# A neural-network fluid closure for the Euler-Poisson system

Laurent Navoret

IRMA, Université de Strasbourg and INRIA Nancy, Tonus

Joint work with Léo Bois, Emmanuel Franck, Vincent Vigon

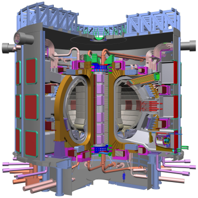
Université de Strasbourg
Tuesday 22 June 2021

# Plan

## Plasma model

**Physical context:**

- **Plasma** = gaz made of charged particles (ions and electrons)
- Fusion by magnetic confinement in a tokamak



**Different description**

- Kinetic description for collisionless plasma
  distribution function $f(x, v, t)$, with $x \in [0, L]$, $v \in \mathbb{R}$, $t \geqslant 0$
- Fluid description for collisional plasma
  density $\rho(x, t)$, velocity $u(x, t)$, temperature $T(x, t)$

→ Knudsen number $\varepsilon$: mean free path between two collisions / L

→ fluid description are cheaper

→ extend the range of validity of fluid models to weakly collisional plasma

**One-dimensionnal Vlasov-Poisson model on $[0, L]$:**

$$\partial_t f + v\partial_x f - E\partial_v f = Q(f)$$

$$E = -\partial_x \phi, \quad -\partial_{xx}\phi = \rho - \frac{1}{L}\int_0^L \rho(t,x)\,dx$$

+ spatial periodic boundary conditions

→ $f(x, v, t)$: distribution function
→ $E(x, t)$: electric field
→ $\phi(x, t)$: electric potential

BGK collision operator: $Q(f) = \frac{1}{\varepsilon}(M(f) - f)$

- relaxation toward a Maxwellian: $M(f)(x, v, t) = \frac{\rho(x,t)}{\sqrt{2\pi T(x,t)}}\, e^{-\frac{(v-u(x,t))^2}{2T(x,t)}}$
- $\rho$, $u$, $T$: moments of the distribution function $f$

[density]

$$\rho(x,t) = \int_{\mathbb{R}} f(x,v,t)dv$$

[pressure]

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

[momentum]

$$\rho(x,t)u(x,t) = \int_{\mathbb{R}} f(x,v,t)v dv$$

[temperature]

$$\rho(x,t)T(x,t) = p(x,t)$$

**One-dimensionnal Vlasov-Poisson model on $[0, L]$:**

$$\partial_t f + v\partial_x f - E\partial_v f = Q(f)$$

$$E = -\partial_x \phi, \quad -\partial_{xx}\phi = \rho - \frac{1}{L}\int_0^L \rho(t,x)\,dx$$

+ spatial periodic boundary conditions

→ $f(x, v, t)$: distribution function
→ $E(x, t)$: electric field
→ $\phi(x, t)$: electric potential

BGK collision operator: $Q(f) = \frac{1}{\varepsilon}(M(f) - f)$

- relaxation toward a Maxwellian: $M(f)(x, v, t) = \frac{\rho(x,t)}{\sqrt{2\pi T(x,t)}}\, e^{-\frac{(v-u(x,t))^2}{2T(x,t)}}$
- $\rho$, $u$, $T$: moments of the distribution function $f$
- conservation of mass, momentum, energy: $\int_{\mathbb{R}} Q(f) \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} dv = 0$

**Three first moments**

$$\begin{bmatrix} \rho(x,t) \\ \rho u(x,t) \\ w(x,t) \end{bmatrix} = \int_{\mathbb{R}} f(x,v,t) \begin{bmatrix} 1 \\ v \\ v^2/2 \end{bmatrix} dv$$

$w$: energy
$w = \rho u^2/2 + p/2$

**Integrate the Vlasov equation against** $(1, v, v^2/2)^T$

$$\int_{\mathbb{R}} (\partial_t f + \ v\partial_x f - \ E\partial_v f) \begin{bmatrix} 1 \\ v \\ v^2/2 \end{bmatrix} dv = 0$$

**Fluid equation:**

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = -E\rho \\ \partial_t w + \partial_x(wu + pu + q) = -E\rho u \end{cases}$$

Fluid equation:

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = -E\rho \\ \partial_t w + \partial_x(wu + pu + q) = -E\rho u \end{cases}$$

→ heat flux: $q(x,t) = \int_{\mathbb{R}} \frac{1}{2} f(x,v,t)(v - u(x,t))^3 dv$

→ system not closed

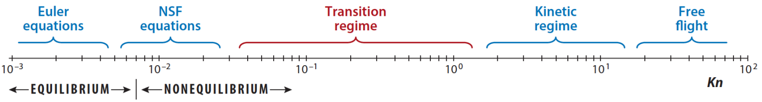
→ **Closure: expression of $q$ as a function of the other moments**

