Hybrid modeling and numerical methods for Vlasov equation

L. Bois, E. Franck, V. Michel-Dansac, L. Navoret, V. Vigon

MACARON project-team, Université de Strasbourg, CNRS, Inria, IRMA, France

16/06/2025, PASC 2025

1. Introduction

Vlasov equation

- We introduce the distribution of particles: $f(t, \mathbf{x}, \mathbf{v})$. The **Vlasov** equation is given by:

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + \mathbf{E}(\mathbf{x}) \cdot \partial_{\mathbf{v}} f = \frac{1}{\varepsilon} Q(f, f)$$

- with $Q(f, f)$ the collisional operator and ε the Knudsen number.

Difficulties:

- Multi-scale PDE in time and space
- High-dimensional PDE
  - the classical numerical methods suffer from the curse of dimensionality. For an error in $O(\varepsilon)$ we need

$$N_{dof} = O\left(\left(\frac{| u^{(p+1)}(x) |_\infty}{\varepsilon} \right)^{\frac{d}{p}} \right)$$

Objectives:

- propose reduced models to reduce the CPU/memory cost
- propose numerical methods more efficient in high dimension

2. hybrid reduced model

Moments model

Objectives : propose new reduced model for Vlasov equation using Machine learning methods

- In the following we will assume that

$$Q(f,f) = (\mathcal{M}(\rho,u,T) - f(t,x,v))$$

with

$$\rho(t,x) = \int_{\mathbb{R}} f(t,x,v)dv, \quad (\rho u)(t,x) = \int_{\mathbb{R}} f(t,x,v)vdv, \quad T(t,x) = \int_{\mathbb{R}} (v-u)^2 f(t,x,v)dv$$

- To construct a reduced model we compute the moments $\left(1, v, \frac{1}{2}v^2\right)$ of Vlasov equation:

$$\partial_t \rho + \partial_x \rho u = 0$$
$$\partial_t \rho u + \partial_x (\rho u^2 + p) = -\rho E$$
$$\partial_t \mathcal{E} + \partial_x (\mathcal{E} u + p u) + \partial_x q = -\rho u E$$

- with $\mathcal{E} = \frac{1}{2}\rho u^2 + \frac{p}{\gamma - 1}$ the total energy of the system and

$$q = \int_{\mathbb{R}} (v-u)^3 f(t,x,v)dv$$

the heat flux. Currently the system is closed only if we know q.

Closure

Definition (closure): A closure ins relation which expresses non-resolution scales as a function of system variables.

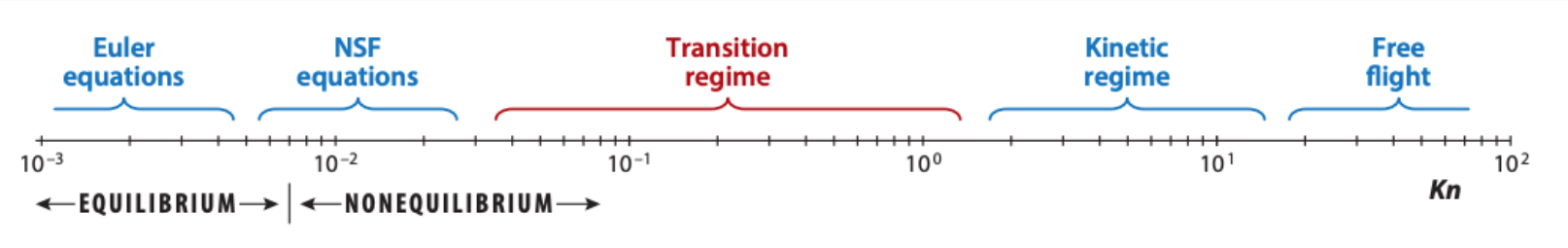
- Here a closure is a relation of the form:

$$q = \mathcal{C}(\rho, u, T, \varepsilon)$$

- The first closures were obtained by asymptotic analysis:
  - $f(t, x, v) = \mathcal{M}(\rho, u, T) \longrightarrow q(t, x) = 0$. We obtain the Euler equations.
  - $f(t, x, v) = \mathcal{M}(\rho, u, T) + \varepsilon g(t, x, v) \longrightarrow q(t, x) = -\frac{3}{2}\varepsilon p \partial_x T$. We obtain the Navier-Stokes equations.
- There is also nonlocal closure to capture specific kinetic effects: Hammett-Perkins, Ghendrih and al .

Objectives: Learn a efficient nonlocal closure using neural network (able to deal with large dimension inputs function).

Nonlocal closure learning I



Definition (learned closure): The learned closure is of the form:

$$q(t, x) = \mathcal{C}_\theta(\rho(), u(), T(), \varepsilon)$$

with θ the neural network parameters.

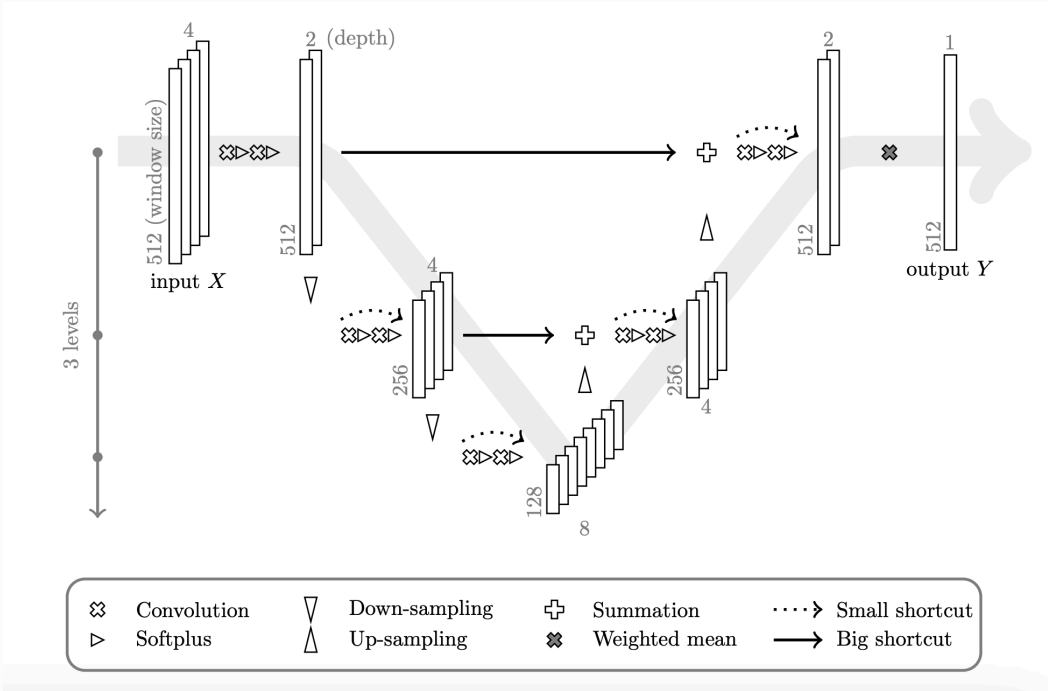


Figure 2: Network used: convolutional NN

Nonlocal closure learning II

- Methodology to produce the data using kinetic simulations:

