Model order reduction method for Hamiltonian dynamics using deep learning

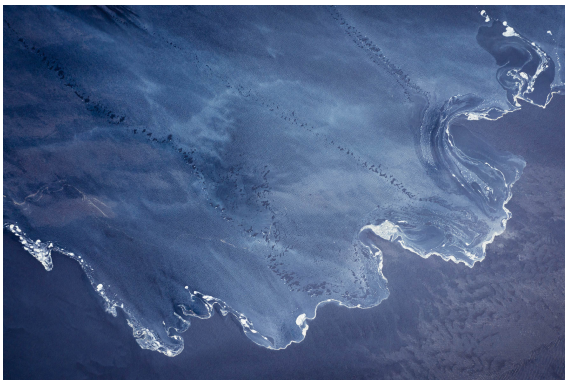
Guillaume Steimer

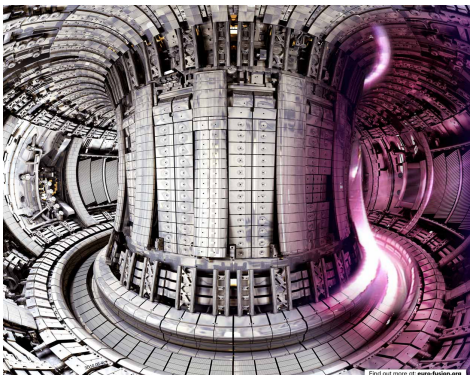
supervised by

Raphaël Côte, Emmanuel Franck, Laurent Navoret and Vincent Vigon

IRMA, University of Strasbourg, September 12, 2025

Motivation





Motivation

- Hamiltonian systems possess a **geometric structure**,
- enforce the conservation of the energy (and possibly other invariants),
- offer guarantees for long-time stability and physical relevance.

Motivation

- Hamiltonian systems possess a **geometric structure**,
- enforce the conservation of the energy (and possibly other invariants),
- offer guarantees for long-time stability and physical relevance.

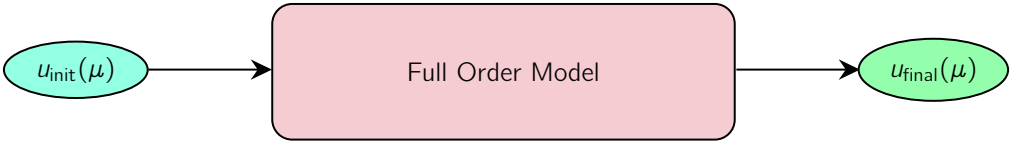
- Full order models (FOMs) are high-dimensional **parametrized** ODEs derived from PDEs,
- parameters can be geometric, physical, the initial condition, etc.,
- Hamiltonian structure preserved with symplectic (implicit) methods.

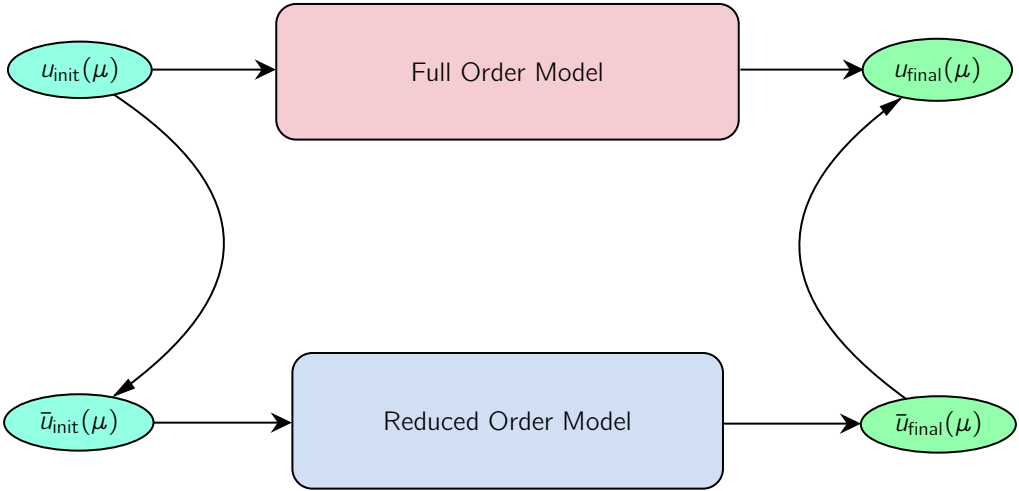
Motivation

- Hamiltonian systems possess a **geometric structure**,
- enforce the conservation of the energy (and possibly other invariants),
- offer guarantees for long-time stability and physical relevance.

- Full order models (FOMs) are high-dimensional **parametrized** ODEs derived from PDEs,
- parameters can be geometric, physical, the initial condition, etc.,
- Hamiltonian structure preserved with symplectic (implicit) methods.

- In many-query or real-time settings, solvers often fail to scale for fast resolution or repeated evaluations across multiple parameters,
- a solution: build a **Hamiltonian** Reduced Order Model (ROM) trading accuracy for computational efficiency,
- efficient over a given time and parameter domain.


$u_{\text{init}}(\mu)$
Full Order Model
$u_{\text{final}}(\mu)$


$u_{\text{init}}(\mu)$
Full Order Model
$u_{\text{final}}(\mu)$
$\bar{u}_{\text{init}}(\mu)$
Reduced Order Model
$\bar{u}_{\text{final}}(\mu)$

Thesis objective

- Develop new model order reduction techniques,
  - preserving the Hamiltonian structure,
  - with deep learning (neural networks),
- test it against several Hamiltonian models,
  - mesh-based models (wave equation, shallow-water system),
  - particle-based model (Vlasov-Poisson system).

- Evolution of a field $u(x, t; \mu) \in V$ depending on space $x \in \Omega \subset \mathbb{R}^d$, time $t \in \mathcal{T} = [0, T]$ and parameters $\mu \in \Gamma \subset \mathbb{R}^p$ is given by

$$\frac{\partial u}{\partial t} = \mathcal{J}(u) \frac{\delta \mathcal{H}}{\delta u}(u),$$

  with $\delta \mathcal{H} / \delta u$ the functional derivative of $\mathcal{H}$ with respect to u,

- $\mathcal{H} : V \to \mathbb{R}$ is the **Hamiltonian** of the system, often the total energy,

- $\mathcal{J}(u) : V \to V$ is a skew-adjoint operator called the **Poisson structure operator**.

- In the canonical case, we denote $u(x,t;\mu) = (q(x,t;\mu), p(x,t;\mu)) \in \mathbb{R}^{2N}$ the canonical coordinates with generalized coordinates $q(x,t;\mu)$ and conjugate momentum $p(x,t;\mu)$,
- the Poisson structure operator $\mathcal{J}$ becomes

$$\mathcal{J} = \begin{pmatrix} 0 & \mathrm{id} \\ -\mathrm{id} & 0 \end{pmatrix},$$

- and the system rewrites

$$\frac{\partial u}{\partial t} = \mathcal{J}\frac{\delta \mathcal{H}}{\delta u}(u) \iff \begin{cases} \partial_t q = \dfrac{\delta \mathcal{H}}{\delta p}(q,p), \\ \partial_t p = -\dfrac{\delta \mathcal{H}}{\delta q}(q,p). \end{cases}$$

- Example : linear wave equation (dim. $N = 1$) on a periodic domain of length 1,

$$\partial_{tt} q(x, t; \mu) - \mu^2 \partial_{xx} q(x, t; \mu) = 0,$$

with the displacement $q(x, t; \mu)$ and the parametrized propagation speed $\mu \in \Gamma \subset \mathbb{R}$ ($p = 1$),

- Example : linear wave equation (dim. $N = 1$) on a periodic domain of length 1,

$$\partial_{tt} q(x, t; \mu) - \mu^2 \partial_{xx} q(x, t; \mu) = 0,$$

  with the displacement $q(x, t; \mu)$ and the parametrized propagation speed $\mu \in \Gamma \subset \mathbb{R}$ ($p = 1$),
- the Hamiltonian (total energy) is

$$\mathcal{H}[q, p] = \frac{1}{2} \int_0^1 \left(p^2 + \mu^2 (\partial_x q)^2 \right) \, dx.$$

- denoting $p(x, t; \mu) := \partial_t q(x, t; \mu)$, the equation rewrites

$$\begin{cases} \partial_t q(x, t; \mu) = p(x, t; \mu), \\ \partial_t p(x, t; \mu) = \mu^2 \partial_{xx} q(x, t; \mu). \end{cases}$$

- Before any reduction technique is applied, the EDP is semi-discretized,

- Before any reduction technique is applied, the EDP is semi-discretized,
- with finite element or finite differences e.g. on a mesh $(x_i)_{i\in\{1,\dots,N\}}$ of size h of Ω, $u_i(t;\mu) \approx u(x_i, t;\mu)$

$$\partial_x u(x_i, t;\mu) \approx \frac{1}{2h}[u_{i+1}(t;\mu) - u_{i-1}(t;\mu)]\,, \text{etc.},$$

- need to preserve the Hamiltonian structure : $\mathcal{J}_h$ becomes a skew-symmetric matrix (+ Jacobi identity),

- Before any reduction technique is applied, the EDP is semi-discretized,
- with finite element or finite differences e.g. on a mesh $(x_i)_{i\in\{1,\dots,N\}}$ of size h of Ω, $u_i(t;\mu) \approx u(x_i, t;\mu)$

$$\partial_x u(x_i, t;\mu) \approx \frac{1}{2h}[u_{i+1}(t;\mu) - u_{i-1}(t;\mu)]\,, \text{etc.},$$

- need to preserve the Hamiltonian structure : $\mathcal{J}_h$ becomes a skew-symmetric matrix (+ Jacobi identity),
- In the canonical case, we derive a $2N$-dimensional ODE, $N \gg 1$

$$\frac{du_h(t;\mu)}{dt} = \mathcal{J}_h \nabla \mathcal{H}_h(u_h(t;\mu)),\ \mathcal{J}_h = \begin{pmatrix} 0 & I_N \\ -I_N & 0 \end{pmatrix} \in \mathcal{M}_{2N}(\mathbb{R})$$

with $\mathcal{H}_h : \mathbb{R}^{2N} \to \mathbb{R}$ the (semi-discretized) Hamiltonian.

