Apprentissage de viscosité artificielle pour les schémas LBM

L. Bois[3] E. Franck[1,2]
L. Navoret[1,2] V. Vigon[1,2]

[1] Institut de Recherche Mathématique Avancée, UMR 7501, Université de Strasbourg et CNRS, 7 rue René Descartes, 67000 Strasbourg, France

[2] INRIA Nancy-Grand Est, équipe MACARON, Strasbourg, France

[3] INRIA Nancy-Grand Est, équipe MIMESIS

Séminaire LBM

Sommaire

Méthodes numériques et apprentissage

- Objectif du ML et des méthodes numériques.
  - On considère une fonction inconnue
$$y = f(x), \quad x \in V \subset \mathbb{R}^d, \quad y \in W \subset \mathbb{R}^p$$
  - **Objectif** : Trouver $f_h \in H$ une approximation de f avec H un espace de fonctions.
  - **Difficulté** : on cherche un objet en dimension infinie
- **Solution** : les modèles paramétriques
  - On se donne une fonction $f_\theta(x)$ connue avec θ des paramètres inconnus
  - Le problème devient
$$\text{trouver } \theta, \text{ tel que } \| f_\theta - f \|_H \leq \epsilon$$

- Approche ML : on construit θ en contraignant l'approximation par des données
- On suppose qu'on a des données $\{(x_1, f_1), \ldots, (x_N, f_N)\}$ telles que :
$$f_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$
- Les paramètres θ sont tels que f_θ représente bien f sur les données : on cherche
$$\arg\min_\theta \sum_{i=1}^{N} d(u_i, u_\theta(x_i))$$

- Méthodes numériques : on construit θ en contraignant par une équation physique
- L'objectif d'une méthode numérique :
$$L(u(x)) = f(x) \Longrightarrow A(\theta) = b(\theta)$$
avec L un opérateur différentiel ou intégrale et A, b un système linéaire.
- D'une méthode à l'autre, le modèle paramétrique et la forme de l'équation change.

Révolution de l'apprentissage profond

- **Modèles classiques en ML** : modèle polynomiaux, à noyau etc

$$f_\theta(x) = \sum_{i=1}^{n} \theta_i P_i(x), \quad f_\theta(x) = \sum_{i=1}^{n} \theta_i K(x, x_i)$$

- **Révolution du deep learning** : Combinaison entre l'utilisation massive de GPU, l'énorme quantité de données et un autre type de modèle.
- Les modèles deviennent nonlinéaires par rapport aux paramètres.
- Conséquences :
  - permet d'attaquer des problèmes en grandes dimensions
  - on doit utiliser des méthodes de gradients stochastiques élaborées car les problèmes deviennent non convexes.
  - On projette sur des variétés de dimension finies plutôt que sur des espaces vecrtoriels.
- Exemple simple de modèles nonlinéaires :

$$f_{\alpha,\mu,\Sigma}(x) = \sum_{i=1}^{N} \alpha_i \mathcal{N}(x; \mu_i, \Sigma_i)$$

avec $\mathcal{N}$ une Gaussienne.

Réseaux de neurones I

- Réseau de neurones généraux (NN).
  - **Couche** : une couche est une fonction $L_l(\mathbf{x}_l) : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^{d_l+1}$ tel que :

    $$L_l(\mathbf{x}_l) = \sigma(A_l \mathbf{x}_l + \mathbf{b}_l),$$

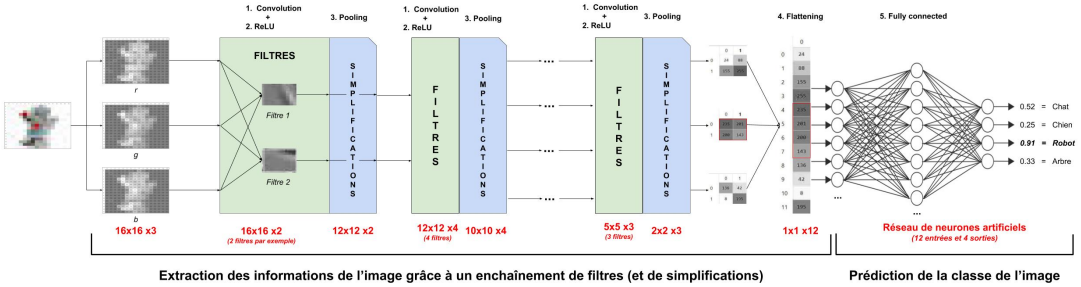
    avec $A_l \in \mathbb{R}^{d_l+1,d_l}$, $\mathbf{b}_l \in \mathbb{R}^{d_l+1}$ les paramètres optimisable t $\sigma()$ une fonction locale appliquée composante par composante.
  - **Réseaux de neurones** : il s'agit d'une fonction paramétrique obtenue par composition de plusieurs couches avec des dimensions d'entrée et de sortie qui peuvent être différentes :

    $$f_\theta(\mathbf{x}) = L_n \circ \ldots . \circ L_1(\mathbf{x})$$

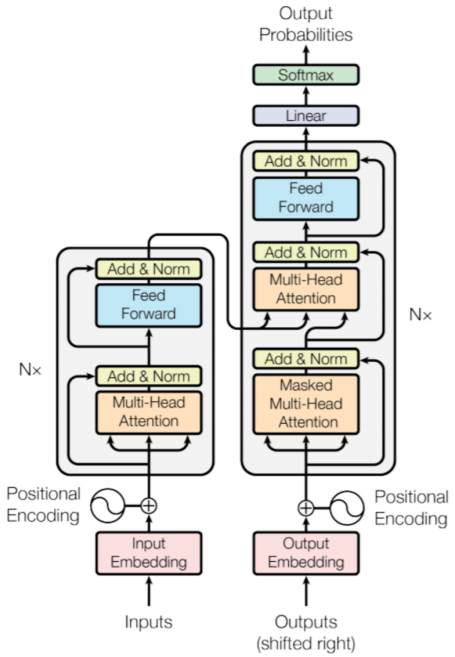
    avec θ l'ensemble des paramètres optimisables qui est constitué de l'ensemble $A_{l,l+1}$ et des $\mathbf{b}_l$.
- Les fonctions d'activation peuvent aussi dépendre de paramètres optimisables.
- Si les matrice A sont denses on parle de réseaux .
- On remplace des modèles qui somment des fonctions simples par des modèles qui composent des fonctions simples.

Réseaux de neurones II

- **Réseaux pour les images** : réseaux convolutifs qui utilisent des matrices creuse de Toeplitz. Cela revient a enchainer les filtres locaux sur une images.

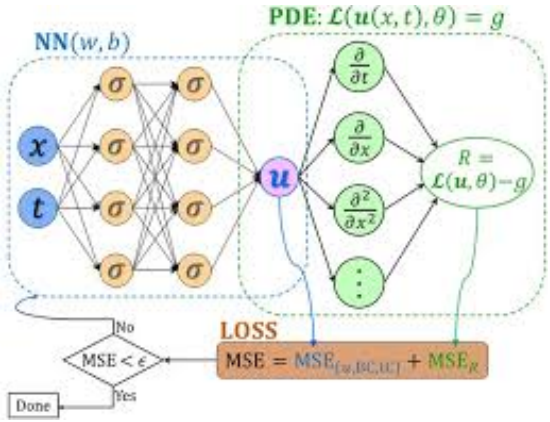


- Ils encodent : l'invariance par translation, la stabilité aux petites déforrmations et une certaine analyse multi-échelle (S. Malat Collègue de France)
- **Réseaux pour les textes** : transformers qui construire des lois de probabilités de correlation entre toutes les entrées (méga, méga simplifié).

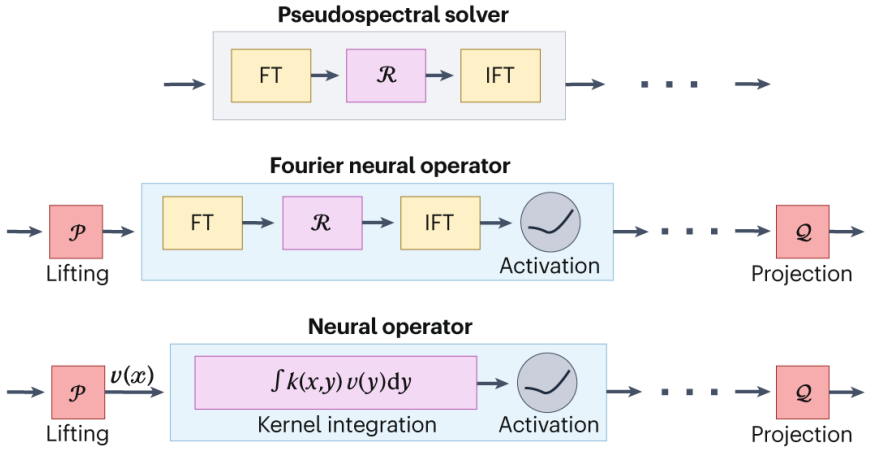


Réseaux de neurones III

- **Réseau pour les données sur graphes/maillages** : réseaux convolutifs sur graphe. Utilise un paradigme de "diffusion de l'information" sur le graphe.
- Pour les EDP :
  - pour représenter une solution on peut remplacer une approximation EF/VF par un réseau totalement connecté qui prend en entrée les variables physiques x, t etc. On parle de PINNs, neural Galerkin etc

