Hybrid numerical methods and models for plasma physics and gas dynamic

L. Bois[12], E. Franck[12], V. Michel Dansac[12], L. Navoret[12], G. Steimer[12], V. Vigon[12]

Workshop Simulation and machine learning hybrid modeling, Palaiseau

[1]Inria Nancy Grand Est, France
[2]IRMA, Strasbourg university, France

Outline

Physical and mathematical context

ANN for numerical methods applied to hyperbolic PDE's

Reduced models for kinetic equations

Physical and mathematical context

Plasma Physics

TONUS team:

- **Objectives**: construct new models (PDE) and numerical methods for problems in Nuclear fusion (Plasma physics).
- **Plasma**: gas at very large temperature which is electrically charged. Coupling between compressible fluid dynamic and electromagnetic.

- **Kinetic models**:
  - $$\partial_t f + \mathbf{v} \cdot \nabla f + (\mathbf{E} + \mathbf{v} \times \frac{\mathbf{B}}{\epsilon}) \cdot \nabla_{\mathbf{v}} f = \frac{1}{\tau} Q(f, f)$$
    with $f(t, \mathbf{x}, \mathbf{v})$ the seven dimensional particles distribution and $\mathbf{E}$, $\mathbf{B}$ the electric and magnetic fields given by the Maxwell equations.
  - Large dimensionnal multiscale PDEs with admit geometric structures to preserve and few diffusion process (less stable). So we need reduced models and adapted numerical methods.
- **Fluid models**:
  - $$\partial_t \mathbf{U} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \epsilon \nabla \cdot (\mathbf{G}(\mathbf{U}) \nabla \mathbf{U})$$
    with $\mathbf{U}(t, \mathbf{x}) \in \mathbb{R}^n$ the macroscopic quantities (density, velocity etc).
  - Strongly nonlinear, multiscale PDE with admit discontinuous solutions in the low diffusion regime. So we need adapted numerical methods. Tricky point: stability.

ANN for numerical methods applied to hyperbolic PDE

Hyperbolic equations and shocks

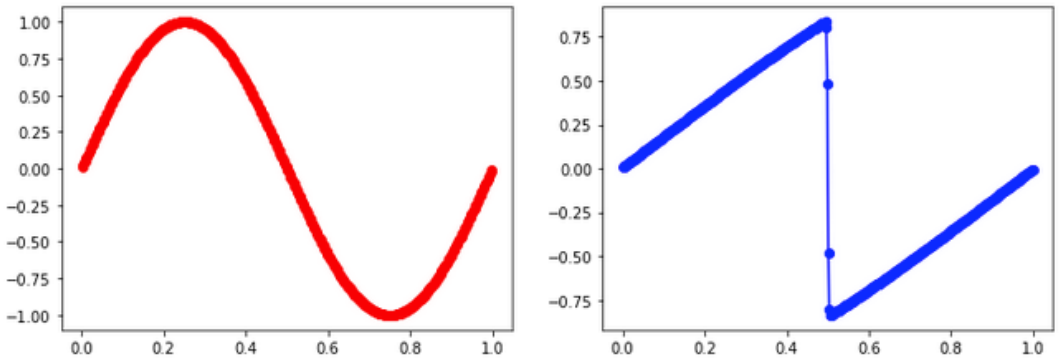
- In plasma physics and gas dynamics (our applications) we use conservation laws:

$$\partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) = 0$$

  with $\mathbf{U}(x) : \Omega \to \mathbb{R}^n$.
- **Specificity**: no regularization effect. **Worse**: discontinuous solution appears in time with a continuous initial data.
- Example: **Burgers equation** used for trafic flow:

$$\partial_t \rho + \partial_x \left(\frac{\rho^2}{2} \right) = 0$$

- Example of simulation. Left: initial data, Right solution at $T = 0.4$



- This property generate lot of difficulties at numerical level.

Finite volumes:

- Natural method to treat conservation laws with piecewise constant approximations.
- Conservation law (Green theorem):

$$\partial_t \int_a^b \mathbf{U} dx = -\int_a^b \partial_x \mathbf{F}(\mathbf{U}) dx = -\left(\mathbf{F}(\mathbf{U})(b) - \mathbf{F}(\mathbf{U})(a)\right) = 0$$

if no information enters the domain.

- We define a mesh with $N+1$ nodes and N cells. We call x_j th cell center and $x_{j\pm\frac{1}{2}}$ the left and right interface of the cell j.
- We integrate on the cell Ω_j:

$$\partial_t \int_{x_{j-\frac{1}{2}}}^{j+\frac{1}{2}} \mathbf{U} dx + \int_{x_{j-\frac{1}{2}}}^{j+\frac{1}{2}} \partial_x \mathbf{F}(\mathbf{U}) dx = 0$$

- We define as unknown the average value in the cell: $\mathbf{U}_j(t) = \frac{1}{\Delta x_j} \int_{x_{j-\frac{1}{2}}}^{j+\frac{1}{2}} \mathbf{U} dx$ and we have

$$\partial_t \mathbf{U}_j(t) + \left(\mathbf{F}(\mathbf{U})(x_{j+\frac{1}{2}}) - \mathbf{F}(\mathbf{U})(x_{j-\frac{1}{2}})\right) = 0$$

- To close the problem, the scheme make the following approximation:

$$\mathbf{F}(\mathbf{U})(x_{j+\frac{1}{2}}) \approx G(\mathbf{U}_j, \mathbf{U}_{j+1})$$

