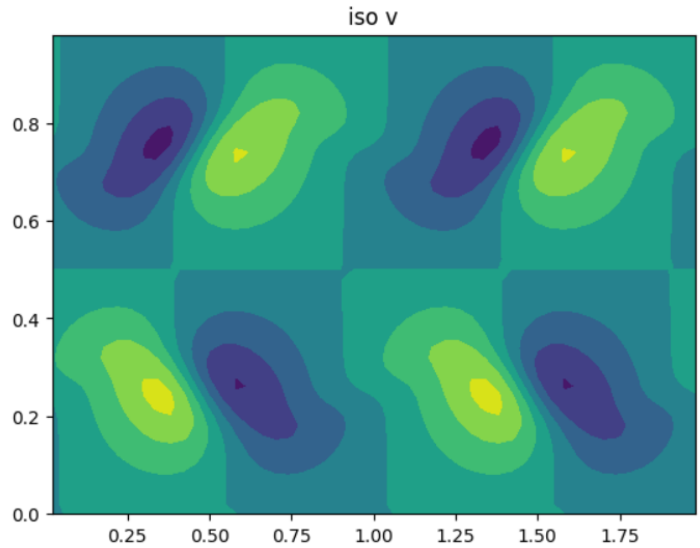
t pour top $\frac{1}{2}(u_{i,j+1} + u_{ij})$
b bottom $1/2(u_{i,j-1}\ u_{ij})$
l left
r right de même mais selon x