- Full order model = $2N$-dimensional ODE of solution $u(t;\mu) \in \mathbb{R}^{2N}$ and Hamiltonian $\mathcal{H} : \mathbb{R}^{2N} \to \mathbb{R}$

$$\begin{cases} \dfrac{du(t;\mu)}{dt} = \mathcal{J}_{2N} \nabla_u \mathcal{H}(u(t;\mu)), \\ u(0;\mu) = u_{\text{init}}(\mu), \end{cases}$$

  with $\mathcal{J}_{2N} = \begin{pmatrix} 0 & I_N \\ -I_N & 0 \end{pmatrix} \in \mathcal{M}_{2N}(\mathbb{R})$,

- and its flow $\phi_t : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$

$$\phi_t(u_{\text{init}}(\mu)) := u(t;\mu).$$

- Hamiltonian ODEs benefits from numerous properties
  - preservation of the Hamiltonian along the flow

$$\frac{d}{dt}\mathcal{H}(u(t;\mu)) = 0,$$

- Hamiltonian ODEs benefits from numerous properties
  - preservation of the Hamiltonian along the flow

$$\frac{d}{dt}\mathcal{H}(u(t;\mu)) = 0,$$

  - the flow is a symplectic map

$$\left(D\phi_t(u)\right)^T \mathcal{J}_{2N} \left(D\phi_t(u)\right) = \mathcal{J}_{2N}$$

    with $D\phi_t$ the Jacobian of the flow ϕ_t,

- Hamiltonian ODEs benefits from numerous properties
  - preservation of the Hamiltonian along the flow

$$\frac{d}{dt}\mathcal{H}(u(t;\mu)) = 0,$$

  - the flow is a symplectic map

$$\left(D\phi_t(u)\right)^T \mathcal{J}_{2N} \left(D\phi_t(u)\right) = \mathcal{J}_{2N}$$

    with $D\phi_t$ the Jacobian of the flow ϕ_t,
  - time reversibility, volume preservation,
  - etc.,

- Hamiltonian ODEs benefits from numerous properties
  - preservation of the Hamiltonian along the flow

$$\frac{d}{dt}\mathcal{H}(u(t;\mu)) = 0,$$

  - the flow is a symplectic map

$$\left(D\phi_t(u)\right)^T \mathcal{J}_{2N} \left(D\phi_t(u)\right) = \mathcal{J}_{2N}$$

    with $D\phi_t$ the Jacobian of the flow ϕ_t,
  - time reversibility, volume preservation,
  - etc.,
- provide guarantees for long-time stability, physical relevance.

- Last step : deriving numerical solutions,

[1] Hairer et al. 2006.

- Last step : deriving numerical solutions,
- discretization of $[0, T]$ with time steps $t^n = n\Delta t$,
- compute approximated solution at each time step with numerical integration,

$$u(t^{n+1}; \mu) = u(t^n; \mu) + \int_{t^n}^{t^{n+1}} \mathcal{J}_{2N} \nabla_u \mathcal{H}(u(t; \mu))\, dt.$$

- quadrature choice for $\int_{t^n}^{t^{n+1}} \mathcal{J}_{2N} \nabla_u \mathcal{H}(u)\, dt$ = numerical scheme,

[1] Hairer et al. 2006.

- Last step : deriving numerical solutions,
- discretization of $[0, T]$ with time steps $t^n = n\Delta t$,
- compute approximated solution at each time step with numerical integration,

$$u(t^{n+1}; \mu) = u(t^n; \mu) + \int_{t^n}^{t^{n+1}} \mathcal{J}_{2N} \nabla_u \mathcal{H}(u(t; \mu))\, dt.$$

- quadrature choice for $\int_{t^n}^{t^{n+1}} \mathcal{J}_{2N} \nabla_u \mathcal{H}(u)\, dt$ = numerical scheme,
- need to use a symplectic numerical scheme[1] to safeguard the system properties : long time stability, physical relevance, etc.,
- in practice : implicit midpoint or Störmer Verlet.

[1] Hairer et al. 2006.

1 Hamiltonian systems

2 Model Order Reduction for Hamiltonian systems
- Proper Symplectic Decomposition (PSD)
- Deep learning based Hamiltonian reduction

3 An application to the shallow-water system

4 Reduced Particle in Cell method for the Vlasov-Poisson system

- We consider the symplectic solution manifold

$$\mathcal{M} = \{u(t;\mu) \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\} \subset \mathbb{R}^{2N}$$

- We consider the symplectic solution manifold

$$\mathcal{M} = \{u(t;\mu) \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\} \subset \mathbb{R}^{2N}$$

- fundamental reduction postulate : $\mathcal{M}$ can be **approximated by a trial manifold**

$$\widehat{\mathcal{M}} := u_{\text{ref}}(\mu) + \{\mathcal{D}\left[\bar{u}(t;\mu)\right] \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\}$$

with $\mathcal{D} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$, $K \ll N$ a reconstruction/decoding operator or **decoder**, $u_{\text{ref}}(\mu) \in \mathbb{R}^{2N}$ a reference state, $\bar{u}(t;\mu) \in \mathbb{R}^{2K}$ a **reduced state**,

- that is

$$\hat{u}(t;\mu) = u_{\text{ref}}(\mu) + \mathcal{D}\left[\bar{u}(t;\mu)\right] \approx u(t;\mu).$$

- We consider the symplectic solution manifold

$$\mathcal{M} = \{u(t;\mu) \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\} \subset \mathbb{R}^{2N}$$

- fundamental reduction postulate : $\mathcal{M}$ can be **approximated by a trial manifold**

$$\widehat{\mathcal{M}} := u_{\text{ref}}(\mu) + \{\mathcal{D}\left[\bar{u}(t;\mu)\right] \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\}$$

with $\mathcal{D} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$, $K \ll N$ a reconstruction/decoding operator or **decoder**, $u_{\text{ref}}(\mu) \in \mathbb{R}^{2N}$ a reference state, $\bar{u}(t;\mu) \in \mathbb{R}^{2K}$ a **reduced state**,

- that is

$$\hat{u}(t;\mu) = u_{\text{ref}}(\mu) + \mathcal{D}\left[\bar{u}(t;\mu)\right] \approx u(t;\mu).$$

- for illustration $N = 10^5$, $K = 30$ and $\dim(\Gamma) = 2$,

- Assume the trial manifold is a linear subspace

$$\widehat{\mathcal{M}} = \text{span}(a_i, i \in \{1, \dots, 2K\}) \iff \mathcal{D}\left[\bar{u}(t;\mu)\right] = A\bar{u}(t;\mu)\,,\ A \in \mathcal{M}_{2N,2K}(\mathbb{R})$$

- Assume the trial manifold is a linear subspace

$$\widehat{\mathcal{M}} = \text{span}(a_i, i \in \{1, \dots, 2K\}) \iff \mathcal{D}\left[\bar{u}(t;\mu)\right] = A\bar{u}(t;\mu)\,,\ A \in \mathcal{M}_{2N,2K}(\mathbb{R})$$

- constraint that the decoder $\bar{u} \mapsto A\bar{u}$ is a symplectic map

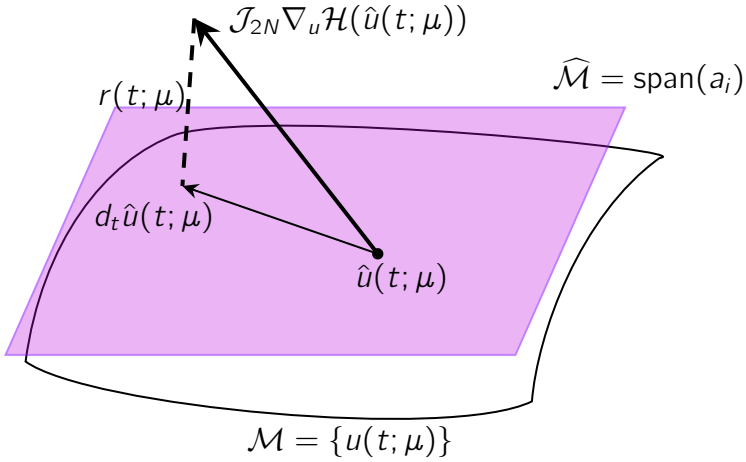
$$A^T \mathcal{J}_{2N} A = \mathcal{J}_{2K}$$

- symplectic inverse of the decoder = encoder $\mathcal{E}$

$$\mathcal{E}\left[u(t;\mu)\right] = A^+ u(t;\mu)$$

with $A^+ \in \mathcal{M}_{2K,2N}(\mathbb{R})$ such that $A^+ = \mathcal{J}_{2K}^T A^T \mathcal{J}_{2N}$ the symplectic inverse of A ($A^+ A = I_{2K}$).

- What is the dynamics of the reduced state $\bar{u}(t;\mu)$?