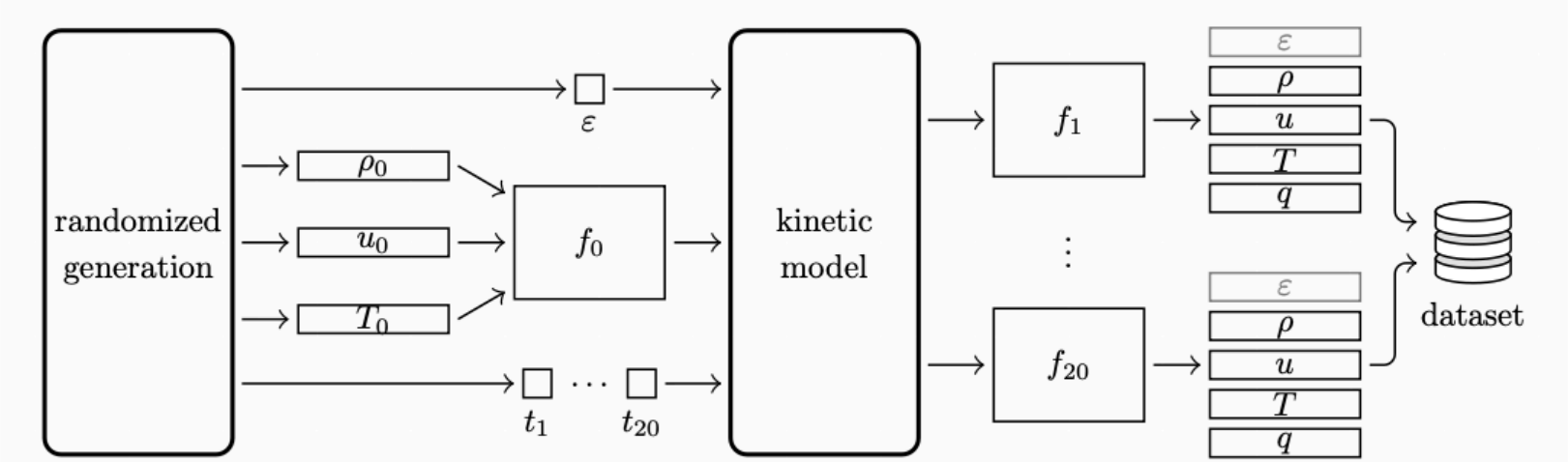


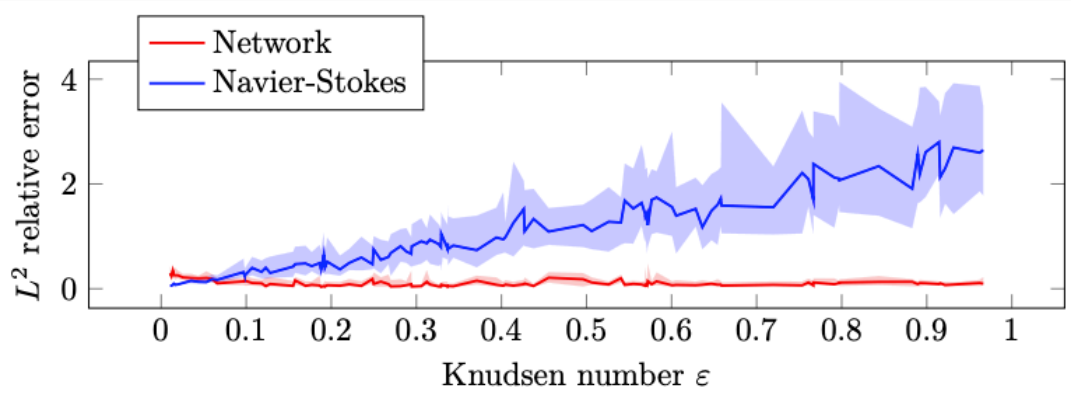
Figure 3: Network used: convolutional NN

- We minimize a classical L2 loss on the heat flux:

$$\min_{\theta}\left(\sum_{i}^{m}\int_{\Omega}\mid q_i(t^n,x)-\mathcal{C}_\theta(\rho_i(t^n,x),u_i(t^n,x),T_i(t^n,x),\varepsilon_i)\mid^2 dx\right)$$

Results I

- Results of L. Bois PhD (Inria).
- Statistical results for the error on q compared to the Knudsen number:



- The NN is less efficient for $\varepsilon < 0.05$

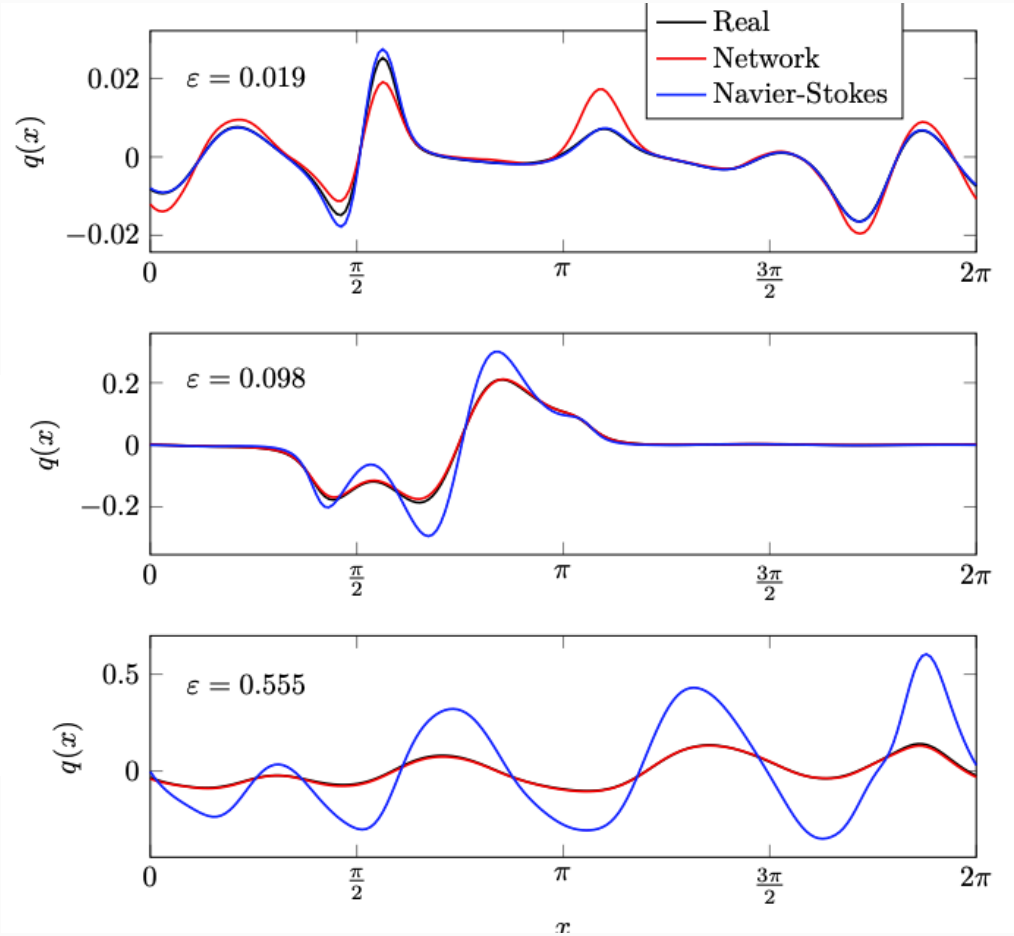
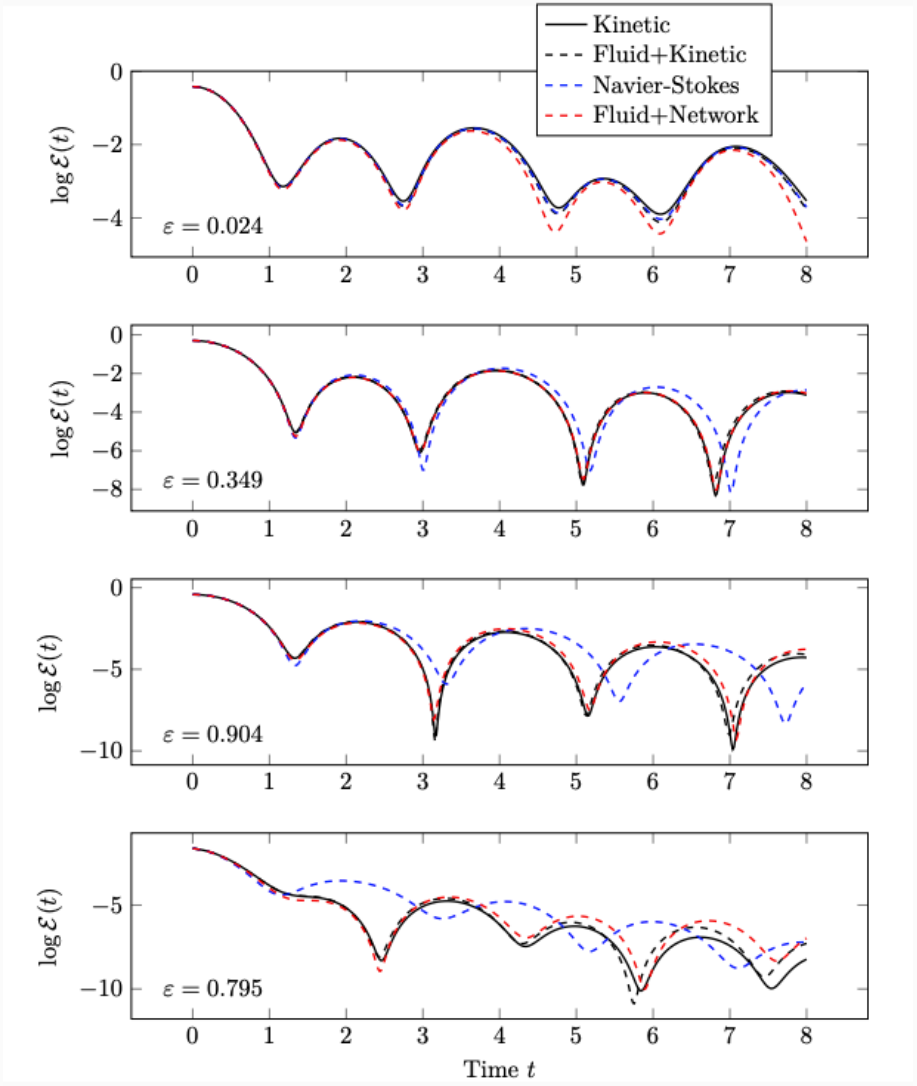


Figure 5: Example of prediction

Results II

- We compare:
  - A kinetic simulation,
  - A fluid simulation with q computed by a kinetic code,
  - A fluid simulation with NS closure,
  - A fluid simulation with NN closure
- We plot the time electrical energy decay for random initial data (random Fourier coefficient).

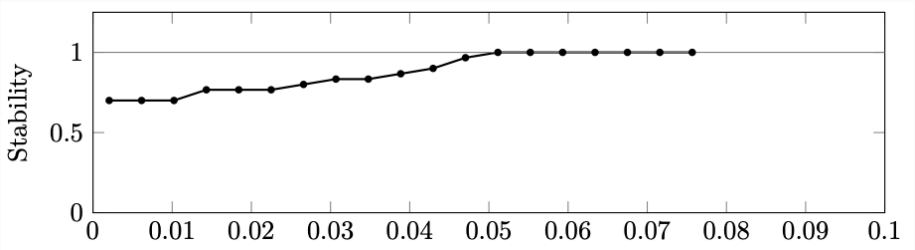


Stability

Question: The fluid model with NN closure is always stable or not ? no.

- In practice, we often observe some crashes due to negative pressure.
- To limit this effect, we smooth the output of the network with a Gaussian convolution of standard deviation σ.
- Sufficient for Vlasov in 1D, but not sufficient for Boltzmann in 2D.

▸ Results of stability agains σ:



▸ Results in 4D for BGK equation:

- The stability is critic especially for the neutral 4D case.

Future: structure preserving closures

Question: How obtain/assure the stability ?

- Unrolled/End to End training.
  - We train using the effect of the closure in the scheme during many time step. It can be view a close loop optimal control approach with NN.
- structure preserving learning and numerical discretization.
  - entropy dissipation or Hamiltonian conservation is imposed directly the neural network structure.

Objectives (closure Agnostic to the grid):

- replace Unet by Neural operators (FNO, Transformer)
- use symbolic regression to obtain analytic closure

3. hybrid numerical method

Classical numerical methods

Definition (numerical method): Restrict the research of PDE solution to a finite dimensional approximation space.