Finite volumes II

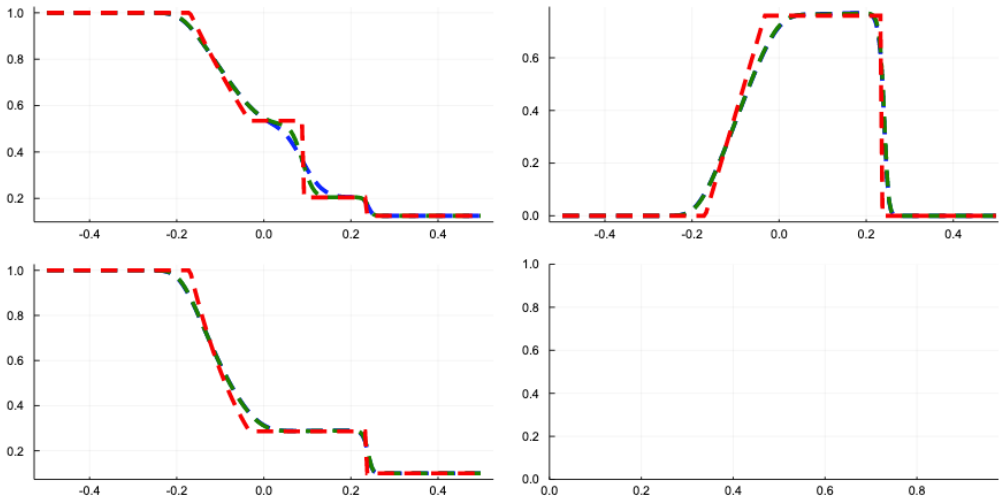
- **Main problem in FV**: How to choose $G(\mathbf{U}_j, \mathbf{U}_{j+1})$ (the flux) ?
  - *Central flux*: $G(\mathbf{U}_j, \mathbf{U}_{j+1}) = \frac{1}{2}\left(\mathbf{U}_j + \mathbf{U}_{j+1}\right)$. Accurate but unstable. Finite time blow-up of the solution.
  - *Rusanov flux*:

$$G(\mathbf{U}_j, \mathbf{U}_{j+1}) = \frac{1}{2}\left(\mathbf{U}_j + \mathbf{U}_{j+1}\right) + \frac{\lambda}{2}\left(\mathbf{U}_j - \mathbf{U}_{j+1}\right).$$

    Stable with condition on λ. Adding large numerical dissipation so not accurate.
  - *Viscosity-form flux*:

$$G(\mathbf{U}_j, \mathbf{U}_{j+1}) = \frac{1}{2}\left(\mathbf{U}_j + \mathbf{U}_{j+1}\right) + \frac{A(\mathbf{U})}{2}\left(\mathbf{U}_j - \mathbf{U}_{j+1}\right).$$

    Potentially stable and accurate scheme with the good $A(\mathbf{U})$. $A(\mathbf{U})$ difficult to find.
- Multiscale problem like Euler equations (gas dynamics):



- simulation with 200 cells.

DG code and dispersion

Discontinuous Galerkin method:

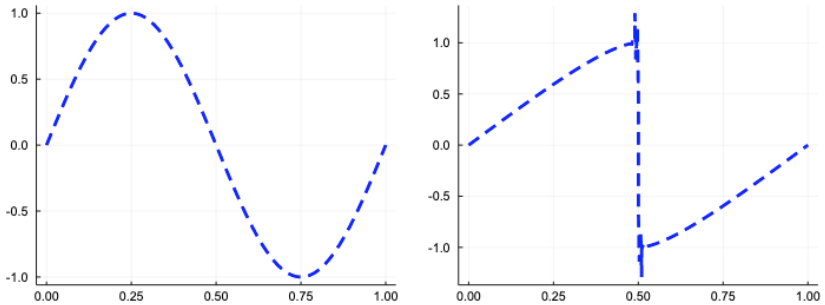
- Same principle but polymonial approximation of degre q and not $q = 0$.

$$\rho\,|_{\Omega_j}\,(t,x) = \sum_{i=1}^{q+1} \rho_i^j(t)\phi_i(x)$$

- We use the weak form of the equation:

$$\partial_t \int_{\Omega_j} \rho\phi(x) + \int_{\Omega_j} \partial_x \rho\phi(x) dx = 0$$

- We plug the polynomial extension of the unknowns, choose $\phi = \phi_j(x)$ integrate by part to obtain the scheme.
- Integration by part and discontinuity between cells $\rightarrow$ boundary terms at interface and we need numerical flux.
- **Remark**: since the method is more accurate the choice of the flux is less important.



- **Drawback**: polynomial reconstruction= Gibbs phenomenon.

Artificial viscosity

- To limit the spurious oscillations in DG scheme: limiter, artificial viscosity etc.
- **Artificial viscosity**: we solve

$$\partial_t \rho + \partial_x \left(\frac{\rho^2}{2} \right) = h^q \partial_x (D(\rho) \partial_x \rho)$$

with h the step mesh. The diffusion term is here to add regularization effect on the solution.

- **Aim**: design $D(\rho)$ to dissipate mainly the numerical oscillations.
- Exemple: derivative-based viscosity

$$q = 2, \quad D(\rho) = \lambda_{max} \mid \nabla \rho \mid$$

- Efficient in short time, too dissipative for long time computation.

Aim:

- Design good finite volumes fluxes or DG artificial viscosity (AV) with neural networks:

$$\mathbf{G}_\theta(\mathbf{U}_l, \mathbf{U}_r), \quad D_\theta(\rho(.))$$

Supervised approach

- A first approach is the **supervised approach**.
- There is many fluxes and artificial viscosities in the litterature.
- **Idea**: build a flux that interpolates between the known flows by choosing the best flux as the learning output.
- For some test cases, we compute the best flux $\mathbf{G}_{best}(\mathbf{U}_l, \mathbf{U}_r)$ testing all the fluxes and we solve

$$\theta^* = \text{argmin}_\theta \sum_j \| \mathbf{G}_\theta(\mathbf{U}_l^j, \mathbf{U}_r^j) - \mathbf{G}_{best}(\mathbf{U}_l^j, \mathbf{U}_r^j) \|_2$$

- Drawbacks:
  - which metrics to determinate the best flux,
  - perhaps not able to learn new type of fluxes,
  - difficult to assure long time accuracy and stability.
- Approach tested for artificial viscosity (J. Hesthaven and al, JCP 2020)

Our problem like an RL/optimal control problem

- To avoid the two last drawbacks we can use Reinforcement learning.
- **RL**: closed-loop optimal control interacting with an environment (unknown model).
- Framework:
  - State and action:

$$s_t = \rho^n, \quad a_t = \text{flux values or AV value}$$

  - Policy (to determine):

$$\mu_\theta(s_t) = \text{flux or AV functions}$$

  - The environment

$$r_{t+1}, s_{t=1} = Env(s_t, a_t) = \text{time scheme}$$

  - Reward:

$$r_{t+1} = \| \rho^{n+1} - \rho^{n+1}_{ref} \|_E + \lambda \| a_t \|_2^2$$

- Solving this reinforcement problem is equivalent to solve an inverse problem with closed-loop optimal control where we want to fit a reference trajectory in time.
- The reference trajectory is computed with a Finite Volumes code on very fine grids for example.

