Neural networks and reduced models for plasmas

Laurent Navoret

IRMA, Université de Strasbourg and INRIA Nancy-Grand Est, Tonus

Joint work with Léo Bois, Nicolas Crouseilles, Emmanuel Franck, Guillaume Steimer, Vincent Vigon

GDR Emili, Ecole Polytechnique
Wednesday 27 October 2021
Context

Plasmas dynamics

- distribution function $f(x,v,t)$ in phase space (high-dimension)
- multi-scale (mass ratio, quasineutrality, collisions, anisotropy)

→ full simulations: time and memory demanding
→ need for reduced models for real-time simulations for diagnostics or control

Goal

- reduced models based on neural networks
- study of the stability and accuracy once used numerically
- one-dimensional test
Plan

1. Neural networks
2. Fluid closure
3. Particle reduced model
4. Conclusion
Plan

1. Neural networks
2. Fluid closure
3. Particle reduced model
4. Conclusion
Neural Network: parametrized function

\[ \mathcal{F}_\theta(X) \]

approximation of a real unknown function \( \mathcal{F}(X) \) (ex: physical quantity)

Fit data (supervised learning)

\[ \hat{\theta} = \arg\min_{\theta \in \Theta} \sum_i \text{distance} \left( \mathcal{F}(X_i), \mathcal{F}_{\theta}(X_i) \right) \]

data: \((X_i, \mathcal{F}(X_i))_i\)
\rightarrow optimization algorithm

Difficulty
\rightarrow X: a vector of large dimension (ex: image from simulations)
Neural Network: parametrized function

\[ Y = \mathcal{F}_\theta(X) = \sigma \left( W^{(d)} \sigma \left( W^{(d-1)} \sigma \left( \ldots \sigma \left( W^{(0)} X \right) \right) \right) \right) \]

- succession of layers
- one layer: linear combination followed by a non-linear activation function

\[ Y^{(p+1)} = \sigma \left( W^{(p)} Y^{(p)} \right) \]

\( W^{(p)} \): weights matrices
\( \sigma(a) = \max(a, 0) \)

- parameters: \( \theta = (W^{(p)})_p \)

Example: fully connected neural network
A lot of applications in signal analysis

- image classification
- image segmentation
- speech recognition

→ based on GPU implementation
→ large amount of data
→ user-friendly library (keras)
→ mathematical properties (separation, symmetries, multi-scale) [Mallat, 2015]

**Used for the construction of reduced models in physics: 2 examples**

1. Fluid closure
2. Particle reduced model

→ specific architecture
→ specific data processing
Plan

1. Neural networks
2. Fluid closure
3. Particle reduced model
4. Conclusion
Different description

- Kinetic description for collisionless plasma \((\varepsilon > 1)\)
  distribution function \(f(x, v, t)\), with \(x \in [0, L], v \in \mathbb{R}, t \geq 0\)

- Fluid description for collisional plasma \((\varepsilon < 10^{-2})\)
  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
Kinetic model

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

$$\partial_t f + v \partial_x f - E \partial_v f = \frac{1}{\varepsilon} \left( M(f) - f \right)$$

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

+ spatial periodic boundary conditions

BGK collision operator

- relaxation to 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 $f$

<table>
<thead>
<tr>
<th>Density</th>
<th>Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho(x,t) = \int_{\mathbb{R}} f(x,v,t) , dv$</td>
<td>$p(x,t) = \int_{\mathbb{R}} f(x,v,t) (v-u(x,t))^2 , dv$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Momentum</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho(x,t)u(x,t) = \int_{\mathbb{R}} f(x,v,t) , dv$</td>
<td>$\rho(x,t)T(x,t) = p(x,t)$</td>
</tr>
</tbody>
</table>
From kinetic to fluid

Fluid equations satisfied by the moments \((\rho, \rho u, w)\):

\[
\begin{align*}
\partial_t \rho + \partial_x (\rho u) &= 0 \\
\partial_t (\rho u) + \partial_x (\rho u^2 + p) &= -E \rho \\
\partial_t w + \partial_x (wu + pu + q) &= -E \rho u
\end{align*}
\]

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

→ 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

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

→ first possibilities

- [Euler closure] \(f = M(f) + O(\varepsilon)\) \(\Rightarrow \hat{q} = 0\)
- [Navier-Stokes closure] \(f = M(f) + \varepsilon g + O(\varepsilon^2)\) \(\Rightarrow \hat{q} = -\frac{3}{2} \varepsilon p \partial_x T\)
Validity model [Torrilhon, 2016]

Extend range of validity of fluid models
- higher order terms in Chapman-Enskog
- higher order moments (Grad 13 model)
- higher order moments based on entropic closure (Levermore 14 moment)
  → ill-posed systems