- We consider a elliptic PDE

$$\mathcal{L}(u(\mathbf{x})) = f(\mathbf{x}),\ \mathbf{x} \in \Omega$$
$$\mathcal{B}(u(\mathbf{x})) = g(\mathbf{x}),\ \mathbf{x} \in \partial\Omega$$

- How find a approximation of $u(\mathbf{x})$?
  - We define an approximation space:

$$V_n = \left\{ u_\theta(\mathbf{x}) = \sum_{i=1}^{N} \theta_i \varphi_i(\mathbf{x}) \right\}$$

  - We define a test space (constrains space) $W_n = \text{Span}\ (\psi_1, ..., \psi_n)$ and project the the residual on V_n orthgonaly to W_n:

$$\min_\theta \int_\Omega |\ (\mathcal{L}(u(\mathbf{x})) - f(\mathbf{x}))\psi_i(\mathbf{x})\ |^2\ d\mathbf{x}$$

- Solving the Least squrare problem we obtain the classical Petrov-Galerkin method:

$$\langle \mathcal{L}(u_\theta), \psi_j \rangle_{L^2} = \langle f, \psi_j \rangle_{L^2}$$

Classical numerical methods II

Definition (boundary condition): The BC can be treat weakly (penalization) adding the BC residual in the minimisation problem or strongly imposed in the space.

- How treat the time problem like $\partial_t u + \mathcal{L}(u) = f(x)$. We consider the same space but with involving in time.

$$V_n = \left\{ u_\theta(t, \mathbf{x}) = \sum_{i=1}^{N} \theta_i(t) \varphi_i(\mathbf{x}) \right\}$$

- Using the same principle we want:

$$\min_{\theta(t)} \int_\Omega | \left(\partial_t u_{\theta(t)}(\mathbf{x}) + \mathcal{L}\left(u_{\theta(t)}(\mathbf{x})\right) - f(\mathbf{x}) \right) \psi_i(\mathbf{x}) |^2 \, d\mathbf{x}$$

- **Time sequential approach**: we assume that we known $\theta(t_n)$ and we discretise in time:

$$\min_{\theta_{t_{n+1}}} \int_\Omega | \left(u_\theta(\mathbf{x}) - u_{\theta(t_n)}(\mathbf{x}) + \Delta t \mathcal{L}\left(u_{\theta(t_n)}(\mathbf{x})\right) - \Delta t f(\mathbf{x}) \right) \psi_i(\mathbf{x}) |^2 \, d\mathbf{x}$$

  Which gives:

$$M\theta(t_{n+1}) = M\theta(t_n) + \Delta t \mathbf{b}(\theta(t_n))$$

Nonlinear approach

- If $\psi_i(\mathbf{x}) = \varphi_i(\mathbf{x})$ we obtain **Galerkin** methods (Finite element, spectral, radial basis method etc).
- If $\psi_i(\mathbf{x}) = \delta_{\mathbf{x}_i}(\mathbf{x})$ we obtain **collocation methods**.

Objectives: replace the **linear approximation space** V_n by a **nonlinear approximation space**

- Possible nonlinear approximation space
  - Neural network:

$$u_{\mathbf{b},\mathbf{W}}(\mathbf{x}) = \varphi_L \circ \dots \varphi_0, \quad \varphi_i(\mathbf{x}) = \sigma(W_i\mathbf{x} + \mathbf{b}_i)$$

  - Nonlinear kernel and spectral method:

$$u_{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i e^{-(\mathbf{x}-\boldsymbol{\mu}_i)\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}, \qquad u_{\boldsymbol{\alpha},\boldsymbol{\omega}}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \sin(\omega_i \mathbf{x})$$

  - Tensor method:

$$u_\theta(\mathbf{x}) = \sum_{i=1}^{n} \varphi_i^1(x_1)\varphi_i^2(x_2)\varphi_i^3(x_3, x_4)$$

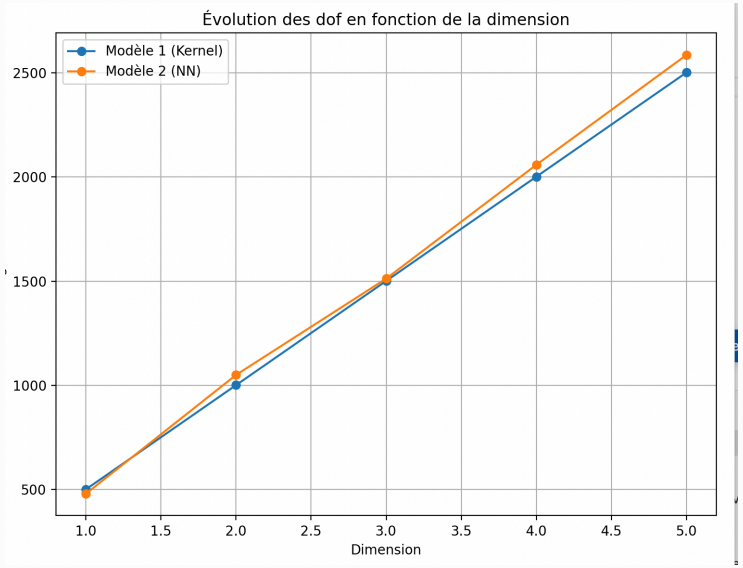
Remark: The optimisation problem is not quadratic so we cannot reduced to problem to a linear system.

Large Dimension

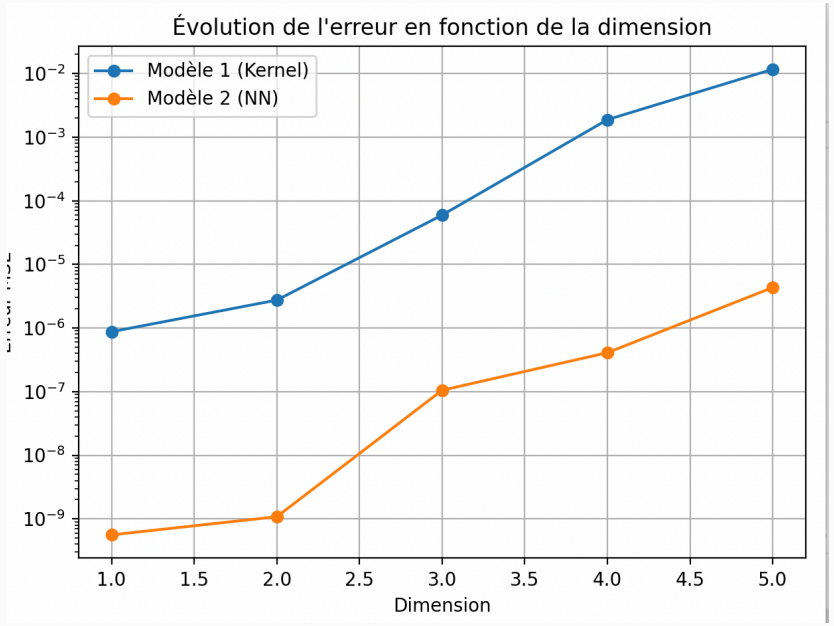
- We approximate

$$u(t, \mathbf{x}) = \prod_{i=1}^{d} \sin(2\pi x_i)$$

- Gaussian radial basis vs neural network:
  - Trajectory



  - Final time: analytical vs numerical solutions



- **Radial basis**: spectral convergence for this case $\| u - u_\theta \| \leq e^{-CN^{\frac{1}{d}}}$. Need problably $O(10^5)$ dofs.