DDPG and defaults

- **Flux case**: the state space S and action space are subspaces of $\mathbf{R}^{n_v}$ with n_v the number of variables (Y. Wang and al 2019),
- **DG case**: the state space S and action space are subspaces of $\mathbf{R}^N$ with N the number of cells.

DDPG:

- Most common algorithm for Reinforcement in continuous action and state space is DDPG.
- **DDPG**: actor-critic algorithm, with Q function construct using Bellamn + policy gradient. Deterministic policy.
- **Exploration**: add random process to the action obtained by the policy or add random process in the weights of the ANN policy.

Drawback:

- The exploration in the high dimensional continuous action space is very hard. How to explore a space of discrete spatial functions like in the DG case ?

- The reinforcement learning is made for the case where the model is unknown (contrary to dynamic programming).
- DDPG is a model-free approach. Alternative: model-based approach.

Model-based RL approach

Model based approach: back-propagation

- We learn the model with random exploration and compute the policy using back-propagation. We want maximize:

$$V(s_0) = \sum_{t=1}^{T} \gamma^t r_{t+1}$$

If $r_{t+1} = r(s_t, a_t)$ and $s_{t+1} = f(s_t, a_t)$ and $a_t = \mu_\theta(s_t)$ then

$$V(s_0) = r(s_0, \mu_{\theta(s_0)} + r(f(s_0, \mu_{\theta(s_0)}), \mu_\theta(f(s_0, \mu_{\theta(s_0)}))) + ...$$

- The gradient can be computed if f and r differentiable.
- For our problems f, r are known.

Our algorithm

- Wee choose randomly ρ_0.
- We compute two trajectories $(\rho^0, ..., \rho^T)$ and $(\rho^0_{ref}, ..., \rho^T_{ref})$.
- We update the weights:

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

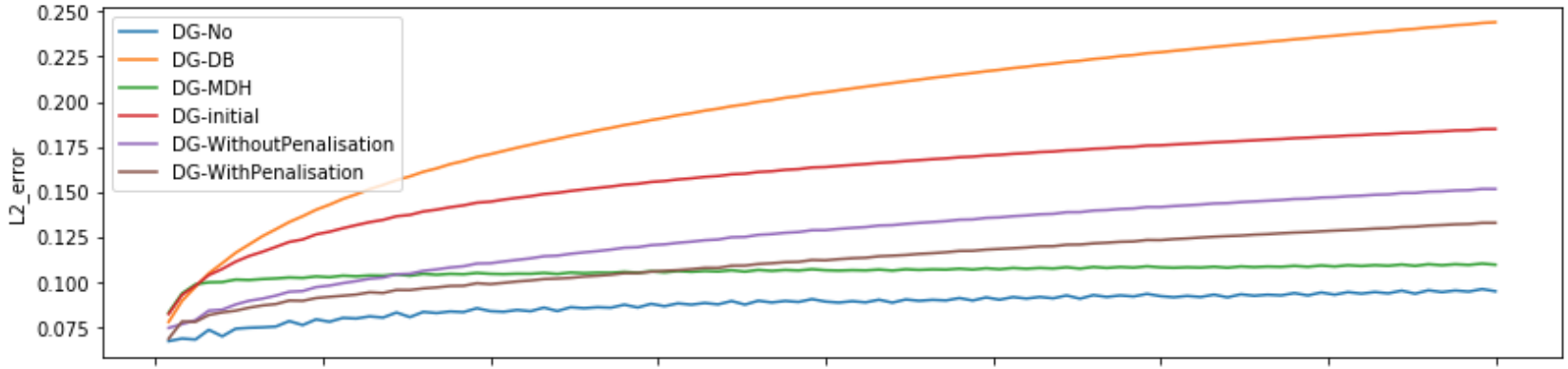
with $J(\theta) = \| \rho^T - \rho^T_{ref} \|_E + \lambda \| a_t \|_2^2$ and the gradient computed by back-propagation (model is known).

Preliminary results for DG case

- We propose to learn an artificial viscosity for

$$\partial_t u + a\partial_x u = \partial_x(D_\theta(u)(\partial_x u))$$

- Oscillations are not critical unlike in the nonlinear case.
- We compare with "derivative-based (DB)" and "MDH" (J. Hestaven papers).
- Error L_2:



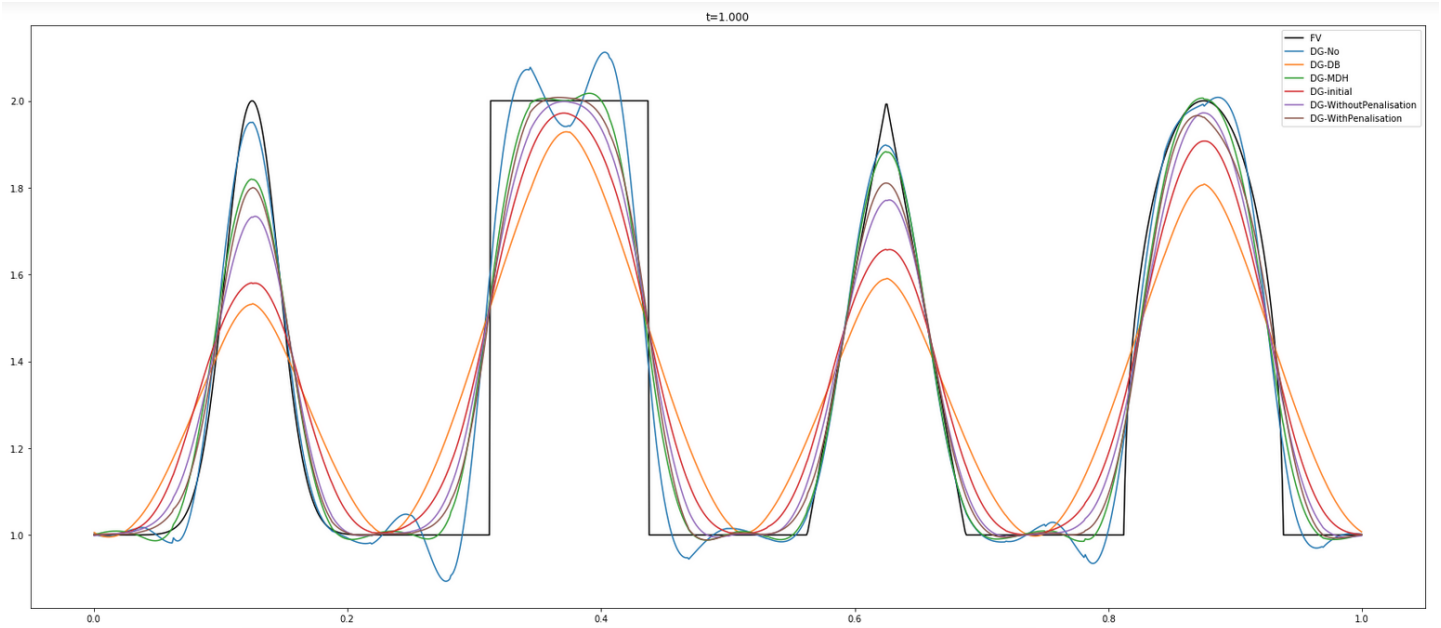
- Our viscosity is a compromise between "MDH" (few dissipation but small oscillations) and "DB" (too dissipative). Parameters perhaps non optimal.
- **Difficulties**: the loss does not "see sufficiently the oscillations" compare to diffusion error.