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

→ **first possibilities: from Chapman-Enskog expansion in $O(\varepsilon)$**

- [Euler closure] $f = M(f) + O(\varepsilon) \quad \Rightarrow \quad \hat{q} = 0$
- [Navier-Stokes closure] $f = M(f) + \varepsilon g + O(\varepsilon^2) \quad \Rightarrow \quad \hat{q} = -\frac{3}{2}\varepsilon\, p\, \partial_x T$

**Validity model** [Torrilhon, 2016]




**Extend range of validity of fluid models**

- higher order terms in Chapman-Enskog
  - → Burnett and Super-Burnett systems
  - → ill-posed systems
- higher order moments
  - → Grad 13 model
  - → reduced hyperbolicity region
- higher order moments based on entropic closure
  - → Levermore 14 moment
  - → entropic minimization not always well-posed

**Add specific kinetic**

- Landau damping effect
  - → phase mixing
  - → damping of spatial modes ⇒ damping of electrostatic energy
- Hammett-Perkins closure [Hammett, Perkins 90, 92]
  - → fitting dispersion relation of the linearized equation
  - → $q$ as Hilbert transform of the temperature

$$\hat{q}_k = -in_0\sqrt{\frac{8}{\pi}}i \,\mathsf{sign}(k)\hat{T}_k$$

  - → non-local closure
- Many extensions in the case of magnetized plasmas

**Neural network closures:**

- turbulent flows [Zhou et al. 2020]
- higher moments for neutral fluid [Han et al, 2019]
- learning known plasma closures [Ma et al. 2020] [Maulik et al. 2020]

**Goal : insert a data driven closures into fluid solvers for** $\varepsilon \in [0.01, 1]$

$\rightarrow$ Off-line phase: supervised learning from kinetic simulations

$\rightarrow$ On-line phase: compute the closure at each time step of the fluid solver

# Plan

**Non-local Neural Network closure**

$$X = (\varepsilon, \rho, u, T) \in (\mathbb{R}^{N_x})^4 \quad \longrightarrow \quad \hat{q} = C_{\hat{\theta}}(\varepsilon, \rho, u, T) \in (\mathbb{R}^{N_x})^4$$

$\hat{\theta} \in \Theta$: set of parameters

**Training: solve the optimization problem (gradient method)**

$$\hat{\theta} = \underset{\theta \in \Theta}{\text{argmin}} \ \text{Loss}(\theta)$$

with loss function:

$$\text{Loss}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(X;q) \in \mathcal{D}} \frac{1}{N_x} \sum_{i=1}^{N_x} |C_\theta(X)_i - q_i|,$$

$\mathcal{D}$ data set
$C_\theta(X)$: prediction of the neural network
$q$: true heat flux

→ Define the architecture of the network
→ Generate data

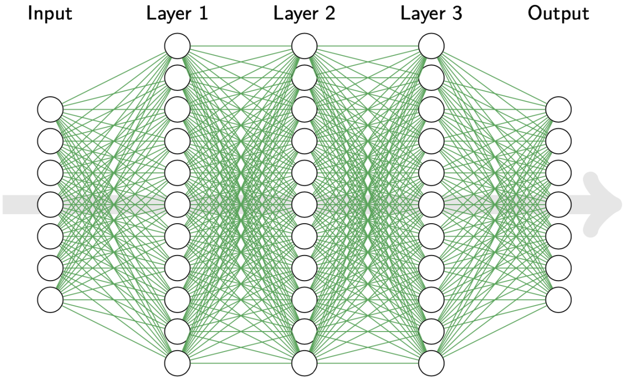
**Neural Network**

- Layer: linear combination followed by a non-linear activation function

$$Y^{(0)} = X, \quad Y^{(p+1)} = \sigma\left(W^{(p)} Y^{(p)}\right)$$

$W^{(p)}$: weights matrices
$\sigma$: activation function

- Example: fully connected neural network

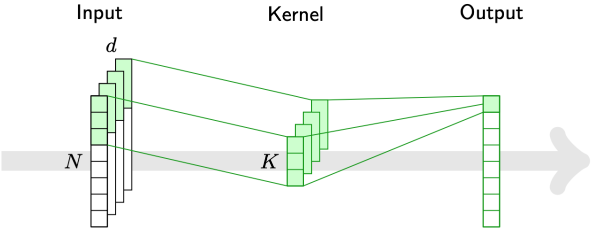


**Convolutional neural network**

→ sparse neural networks
→ very efficient for structured data (image, signals)
→ each layer: several 1D convolutions with small kernels followed by activation functions