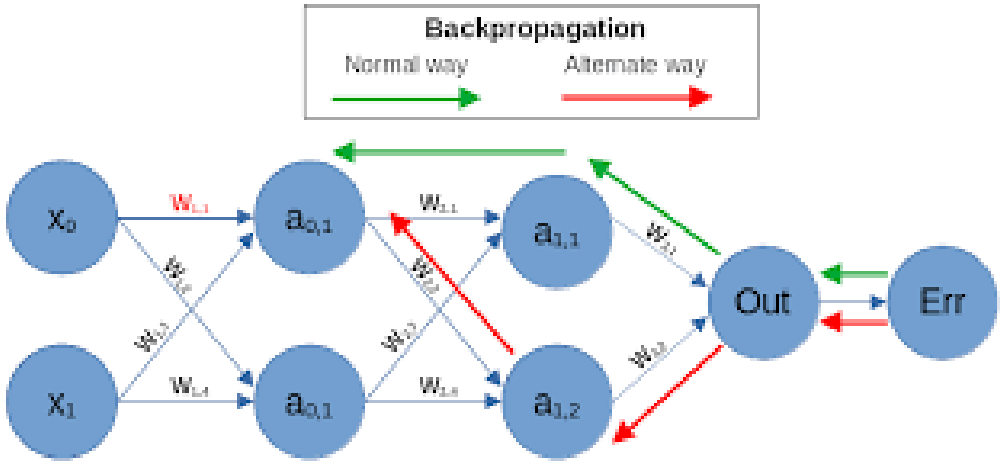


  - pour transformer une fonction en une autre on utilise des **opérateurs neuraux**. On enchaine des opérateurs intégraux linéaires et des fonctions activation et on paramètre les opérateurs afin de rendre l'apprentissage indépendant de la discrétisation.



Apprentissage et différentiation automatique

- L'ensemble des réseaux de neurones sont différentiables.
- On optimise les poids par des méthodes de gradients type Adam.
- Comment est calculé le gradient ?
- On utilise la rétro-propagation pour calculer le gradient d'une composition de fonction (règle de la chaine) :



- De façon général il existe plusieurs frameworks de développement "différentiable" qui permettent d'écrire des codes ou toutes les fonctions sont différentiables. Utile pour apprendre des bouts de schéma ou faire du contrôle sans calcul d'adjoint.
- Exemple avec Taichi :

$$\frac{d}{dt} S(t) = -\beta SI$$
$$\frac{d}{dt} I(t) = \beta SI - \gamma I$$
$$\frac{d}{dt} R(t) = \gamma I$$

- On cherche a retrouver β à partir d'une solution de référence.

Sommaire

Schéma LBM I

- On considère des équations hyperboliques.
- On introduit le cas scalaire, mais tout ce qui suit fonctionne de la même façon pour le cas vectoriel.

$$\partial \rho + \nabla \cdot (\boldsymbol{f}(\rho)) = 0.$$

- Schéma LBM $[D2Q4]$:
  1. On introduit 4 variables $f_1(t,x), \ldots, f_4(t,x)$
  2. On introduit 4 vitesses :

$$\left[\boldsymbol{v}_1 = \begin{pmatrix} \lambda \\ 0 \end{pmatrix}, \boldsymbol{v}_2 = \begin{pmatrix} -\lambda \\ 0 \end{pmatrix}, \boldsymbol{v}_3 = \begin{pmatrix} 0 \\ -\lambda \end{pmatrix}, \boldsymbol{v}_4 = \begin{pmatrix} 0 \\ \lambda \end{pmatrix} \right]$$

  3. On résout à chaque pas de temps :

$$f_i^*(t_n, \boldsymbol{x}_j) = f_i(t_n, \boldsymbol{x}_j - \Delta t \boldsymbol{v}_i), \quad \forall 1 \leq i \leq 4$$

  4. On applique une étape de collision

$$f_i(t_{n+1}, \boldsymbol{x}_j) = f_i^*(\boldsymbol{x}_j) + (f_i^{eq}(\rho(\boldsymbol{x}_j)) - f_i^*(\boldsymbol{x}_j))$$

  avec $\rho(\boldsymbol{x}_j) = \sum_{i=1}^{4} f_i^*(\boldsymbol{x}_j)$.

- Schéma LBM $[D2Q4]^n$ permet de résoudre :

$$\partial_t \mathbf{U} + \nabla \cdot (\boldsymbol{f}(\mathbf{U})) = 0.$$

- On écrit un schéma D2Q4 par variable. Le couplage entre les variables physique à lieu uniquement par l'équilibre.

Schéma LBM II

- Une variante consiste à ne pas relaxer les variables toutes à la même vitesses.
- On parle dans le cas LBM standard (pas vectoriel) de méthode MRT.
- Extension au cas vectoriel. L'étape de collision devient :

$$f_i(t_{n+1}, \boldsymbol{x}_j) = f_i^*(\boldsymbol{x}_j) + W(\rho(\boldsymbol{x}_j))(f_i^{eq}(\rho(\boldsymbol{x}_j)) - f_i^*(\boldsymbol{x}_j))$$

avec $\rho(\boldsymbol{x}_j) = \sum_{i=1}^{4} f_i^*(\boldsymbol{x}_j)$, M une matrice de passage et

$$W(\rho) = M^{-1} \left(\begin{array}{c|cc|c} 1 & 0 & 0 & 0 \\ \hline 0 & & & 0 \\ 0 & & \Omega(\rho) & 0 \\ \hline 0 & 0 & 0 & \tau \end{array} \right) M.$$

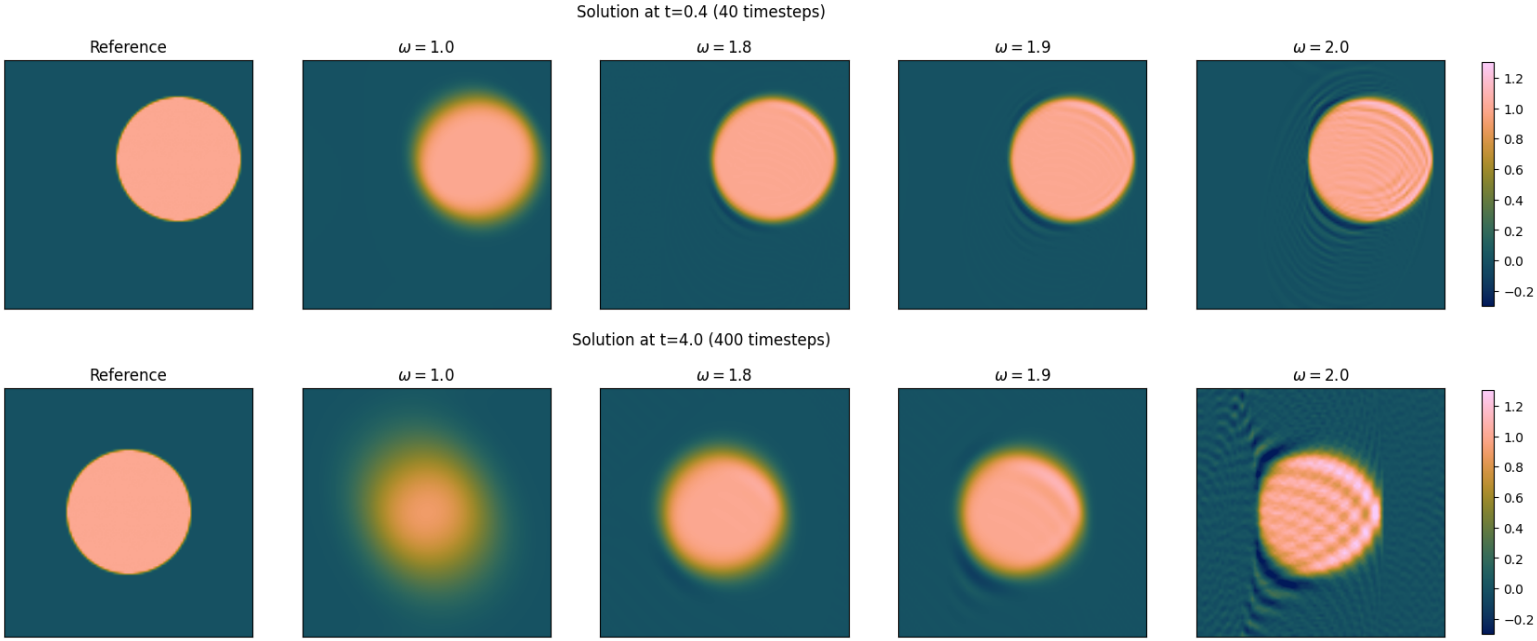
- Dans les approches de type MRT on prend en général Ω constant. Ici on se propose de le rendre variable.
- Par contre on se propose de prendre $\tau = 1$.
- Pour le schéma LBM $[D2Q4]^n$: la matrice $\Omega(\mathbf{U})$ couple les variables physiques.