Preliminary results for DG case

- We propose to learn an artificial viscosity for

$$\partial_t u + a\partial_x u = \partial_x(D_\theta(u)(\partial_x u))$$

- Oscillations are not critical unlike in the nonlinear case.
- We compare with "derivative-based (DB)" and "MDH" (J. Hestaven papers).
- Time $T_f = 1$:



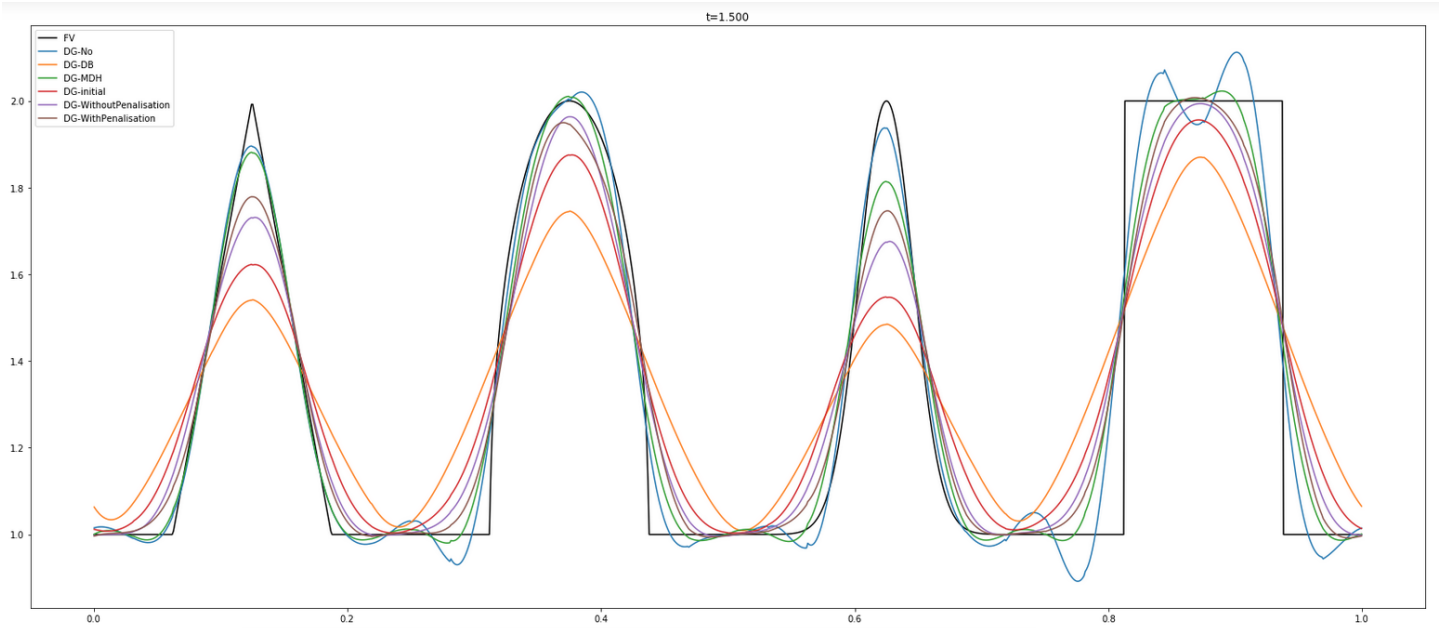
- Our viscosity is a compromise between "MDH" (few dissipation but small oscillations) and "DB" (too dissipative). Parameters perhaps non optimal.
- **Difficulties**: the loss does not "see sufficiently the oscillations" compare to diffusion error.