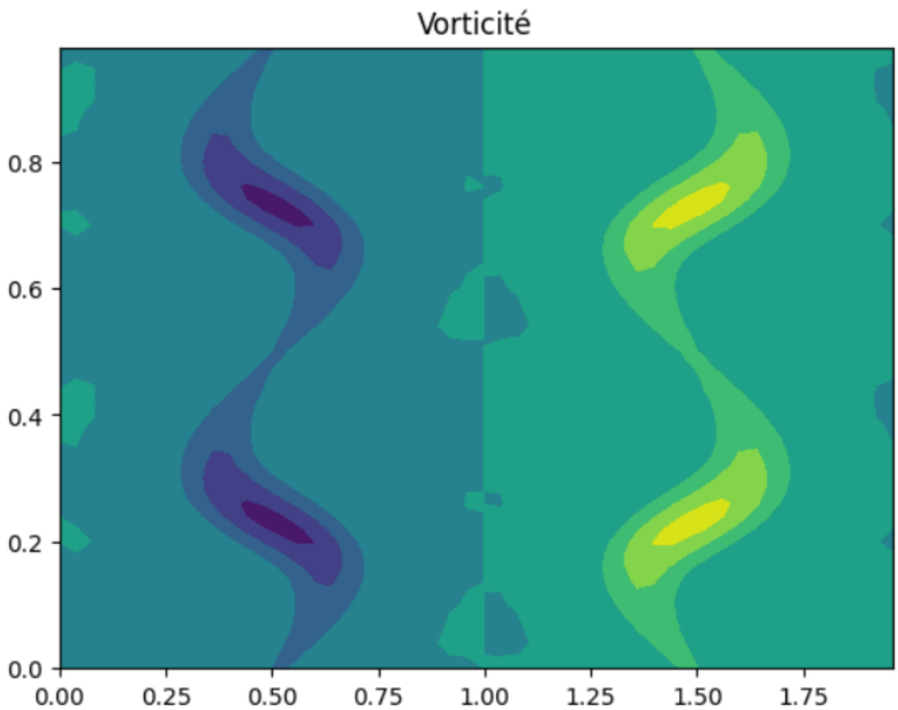
SAMMAN Mohamad

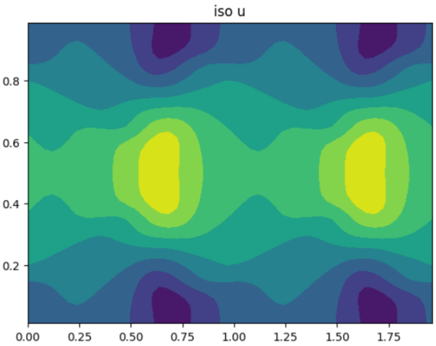
Projet Navier-Stockes 2D

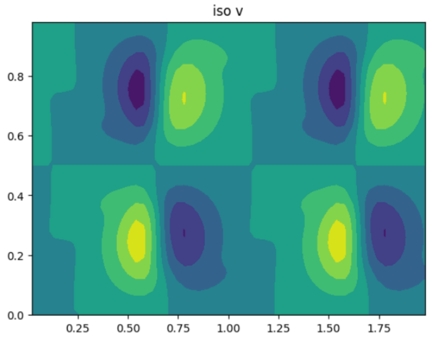
Après execution du code KH-avec-graph.py on obtient les graphiques suivants respectivement en **t = 0.45 , t = 0.85 , t = 1.05 , t = 1.85** (50 points, Lx = 2, Ly=1, nu = 1e-3)