Schéma LBM

- Erreur des schémas LBM :

$$\partial\rho + \nabla\cdot(\boldsymbol{f}(\rho)) = \Delta t \nabla\cdot\left(\underbrace{\left(\Omega^{-1}(\rho) - \frac{1}{2}I\right)\left(\frac{\lambda^2}{2}I - \boldsymbol{f}'(\rho)\otimes\boldsymbol{f}'(\rho)\right)}_{D(\rho)}\nabla\rho\right) + O(\Delta t^2)$$

- $\Omega = I_d$ donne le schéma d'ordre 1 classique. $\Omega = 2I_d$ donne le schéma d'ordre 2.
- Stabilité approchée : $D(\rho) \geq 0$
- Le schéma d'ordre 2 oscille notamment autour des discontinuités, ce qui génère des problèmes de stabilité.
- Exemple ci-dessous avec $\boldsymbol{f}(\rho) = \begin{pmatrix} a \\ b \end{pmatrix}\rho$ avec 40 et 400 étapes en temps :



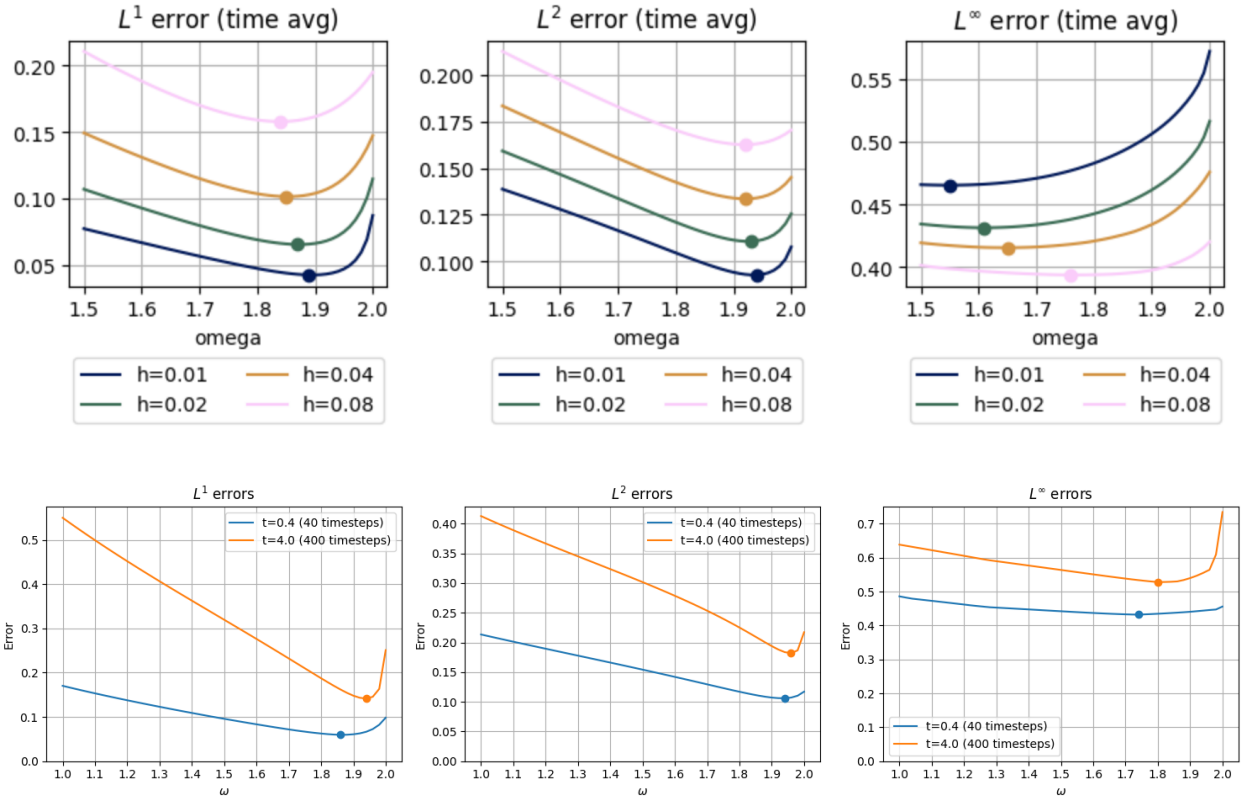
Etude du choix de Ω I

- Le choix de Ω n'est pas simple.
- Exemple : on résout

$$\partial \rho + \nabla \cdot (\mathbf{a}\rho) = 0$$

with $\mathbf{a} = (1, 0.5)^t$

- Advection d'un disque.



- Erreur en fonction de $\Omega = \omega I_d$.
- Il y a compétition :
  - entre l'erreur de diffusion et l'erreur de dispersion $O(\Delta t^2 \partial_{xxx} \dots)$,
  - entre l'erreur de consistance et la stabilité.

Idée :

- **But** : apprendre le meilleur Ω en apprenant sur des simulations en temps.
- Question 1 ? quel réseau (architecture, nombre de paramètres, données d'entrées etc)
- Plusieurs type de données d'entrées :

$$\Omega(\rho(x_i)) = f_\theta(\rho(x_i))I_d, \text{ ou } f_\theta(\rho(x_{i-1}), \rho(x_i), \rho(x_{i+1}))I_d, \text{ ou } = A_\theta(\rho(x_i))$$

- Le dernier cas génère une diffusion anisotrope en espace.
- Pour les systèmes on peut aussi avoir une matrice qui couple les variables et génère une diffusion qui change selon les variables.
- Question 2 ? quel type d'apprentissage et de données.

Sommaire

Contrôle optimal

- Classiquement en apprentissage on cherche $y = f_\theta(x)$ en connaissant des exemples (x_i, y_i).
- Pour faire cela ici il faudrait :
  - faire des centaines de simulations pour différent ωI_d
  - déterminer le meilleur (celui qui produit moins d'erreur) associé à chaque donnée d'entrrée (solution en temps)
  - et l'apprendre
- Stratégie couteuse et pas forcement très efficace. Ne tient pas compte de la stabilité.
- On propose une approche contrôle optimal :

$$\min_\theta \sum_{i=1}^{m} \sum_{n=1}^{N} \| \rho_n^i - \rho_{ref,n}^i \|^2$$

  sous la contrainte que ρ_n^i soit solution du schéma LBM avec $\rho_0^i = \rho_i(x)$.
- Les conditions initiales sont générées aléatoirement.
- Il faut donc calculer l'adjoint du schéma ce qui peut être compliqué et le combiner avec le gradient du réseau.
- **Autre solution** : utiliser les frameworks différentiables pour calculer le gradient de la solution directement.
- En pratique pour éviter un gradient trop lourd on optimise sur des sous trajectoires de time raisonnable

Algorithme final

Algorithm 1: Training algorithm

Set the total number of timesteps desired N
Compute reference solutions $\{\mathbf{u}_{\text{ref},i}\}_{i=1}^{m}$ on all timesteps t^0, ..., t^N
Build the neural network π_θ and initialize its parameters θ
Set the number of timesteps for the training n
while *True* **do**
    for $k \in \{1, ..., \textit{batch size}\}$ **do**
        for $i \in \{1, ..., m\}$ **do**
            Draw a random starting time $t^i \in \{t^0, ..., t^{N-n}\}$
            Compute the numerical solutions $\mathbf{u}_{\theta,i}^{[t^i, t^{i+n}]}$
            Compute the error $E_i(\theta) = \|\mathbf{u}_{\theta,i}^{[t^i, t^{i+n}]} - \mathbf{u}_{\text{ref},i}^{[t^i, t^{i+n}]}\|$
        end
        Compute the approximated loss for this sample $\mathcal{L}_k(\theta) = \sum_{i=1}^{m} E_i(\theta)$
    end
    Compute the approximated loss for this batch $\mathcal{L}(\theta) = \sum_{k=1}^{\text{batch size}} \mathcal{L}_k(\theta)$
    Update the parameters θ with $\nabla\mathcal{L}(\theta)$
end

Réseau

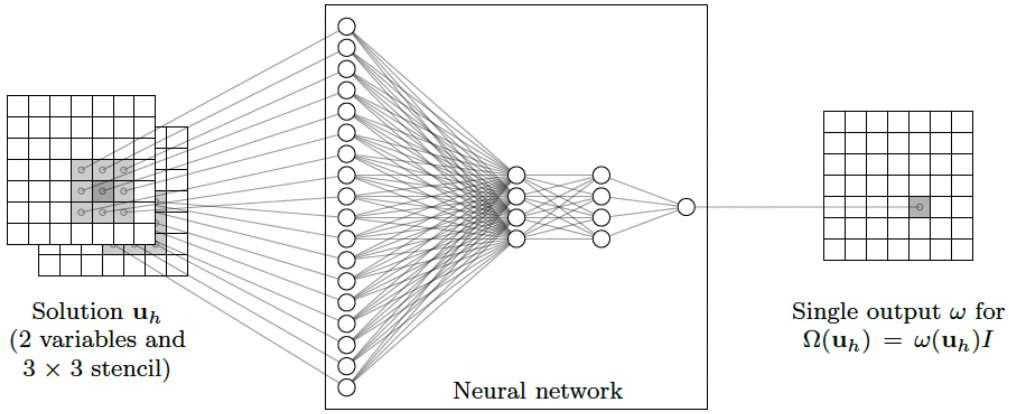


Figure 1: Example of neural network, for a system of 2 equations solved using a scalar relaxation matrix.