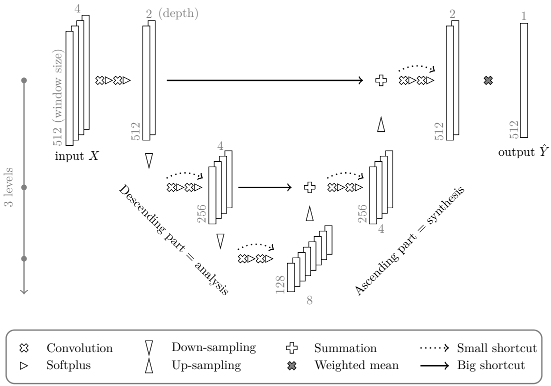
input: $X$ of shape $(N, d)$
output: $Y$ of shape $(N, d')$
kernel: $K$ of shape $(p, d, d')$ size $p$
activation: $\sigma$

$$Y_{i,k} = \sigma\left(\sum_{j=1}^{d}\sum_{di=1}^{p} X_{i+di,j} K_{di,j,k}\right)$$



**One-dimensional V-net architecture** [Ronneberger et al., 2015] [Milletari et al., 2016]

- multi-scale analysis (like in wavelet analysis)
- based on up-samplings and dow-samplings
  - → down-sampling: decrease the size of the signals / increase the number of channels
  - → up-samping: increase the size of the signals / decrease the number of channels
- shortcut: add the input to output for accelarating the training process



**Choice of the hyperparameters:**

| Hyper-parameter | Value |
|---|---|
| size of the input window ($N$) | 512 |
| number of levels ($\ell$) | 5 |
| depth ($d$) | 4 |
| size of the kernels ($p$) | 11 |
| activation function | softplus |

softplus: $\sigma(x) = \ln(1 + \exp(x))$

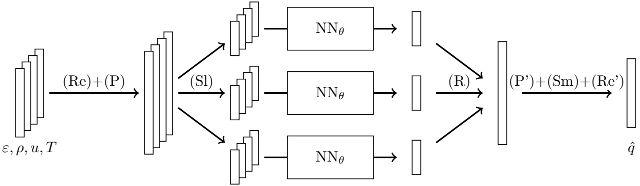
$\rightarrow$ $15$ layers

**Neural network parameters to learn**

- $O(2^\ell d^2 pN)$
- Here: $161\,937$ parameters

For learning and flexibility:

1. Resampling to a given resolution $N'_x$ and preprocessing (standardization of the data)
2. Slicing into overlapping "windows" of size $N = 512$
3. Neural network
4. Reconstructing
5. Post-processing, smoothing and resampling

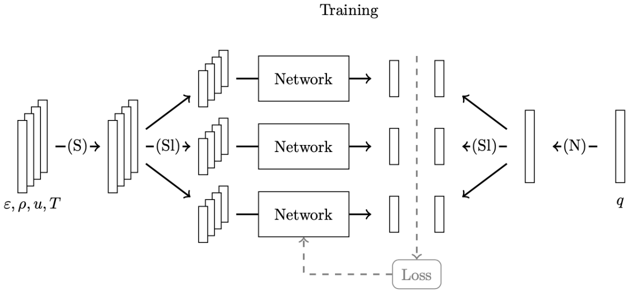



$$C_\theta : \quad X \xmapsto{\text{(Re)+(P)}} X^{(P)} \xmapsto{\text{(Sl)}} (X_j^{(P)})_j \xmapsto{(\mathrm{NN}_\theta)} (\hat{Y}_j^{(P)})_j \xmapsto{\text{(R)}} \hat{Y}^{(P)} \xmapsto{\text{(P')+(Sm)+(Re)}} \hat{Y}.$$

**To train the neural network:**

→ do the same Resampling and preprocessing

→ do the same Slicing



**Data generation by kinetic solver: for each simulation**

- initialization: $f_0(x) = M(\rho, u, T)$, with $\rho$, $u$ and $T$ as Fourier series :

$$\alpha \times \left( \frac{a_0}{2} + 0.5 \sum_{n=1}^{20} (a_n \cos(nx) + b_n \sin(nx)) \right), \quad x \in [0, 2\pi].$$