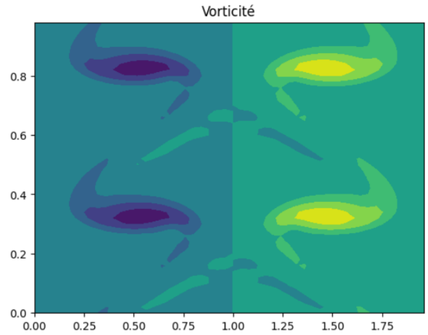


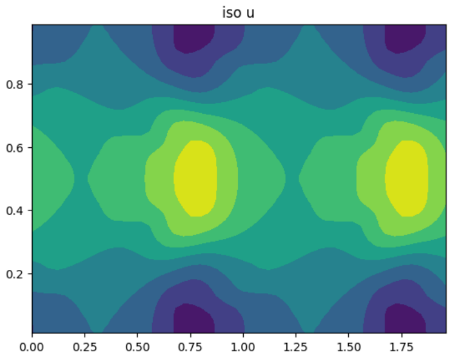


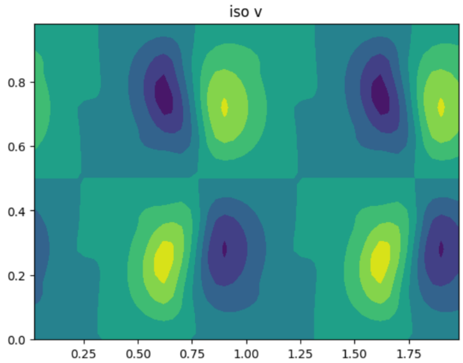


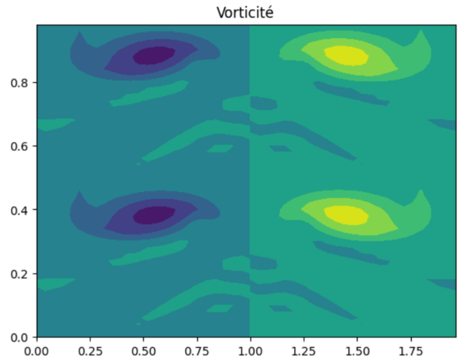

iso u

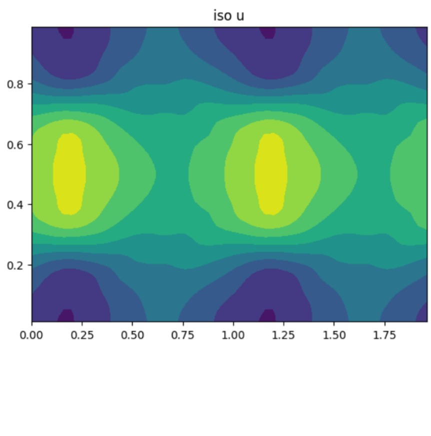

iso v

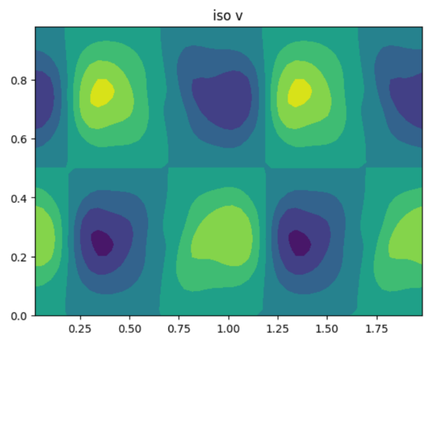