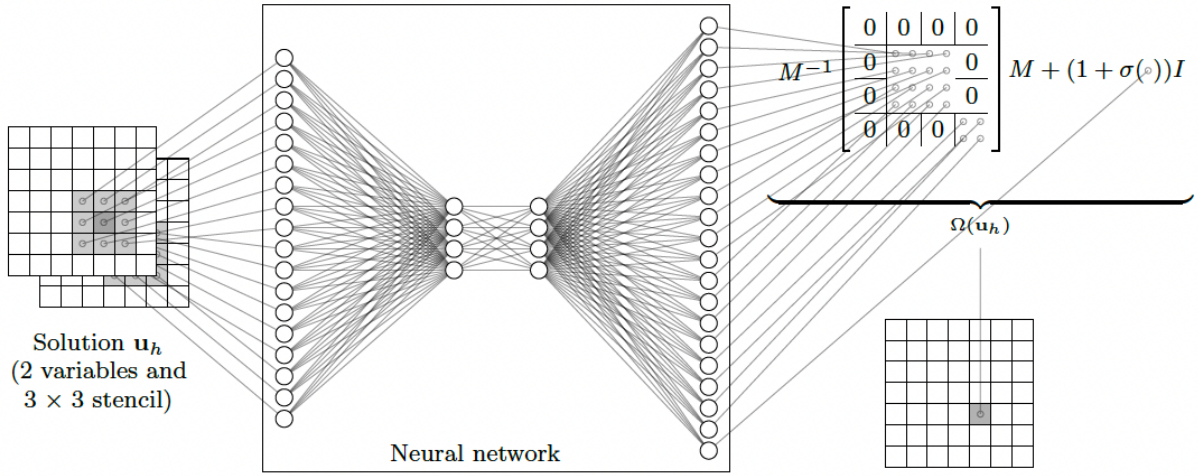
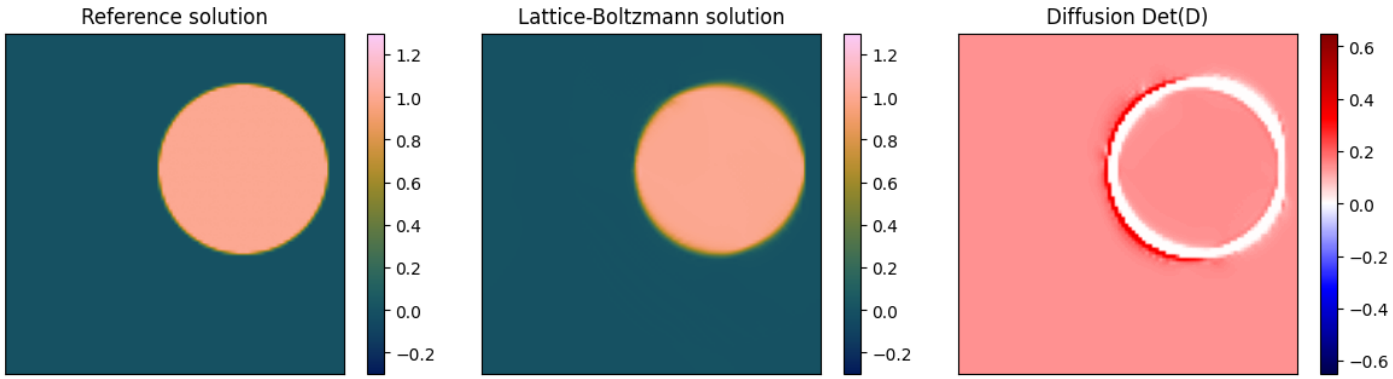


Figure 2: Example of neural network, for a system of 2 equations solved using a scalar relaxation matrix.

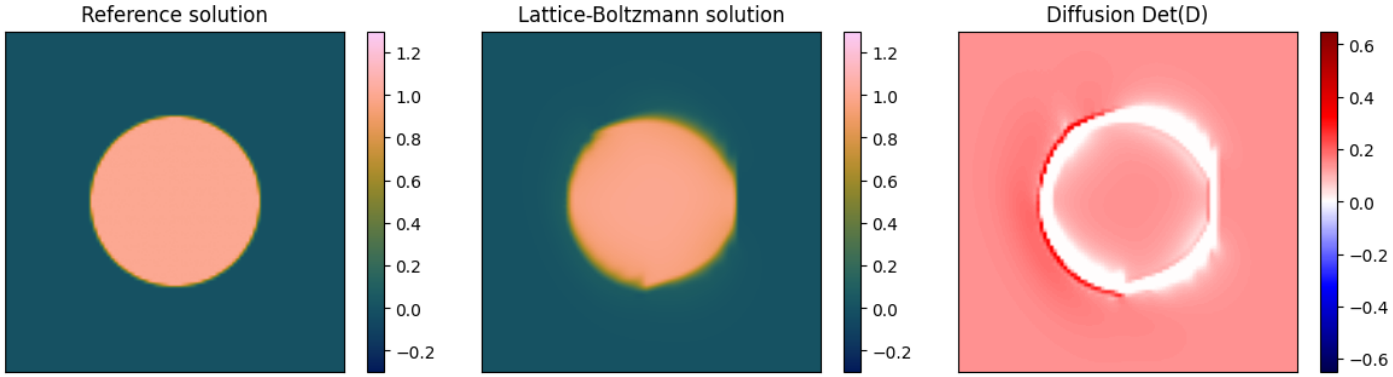
Sommaire

Équation de transport I : une condition initiale

- Équation de transport. On apprend sur la condition initiale considérée.
- On apprend la matrice $\Omega(\rho) = I_d \omega(\rho)$
- Calcul sur 40 pas de temps

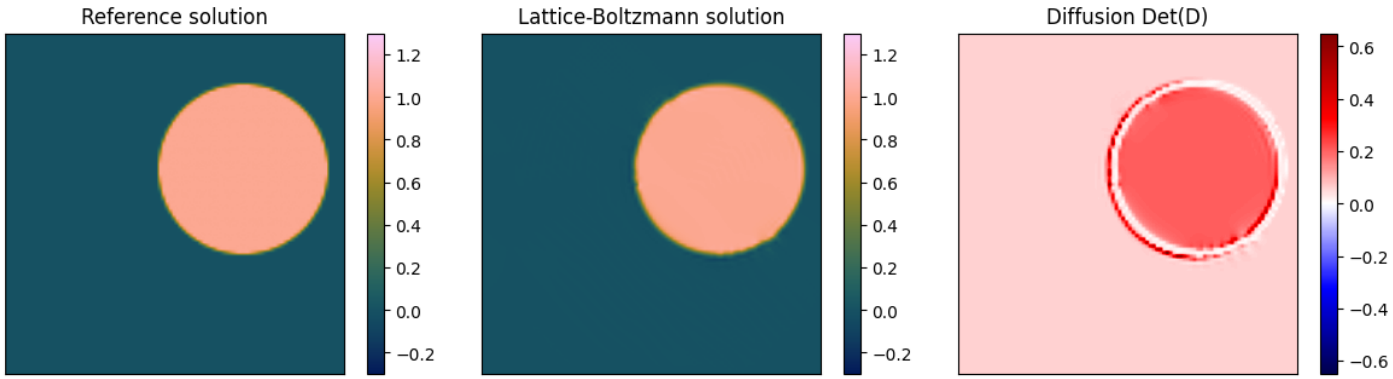


- Calcul sur 400 pas de temps

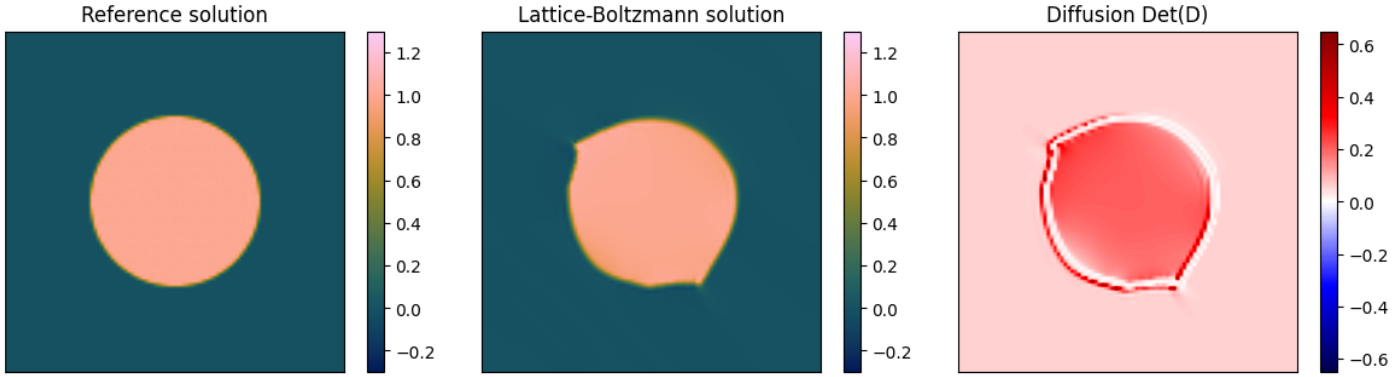


Équation de transport II : une condition initiale

- Équation de transport. On apprend sur la condition initiale considérée.
- On apprend la matricer Matrice $\Omega(\rho)$
- Calcul sur 40 pas de temps