Semi Lagrangian approach

- For transport equation like

$$\partial_t u(t,\mathbf{x}) + a(t,\mathbf{x}) \cdot \nabla u = 0$$

we can designed specific method called **semi Lagrangian** methods.

Definition (Characteristic): The solution of the transport equation is given by the characteristics:

$$u(t,\mathbf{x}) = u(t_0, \mathbf{X}(t_0,t,\mathbf{x}))$$

where

$$\frac{d}{dt}X(t_0,t,\mathbf{x}) = a(s,\mathbf{X}(t_0,t,\mathbf{x})), \quad X(t_0,t,\mathbf{x}) = \mathbf{x}$$

- It's allows to obtain a **sequential time scheme**:

$$\min_{\theta_{t_{n+1}}} \int_\Omega | \ (u_\theta(\mathbf{x}) - u_{\theta(t_n)}(\mathbf{X}(t_n,t_{n+1},\mathbf{x})) \ |^2 \ d\mathbf{x}$$

- We speak about **Semi lagrangian method**.

Semi Lagrangian II

Result (Consistency and stability): The order in time is the order of **ODE solver**. The scheme is explicit without CFL

- Possible extension to advection diffusion problem. More than one direction.
- We define the direction:

$$\mathbf{v}_0 = \frac{1}{\sqrt{d}}\mathbb{1}_d, \quad \mathbf{v}_i = \left(\sqrt{1+\frac{1}{d}}\right)e_i - \frac{1+\sqrt{d+1}}{d^{\frac{3}{2}}}\mathbb{1}_d$$

- We define the modified characteristic:

$$\mathbf{X}_i(t_n, t_{n+1}, \mathbf{x}) = \mathbf{X}(t_n, t_{n+1}, \mathbf{x}) + \sqrt{2d\sigma\Delta t}\mathbf{v}_i$$

with $\mathbf{X}(t_n, t_{n+1}, \mathbf{x})$ the transport characteristic

Remark: Order in time $\Delta^{\frac{1}{2}}$. Order 1 or 2 possible with more direction.

Neural Semi Lagrangian I

Idea (Neural SL): Since neural networks are good for high-dimensional problems, we use them to perform the regression step in SL with a neural network space.

- Can be used for parametric problems.
- **Algorithm**
  - In practice we know $u_{\theta(t_n)}(\mathbf{x}, \boldsymbol{\mu})$ at time t_n and we want to compute it at time $t_n + \Delta t$,
  - We randomly choose n collocation points using $p\left(\mathbf{x} \mid u_{\theta(t_n)}\right)$,
  - We solve the transport on the characteristic curves with RK4 for all the collocation points,
  - We modify the characteristic to deal with diffusion,
  - We solve

$$\theta(t_n + \Delta t) = \operatorname{argmin}_{\theta \in \mathbb{R}^p} \sum_{j=1}^{n} \left(u_\theta(\mathbf{x}_j, \boldsymbol{\mu}_j) - \sum_{i=0}^{d} u_{\theta(t_n)}(\mathbf{X}_i(t_n, t_{n+1}, \mathbf{x}_j, \boldsymbol{\mu}_j)) \right)^2$$

  initializing the optimisation with $\theta(t_n)$.

Neural Semi Lagrangian II

Result (Error): We can compute some basic error estimates like:

$$\| u_{\theta(t_n)}(\mathbf{x}) - u\left(t_{n(\mathbf{x})}\right) \|^2 \leq \sum_{k=0}^{k} C_n\left(\varepsilon_{approx,k} + \varepsilon_{opti,k} + \varepsilon_{mc,k}\right) + C\left(\frac{\Delta t^p}{n_t}\right)$$

with C and C_n depending of the flow.

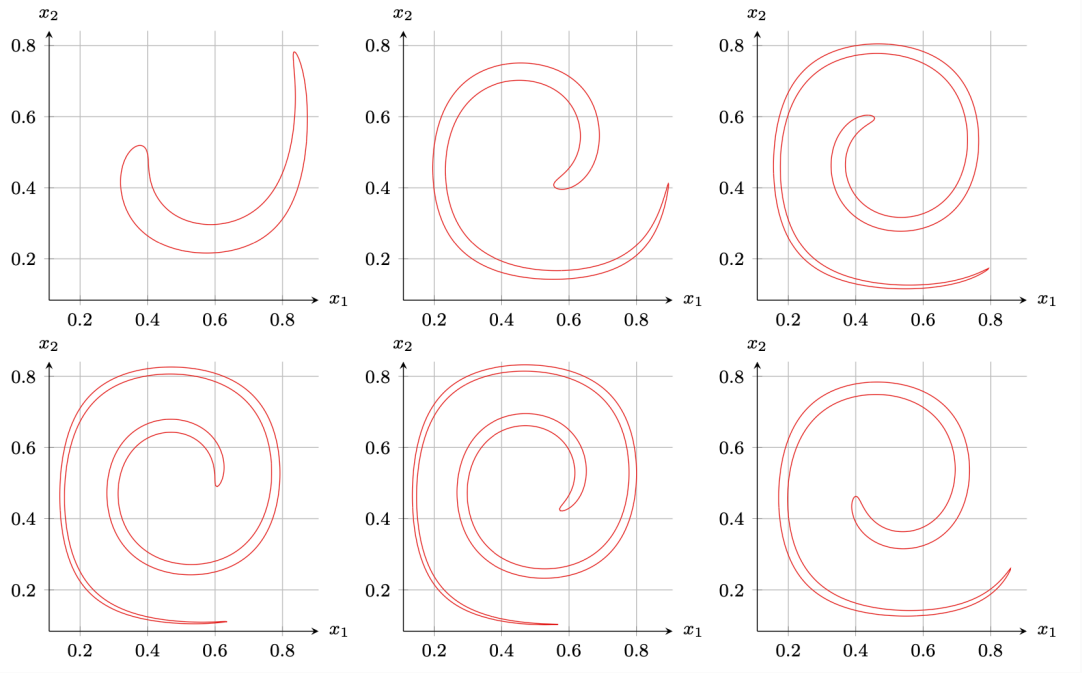
- **Key point**: in general $\varepsilon_{opti,k}$ dominate the other error terms.
- How limit this ? **Natural Gradient**:
  - Gradient descent

$$\theta_{k+1} = \theta_k - \eta_k \left(H_{\theta_k}\right)^{-1} \nabla_\theta J(\theta_k)$$