Vorticité

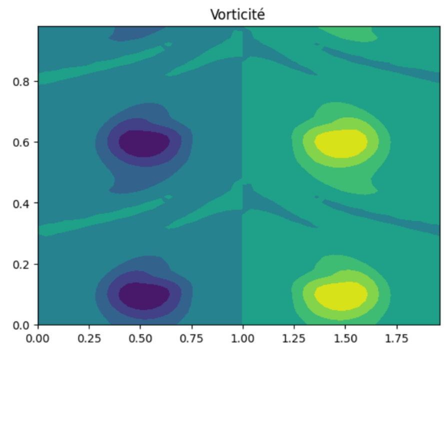

iso u


iso v


Vorticité

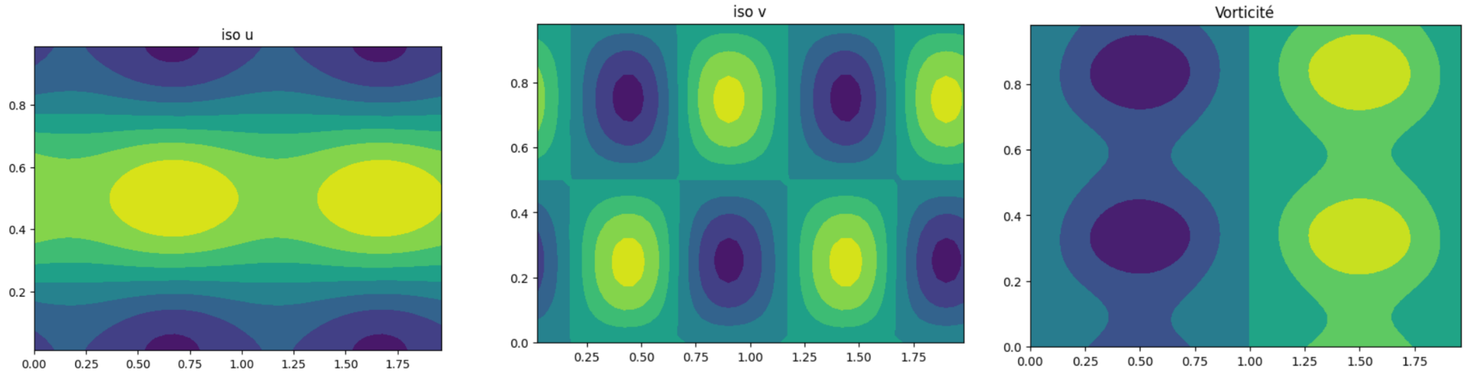

iso u


iso v

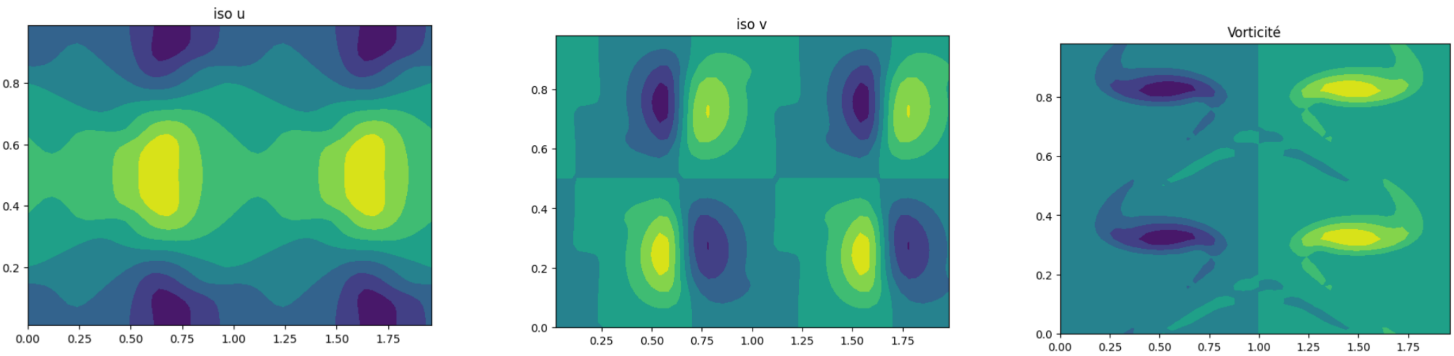

Vorticité

J'ai ensuite essayé de faire varier nu (t = 0.85):