- Calcul sur 400 pas de temps



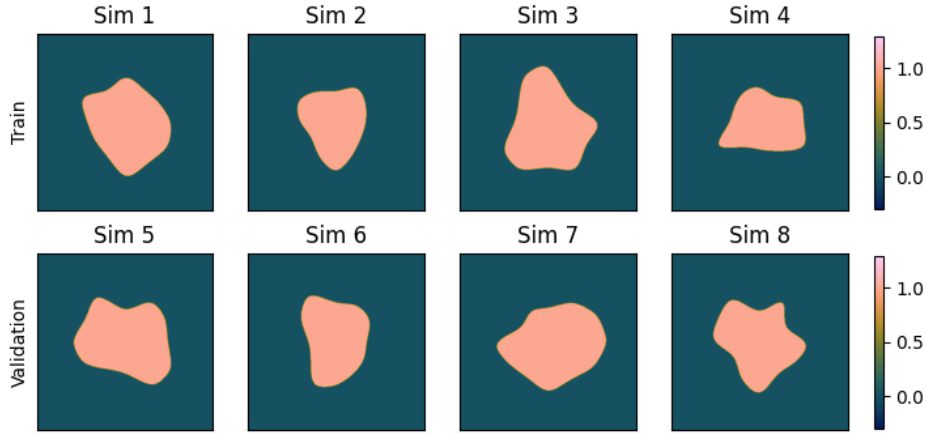
Équation de transport : Erreur

Model	L^1	L^2	L^∞
Constant $= 1.9$	6.08e-02	1.07e-01	4.41e-01
1×1 stencil, $m = 40$ timesteps	4.06e-02	8.63e-02	4.77e-01
3×3 stencil, $m = 40$ timesteps	3.19e-02	7.08e-02	3.41e-01
5×5 stencil, $m = 40$ timesteps	3.20e-02	7.14e-02	3.37e-01
7×7 stencil, $m = 40$ timesteps	3.28e-02	7.09e-02	3.38e-01
3×3 stencil, $m = 5$ timesteps	4.32e-02	7.83e-02	3.89e-01
3×3 stencil, $m = 20$ timesteps	3.34e-02	6.97e-02	3.19e-01
3×3 stencil, $m = 40$ timesteps	3.19e-02	7.08e-02	3.41e-01
3×3 stencil, $m = 80$ timesteps	3.14e-02	7.39e-02	3.80e-01

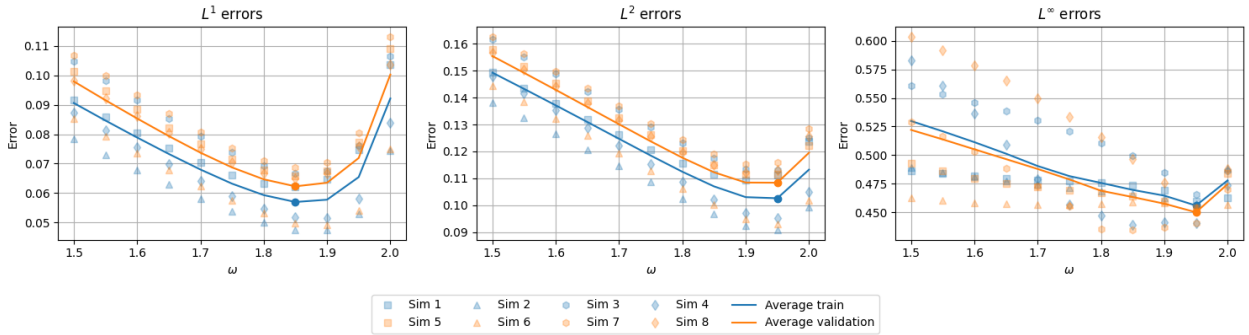
Table – Errors at $t = 0.4$ (40 timesteps) using different scalar relaxations.

Équation de transport I : plusieurs conditions initiales

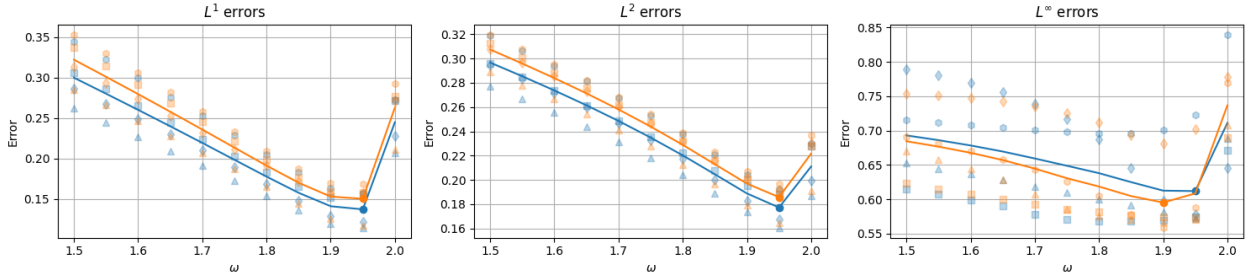
- Exemples :