Add specific kinetic
- Landau damping effect
- Hammett-Perkins closure [Hammett, Perkins 90, 92]
  → fitting dispersion relation of the linearized equation
  → \( q \) as Hilbert transform of the temperature

\[
\hat{q}_k = -in_0 \sqrt{\frac{8}{\pi}} i \ \text{sign}(k) \hat{T}_k
\]
  → 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 $\varepsilon \in [0.01, 1]$

→ Off-line phase: supervised learning from kinetic simulations
→ On-line phase: compute the closure at each time step of the fluid solver
Non-local Neural Network closure

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

\( \hat{\theta} \in \Theta \): set of parameters

Training: solve the optimization problem (gradient method)

\[ \hat{\theta} = \arg \min_{\theta \in \Theta} \frac{1}{|D|} \sum_{(X;q) \in D} \frac{1}{N_x} \sum_{i=1}^{N_x} |C_{\theta}(X)_i - q_i| \]

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

→ Define the architecture of the network
→ Generate data
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 $p$

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

- scalar product with the kernel: measure of similarity
Architecture

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 down-samplings
  → down-sampling: decrease the size of the signals / increase the number of channels
  → up-sampling: increase the size of the signals / decrease the number of channels
- shortcut: add the input to output for accelerating the training process
Choice of the hyperparameters:

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>size of the input window ((N))</td>
<td>512</td>
</tr>
<tr>
<td>number of levels ((\ell))</td>
<td>5</td>
</tr>
<tr>
<td>depth ((d))</td>
<td>4</td>
</tr>
<tr>
<td>size of the kernels ((p))</td>
<td>11</td>
</tr>
<tr>
<td>activation function</td>
<td>softplus</td>
</tr>
</tbody>
</table>

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

\(\rightarrow 15\) layers

Neural network parameters to learn

- \(O(2^\ell d^2 p N)\)
- Here: 161,937 parameters
Full closure

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

$$C_{\theta} : X \xrightarrow{(\text{Re})+(P)} X^{(P)} \xrightarrow{(\text{SI})} (X_j^{(P)})_j \xrightarrow{(\text{NN}_{\theta})} (\hat{Y}_j^{(P)})_j \xrightarrow{(R)} \hat{Y}^{(P)} \xrightarrow{(P')+(\text{Sm})+(\text{Re'})} \hat{Y}.$$
Data generation

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

- \( \varepsilon \in [0.01, 1] \): 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 / Finite Element method in velocity [Helluy et al., 2014]

\[
\begin{align*}
20 \times 500 &= 10 000 \text{ different spatial data for training} \\
20 \times 500 &= 10 000 \text{ different spatial data for validation}
\end{align*}
\]
Data generation

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:

\[
q_{\text{norm}}^{k_0} = \begin{cases} 
q_0^{k_0} \times \frac{\theta}{q_{NS}^{k_0}}, & \text{if } 0 < q_{NS}^{k_0} \leq \theta, \\
q_0^{k_0}, & \text{otherwise},
\end{cases}
\]
Learning results

Examples from the validation set:

For large $\varepsilon$: neural network closure better than Navier-Stokes one

$L^2$ relative error on the validation set:

$\Rightarrow$ relative error independent of the Knudsen number $\approx 10^{-1}$
Fluid model with neural network

\[ \hat{q} = C_\theta(\varepsilon, \rho, u, T) \]

Fluid solver + Network compared with:

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

→ for large \( \varepsilon \): good results for Fluid + Network
→ error on Fluid + Kinetic due to numerical approximations
Fluid model with neural network

Fluid solver + Network: \( \hat{q} = C_\theta(\varepsilon, \rho, u, T) \) compared with:

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

\( L^2 \) error on density, momentum, energy on 200 simulations

\( \rightarrow \) cannot expect better than Fluid + Kinetic
\( \rightarrow \) relative error \( \approx 0.2 \)
\( \rightarrow \) 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 \( \sigma \)

Numerical results: proportion of simulations reaching final time

\[ \sigma = 0.06 \] leads to stable numerical simulations
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)
Plan

1. Neural networks
2. Fluid closure
3. Particle reduced model
4. Conclusion
Particle In Cell method: \( N \) macro-particles \((X_j, V_j) \in \mathbb{R}^2\)

\[
\forall j \in \{1, \ldots, N\}, \quad \frac{dX_j}{dt} = V_j \\
\frac{dV_j}{dt} = \frac{q}{m} (E_h + E_{\text{ext}})(X_j)
\]

\( E_h = -\nabla \phi_h \): approximated self-consistent electric field
\( E_{\text{ext}} = -\nabla \phi_{\text{ext}} \): external electric field