$\mathcal{J}_{2N}\nabla_u\mathcal{H}(\hat{u}(t;\mu))$
$\widehat{\mathcal{M}} = \text{span}(a_i)$
$r(t;\mu)$
$d_t\hat{u}(t;\mu)$
$\hat{u}(t;\mu)$
$\mathcal{M} = \{u(t;\mu)\}$

- We define the residual $r(t;\mu)$

$$r(t;\mu) = \frac{d\hat{u}(t;\mu)}{dt} - \mathcal{J}_{2N}\nabla_u \mathcal{H}(\hat{u}(t;\mu))$$

- We define the residual $r(t;\mu)$

$$r(t;\mu) = \frac{d\hat{u}(t;\mu)}{dt} - \mathcal{J}_{2N}\nabla_u \mathcal{H}(\hat{u}(t;\mu))$$

- symplectic Galerkin projection = the residual must vanish under the symplectic projection

$$A^+ r(t;\mu) = 0,$$

- We define the residual $r(t;\mu)$

$$r(t;\mu) = \frac{d\hat{u}(t;\mu)}{dt} - \mathcal{J}_{2N}\nabla_u \mathcal{H}(\hat{u}(t;\mu))$$

- symplectic Galerkin projection = the residual must vanish under the symplectic projection

$$A^+ r(t;\mu) = 0,$$

- results in a Hamiltonian reduced model

$$\begin{cases} \dfrac{d\bar{u}(t;\mu)}{dt} = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}(\bar{u}(t;\mu)), \\ \bar{u}(0;\mu) = A^+ u_{\text{init}}(\mu), \end{cases}$$

with the reduced Hamiltonian $\bar{\mathcal{H}} = \mathcal{H} \circ A$,

- Hamiltonian reduced model

$$\begin{cases} \dfrac{d\bar{u}(t;\mu)}{dt} = \mathcal{J}_{2K} \nabla_{\bar{u}} \bar{\mathcal{H}}(\bar{u}(t;\mu)), \\ \bar{u}(0;\mu) = A^{+} u_{\text{init}}(\mu), \end{cases}$$

with the reduced Hamiltonian $\bar{\mathcal{H}} = \mathcal{H} \circ A$,

- Hamiltonian reduced model

$$\begin{cases} \dfrac{d\bar{u}(t;\mu)}{dt} = \mathcal{J}_{2K} \nabla_{\bar{u}} \bar{\mathcal{H}}(\bar{u}(t;\mu)), \\ \bar{u}(0;\mu) = A^{+} u_{\text{init}}(\mu), \end{cases}$$

  with the reduced Hamiltonian $\bar{\mathcal{H}} = \mathcal{H} \circ A$,

- the method provides both reduced states and reduced dynamics,

- Hamiltonian reduced model

$$\begin{cases} \dfrac{d\bar{u}(t;\mu)}{dt} = \mathcal{J}_{2K} \nabla_{\bar{u}} \bar{\mathcal{H}}(\bar{u}(t;\mu)), \\ \bar{u}(0;\mu) = A^{+} u_{\text{init}}(\mu), \end{cases}$$

  with the reduced Hamiltonian $\bar{\mathcal{H}} = \mathcal{H} \circ A$,

- the method provides both reduced states and reduced dynamics,
- called the **Proper Symplectic Decomposition (PSD)** = linear method for symplectic reduction, from Peng and Mohseni (2016),

- Hamiltonian reduced model

$$\begin{cases} \dfrac{d\bar{u}(t;\mu)}{dt} = \mathcal{J}_{2K} \nabla_{\bar{u}} \bar{\mathcal{H}}(\bar{u}(t;\mu)), \\ \bar{u}(0;\mu) = A^{+} u_{\text{init}}(\mu), \end{cases}$$

  with the reduced Hamiltonian $\bar{\mathcal{H}} = \mathcal{H} \circ A$,

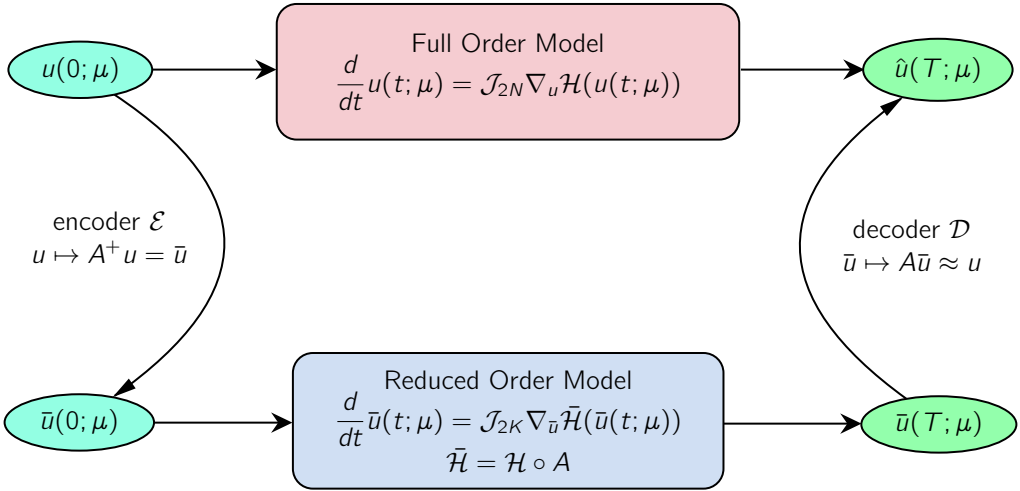
- the method provides both reduced states and reduced dynamics,
- called the **Proper Symplectic Decomposition (PSD)** = linear method for symplectic reduction, from Peng and Mohseni (2016),
- PSD = symplectic variant of the Proper Orthogonal Decomposition (POD),

- Hamiltonian reduced model

$$\begin{cases} \dfrac{d\bar{u}(t;\mu)}{dt} = \mathcal{J}_{2K} \nabla_{\bar{u}} \bar{\mathcal{H}}(\bar{u}(t;\mu)), \\ \bar{u}(0;\mu) = A^{+} u_{\text{init}}(\mu), \end{cases}$$

  with the reduced Hamiltonian $\bar{\mathcal{H}} = \mathcal{H} \circ A$,

- the method provides both reduced states and reduced dynamics,
- called the **Proper Symplectic Decomposition (PSD)** = linear method for symplectic reduction, from Peng and Mohseni (2016),
- PSD = symplectic variant of the Proper Orthogonal Decomposition (POD),
- warning: no complexity improvement in general, need for hyper-reduction techniques (Discrete Empirical Interpolation Method (DEIM), etc., not discussed).


$u(0;\mu)$
Full Order Model
$\frac{d}{dt}u(t;\mu) = \mathcal{J}_{2N}\nabla_u \mathcal{H}(u(t;\mu))$
$\hat{u}(T;\mu)$
encoder $\mathcal{E}$
$u \mapsto A^+ u = \bar{u}$
decoder $\mathcal{D}$
$\bar{u} \mapsto A\bar{u} \approx u$
$\bar{u}(0;\mu)$
Reduced Order Model
$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}(\bar{u}(t;\mu))$
$\bar{\mathcal{H}} = \mathcal{H} \circ A$
$\bar{u}(T;\mu)$

- How to build A ? The solution manifold $\mathcal{M}$ is unknown !

[2] Lange 2010.

- How to build A ? The solution manifold $\mathcal{M}$ is unknown !
- Solution: from numerical solution snapshots/samples

$$U = \begin{bmatrix} u(t_1;\mu_1) & \dots & u(t_P;\mu_P) \end{bmatrix} \in \mathcal{M}_{2N,P}(\mathbb{R}),$$

- A minimizes the reconstruction error on the snapshots,

$$\min_{A^T \mathcal{J}_{2N} A = \mathcal{J}_{2K}} \left\| U - AA^+ U \right\|_F$$

[2] Lange 2010.

- How to build A ? The solution manifold $\mathcal{M}$ is unknown !

- Solution: from numerical solution snapshots/samples

$$U = \begin{bmatrix} u(t_1;\mu_1) & \dots & u(t_P;\mu_P) \end{bmatrix} \in \mathcal{M}_{2N,P}(\mathbb{R}),$$

- A minimizes the reconstruction error on the snapshots,

$$\min_{A^T \mathcal{J}_{2N} A = \mathcal{J}_{2K}} \left\| U - AA^+ U \right\|_F$$

- in practice, the **Singular Value Decomposition** (SVD)[2] of U on a modified minimization problem is used.

[2] Lange 2010.

- The SVD of U is a decomposition of the form

$$U = W\Sigma V^*$$

with $W \in \mathcal{U}_{2N}(\mathbb{C})$, $V \in \mathcal{U}_P(\mathbb{C})$ unitary matrices, $\Sigma = (\sigma_i)_{i,i} \in \mathcal{M}_{2N,P}(\mathbb{C})$ a rectangular diagonal matrix, V^* is the conjugate-transpose of V,

- W, V columns are orthonormal bases called left and right singular vectors, respectively, $(\sigma_i)_{i,i}$ are singular values,

[3] Peng and Mohseni 2016.

- The SVD of U is a decomposition of the form

$$U = W\Sigma V^*$$

with $W \in \mathcal{U}_{2N}(\mathbb{C})$, $V \in \mathcal{U}_P(\mathbb{C})$ unitary matrices, $\Sigma = (\sigma_i)_{i,i} \in \mathcal{M}_{2N,P}(\mathbb{C})$ a rectangular diagonal matrix, V^* is the conjugate-transpose of V,

- W, V columns are orthonormal bases called left and right singular vectors, respectively, $(\sigma_i)_{i,i}$ are singular values,
- **best rank** r **approximation** of U:

$$U_r = W_r \Sigma_r V_r^*$$

with $W_r = W[:, : r]$, $V_r = V[:, : r]$, $\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$,

- A depends on[3] W_K.

[3] Peng and Mohseni 2016.

- How to make it efficient ?