- Erreur des schémas classiques sur 40 pas de temps

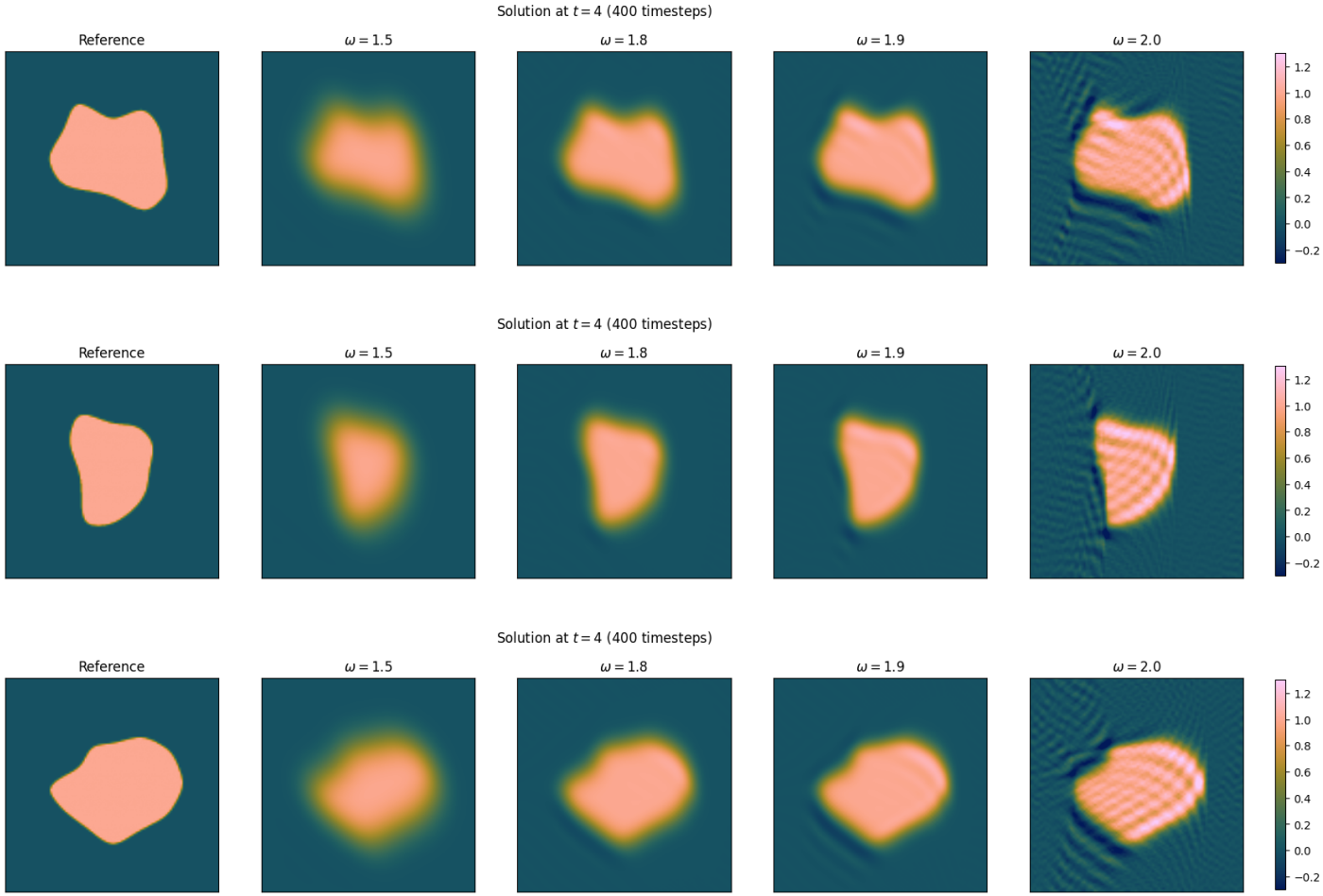


- Erreur des schémas classiques sur 400 pas de temps



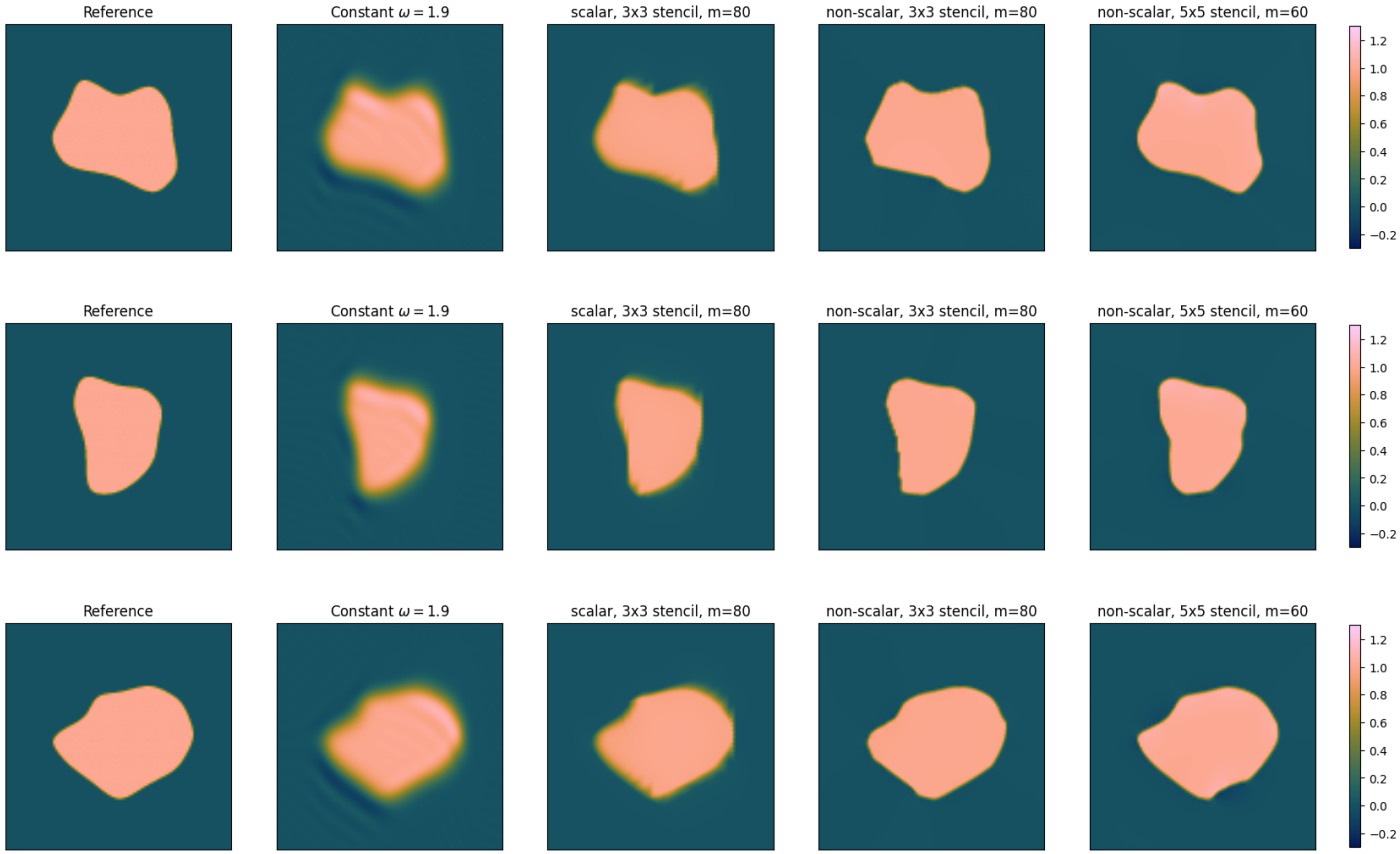
Équation de transport II : plusieurs conditions initiales

- Exemples avec le schéma classique :



Équation de transport III : plusieurs conditions initiales

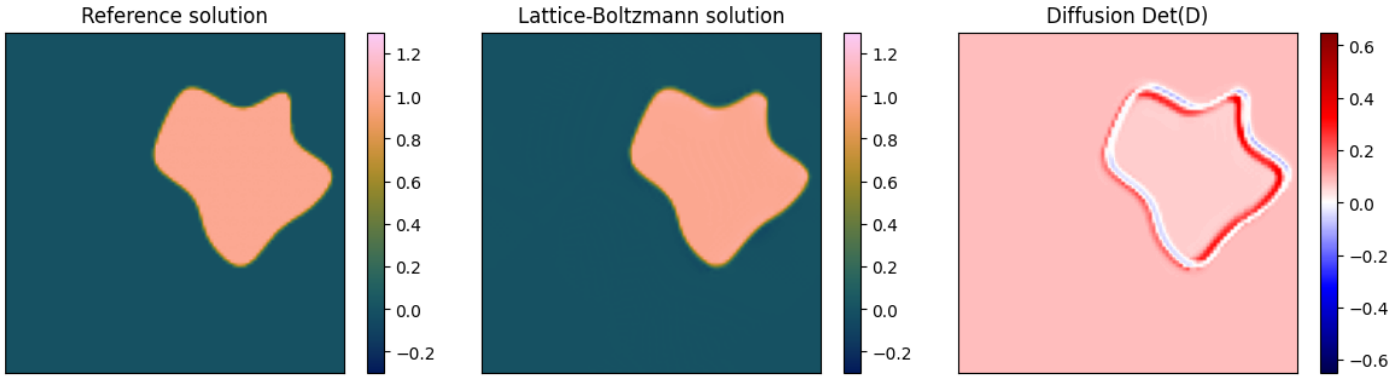
- Exemples avec des viscosités apprises :



- la forme peut être légèrement déformée par le schéma. Manque de contrainte sur la viscosité ?

Équation de transport IV : erreur

- Exemples avec la viscosité matricielle en temps cours :

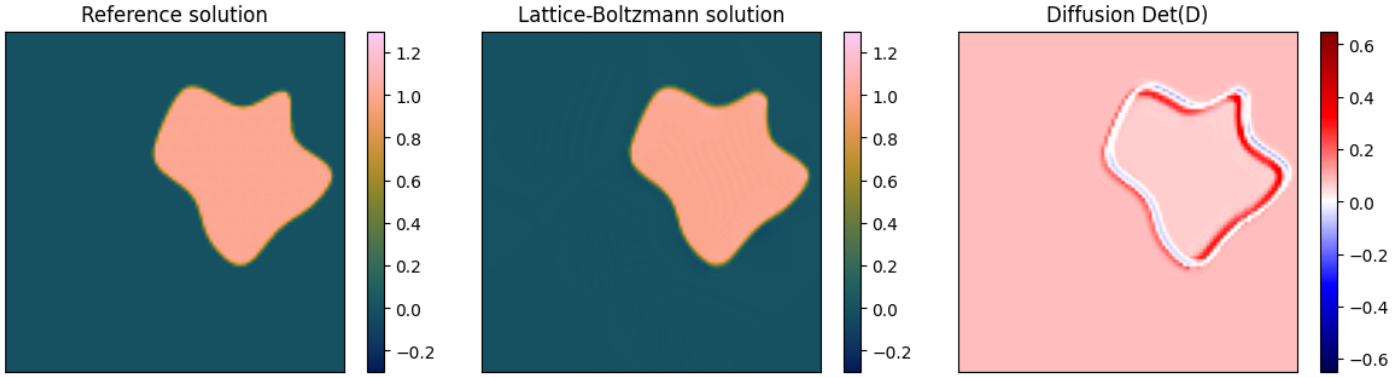


- Anti-diffusion ?
- Erreur :
- Cas scalaire

Model	L^1		L^2		L^∞	
	$t = 0.4$	$t = 4.0$	$t = 0.4$	$t = 4.0$	$t = 0.4$	$t = 4.0$
Constant $= 1.9$	6.35e-02	1.53e-01	1.08e-01	1.97e-01	4.58e-01	5.95e-01
1×1, 40	4.38e-02	1.19e-01	8.93e-02	1.61e-01	5.07e-01	8.63e-01
3×3, 40	3.33e-02	1.01e-01	7.23e-02	1.33e-01	4.07e-01	6.61e-01
5×5, 40	3.35e-02	1.08e-01	7.19e-02	1.39e-01	3.92e-01	7.01e-01
7×7, 40	3.37e-02	1.11e-01	7.09e-02	1.44e-01	3.77e-01	7.27e-01
3×3, 20	3.64e-02	1.30e-01	7.11e-02	1.44e-01	4.05e-01	7.88e-01
3×3, 40	3.33e-02	1.01e-01	7.23e-02	1.33e-01	4.07e-01	6.61e-01
3×3, 60	3.32e-02	9.72e-02	7.38e-02	1.33e-01	4.11e-01	6.48e-01
3×3, 80	3.29e-02	8.85e-02	7.52e-02	1.33e-01	4.32e-01	6.45e-01
3×3, 100	3.32e-02	8.95e-02	7.58e-02	1.34e-01	4.30e-01	6.41e-01

Équation de transport IV : erreur

- Exemples avec la viscosité matricielle en temps cours :