Preliminary results for DG case

- We propose to learn an artificial viscosity for

$$\partial_t u + a\partial_x u = \partial_x(D_\theta(u)(\partial_x u))$$

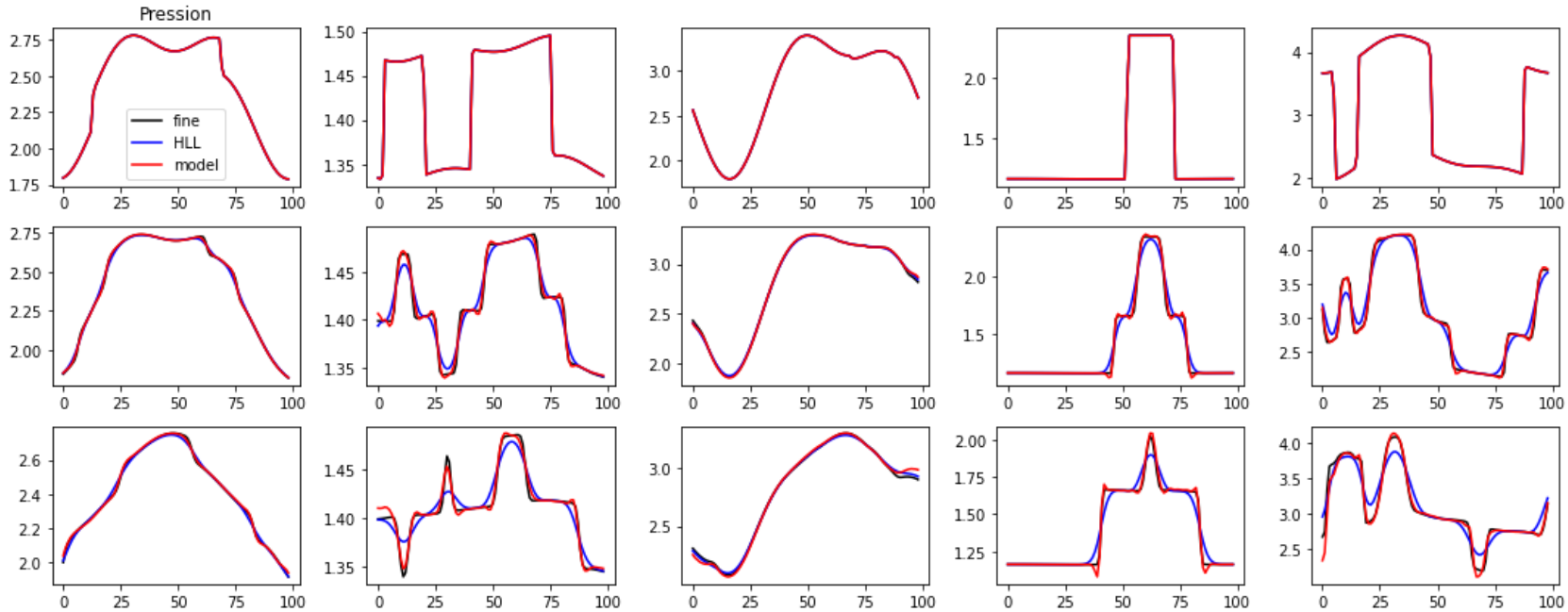
- Oscillations are not critical unlike in the nonlinear case.
- We compare with "derivative-based (DB)" and "MDH" (J. Hestaven papers).
- Time $T_f = 1.5$:



- Our viscosity is a compromise between "MDH" (few dissipation but small oscillations) and "DB" (too dissipative). Parameters perhaps non optimal.
- **Difficulties**: the loss does not "see sufficiently the oscillations" compare to diffusion error.

Preliminary results for Finite volumes case

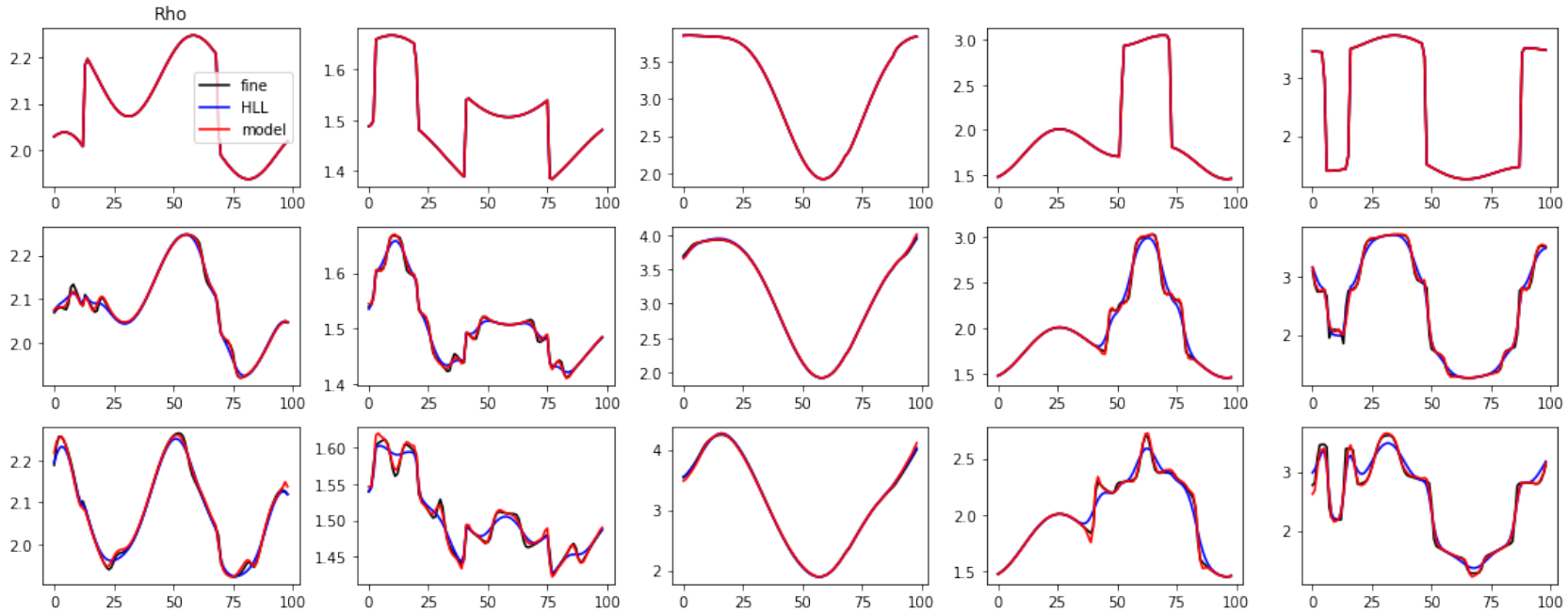
- We propose to learn a correction to the classical HLL flux.
- Results on general test cases:
- Pressure:



- **Conclusion**: very good results on non-trivial solutions and bad results on Sod-like problems. Is a physical prior needed ?