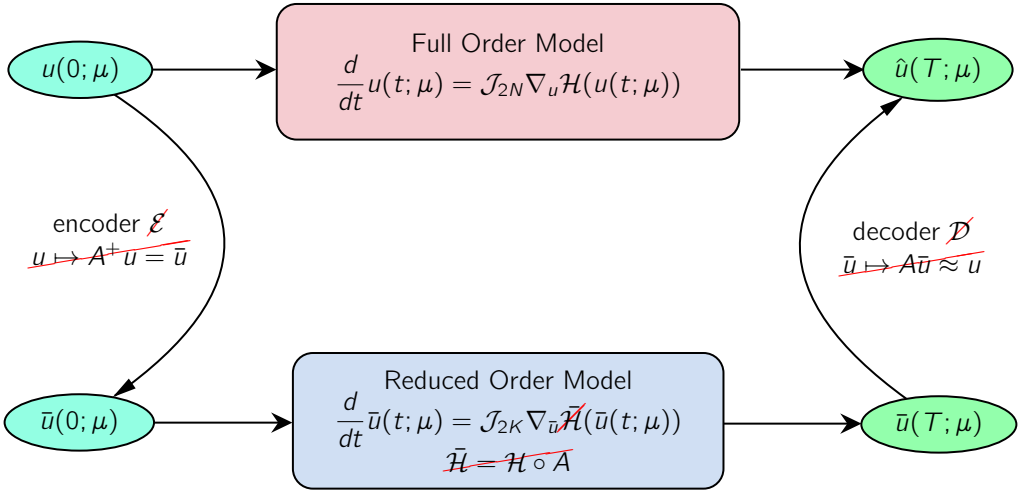
- How to make it efficient ?
- Offline/online decomposition:

- How to make it efficient ?
- Offline/online decomposition:
  - **offline stage**: computationally expensive, parametrically independent, performed once (building models, precompute quantities e.g. snapshots, reduced basis A, choose K, etc.),

- How to make it efficient ?
- Offline/online decomposition:
  - **offline stage**: computationally expensive, parametrically independent, performed once (building models, precompute quantities e.g. snapshots, reduced basis A, choose K, etc.),
  - **online stage**: fast computation, done for every new parameter, use offline precomputation to accelerate the reduced simulation.

- Statements on the linear model order reduction:
  - works well in linear and quasi-linear regimes,
  - interpolation/approximation strategies (DEIM, etc.)[4] in nonlinear regimes,
  - struggles in strongly nonlinear regimes,
- idea : replace the encoder, decoder, and eventually the reduced model by neural networks, as presented in Côte, Franck, Navoret, S., and Vigon (2025).

[4] Peng and Mohseni 2016; Hesthaven et al. 2024.


$u(0;\mu)$
Full Order Model
$\frac{d}{dt}u(t;\mu) = \mathcal{J}_{2N}\nabla_u \mathcal{H}(u(t;\mu))$
$\hat{u}(T;\mu)$
encoder $\mathcal{E}$
$u \mapsto A^+ u = \bar{u}$
decoder $\mathcal{D}$
$\bar{u} \mapsto A\bar{u} \approx u$
$\bar{u}(0;\mu)$
Reduced Order Model
$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}(\bar{u}(t;\mu))$
$\bar{\mathcal{H}} = \mathcal{H} \circ A$
$\bar{u}(T;\mu)$

- Neural network = **parametric function** g_θ of parameters $\theta \in \Theta$,
- g_θ = **composition of** c **simple functions** $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ = layer,

$$g_\theta = g_c \circ \cdots \circ g_1,$$

- e.g.:
  - dense layer $g_i(x) = \sigma\left(W^{[i]}x + b^{[i]}\right)$ with $W^{[i]} \in \mathcal{M}_{n_{i+1},n_i}(\mathbb{R})$, $b^{[i]} \in \mathbb{R}^{n_{i+1}}$,
  - convolutional layer $g_i(x) = \sigma\left(W^{[i]} * x + b^{[i]}\right)$ with $*$ a convolution with a kernel $W^{[i]}$,
- σ non-linear function, $\theta = \left\{W^{[i]}, b^{[i]}, i \in \{1, \cdots, c\}\right\}$.

- Neural network = **parametric function** g_θ of parameters $\theta \in \Theta$,
- g_θ **fitted to a target function** g : $g_\theta \sim g$,
- on snapshots U, according to a cost function / loss $\mathcal{L}$,

$$\theta^* = \text{argmin}_{\theta\in\Theta}\, \mathcal{L}(g, g_\theta),$$

  e.g. $\mathcal{L}(g, g_\theta) = \sum_{u\in U} \|g(u) - g_\theta(u)\|_2^2$,
- with a gradient descent (*Adam* algorithm...),

$$\theta^{[k+1]} = \theta^{[k]} - \eta^{[k]} \nabla_\theta \mathcal{L}(g, g_{\theta^{[k]}}),$$

  with the learning rate $\eta^{[k]}$,
- called the neural network training.

- Compression/decompression managed by a (convolutional) AutoEncoder[5] (AE) = pair of neural networks $\mathcal{E}_\theta : \mathbb{R}^{2N} \to \mathbb{R}^{2K}, \mathcal{D}_\theta : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$ such that $\mathcal{D}_\theta \circ \mathcal{E}_\theta \approx \mathrm{id}$,
- compression $\mathcal{E}_\theta(u) = \bar{u}$ and decompression $\mathcal{D}_\theta(\bar{u}) \approx u$,
- fitted with the loss $\mathcal{L}_{\mathrm{AE}}$

$$\mathcal{L}_{\mathrm{AE}} = \sum_{u \in U} \left\| u - \mathcal{D}_\theta \left(\mathcal{E}_\theta \left(u \right) \right) \right\|^2 ,$$

- **no direct symplecticity constraint** in the architecture or the loss.

[5] Goodfellow et al. 2016.

- What happens to the reduced model ?

$$\frac{d}{dt}\bar{u} = \mathcal{J}_{2K}\left[D_{\mathcal{D}_\theta}(\bar{u})\right]^T \nabla_{\mathcal{D}_\theta(\bar{u})}\mathcal{H}\left[\mathcal{D}_\theta(\bar{u})\right] \stackrel{?}{=} \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}(\bar{u})$$

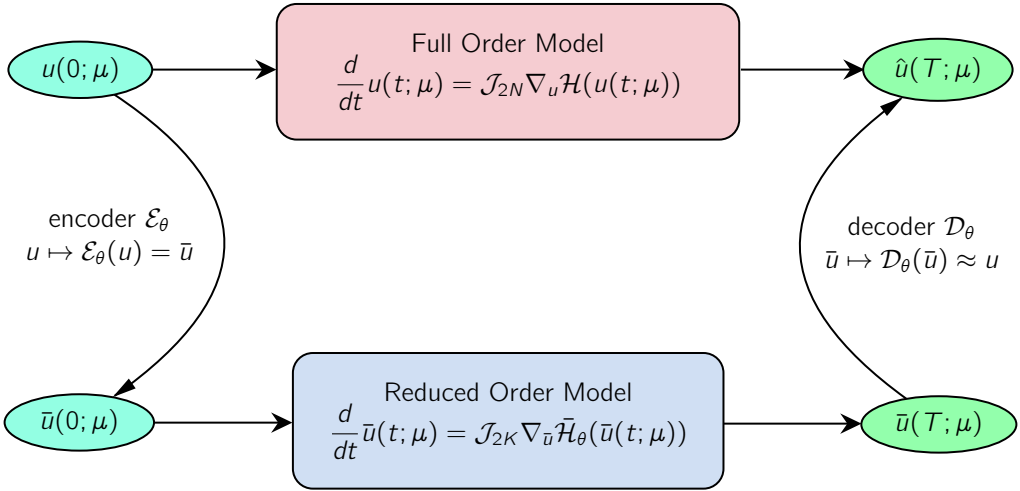
- What happens to the reduced model ?

$$\frac{d}{dt}\bar{u} = J_{2K}\left[D_{\mathcal{D}_\theta}(\bar{u})\right]^T \nabla_{\mathcal{D}_\theta(\bar{u})}\mathcal{H}\left[\mathcal{D}_\theta(\bar{u})\right] \stackrel{?}{=} \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}(\bar{u})$$

- supplant it with a Hamiltonian Neural Network (HNN) $\bar{\mathcal{H}}_\theta : \mathbb{R}^{2K} \to \mathbb{R}$ from Greydanus, Dzamba, and Yosinski (2019),

$$\begin{cases} \frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}_\theta(\bar{u}(t;\mu)) \\ \bar{u}(0;\mu) = \mathcal{E}_\theta(u_{\text{init}}(\mu)), \end{cases}$$

- reduced model is Hamiltonian **by design**.