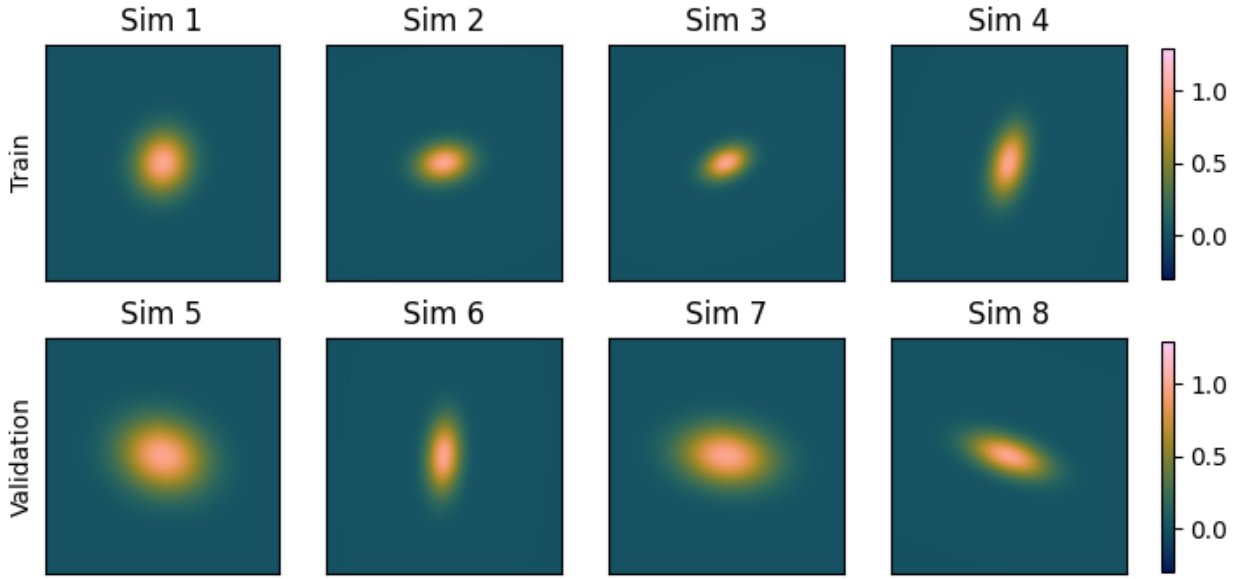


- Anti-diffusion ?
- Erreur :
- Cas non scalaire

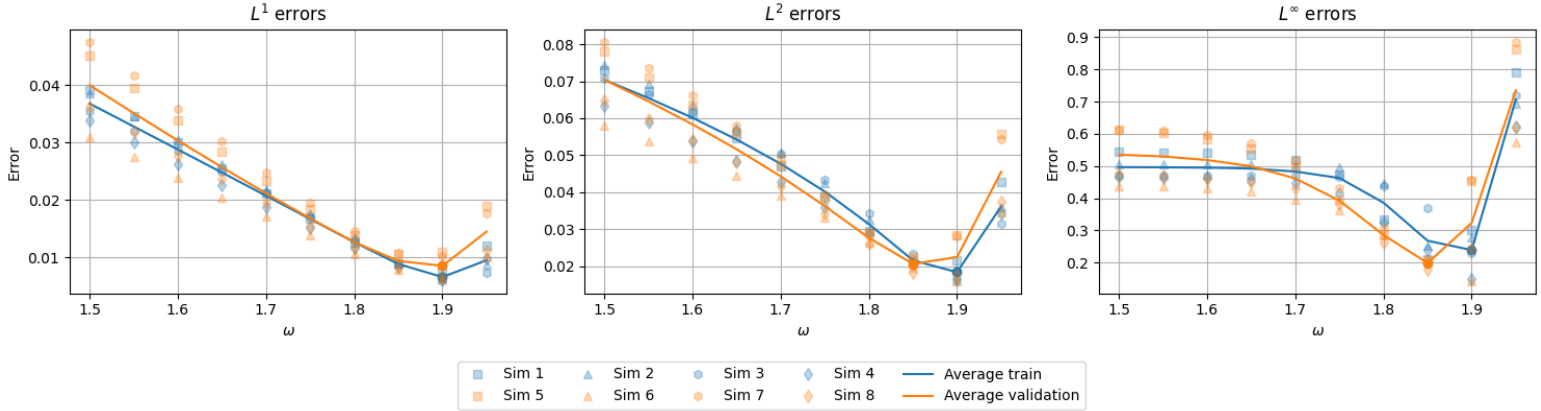
Model	L^1		L^2		L^∞	
	$t = 0.4$	$t = 4.0$	$t = 0.4$	$t = 4.0$	$t = 0.4$	$t = 4.0$
Constant $= 1.9$	6.35e-02	1.53e-01	1.08e-01	1.97e-01	4.58e-01	5.95e-01
3×3, 40	2.17e-02	nan	5.29e-02	nan	4.76e-01	nan
5×5, 40	4.78e-02	1.74e-01	9.49e-02	2.83e-01	4.85e-01	1.73e+00
7×7, 40	5.33e-02	1.46e-01	1.01e-01	1.94e-01	4.54e-01	6.01e-01
3×3, 60	1.96e-02	5.44e-02	4.94e-02	1.28e-01	4.91e-01	1.06e+00
5×5, 60	1.21e-02	3.96e-02	3.19e-02	9.39e-02	3.21e-01	9.80e-01
7×7, 60	2.16e-02	6.62e-02	5.26e-02	1.50e-01	5.40e-01	1.36e+00
3×3, 80	1.29e-02	3.00e-02	3.53e-02	9.05e-02	3.68e-01	9.24e-01
5×5, 80	2.13e-02	5.95e-02	5.25e-02	1.35e-01	5.37e-01	1.31e+00
7×7, 80	2.27e-02	5.58e-02	5.96e-02	1.34e-01	6.76e-01	1.11e+00

Burgers I

- Exemples :