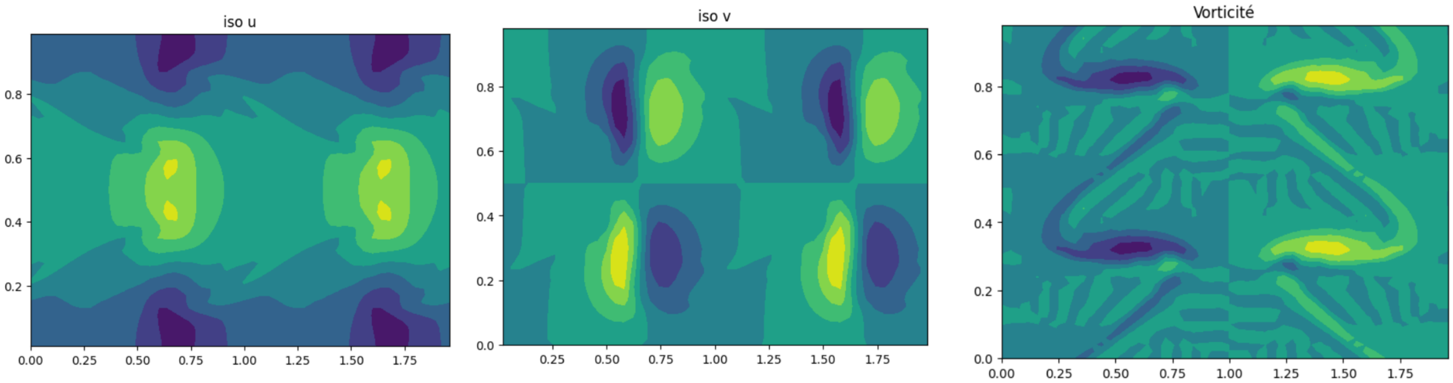
nu = 1e-2



nu = 1e-3



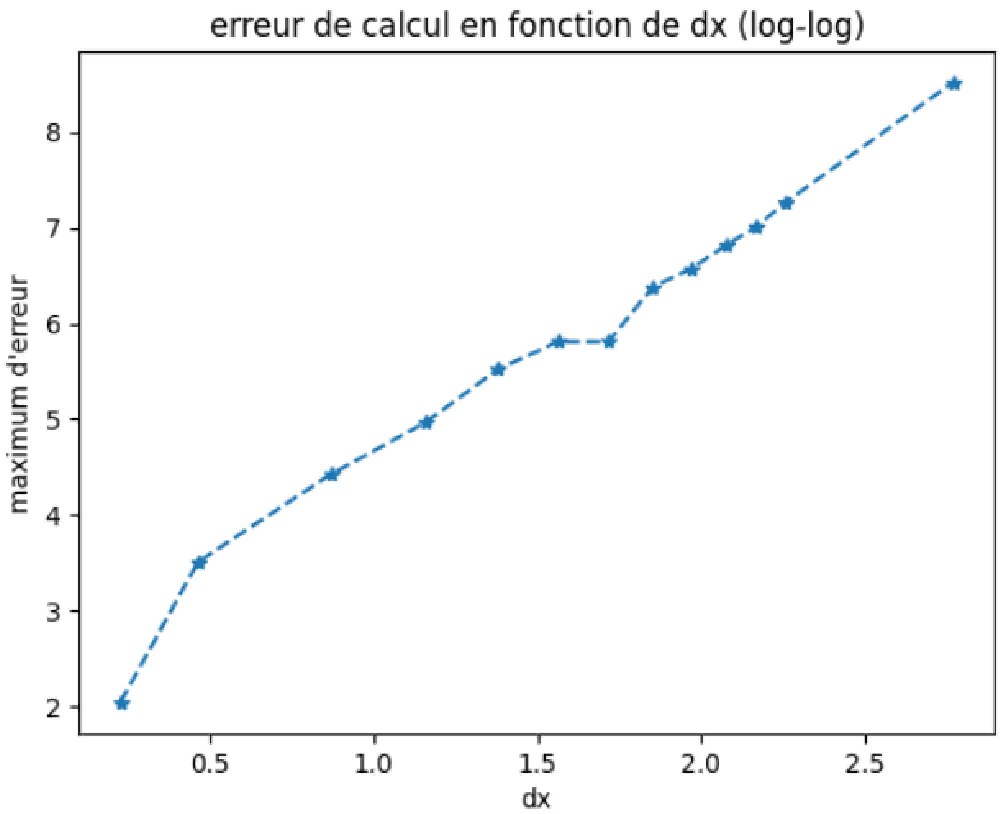
nu = 1e-4



-En ce qui concerne **la précision temporelle**, le Schéma d'Euler explicite est précis à l'ordre 1.

-La précision spatiale est à l'ordre 2 ce que l'on peut vérifier avec le code mms-avec-graph.py (problème stationnaire) en modifiant le nombre de points (cf le graph ci-dessous) .

En effet si on note E1 l'erreur pour un hx donné on a E1 < C * hx². Soit E2 l'erreur pour hx = hx / 2 : E2 < C * (hx/2)² = C * hx² / 4 et donc E2 < E1/4 .



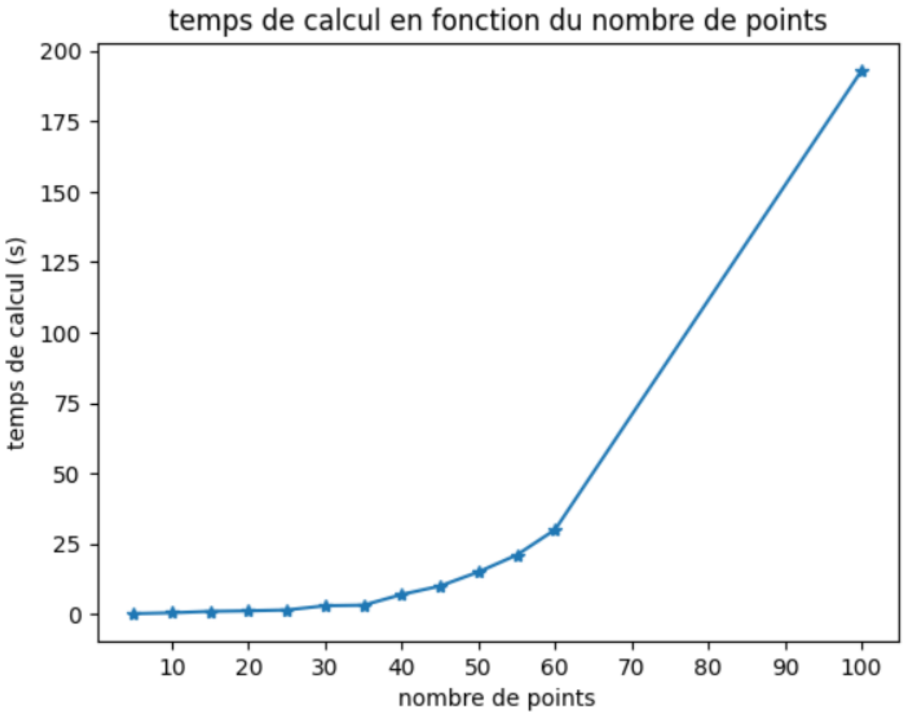
Calcul effectué avec un ryzen 5600g et 16gb de RAM 3200MHz et chronometrés à la main donc la précision n'est pas trés bonne pour un nombre de point petit

points	5	10	15	20	25	30	35	40	45	50	55	60	100
temps (s)	<<1	0.5	1	>1	1.5	3	>3	7	10	15	21	30	193
erreur	0.13	0.03	0.012	0.007	0.004	0.003	0.003	0.0017	0.0014	0.0011	0.0009	0.0007	0.0002