$u(0;\mu)$
Full Order Model
$\frac{d}{dt}u(t;\mu) = \mathcal{J}_{2N}\nabla_u \mathcal{H}(u(t;\mu))$
$\hat{u}(T;\mu)$
encoder $\mathcal{E}_\theta$
$u \mapsto \mathcal{E}_\theta(u) = \bar{u}$
decoder $\mathcal{D}_\theta$
$\bar{u} \mapsto \mathcal{D}_\theta(\bar{u}) \approx u$
$\bar{u}(0;\mu)$
Reduced Order Model
$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}} \bar{\mathcal{H}}_\theta(\bar{u}(t;\mu))$
$\bar{u}(T;\mu)$

- How to learn the reduced dynamics ?

$$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}_\theta(\bar{u}(t;\mu))$$

- How to learn the reduced dynamics ?

$$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}_\theta(\bar{u}(t;\mu))$$

- prediction operator $\mathcal{P}$ = a step from a symplectic scheme (e.g. midpoint):

$$\mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right) \approx \bar{u}^{n+1} = \mathcal{E}_\theta(u^{n+1})$$

- How to learn the reduced dynamics ?

$$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}_\theta(\bar{u}(t;\mu))$$

- prediction operator $\mathcal{P}$ = a step from a symplectic scheme (e.g. midpoint):

$$\mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right) \approx \bar{u}^{n+1} = \mathcal{E}_\theta(u^{n+1})$$

- we add 3 losses:

$$\mathcal{L}_{\overline{\text{pred}}} = \sum_{u^n,u^{n+1}\in U} \left\|\bar{u}^{n+1} - \mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right)\right\|^2,$$

$$\mathcal{L}_{\overline{\text{stab}}} = \sum_{u^n,u^{n+1}\in U} \left\|\bar{\mathcal{H}}_\theta\left(\bar{u}^{n+1}\right) - \bar{\mathcal{H}}_\theta\left(\bar{u}^n\right)\right\|^2,$$

$$\mathcal{L}_{\text{pred}} = \sum_{u^n,u^{n+1}\in U} \left\|u^{n+1} - \mathcal{D}_\theta\left(\mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right)\right)\right\|^2.$$

- How to learn the reduced dynamics ?

$$\frac{d}{dt}\bar{u}(t;\mu) = \mathcal{J}_{2K}\nabla_{\bar{u}}\bar{\mathcal{H}}_\theta(\bar{u}(t;\mu))$$

- prediction operator $\mathcal{P}$ = a step from a symplectic scheme (e.g. midpoint):

$$\mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right) \approx \bar{u}^{n+1} = \mathcal{E}_\theta(u^{n+1})$$

- we add 3 losses:

$$\mathcal{L}_{\overline{\text{pred}}} = \sum_{u^n,u^{n+1}\in U} \left\|\bar{u}^{n+1} - \mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right)\right\|^2,$$

$$\mathcal{L}_{\overline{\text{stab}}} = \sum_{u^n,u^{n+1}\in U} \left\|\bar{\mathcal{H}}_\theta\left(\bar{u}^{n+1}\right) - \bar{\mathcal{H}}_\theta\left(\bar{u}^n\right)\right\|^2,$$

$$\mathcal{L}_{\text{pred}} = \sum_{u^n,u^{n+1}\in U} \left\|u^{n+1} - \mathcal{D}_\theta\left(\mathcal{P}\left(\bar{u}^n;\bar{\mathcal{H}}_\theta\right)\right)\right\|^2.$$

- remark : losses linked to AE inputs/outputs $\rightarrow$ constrain AE-HNN.

- Reduced variables and reduced dynamics constructed separately ($\neq$ PSD) + lack of a symplectic AE,

- solution: **joint training of AE and HNN**,

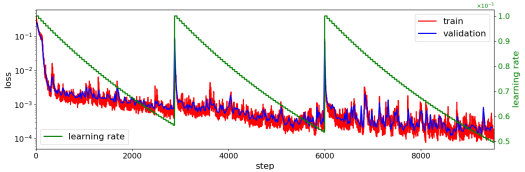
- Reduced variables and reduced dynamics constructed separately ($\neq$ PSD) + lack of a symplectic AE,
- solution: **joint training of AE and HNN**,
- the 4 losses are weighted and coupled during training

$$\min_{\theta} \quad \omega_{\mathrm{AE}}\, \mathcal{L}_{\mathrm{AE}}(\theta) + \omega_{\overline{\mathrm{pred}}}\, \mathcal{L}_{\overline{\mathrm{pred}}}(\theta) + \omega_{\overline{\mathrm{stab}}}\, \mathcal{L}_{\overline{\mathrm{stab}}}(\theta) + \omega_{\mathrm{pred}}\, \mathcal{L}_{\mathrm{pred}}(\theta),$$

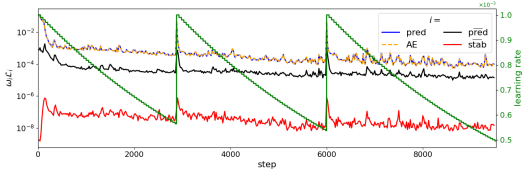
- the 4 losses are weighted and coupled during training

$$\min_{\theta} \quad \omega_{\mathrm{AE}}\,\mathcal{L}_{\mathrm{AE}}(\theta) + \omega_{\overline{\mathrm{pred}}}\,\mathcal{L}_{\overline{\mathrm{pred}}}(\theta) + \omega_{\overline{\mathrm{stab}}}\,\mathcal{L}_{\overline{\mathrm{stab}}}(\theta) + \omega_{\mathrm{pred}}\,\mathcal{L}_{\mathrm{pred}}(\theta),$$

- e.g. $\omega_{\mathrm{AE}} = 1, \quad \omega_{\overline{\mathrm{pred}}} = 10, \quad \omega_{\overline{\mathrm{stab}}} = 1\times10^{-4}, \quad \omega_{\mathrm{pred}} = 1$



(a) Training loss function (blue) and validation loss function (red) history.



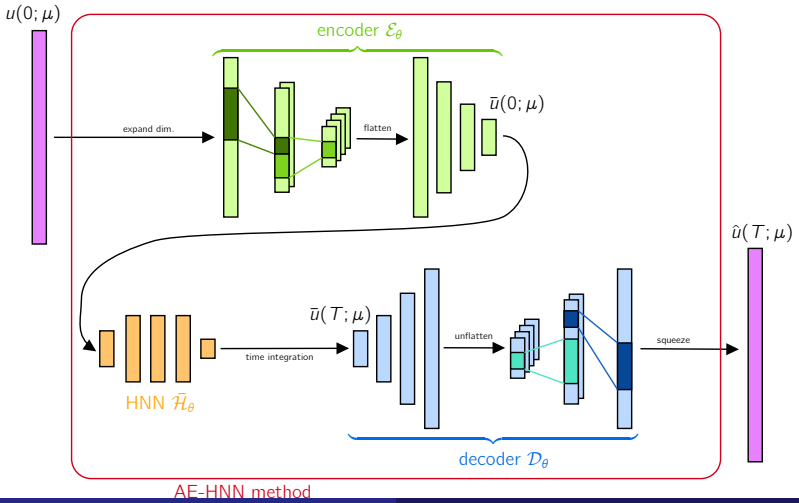
(b) All the weighted loss functions as functions of the training step.

Figure: Example of loss history during a training, overlaid with the evolution of the learning rate (green).

- Key elements on neural networks construction
  - large set of hyperparameters (architecture, layer number, layer size, activation function, Newton solver, etc.)
  - chosen from experience, grid search or random search,
- and training
  - scheduled learning rate, warm restart,
  - AE pretraining, variable loss weights.

- Updated offline stage:
  - build full order model, reduced model, select hyperparameters and a minimal reduced dimension K for correct accuracy,
  - train the AE and HNN together with full order snapshots as dataset.

AE-HNN online stage



- Evolution of a free surface of water on a flat bottom,
- $\chi, \phi : \mathbb{R}^2/(L\mathbb{Z}^2) \times [0, T] \times \Gamma \to \mathbb{R}$ are the perturbation from the equilibrium and the scalar velocity potential, Ω is a periodic square domain on size L,
- $u(x, t; \mu) = (\chi, \phi)^T(x, t; \mu)$

$$\begin{cases} \partial_t \chi + \nabla \cdot ((1 + \chi) \nabla \phi) = 0, \\ \partial_t \phi + \frac{1}{2} |\nabla \phi|^2 + \chi = 0, \end{cases}$$

- with the Hamiltonian

$$\mathcal{H}[\chi, \phi] = \frac{1}{2} \int_{\mathbb{R}^2/(L\mathbb{Z}^2)} \left((1 + \chi) |\nabla \phi|^2 + \chi^2 \right) \, dx.$$

- Domain $\Omega = \mathbb{R}^2/(L\mathbb{Z}^2)$ discretized with a mesh $(x_i, y_j)_{i,j}$ of N nodes,
- discretized state $\chi_h(t;\mu), \phi_h(t;\mu) \in \mathbb{R}^N$, $(\chi_h)_m(t;\mu) = \chi_{i,j}(t;\mu) \approx \chi(x_i, y_j, t;\mu)$

- Domain $\Omega = \mathbb{R}^2/(L\mathbb{Z}^2)$ discretized with a mesh $(x_i, y_j)_{i,j}$ of N nodes,
- discretized state $\chi_h(t;\mu), \phi_h(t;\mu) \in \mathbb{R}^N$, $(\chi_h)_m(t;\mu) = \chi_{i,j}(t;\mu) \approx \chi(x_i, y_j, t;\mu)$
- with finite differences $\rightarrow$ high dimensional Hamiltonian ODE of solution $u_h = (\chi_h, \phi_h)$

$$\mathcal{H}(\chi_h, \phi_h) = \frac{1}{2}\sum_{i,j=0}^{M-1}\left((1+\chi_{i,j})\left[\left(\frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}\right)^2 + \left(\frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta y}\right)^2\right] + \chi_{i,j}^2\right),$$

$$\begin{cases} \dfrac{d}{dt}\chi_h = -D_x\left([1+\chi_h]\odot D_x\phi_h\right) - D_y\left([1+\chi_h]\odot D_y\phi_h\right), \\ \dfrac{d}{dt}\phi_h = -\dfrac{1}{2}\left[(D_x\phi_h)^2 + (D_y\phi_h)^2\right] - \chi_h, \end{cases}$$

with $D_x, D_y \in \mathcal{M}_{2N}(\mathbb{R})$ finite difference matrices,

- cost of $\mathcal{O}(N^2)$ for each $(t,\mu) \in \mathcal{T}\times\Gamma$.