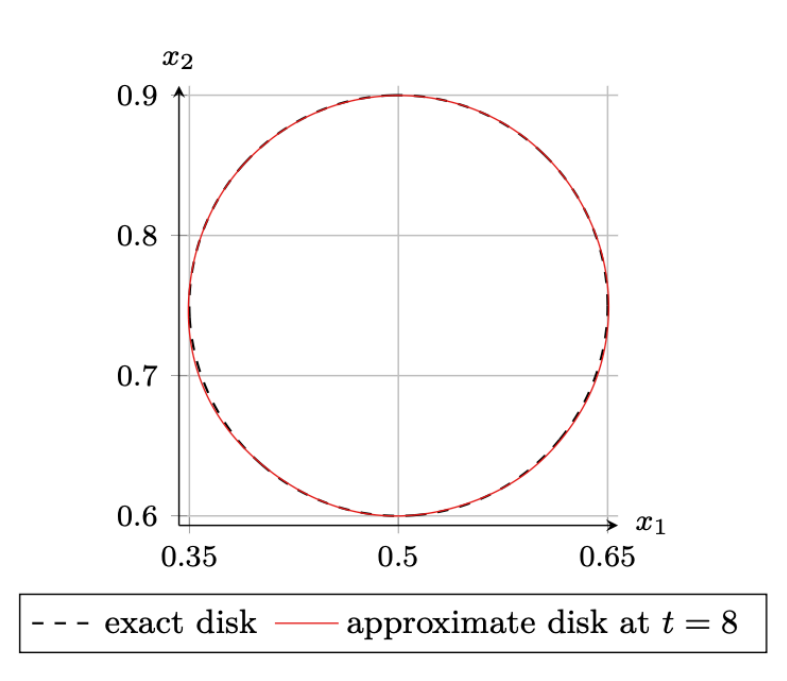
with $H_{\theta_k} = \sum_i^n \nabla_\theta nn_\theta(\mathbf{x}_i) \nabla_\theta nn_\theta(\mathbf{x}_i)^t$ and a line search method for η_k.

Results I

- We solve an advection of level set function with the Neural SL method.
  - Trajectory



  - Final time: analytical vs numerical solutions



- **Informations**: 3800 parameters, 60 000 collocation points, 250 epoch by time step

Results II

- We solve a advection diffusion problem in dimension d.

- Sinus test case

dimension d	computation time		relative error	
	init.	iter.	e_{L^2}	e_{L^∞}
1	50.12 s	13.81 s	1.56×10^{-3}	2.76×10^{-3}
2	55.96 s	17.07 s	4.04×10^{-3}	7.47×10^{-3}
3	76.06 s	25.12 s	6.36×10^{-3}	1.79×10^{-2}
4	121.85 s	36.62 s	4.01×10^{-3}	1.29×10^{-2}
5	198.20 s	57.18 s	2.27×10^{-3}	9.59×10^{-3}
6	341.12 s	101.95 s	1.17×10^{-3}	9.86×10^{-3}
7	591.64 s	168.27 s	5.60×10^{-4}	5.78×10^{-3}
8	1 000.97 s	262.36 s	8.43×10^{-4}	1.80×10^{-2}

- Gaussian test case

dimension d	computation time		relative error	
	init.	iter.	e_{L^2}	e_{L^∞}
1	33.67 s	49.08 s	1.38×10^{-3}	3.55×10^{-3}
2	39.29 s	41.85 s	2.07×10^{-3}	6.12×10^{-3}
3	46.32 s	51.41 s	1.54×10^{-3}	6.38×10^{-3}
4	58.18 s	66.80 s	8.49×10^{-4}	5.77×10^{-3}
5	78.26 s	89.54 s	3.07×10^{-4}	4.45×10^{-3}
6	106.11 s	122.03 s	9.50×10^{-5}	4.90×10^{-3}
7	146.83 s	168.42 s	1.08×10^{-4}	2.19×10^{-2}
8	217.95 s	243.31 s	3.88×10^{-5}	2.02×10^{-3}

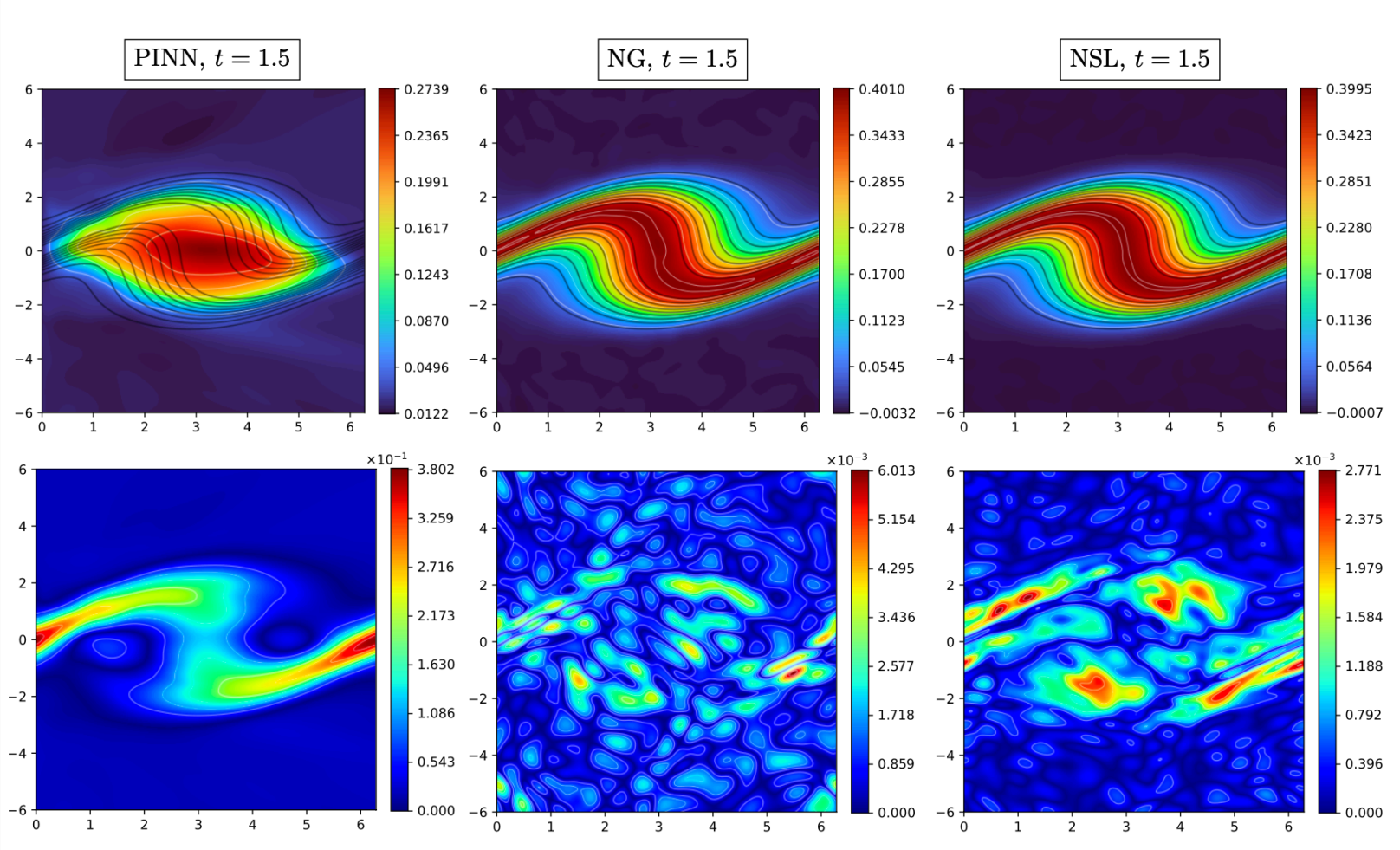
-**Informations**:

- Nb parameters $105\mathrm{d}^2 + 28^{\mathrm{d}} + 1$. For $\mathrm{d} = 1$ 134 parameters. For $\mathrm{d} = 8$: 6945 parameters.
- 100 000 collocations points.
- 200 to 400 epoch for initialization 5 to 5 epoch by time step.

Results III

- Linear Vlasov equation (no natural gradient):

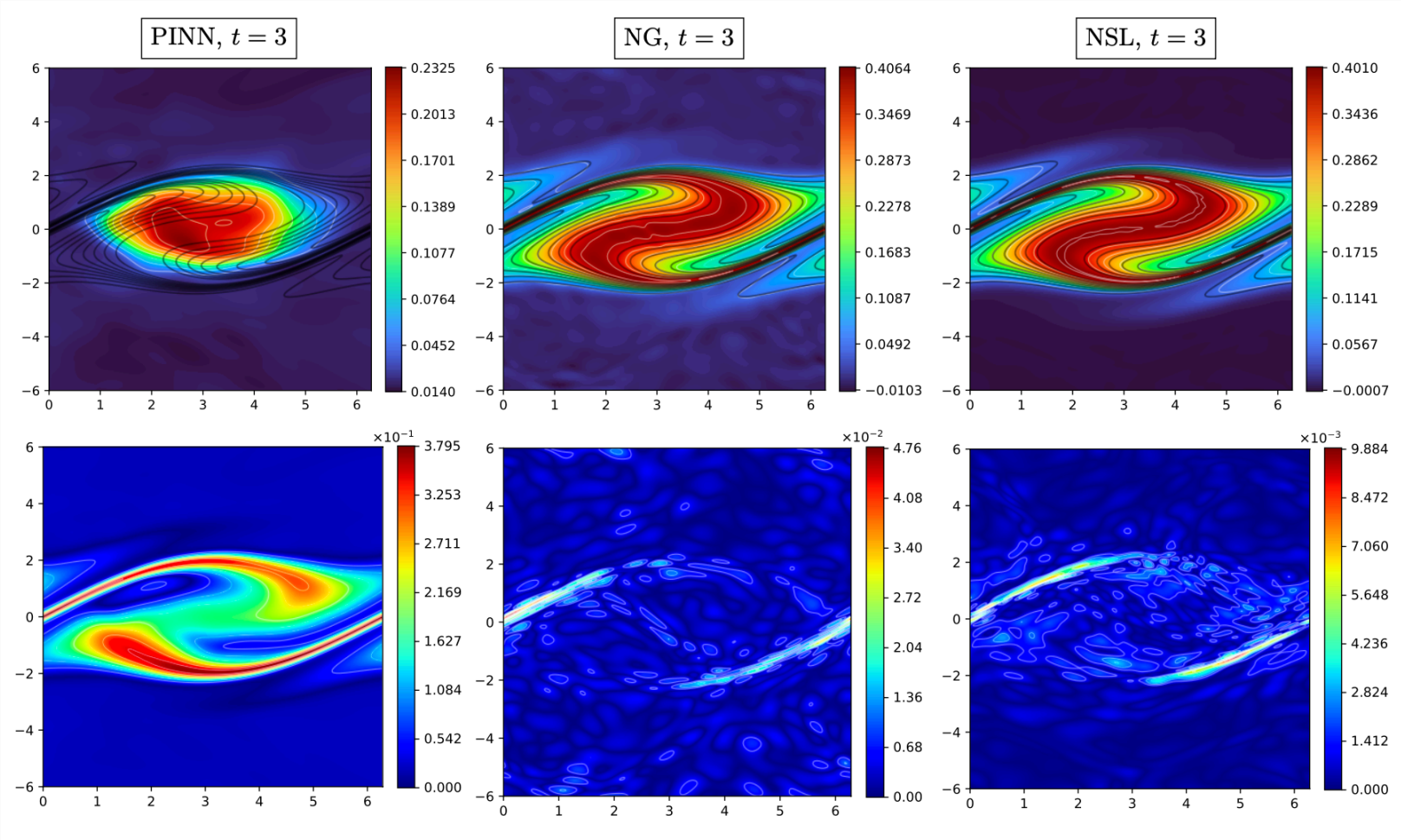
$$\partial_t f(t, x, v) + v\partial_x f + \sin(x)\partial_f = 0$$



Results III

- Linear Vlasov equation (no natural gradient):