Par exemple 0.03 / 0.007 = 4.285.. > 4
0.007 / 0.0017 = 4.117.. > 4

Une façon de vérifier cela numériquement est d'utiliser numpy.polyfit:
print(np.polyfit(np.abs(np.log(hx)) , np.abs(np.log(e)) , deg=1)[1]) où hx est le pas et e est l'erreur. On obtient alors le coeficient directeur le droite de regression linéaire 2.21… et on confirme l'ordre.

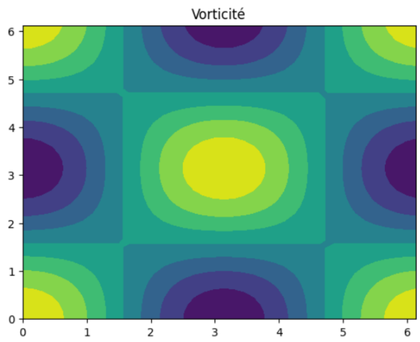
- Le temps de calcul croit de plus en plus quand on rajoute des points (mms-avec-graph.py) .

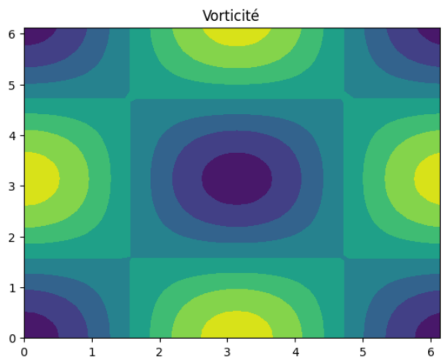


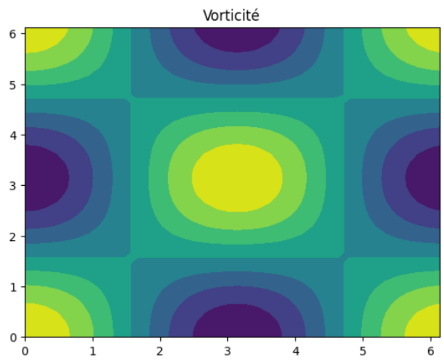
- Pour illustrer la stabilité j'ai testé des CFL differentes et à partir d'un certain temps la solution numérique diverge pour si dt n'est pas assez petit :

dt = 0.25*hx**2/nu , problème instationnaire , stable (CFL)