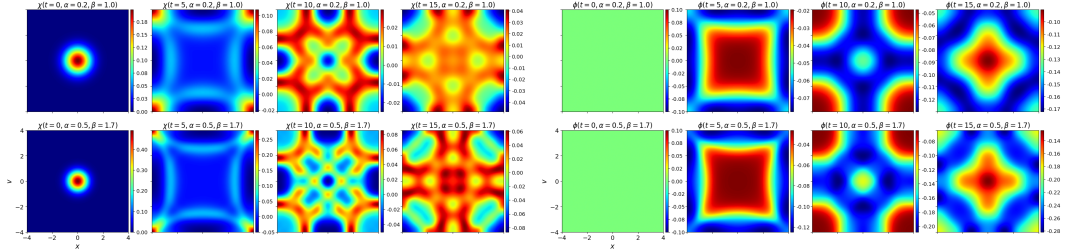
- Discretization with $M = 64$ cells per direction, final time $T = 15$, time step $\Delta t = 1 \times 10^{-3}$, implicit midpoint numerical scheme,

- **parametrized initial condition** with two parameters $\mu = (\alpha, \beta) \in \Gamma = [0.2, 0.5] \times [1, 1.7]$

$$\chi_{\text{init}}(x; \mu) = \alpha \exp\left(-\beta\, x^T x\right), \quad \phi_{\text{init}}(x; \mu) = 0.$$

- Numerical solution with a symplectic scheme

- With a non symplectic scheme



(a) $\chi(t;\mu)$ (b) $\phi(t;\mu)$

Figure: Solutions (χ, ϕ) at different times $t \in \{0, 5, 10, 15\}$ for various parameters $(\alpha, \beta) \in \{(0.2, 1), (0.5, 1.8)\}$.

- $\Gamma = [0.2, 0.5] \times [1, 1.7]$ sampled with 20 snapshots regularly spaced in the segment $[(0.2, 1), (0.5, 1.7)]$,

- $\Gamma = [0.2, 0.5] \times [1, 1.7]$ sampled with 20 snapshots regularly spaced in the segment $[(0.2, 1), (0.5, 1.7)]$,

- $K = 4$ (from $N = 64^2 = 4096$), chosen minimal while preserving sufficient accuracy,

- $\Gamma = [0.2, 0.5] \times [1, 1.7]$ sampled with 20 snapshots regularly spaced in the segment $[(0.2, 1), (0.5, 1.7)]$,
- $K = 4$ (from $N = 64^2 = 4096$), chosen minimal while preserving sufficient accuracy,
- inputs $u(t; \mu) = (\chi, \phi)^T(t; \mu) \in \mathbb{R}^{2N}$ are structured $\rightarrow$ **convolutional** autoencoder, $\sim 10^6$ parameters, used once,
- HNN = small dense neural network $\sim 10^4$ parameters / PSD $\sim 10^4$ parameters.

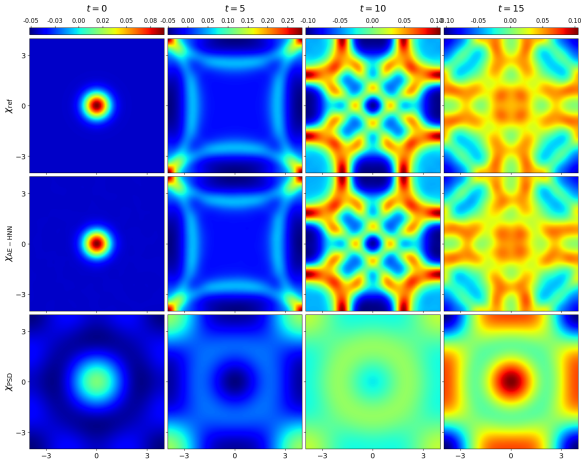


Figure: Solutions $\chi(t;\mu)$ at different times $t \in \{0, 5, 10, 15\}$ on $\mu = (0.51, 1.72) \notin \Gamma$ with $K = 4$, reference solution (top line), AE-HNN solution (middle line) and PSD solution (bottom line).

- Offline time : AE-HNN training $\sim$ 1h / PSD $\sim$ 2-3min (+ snapshots $\sim$ 30min),

- Offline time : AE-HNN training $\sim$ 1h / PSD $\sim$ 2-3min (+ snapshots $\sim$ 30min),
- online time :
  - full order model : 101s,
  - (DEIM hyper-reduced) PSD reduced model with $K = 30$ (comparable error) : 57s,
  - AE-HNN reduced model : 27s,

- Offline time : AE-HNN training $\sim$ 1h / PSD $\sim$ 2-3min (+ snapshots $\sim$ 30min),
- online time :
  - full order model : 101s,
  - (DEIM hyper-reduced) PSD reduced model with $K = 30$ (comparable error) : 57s,
  - AE-HNN reduced model : 27s,
- Limited performance and explainability: different hardware CPU/GPU, different libraries Numpy/Tensorflow, developer expertise.

- convolutional AE: nonlinear projection, convolutions take into account spatial structure,
- HNN based reduced model : Hamiltonian by design,
- joint AE-HNN training: compensates for the absence of a symplectic AE,

- convolutional AE: nonlinear projection, convolutions take into account spatial structure,
- HNN based reduced model : Hamiltonian by design,
- joint AE-HNN training: compensates for the absence of a symplectic AE,
- strengths:
  - improved precision compared to the PSD,
  - great speed: neural networks are efficiently parallelized on GPUs,

- convolutional AE: nonlinear projection, convolutions take into account spatial structure,
- HNN based reduced model : Hamiltonian by design,
- joint AE-HNN training: compensates for the absence of a symplectic AE,
- strengths:
  - improved precision compared to the PSD,
  - great speed: neural networks are efficiently parallelized on GPUs,
- weaknesses:
  - increasing K is not enough to systematically improve precision,
  - lack of errors bounds, no clear guarantees of global convergence.

- System described by the distribution $f(t, x, v; \mu)$ with time $t \in \mathcal{T} = [0, T]$, position $x \in \Omega_x = \mathbb{R}/2\pi\mathbb{Z}$, velocity $v \in \Omega_v \subset \mathbb{R}$ and parameters $\mu \in \Gamma \subset \mathbb{R}^p$, $p > 0$, charge q and mass m,

$$\begin{cases} \partial_t f(t, x, v; \mu) + v\partial_x f(t, x, v; \mu) + \dfrac{q}{m} E(t, x; \mu)\partial_v f(t, x, v; \mu) = 0, \\ \partial_x E(t, x; \mu) = \rho(t, x; \mu), \end{cases}$$

  where $\rho(t, x; \mu) = q \int_{\Omega_v} f(t, x, v; \mu)\, dv$ is the electric density,
- $E(t, x; \mu)$ is the (self-induced) electric field, derives from electric potential $\phi(t, x; \mu)$: $-\partial_x \phi = E$,
- the Poisson equation rewrites

$$-\partial_{xx}\phi(t, x; \mu) = \rho(t, x; \mu),$$

- admits an Hamiltonian structure with a Lie-Poisson bracket[6] (not detailed).

[6] Casas et al. 2017.

- Solution **approximated with** $N \gg 1$ **particles** $(x_k(t), v_k(t))$ in the phase space

$$f_N(t, x, v; \mu) = \sum_{k=1}^{N} \omega \, \delta\left(x - x_k(t)\right) \delta\left(v - v_k(t)\right)$$

- results in a $2N$-dimensional ODE

$$\begin{cases} \dfrac{d}{dt} x_h(t; \mu) = v_h(t; \mu), \\ \dfrac{d}{dt} v_h(t; \mu) = \dfrac{q}{m} E(x_h(t; \mu); \mu), \end{cases}$$

  where $(x_h)_k = x_k, (v_h)_k = v_k$,

- electric field computed with a mesh : (Hamiltonian) **Particle-In-Cell (PIC) method** from Kraus, Kormann, Morrison, and Sonnendrücker (2017).

- Full order model of solution $u = (x\ v)^T \in \mathbb{R}^{2N}$

$$\frac{d}{dt}u(t;\mu) = J_{2N}\nabla_u \mathcal{H}(u(t;\mu))$$

with $J_{2N} = \begin{pmatrix} 0_N & I_N \\ -I_N & 0_N \end{pmatrix}$,

- $\mathcal{H} : \mathbb{R}^{2N} \to \mathbb{R}$ is the Hamiltonian (total energy)

$$\mathcal{H}(u(t;\mu)) = \underbrace{\frac{1}{2}v^T v}_{\text{kinetic energy}} + \underbrace{\frac{1}{2m}q^2\omega\Lambda^0(x(t;\mu))L^{-1}\Lambda^0(x(t;\mu))^T\mathbb{1}_N}_{\text{potential energy}}$$

with Λ^0 a particle-to-grid mapping, L a discrete Laplacian matrix.

- We cannot apply our AE-HNN framework with inputs $u(t;\mu) \in \mathbb{R}^{2N}$: particles are not structured and N too large,

- We cannot apply our AE-HNN framework with inputs $u(t;\mu) \in \mathbb{R}^{2N}$: particles are not structured and N too large,

- idea : preprocess $u(t;\mu) \mapsto \tilde{u}(t;\mu) \in \mathbb{R}^{2M}$, $M \ll N$ while keeping the symplectic structure,

- solution: use the **PSD coupled with the AE-HNN method** for a two steps encoder/decoder, Franck, Navoret, Vigon, Côte, and S. (2025).