Preliminary results for Finite volumes case

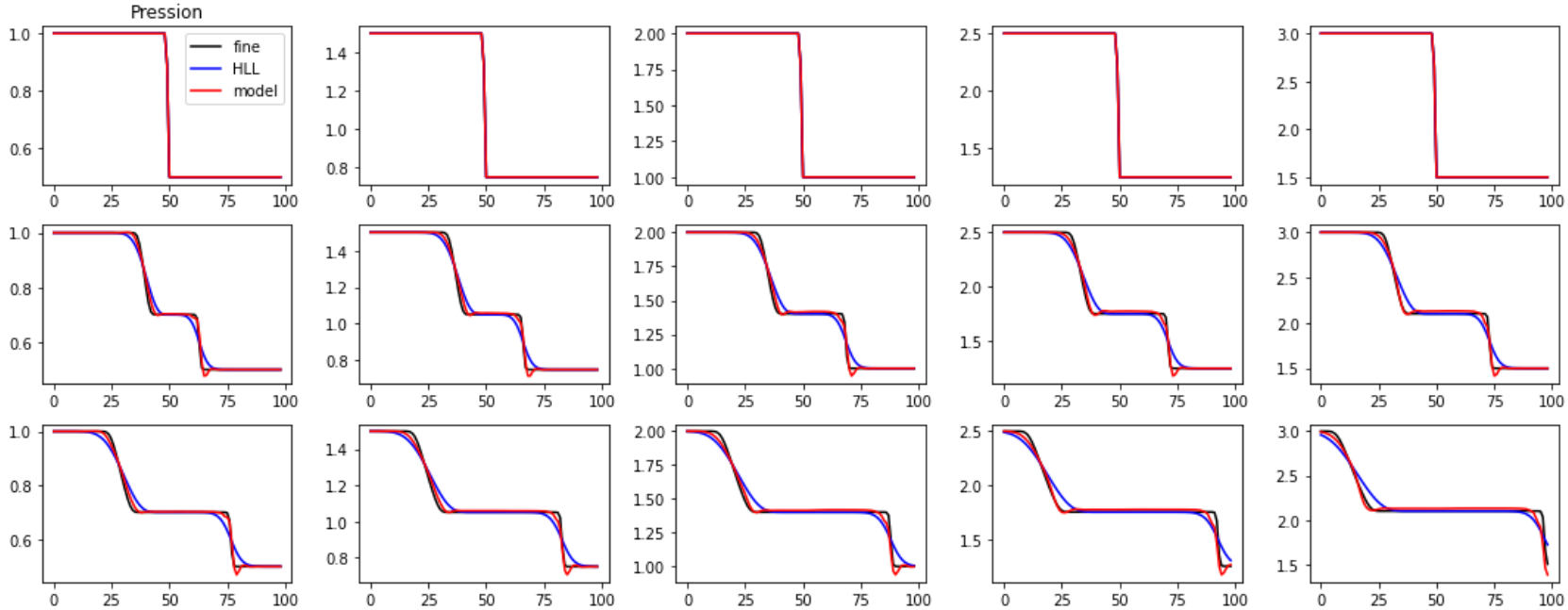
- We propose to learn a correction to the classical HLL flux.
- Results on general test cases:
- Densité:



- **Conclusion**: very good results on non-trivial solutions and bad results on Sod-like problems. Is a physical prior needed ?

Preliminary results for Finite volumes case

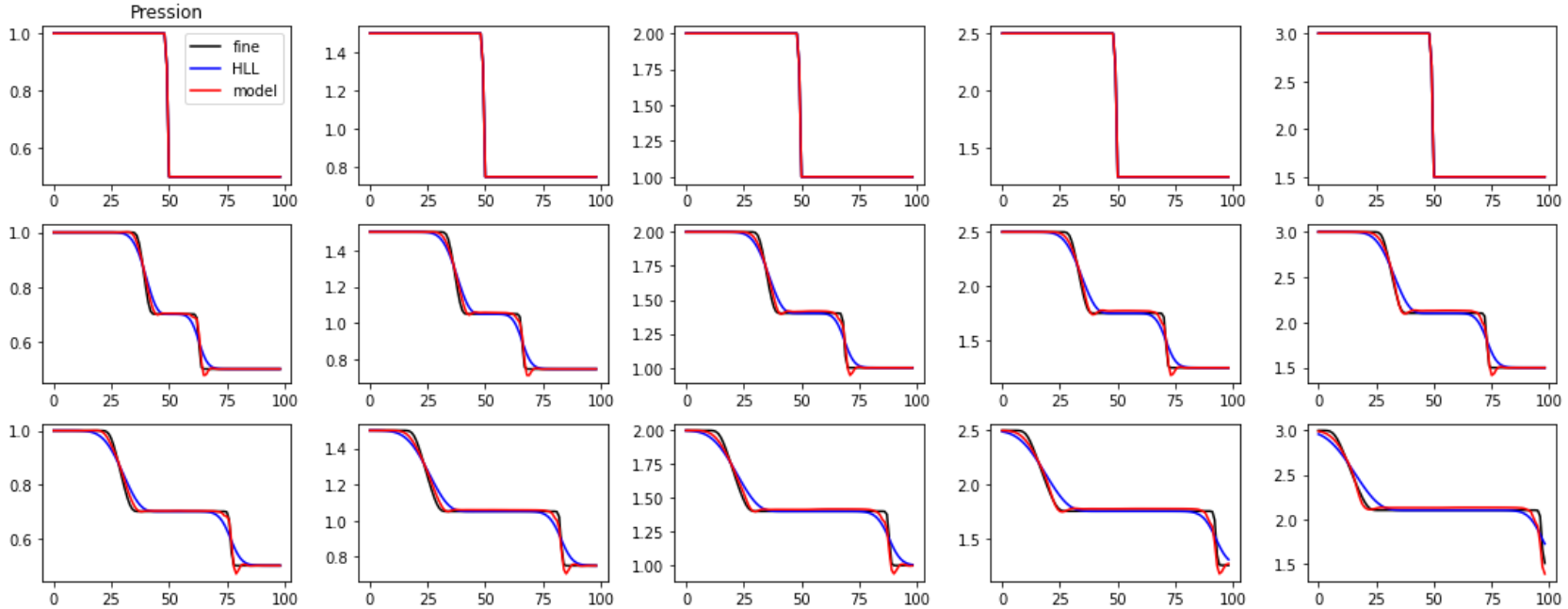
- We propose to learn a correction to the classical HLL flux.
- Results on classical Sod test cases:
- Pressure:



- **Conclusion**: very good results on non-trivial solutions and bad results on Sod-like problems. Is a physical prior needed ?

Preliminary results for Finite volumes case

- We propose to learn a correction to the classical HLL flux.
- Results on classical Sod test cases:
- densité:



- **Conclusion**: very good results on non-trivial solutions and bad results on Sod-like problems. Is a physical prior needed ?

Conclusion

- We propose a simple method based on back-propagation and optimal control view to design or modify numerical schemes.
- Compared to the supervised approach, we can learn new type of terms with good long time behavior.
- **Remark**: to perform better results, we must add additional knownledge. How ?

Limit

- With the back-propagation we can reach only local minimum. Not global mechanics like in DDPG.
- We cannot solve the problem on large time (max around 1000 time step) since the back-propagation becomes instable or to heavy.
- **Next**: propose something between DDPG and back-propagation.
- **Next**: find better metric to detect oscillations.