→ Hamiltonian dynamics
→ costly numerical simulations

→ Goal: describe the dynamics locally around a given trajectory with \( K \ll N \) reduced “particles”
1. **Find out reduced variables:**

\[ u = (X, V) \in \mathbb{R}^{2N} \quad \rightarrow \quad \bar{u} = (\bar{X}, \bar{V}) \in \mathbb{R}^{2K} \quad \rightarrow \quad u = (X, V) \in \mathbb{R}^{2N} \]

\( \rightarrow \) fast compression / decompression

**Linear reduction:** \( \bar{u} = Au \)

- Proper orthogonal decomposition (POD)
- Proper symplectic decomposition (PSD) [Tyranowski, Krauss, 2019]
- analytical dynamics on the reduced variables

\( \rightarrow \) valid on linear regime

\( \rightarrow \) but electric field computed from original variables

\( \rightarrow \) back and forth between reduced and original variables

**Non-linear reduction:** neural networks

\( \rightarrow \) Auto-encoder architecture
Auto-encoder

**Neural network Encoder/Decoder**

\[ \tilde{u} = \mathcal{F}_\theta(u) \]: encoder

\[ u = \mathcal{G}_\theta(\tilde{u}) \]: decoder

**Learning:**

\[ \hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N_d} \| u_i - \mathcal{G}_\theta(\mathcal{F}_\theta(u_i)) \|^2 \]

→ \((u_i)\): extracted from numerical simulations at different times

→ \( \mathcal{G}_\theta \circ \mathcal{F}_\theta \approx \text{Id} \) on the data

**Architecture:**

→ pooling/unpooling

→ light architecture to reduce the number of parameters
Numerical results

Test-case: one numerical simulation with $N = 1000$ particles

**PSD reduction $K = 20$**

$\rightarrow$ error $\approx 10^{-2}$

**Auto-encoder reduction $K = 7$**

$\rightarrow$ error $\approx 10^{-4}$
Learning the reduced dynamics

2. Determine the dynamics of the reduced variables:

\[ \forall j \in \{1, \ldots, K\}, \quad \frac{d\bar{X}_k}{dt} = \nabla_{\bar{V}} H_{\theta}(\bar{X}, \bar{V}) \]

\[ \frac{d\bar{V}_k}{dt} = -\nabla_{\bar{X}} H_{\theta}(\bar{X}, \bar{V}) \]

Neural network Hamiltonian function: \( H_{\theta}(\bar{X}, \bar{V}) \)

\( \rightarrow \) assumption: separable Hamiltonian

\( \rightarrow \) take implicitly into account the electric force

\( \rightarrow \) ensure large time stability of the reduced dynamics

Learning

\[ \hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{N_d} \left\| \frac{\bar{X}_i + \Delta t - \bar{X}_i}{\Delta t} - \nabla_{\bar{V}} H_{\theta}(\bar{X}_i, \bar{V}_i) \right\| + \left\| \frac{\bar{V}_i + \Delta t - \bar{V}_i}{\Delta t} + \nabla_{\bar{X}} H_{\theta}(\bar{X}_i, \bar{V}_i) \right\| \]

data: numerical simulations at different times \((\bar{X}, \bar{V}, \bar{X}+\Delta t, \bar{V}+\Delta t)_i\)

Architecture fully connected
Numerical results

Initial condition:

\[ f(x, v, t) = 1_{[-0.5, 0.5]}(x) \exp(-v^2/2) \]

Numerical method:

1. compression
2. simulation of the reduced model
3. decompression

Phase portrait

Reference (PIC)  PSD reduction (K=35)  Neural Network reduction (K=8)
Numerical results

Initial condition:

\[ f(x, v, t) = 1_{[-0.5, 0.5]}(x) \exp(-v^2/2) \]

Numerical method:

1. compression
2. simulation of the reduced model
3. decompression

→ Hamiltonian reduced dynamics: better control of the numerical error
Numerical results

Initial conditions:

\[ f(x, v, t) = C x^\alpha (1 - x)^{3/2} \exp(-v^2/2) \]

Goal: learn the dynamics for \( \alpha \in [2.2, 3.6] \)

Data: dynamics of \( N = 10^4 \) particles with \( \alpha \in \{2.2, 2.6, 3, 3.4, 3.6\} \)

Error for the whole strategy

\[ \rightarrow \text{reduced method: 25 times faster} \]
Plan

1. Neural networks
2. Fluid closure
3. Particle reduced model
4. Conclusion
Conclusion

Construction of reduced models

- Fluid closure based on a V-net architecture supervised learning
- Particle reduced dynamics based on auto-encoder architecture semi-supervised learning
- Stability properties observed/ensured

Perspectives:

- Extension to dimension 2 or 3
- Add a magnetic field
- Use in real applications

Ongoing ANR project with Max-Planck Institut für Plasma Physics (Garching).