- Two steps projection

$$\begin{array}{ccccc} \mathbb{R}^{2N} & \longrightarrow & \mathbb{R}^{2M} & \longrightarrow & \mathbb{R}^{2K} \\ u(t;\mu) & \xmapsto{A^+ \cdot} & \tilde{u}(t;\mu) & \xmapsto{\mathcal{E}_\theta} & \bar{u}(t;\mu) \end{array}$$

- with an intermediate state of size $2M, K < M \ll N$ e.g. $K = 4, M = 121$,

- Two steps projection

$$\begin{array}{ccccc} \mathbb{R}^{2N} & \longrightarrow & \mathbb{R}^{2M} & \longrightarrow & \mathbb{R}^{2K} \\ u(t;\mu) & \xmapsto{A^+ \cdot} & \tilde{u}(t;\mu) & \xmapsto{\mathcal{E}_\theta} & \bar{u}(t;\mu) \end{array}$$

- with an intermediate state of size $2M, K < M \ll N$ e.g. $K = 4, M = 121$,
- first projection = linear operator $A \in \mathcal{M}_{2N,2M}(\mathbb{R})$ from the PSD such that

$$u = A\tilde{u}, \quad \tilde{u} = A^+ u,$$

- Two steps projection

$$\begin{array}{ccccc} \mathbb{R}^{2N} & \longrightarrow & \mathbb{R}^{2M} & \longrightarrow & \mathbb{R}^{2K} \\ u(t;\mu) & \xmapsto{A^+\cdot} & \tilde{u}(t;\mu) & \xmapsto{\mathcal{E}_\theta} & \bar{u}(t;\mu) \end{array}$$

- with an intermediate state of size $2M$, $K < M \ll N$ e.g. $K = 4, M = 121$,
- first projection = linear operator $A \in \mathcal{M}_{2N,2M}(\mathbb{R})$ from the PSD such that

$$u = A\tilde{u}, \quad \tilde{u} = A^+ u,$$

- second projection = autoencoder $(\mathcal{E}_\theta, \mathcal{D}_\theta)$

$$\bar{u} = \mathcal{E}_\theta(\tilde{u}), \quad \tilde{u} \approx \mathcal{D}_\theta(\bar{u}),$$

- Two steps projection

$$\begin{array}{ccccc} \mathbb{R}^{2N} & \longrightarrow & \mathbb{R}^{2M} & \longrightarrow & \mathbb{R}^{2K} \\ u(t;\mu) & \xmapsto{A^+\cdot} & \tilde{u}(t;\mu) & \xmapsto{\mathcal{E}_\theta} & \bar{u}(t;\mu) \end{array}$$

- with an intermediate state of size $2M$, $K < M \ll N$ e.g. $K = 4, M = 121$,
- first projection = linear operator $A \in \mathcal{M}_{2N,2M}(\mathbb{R})$ from the PSD such that

$$u = A\tilde{u}, \quad \tilde{u} = A^+ u,$$

- second projection = autoencoder $(\mathcal{E}_\theta, \mathcal{D}_\theta)$

$$\bar{u} = \mathcal{E}_\theta(\tilde{u}), \quad \tilde{u} \approx \mathcal{D}_\theta(\bar{u}),$$

- reduced model captured with a HNN $\bar{\mathcal{H}}_\theta$,

- Two steps projection

$$\begin{array}{ccccc} \mathbb{R}^{2N} & \longrightarrow & \mathbb{R}^{2M} & \longrightarrow & \mathbb{R}^{2K} \\ u(t;\mu) & \xmapsto{A^+\cdot} & \tilde{u}(t;\mu) & \xmapsto{\mathcal{E}_\theta} & \bar{u}(t;\mu) \end{array}$$

- with an intermediate state of size $2M$, $K < M \ll N$ e.g. $K = 4, M = 121$,
- first projection = linear operator $A \in \mathcal{M}_{2N,2M}(\mathbb{R})$ from the PSD such that

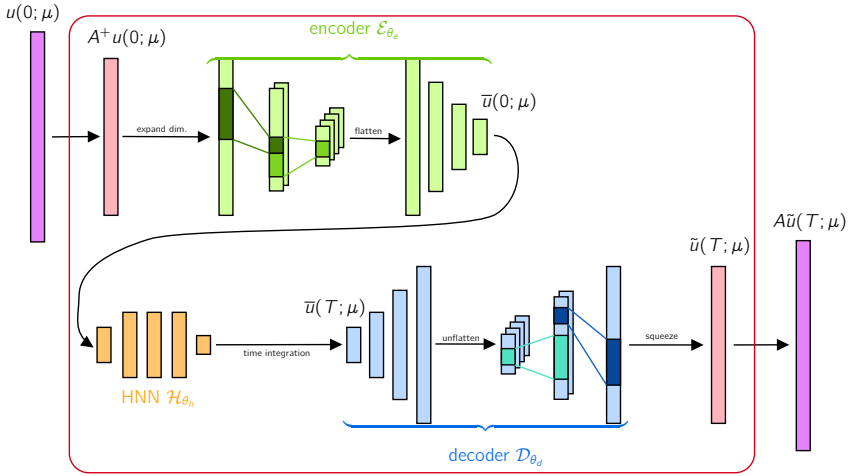
$$u = A\tilde{u}, \quad \tilde{u} = A^+ u,$$

- second projection = autoencoder $(\mathcal{E}_\theta, \mathcal{D}_\theta)$

$$\bar{u} = \mathcal{E}_\theta(\tilde{u}), \quad \tilde{u} \approx \mathcal{D}_\theta(\bar{u}),$$

- reduced model captured with a HNN $\bar{\mathcal{H}}_\theta$,
- offline stage: first PSD then AE-HNN training.

PSD-AE-HNN online stage



- Landau damping : parametrized initial condition $\mu = (\alpha, \sigma)^T \in \Gamma \subset \mathbb{R}^2$

$$f_{\text{init}}(x, v; \mu) = \underbrace{\frac{1}{4\pi}\left(1 + \alpha \cos\left(\frac{x}{2}\right)\right)}_{f_{\text{init},x}(x;\alpha)} \underbrace{\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{v^2}{2\sigma^2}\right)}_{f_{\text{init},v}(v;\sigma)},$$

- $(\alpha, \sigma) \in \Gamma = [0.03, 0.06] \times [0.8, 1]$,
- quantity of interest : damping rate of the electric energy $\frac{1}{2}\|E(x)\|_{L^2}$,
- $N = 10^5$ and $T = 20, \Delta t = 2.5 \times 10^{-3}$,
- $\Gamma = [0.03, 0.06] \times [0.8, 1]$ is sampled over a regular grid of size $8 \times 8 = 64$.

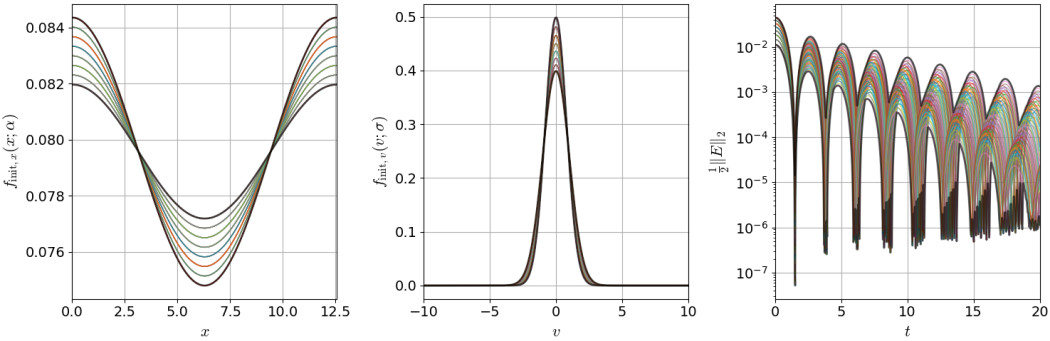


Figure: Initial distribution $f_{\text{init},x}(x;\alpha)$ (left), $f_{\text{init},v}(x;\sigma)$ (middle) and evolution of the electric energy $\frac{1}{2}\|E\|_2\left(x(t;\mu);\mu\right)$ (right) for every $\mu \in \Gamma^{\text{train}}$.

- How to choose $M(= 121)$?
- For example, according to the decay of the snapshots matrix singular values

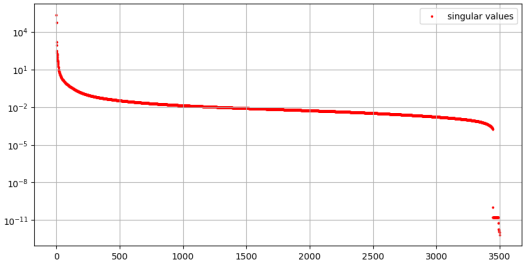


Figure: Singular values $(\sigma_i)_i$ decay.

- in practice:
  - sufficiently small to ensure a fast projection,
  - sufficiently large to provide an intermediate space rich enough for the AE-HNN method.

- How to choose $K(=3=\dim(\Gamma)+1)$? Smallest possible with a sufficient precision,

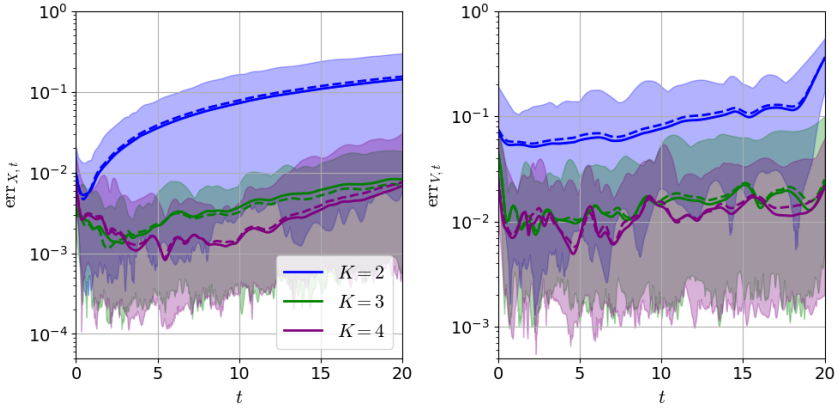


Figure: Mean relative error as a function of time (solid line) for x (left) and v (right), envelopes represents minimum and maximum errors.