Other investigation

- Extension on unstructured grids with GNN's and geometric deep learning theory.
- Very premilinary results are positives.
- **Missing in librairies**: lot of codes for graphs in Spektral and Geometric Pytorch, less for meshes. An engineer to implement the tools for meshes (interesting for different teams) ?

Reduced models for kinetic equations

Vlasov and PIC code

- We consider the 1Dx1D nonlinear Vlasov-Poisson equation for plasma:

$$\partial_t f(t,x,v) + v\partial_x f + (E_{ext}(x) + E(x))\partial_v f = 0$$

with

$$-\Delta\phi(x) = \int_v f(t,x,v')dv' - 1, \quad E(x) = -\nabla\phi(x)$$

- $f(t,x,v)$ is a probability density of the particles.
- **Solver PIC** (Particle In Cells). We approximate the distribution by macro-particles

$$f(t,x,v) \approx f_N(x(t),v(t)) = \sum_{i=1}^{N} w_i \delta(x - x_i(t))\delta(v - v_i(t))$$

where

$$\begin{cases} \dfrac{dx_i}{dt} = v_i, \\ \dfrac{dv_i}{dt} = \frac{q}{m}(E + E_{ext})(x_i(t)), \end{cases} \tag{1}$$

- To compute the electric field, we compute $\int_v f(t,x,v')dv'$ on the mesh, solve the Poisson equation on the mesh, interpolate $E(x)$ at the position of macro-particles.

ROM and POD

- After PIC discretization, we have a large ODE (d^2N with d the dimension and N the number of particles) to solve.
- PIC method converge in $O(\frac{C(f)}{\sqrt{N}})$ so $d^2N >> 1$ (like MC methods).

ROM method

Design reduced models of size $K << N$ for a subset of initial data and parameters.
Assumption:

$$\mathbf{Y}(t) \approx A\mathbf{Z}(t)$$

with $\mathbf{Y}(t) \in \mathbb{R}^{d^2N}$, $\mathbf{Z}(t) \in \mathbb{R}^{2K}$.

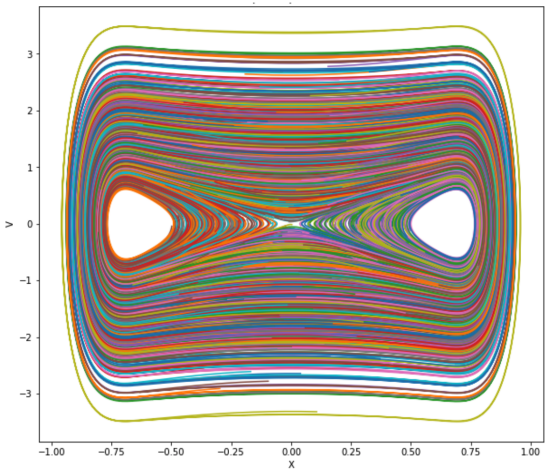
- Here we consider $f(t=0,x,v) = f_\gamma(\alpha,\beta)\mathcal{M}(v)$ with (α,β) the parameters.
- Classical approach: **POD** (see J. Hestaveen papers)
  - We collect some snapshots: $S = [(x(t_1), v(t_1)), ..., (x(t_n), v(t_n))]$.
  - By the SVD method, we compute the K dominant mods (associated to the K largest eigenvalues) and construct: $A \in \mathcal{M}_{K,N}$ (encoder) et $A^+ \in \mathcal{M}_{N,K}$ (decoder).
  - **Reduction**:

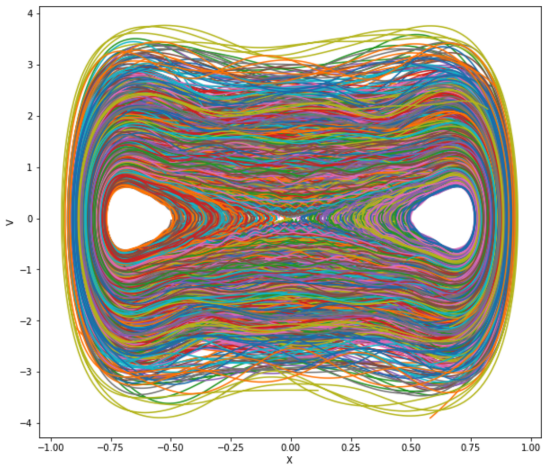
$$\partial_t \mathbf{Y}(t) = \mathbf{F}(\mathbf{Y}(t)) \rightarrow \partial_t \mathbf{Z}(t) = A\mathbf{F}(A^+\mathbf{Z}(t)), \text{ with } \mathbf{Z}(t) = A\mathbf{Y}(t)$$

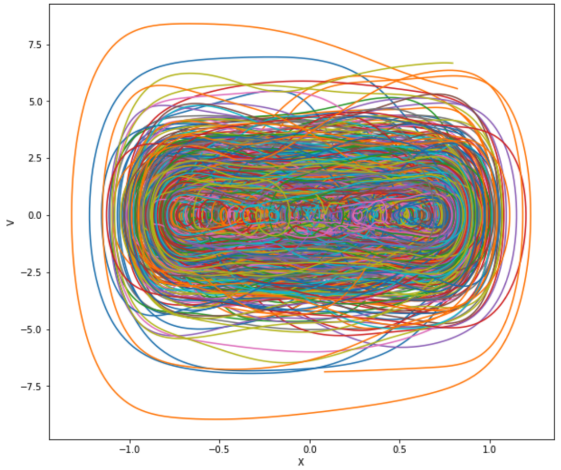
  - The term $A\mathbf{F}(A^+\mathbf{Z}(t))$ can be computed and stored in the linear case and approximated in the nonlinear case.

Nonlinear reduction and Auto-encoder

- Phase-space: PIC code (left), Reduction of PIC (middle), Reduced model (right)







- It doesn't work. Why ?