t = 5.2 t = 10 t = 12.5

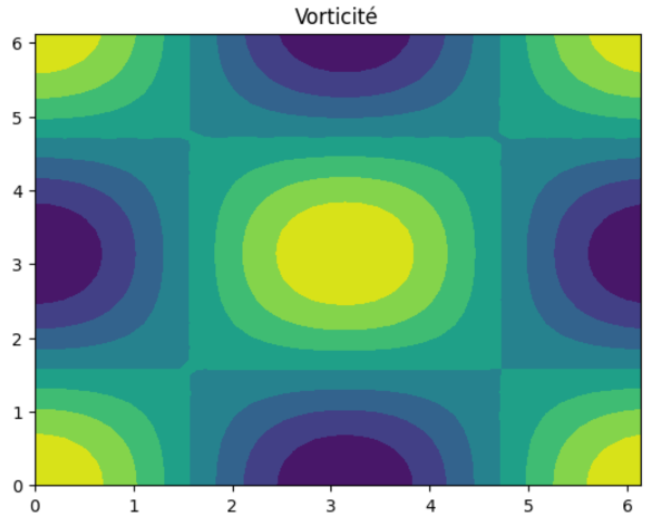




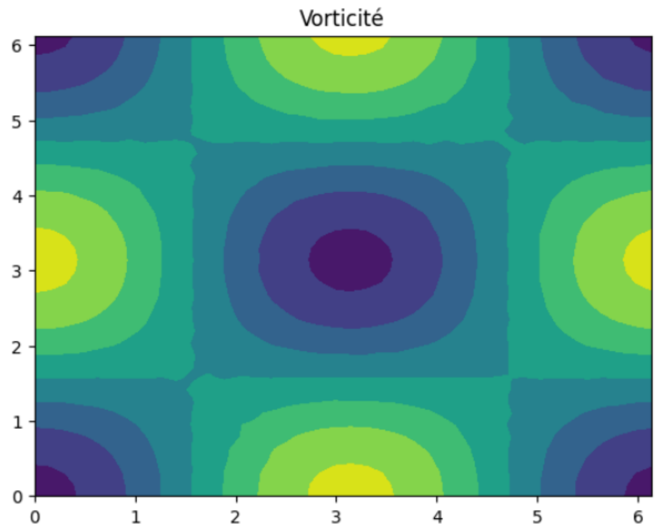


dt = hx/nu , problème instationnaire , instable

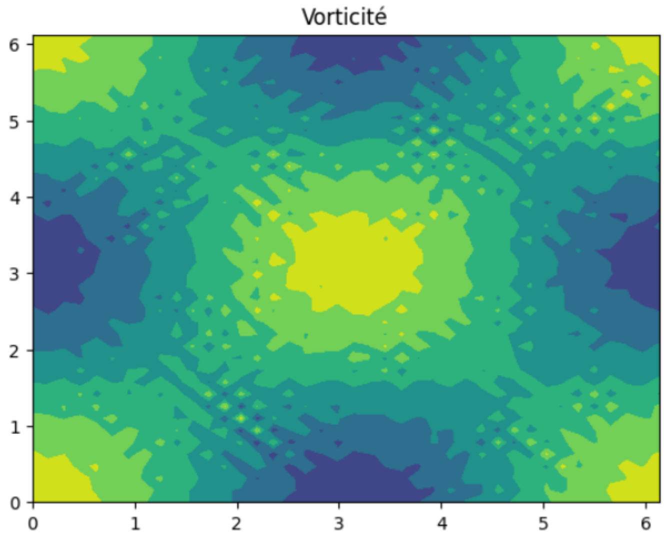
t = 10



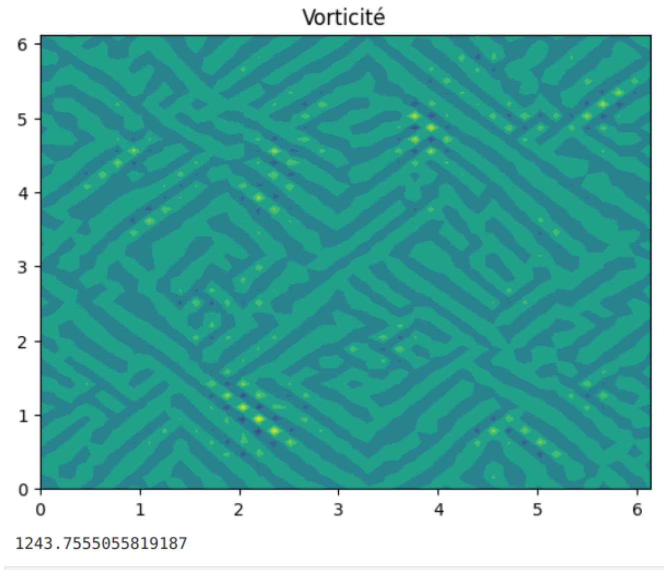
t = 12.5



t = 12.8



t = 17 (erreur trés grande)



Enfin j’ai implémenté le schéma d’Adams-Bashforth (explicite a deux pas et d’ordre 2) pour la discrétisation temporelle dans le fichier **mms_Admas_Bashforth**