  $a_n, b_n$: random in $[-1/n, 1/n]$ for $n \geqslant 1$

  density; $a_0/2 = 1$, $\alpha = 1$
  temperature: $a_0/2 = 1$, $\alpha \in [0.1, 1]$ random
  momentum: $a_0/2 \in [-1, 1]$ random, $\alpha$ s.t. Mach number $\in [10^{-4}, 5.10^{-1}]$

- $\varepsilon \in [0.01, 1]$: non-uniform distribution
- $20$ recording time $t_1, t_2, \ldots, t_{20} \in [0.1, 2]$

→ discretization parameters: $N_x = 1024$, $N_v = 141$

→ Finite Volume in space / Finile Element method in velocity [Helluy et al., 2014]

→ $20 \times$ **500** $= 10\,000$ different spatial data for training
→ $20 \times$ **500** $= 10\,000$ different spatial data for validation

**Data generation by kinetic solver: for each simulation**

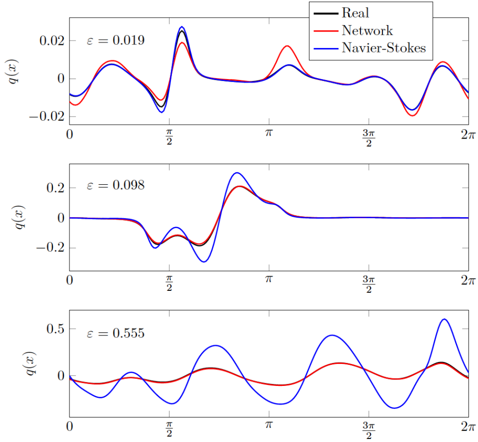
**Output normalization**

- avoid too small values of the heat flux prediction
- normalisation with the Navier-Stokes heat flux:

$$q^{k_0}_{\text{norm}} = \begin{cases} q^{k_0} \times \dfrac{\theta}{q^{k_0}_{NS}}, & \text{if } 0 < q^{k_0}_{NS} \leqslant \theta, \\ \\ q^{k_0}, & \text{otherwise,} \end{cases}$$

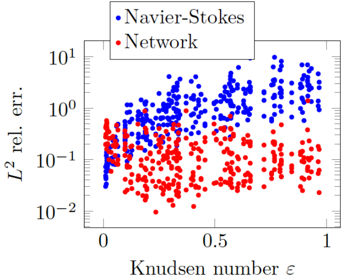
# Plan

**Examples from the validation set:**




→ For large $\varepsilon$: neural network closure better than Navier-Stokes one $\approx 10^{-1}$

$L^2$ **relavtive error on the validation set:**




→ For large $\varepsilon$: neural network closure better than Navier-Stokes one
→ relative error independent of the Knudsen number $\approx 10^{-1}$

**Fluid+Network:** $\hat{q} = C_\theta(\varepsilon, \rho, u, T)$
**solved with explicit finite volume scheme (Lax-Friedrichs flux)**
**compared with :**

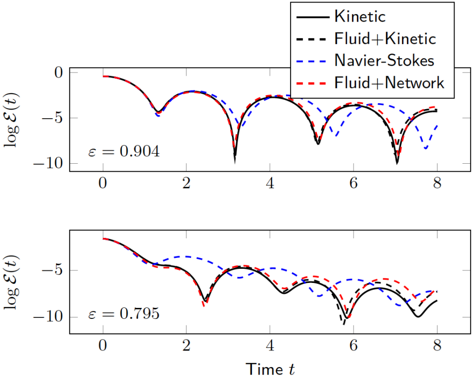
- Kinetic
- Fluid+Kinetic ($\hat{q} = q$)
- Fluid+Navier-Stokes ($\hat{q} = -\frac{3}{2}\varepsilon p\, \partial_x T$)

→ difference between Kinetic and Fluid+Kinetic results from numerical errors

→ cannot expect better than Fluid+Kinetic

# Fluid model with neural network

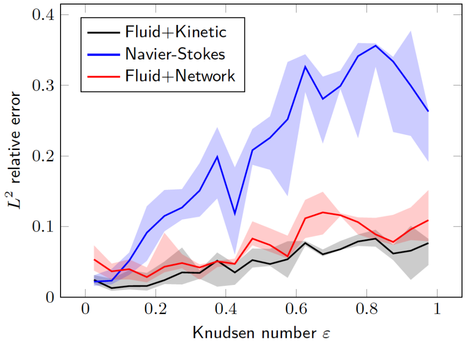
Time evolution of electric energy: $\mathcal{E}(t) = \int_{[0,L]} E^2(x,t)dx$