Nonlinear assumption

For nonlinear transport equation, we can assume that

$$\mathbf{Y}(t) \approx G(\mathbf{Z}(t)), \quad \mathbf{Y}(t) \in \mathbb{R}^N, \mathbf{Z}(t) \in \mathbb{R}^K$$

- Idea: replace SVD by Deep - Auto encoder.
- In practice: light Multi-Percetron architecture (fully connected by packets).

$$\begin{cases} E^x_{\theta_{e_x}}(\mathbf{x}) = \bar{\mathbf{x}} \\ E^v_{\theta_{e_v}}(\mathbf{v}) = \bar{\mathbf{v}} \end{cases}, \begin{cases} D^x_{\theta_{e_x}}(\bar{\mathbf{x}}) = \mathbf{x} \\ D^v_{\theta_{e_v}}(\bar{\mathbf{v}}) = \mathbf{v} \end{cases}$$

Hamiltonian reduced model

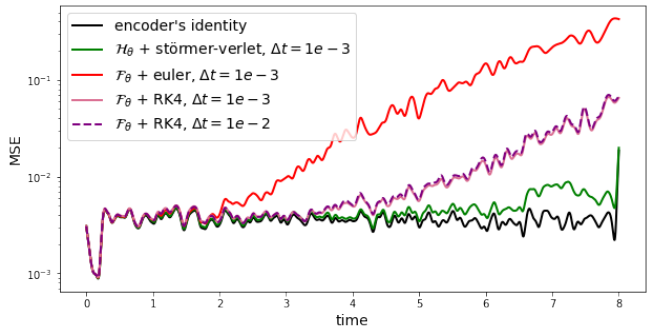
- How to construct the reduced model:
  - **Projection**: as in the linear case. **Drawbacks**: use the Jacobian of E, D which can be larges. We must construct the reduced flux by approximation.
  - **Learning**: we learn the reduce model using reduced trajectories.
- Two ways proposed:
  - Baseline:

$$\min_{\theta_x,\theta_v} \left\| \frac{\bar{\mathbf{x}}^{n+1} - \bar{\mathbf{x}}^{n-1}}{2\Delta t} - F_x(\bar{\mathbf{x}}^n, \bar{\mathbf{v}}^n) + \frac{\bar{\mathbf{v}}^{n+1} - \bar{\mathbf{v}}^{n-1}}{2\Delta t} - F_v(\bar{\mathbf{x}}^n, \bar{\mathbf{v}}^n) \right\|_2^2$$

  - Hamiltonian form (S. Greydanus and al 2019):

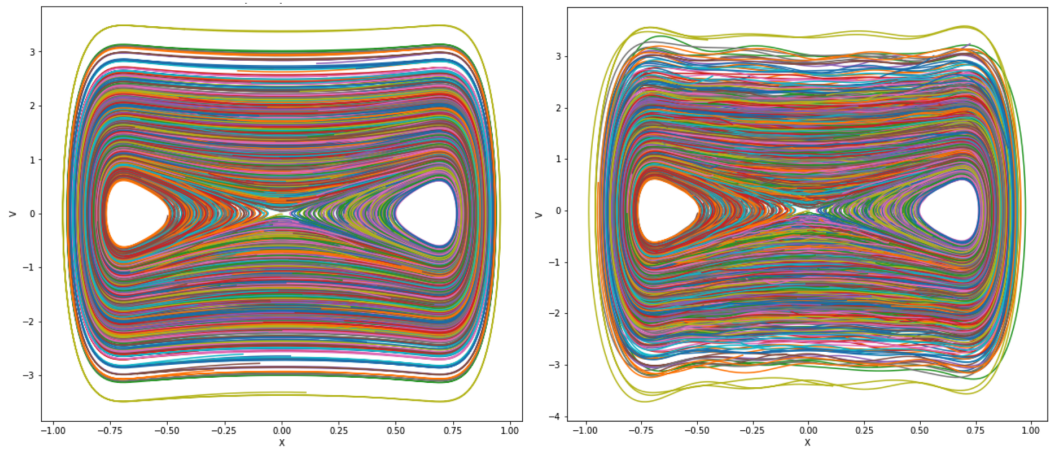
$$\min_{\theta_x,\theta_v} \left\| \frac{\bar{\mathbf{x}}^{n+1} - \bar{\mathbf{x}}^{n-1}}{2\Delta t} - \frac{\partial H_{\theta_v}(\mathbf{v}^n)}{\partial \mathbf{v}} + \frac{\bar{\mathbf{v}}^{n+1} - \bar{\mathbf{v}}^{n-1}}{2\Delta t} + \frac{\partial H_{\theta_x}(\mathbf{x}^n)}{\partial \mathbf{x}} \right\|_2^2$$

- **Advantage**: with good numerical schemes (symplectic integrators) we can assure some stability with Hamiltonian ODE.

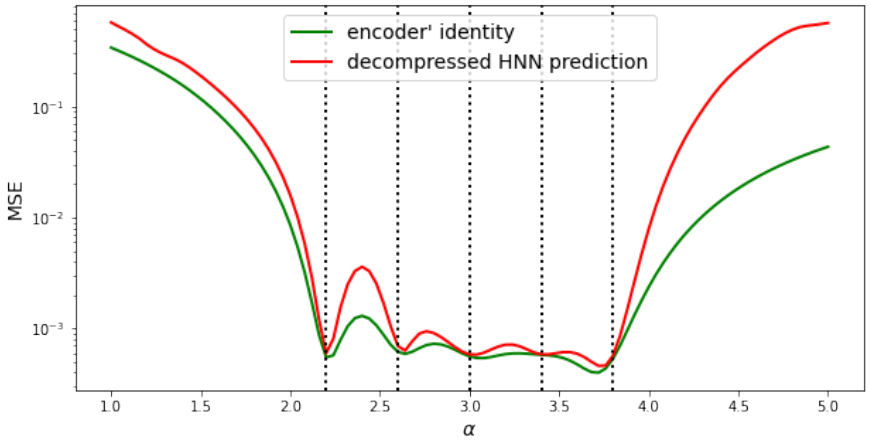


Results

- Results for the reduced model for one trajectory:



- Learning with different initial conditions (varying α):



- **Current step**: varying randomly β and α in the learning step.

Conclusion

Conclusion

- Auto-encoders gives promizing results to construct reduced models for nonlinear transport equation (R Maulik and al 2020, K. Lee and K. Calberg JCP 2022)
- Light Auto-encoder allows to apply reduction for PIC code.
- HNN reduced models allows to ensure some stability.
- General methods for PIC codes.

Next step

- Extend the results on large data set.
- Investigate permutation-invariant neural networks like Transformers or DeepSets.

Project

PRCI ANR project with Max Planck of Plasma Physics + one INRIA PhD on reduced models for Vlasov equation using deep learning:

- ROM approach for more complex problems,
- new reduced models in collisional limit (Léo talk) and strong oscillatory limit,
- space time adaptive modeling.

Two positions of post-docs (two years) in 2022-2023.