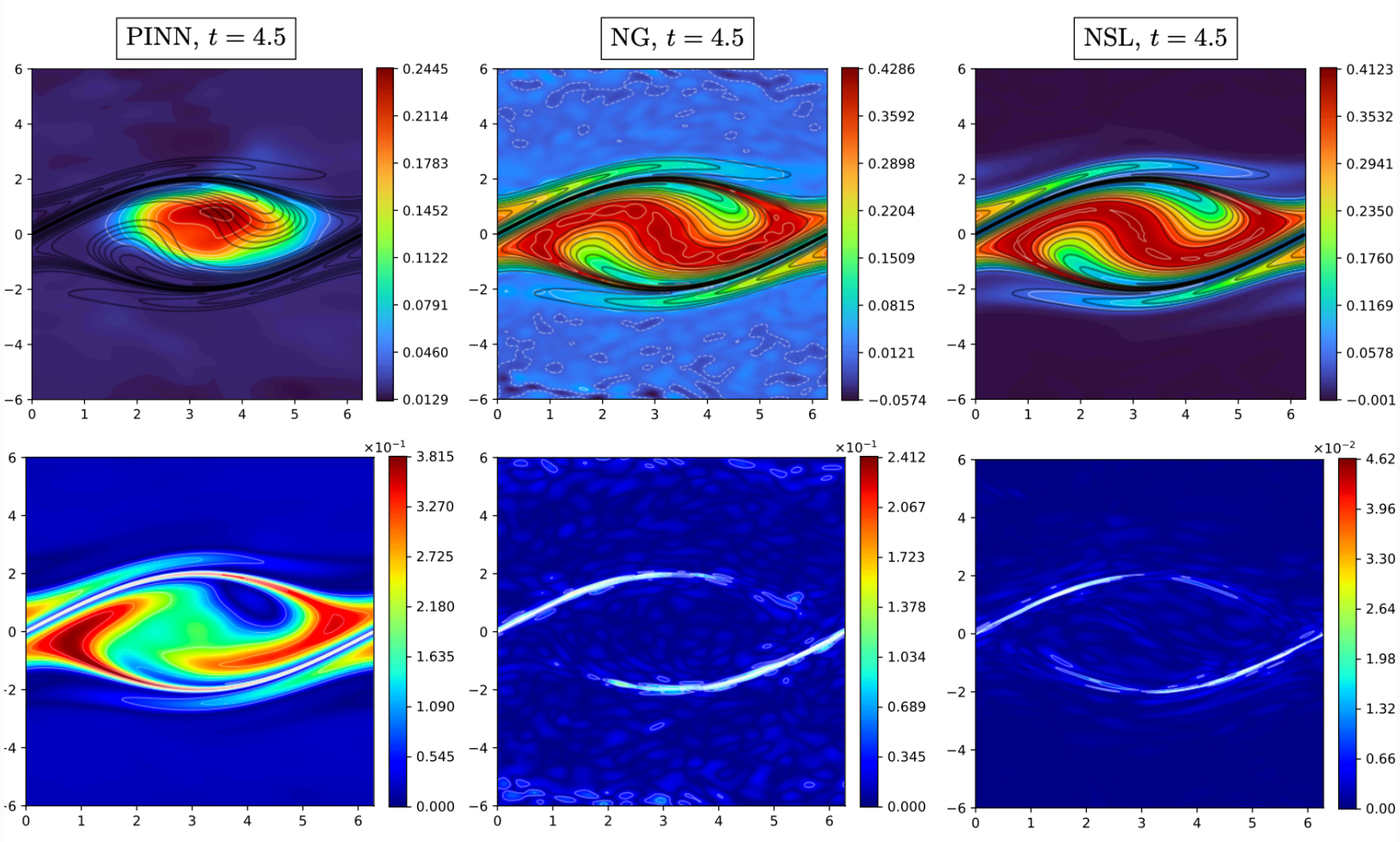
$$\partial_t f(t, x, v) + v\partial_x f + \sin(x)\partial_f = 0$$



Results III

- Linear Vlasov equation (no natural gradient):

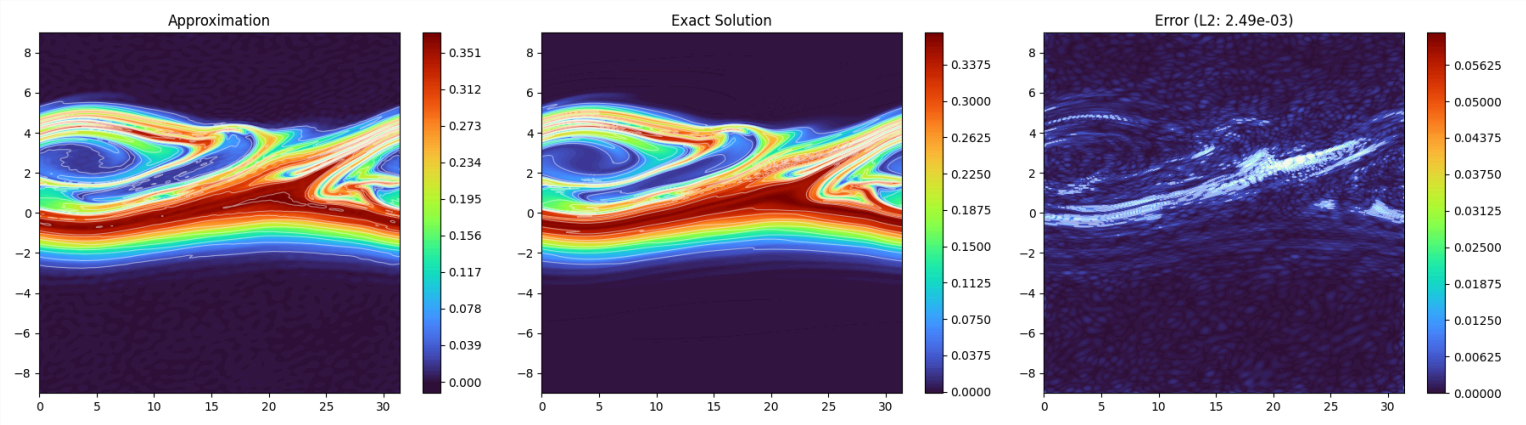
$$\partial_t f(t, x, v) + v\partial_x f + \sin(x)\partial_f = 0$$



Can we make more difficult ?

Remark (Spectral bias): neural networks are low-frequency biased models. Can we treat Vlasov-Poisson ?

- Just a training with a 7000 parameters basic network



4. Conclusion

Conclusion

Result: Coupling the SL method with neural networks approximation space ,good optimizer and adaptive sampling we can treat high dimensional and localized problem with good accuracy and reasonable CPU time.

- **Next**:
  - Vlasov-Poisson fully solved with neural network (on going)
  - Coupling with SL code like Gysela for compression and SL (with V. Grangirard)
  - Coupling with delta-PIC code (with E. Sonnendrücker, M. Campos Pinto and V. Fournet)
  - Fokker-Planck collision (with N. Crouseilles)
  - Adaptive neural network to increase accuracy (with A. Heinlien, V. Dolean)
  - Greedy optimizers (with V. Ehrlacher, F. Salin)