→ for large $\varepsilon$: good results for Fluid+Network

# Fluid model with neural network

$L^2$ error on density, momentum, energy on $200$ simulations



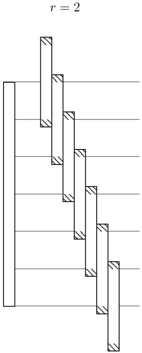

→ relative error below $0.2$

→ Fluid+Network errors vary like the Fluid+Kinetic one

**Stability**

→ no guarantee of stability

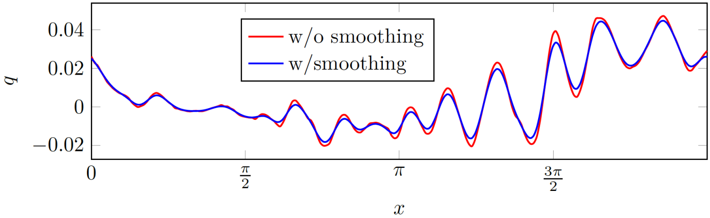
→ instabilities triggered by irregular reconstruction of the heat flux due to slicing




**Smoothing of the output**

$$\tilde{q}(x) = \int_{-3\sigma}^{3\sigma} q(x+t)w(t)\,dt.$$

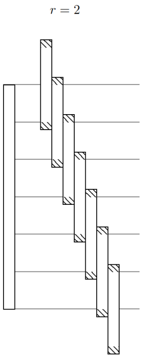
$w$: Gaussian kernel with standard deviation $\sigma$



**Stability**

→ no guarantee of stability

→ instabilities triggered by irregular reconstruction of the heat flux due to slicing

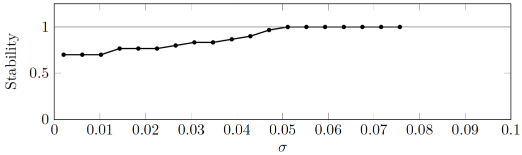



**Smoothing of the output**

$$\tilde{q}(x) = \int_{-3\sigma}^{3\sigma} q(x+t)w(t)\,dt.$$

$w$: Gaussian kernel with standard deviation $\sigma$

**Numerical results: proportion of simulations reaching final time**



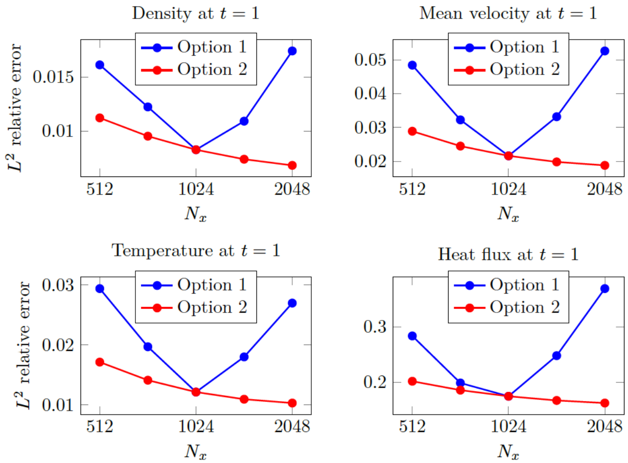

→ $\sigma = 0.06$ leads to stable numerical simulations

# Convergence

**Two options for considering refined grids**

→ option 1: use slicing

→ option 2: use downsampling to the refinement used for learning




→ keep close to the data used in training set (same resolution)

## CPU cost

**Typical simulation time for** $T_f = 8$ **with** $N_x = 512$ **and** $N_v = 101$**:**

| | |
|---|---|
| Kinetique | 70 sec |
| Fluid+Kinetic | 78 sec |
| Fluid+Network | 74 sec |
| Navier-Stokes | 3 sec |

→ Fluid+Network not competitive
→ **but** non optimal implementation ( CPU/GPU communications)
→ **but** would be better in higher dimension

| | |
|---|---|
| V-Net 1D | $O(2^\ell d^2 p N_x)$ |
| V-Net 2D | $O(\ell d^2 p^2 N_x^2)$ |
| V-Net 3D | $O(d^2 p^3 N_x^3)$ |
| Kinetic $m$D | $O(N_v^m N_x^m)$ |

# Plan

**Fluid Neural Network closure**

- based on a V-net architecture
- Good results in the range $\varepsilon \in [0.01, 1]$
- Stability and convergence properties are numerically observed

**Perpectives:**

- Extension to dimension 2 and optimization
- Add a magnetic field
- Closure for models with more moments
- stability with reinforcement learning ?