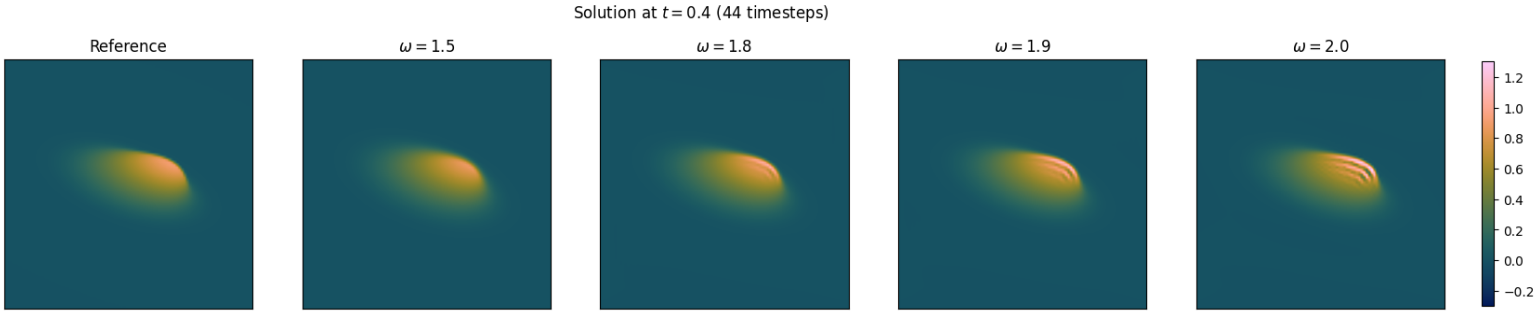


- Erreur des schémas classiques sur 220 pas de temps

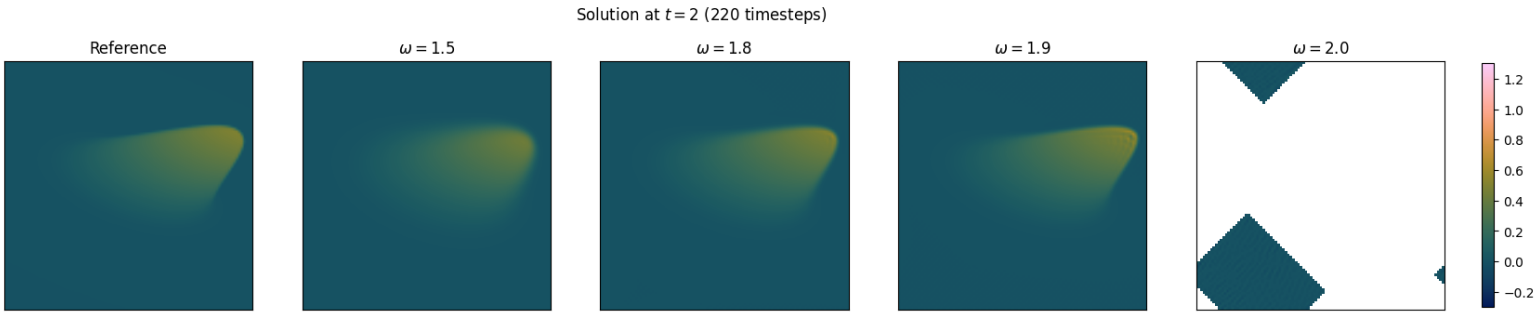


Burgers II

- Schéma classique :
- Erreur des schémas classiques sur 44 pas de temps

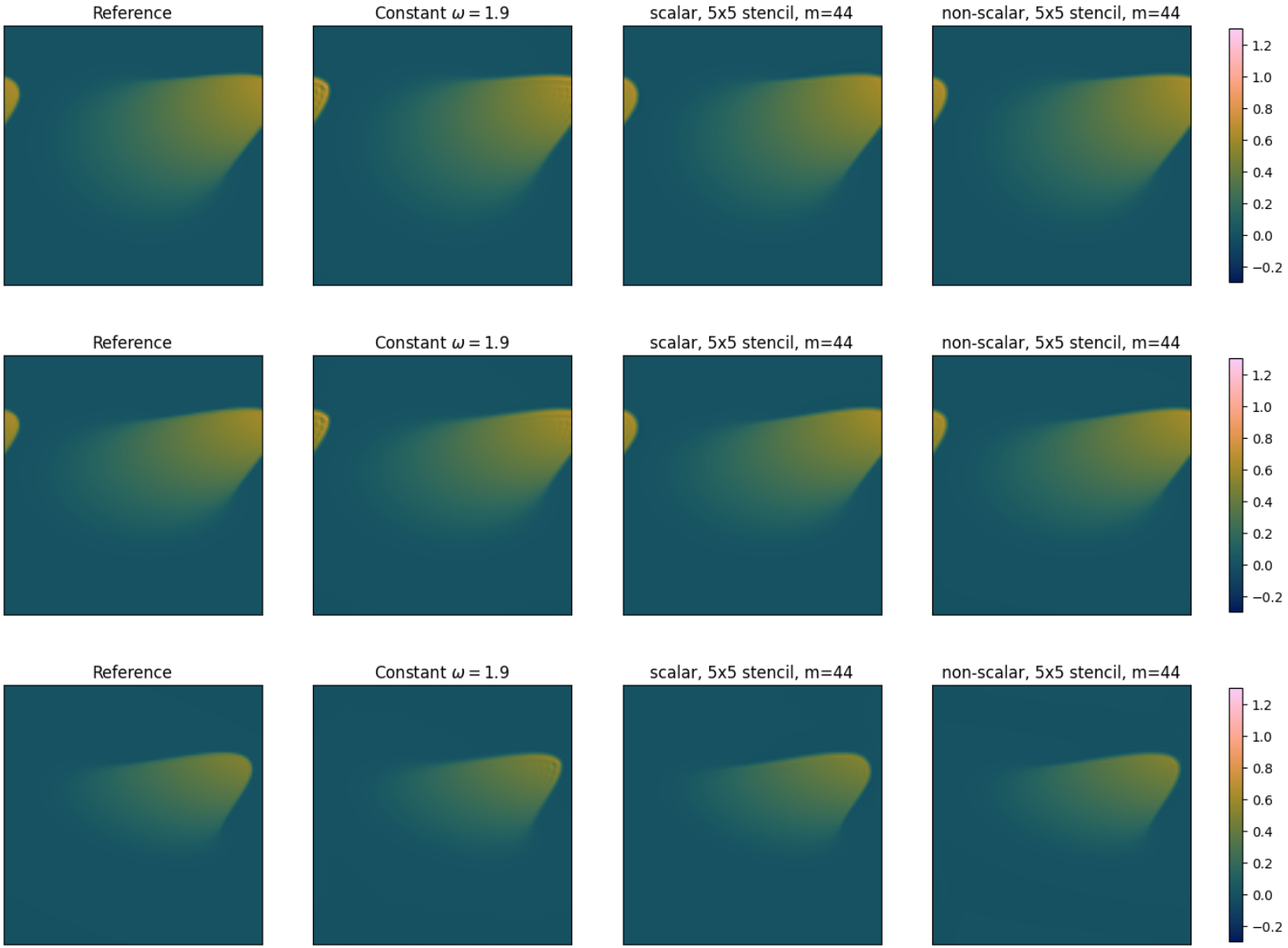


- Erreur des schémas classiques sur 220 pas de temps



Burgers III

- Comparaison :



Burgers IV : erreur

- cas scalaire :

Model	L^1		L^2		L^∞	
	$t = 0.4$	$t = 2.0$	$t = 0.4$	$t = 2.0$	$t = 0.4$	$t = 2.0$
Constant $= 1.9$	6.21e-03	8.53e-03	2.26e-02	2.24e-02	2.54e-01	3.23e-01
3×3, $m = 44$, batch 1	3.38e-03	5.06e-03	1.24e-02	2.06e-02	1.82e-01	3.71e-01
3×3, $m = 44$, batch 5	3.32e-03	5.44e-03	1.32e-02	2.14e-02	1.95e-01	3.72e-01
3×3, $m = 44$, batch 10	3.31e-03	5.51e-03	1.26e-02	1.93e-02	1.85e-01	3.42e-01
1×1, $m = 44$	4.12e-03	6.14e-03	1.37e-02	2.13e-02	1.83e-01	2.87e-01
3×3, $m = 44$	3.38e-03	5.06e-03	1.24e-02	2.06e-02	1.82e-01	3.71e-01
5×5, $m = 44$	2.84e-03	5.06e-03	1.10e-02	1.54e-02	1.61e-01	2.68e-01
7×7, $m = 44$	5.10e-03	6.02e-03	1.70e-02	1.88e-02	2.14e-01	2.76e-01
3×3, $m = 5$	5.31e-03	6.98e-03	1.99e-02	1.98e-02	2.29e-01	2.69e-01
3×3, $m = 11$	4.94e-03	7.56e-03	1.52e-02	2.66e-02	2.01e-01	3.51e-01
3×3, $m = 22$	4.70e-03	5.83e-03	1.69e-02	1.66e-02	1.99e-01	1.84e-01
3×3, $m = 44$	3.38e-03	5.06e-03	1.24e-02	2.06e-02	1.82e-01	3.71e-01
3×3, $m = 88$	3.32e-03	4.74e-03	1.30e-02	1.86e-02	1.84e-01	3.28e-01

- cas non scalaire :

Model	L^1		L^2		L^∞	
	$t = 0.4$	$t = 2.0$	$t = 0.4$	$t = 2.0$	$t = 0.4$	$t = 2.0$
Constant $= 1.9$	6.21e-03	8.53e-03	2.26e-02	2.24e-02	2.54e-01	3.23e-01
1×1, $m = 44$	4.96e-03	6.70e-03	1.43e-02	1.84e-02	1.67e-01	2.47e-01
3×3, $m = 44$	3.63e-03	5.00e-03	1.19e-02	1.37e-02	1.57e-01	1.70e-01
5×5, $m = 44$	3.85e-03	4.82e-03	1.31e-02	1.29e-02	1.76e-01	1.76e-01
7×7, $m = 44$	3.35e-03	3.75e-03	1.27e-02	1.17e-02	1.68e-01	1.41e-01
3×3, $m = 11$	6.61e-03	1.62e-02	2.03e-02	4.30e-02	2.27e-01	6.04e-01
3×3, $m = 22$	6.05e-03	1.33e-02	1.76e-02	3.36e-02	1.98e-01	4.49e-01
3×3, $m = 44$	3.63e-03	5.00e-03	1.19e-02	1.37e-02	1.57e-01	1.70e-01
3×3, $m = 88$	3.36e-03	4.00e-03	1.18e-02	1.20e-02	1.59e-01	1.60e-01

Équations d'Euler

- On s'intéresse aux équations d'Euler

$$\begin{cases} \partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0, \\ \partial_t (\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u} + p I_d) = 0, \\ \partial_t E + \nabla \cdot (E \boldsymbol{u} + p \boldsymbol{u}) = 0. \end{cases}$$

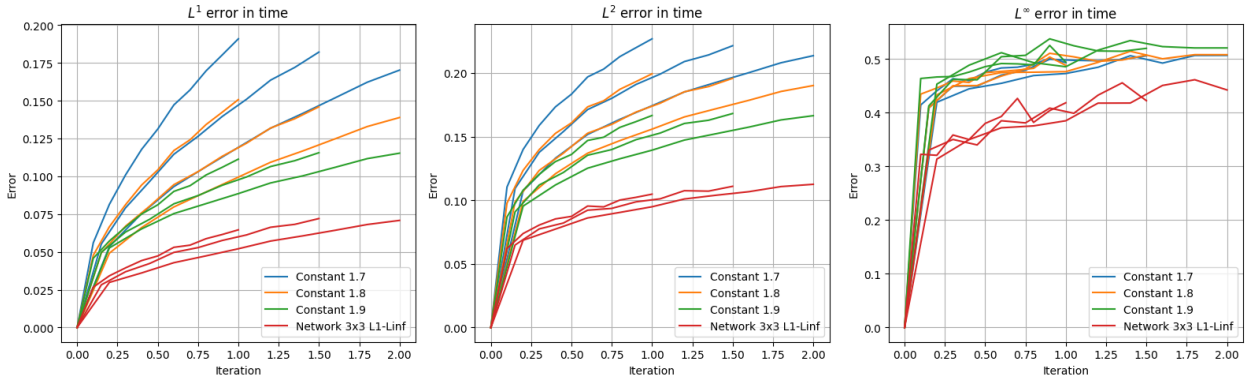
- Si $p = \text{cst}$ et $u = \text{cst}$, alors on a

$$\partial_t \rho + \boldsymbol{u} \cdot \nabla \rho = 0$$

- Pour la stabilité $\lambda > |\boldsymbol{u}| + \sqrt{\frac{\gamma p}{\rho}}$.
- Erreur de diffusion standard sur la densité :

$$E = \Delta t \nabla \cdot (\lambda^2 - |\boldsymbol{u}|^2) \nabla \rho + O(\Delta t)$$

- L'erreur augmente avec p sans que le problème de transport de densité change.



- Trois pressions : $p = 0$, $p = 0.012$, $p = 0.5$.

Conclusion

- **Conclusion** : cette technique fonctionne bien sur des cas limités. Elle permet d'obtenir des bonnes approximations peu oscillantes et peu diffusives.
- **Suite** : faire un papier (Un papier est sorti avec la même approche sur le D2Q9) ? Attaquer les systèmes ?
- Le plus intéressant est d'utiliser des matrices Ω non diagonales pour les systèmes, afin d'obtenir de meilleures viscosités.
- Des problèmes de stabilité compromettent rapidement l'entraînement. Questions :
  - comment détecter les possibles explosions et apprendre ?
  - comment caractériser/explorer les matrices Ω qui assurent la stabilité ? (la dessus on galère).

  1mm
- **Défaut** : la relaxation est plus locale en espace. Proposition de T. Bellotti : une matrice qui reste locale mais dépend de tous les f. Ils vont portés de l'informations sur le gradient.
- **Suite** : un code LBM différentiable dans taichi pour aller plus loin ? neural operator qui prédit la solution et LBM qui corrige ?
- **Autre direction** : version PINNS/Neural Galerkin des schémas cinétiques implicites (LBM ou le transport exact devient du SL).

Merci de votre attention !