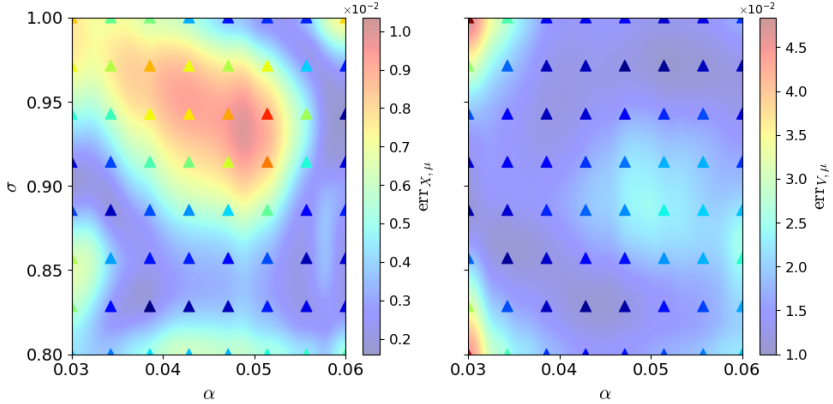


Figure: Errors as a function of the reduction parameters for x (left) and v (right).

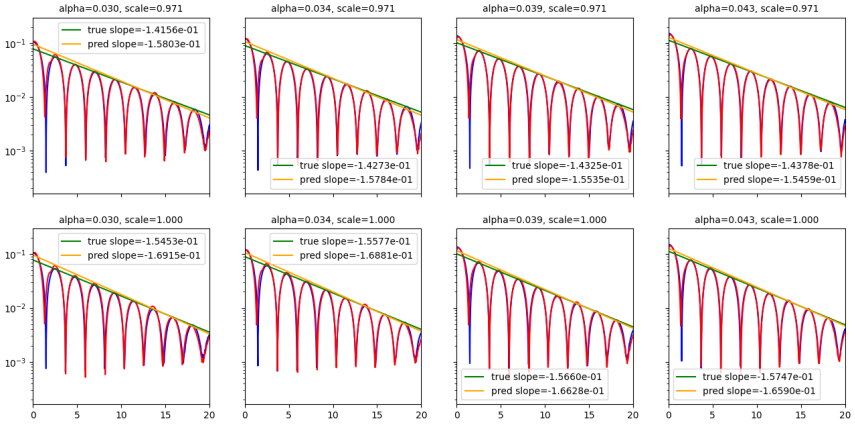


Figure: Some damping rates predictions for various $\mu \in \Gamma$, $K = 3$.

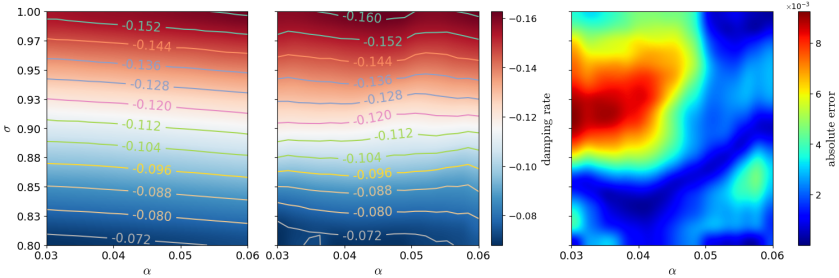


Figure: Electric energy $\frac{1}{2}\|E\|_2\left(x(t;\mu);\mu\right), \mu \in \Gamma$ exponential damping rates of the FOM (left), the ROM (center) and absolute error (right), $K = 3$.

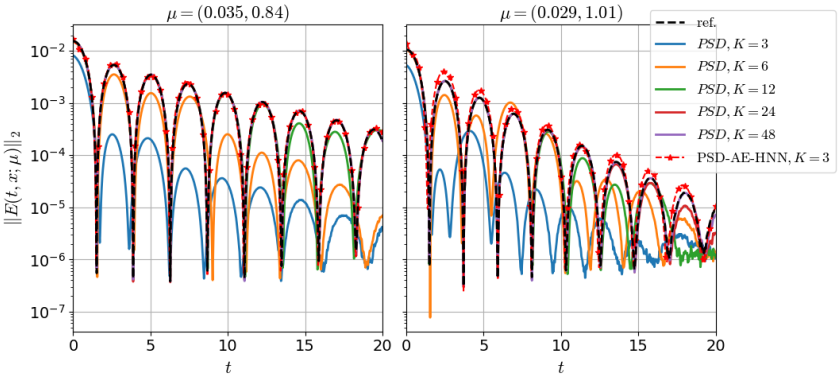


Figure: Electric energies $\frac{1}{2}\|E\|_2\,(x(t;\mu)))$ of the PSD reduced model against our method for $\mu = (0.035, 0.84) \in \Gamma$ (left) and $\mu = (0.029, 1.01) \notin \Gamma$ (right), $K = 3$.

- equivalent precision with $K = 30$ PSD modes.

- small HNN $\sim 10^3$ parameters : competitive,
- offline time :
  - full order PIC : 25s,
  - PIC with comparable accuracy ($N = 7 \times 10^4$) : 11s,
  - PSD-AE-HNN reduced model : 2s,
- Difficult to quantify acceleration: hardware, software, noise, developper expertise.

- What happens if we ignore the Hamiltonian structure ?

- What happens if we ignore the Hamiltonian structure ?

- Learn directly the vector field of the reduced dynamics

$$\frac{d}{dt}\bar{u}(t;\mu) = \bar{\mathcal{F}}_\theta(\bar{u}(t;\mu))$$

- test on the (nonlinear) Landau damping :
  $T = 20 \to 40, \Gamma = [0.46, 0.5] \times [0.96, 1], K = 4$, other parameters unchanged.

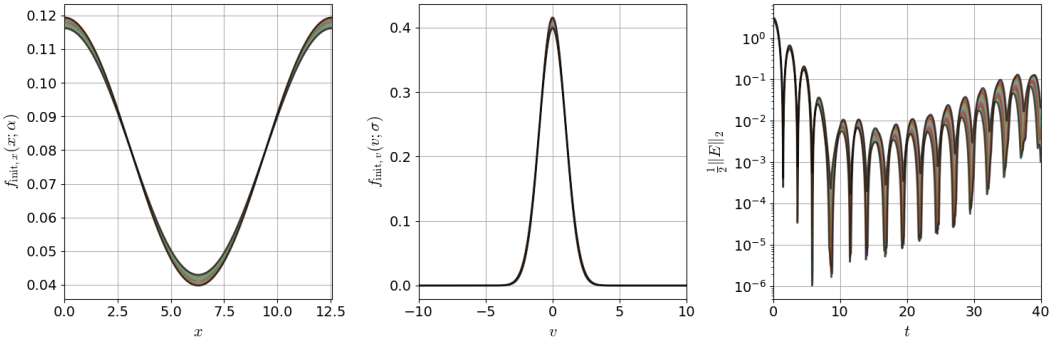


Figure: Initial distribution $f_{\text{init},x}(x;\alpha)$ (left), $f_{\text{init},v}(x;\sigma)$ (middle) and evolution of the electric energy $\frac{1}{2}\|E\|_2\left(x(t;\mu);\mu\right)$ (right) for every $\mu \in \Gamma^{\text{train}}$.

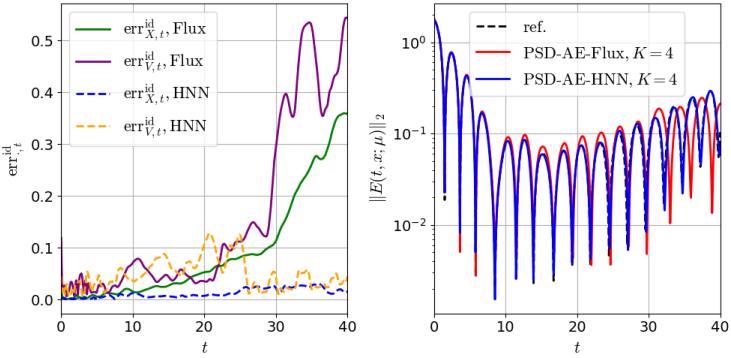


Figure: PSD-AE-Flux prediction for a single test parameter μ compared to the PSD-AE-HNN method. Errors as a function of time (left) and predicted electric energy $\frac{1}{2}\|E\|_2\left(x(t;\mu)\right)$).

- its prediction quickly drifts from the reference.

- Reduction **in the number of particles** of a PIC discretization of the Vlasov Poisson equation,

- two-step mapping combining the PSD and the CAE for an efficient nonlinear compression,

- Reduction **in the number of particles** of a PIC discretization of the Vlasov Poisson equation,
- two-step mapping combining the PSD and the CAE for an efficient nonlinear compression,
- strong performance in each test case and good computational efficiency,

- Reduction **in the number of particles** of a PIC discretization of the Vlasov Poisson equation,
- two-step mapping combining the PSD and the CAE for an efficient nonlinear compression,
- strong performance in each test case and good computational efficiency,
- same strengths/limitations as the AE-HNN.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,
  - neural networks : computationally intensive training, very efficient reduced dynamics prediction, highly parallelizable on GPUs,

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,
  - neural networks : computationally intensive training, very efficient reduced dynamics prediction, highly parallelizable on GPUs,
  - joint training strategy : a workaround to the construction of nonlinear symplectic projections,

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,
  - neural networks : computationally intensive training, very efficient reduced dynamics prediction, highly parallelizable on GPUs,
  - joint training strategy : a workaround to the construction of nonlinear symplectic projections,
- their weaknesses are common in scientific machine learning applications:

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,
  - neural networks : computationally intensive training, very efficient reduced dynamics prediction, highly parallelizable on GPUs,
  - joint training strategy : a workaround to the construction of nonlinear symplectic projections,
- their weaknesses are common in scientific machine learning applications:
  - neural network training = high-dimensional minimization process (non convex, local minima, no clear guarantee of global convergence),

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,
  - neural networks : computationally intensive training, very efficient reduced dynamics prediction, highly parallelizable on GPUs,
  - joint training strategy : a workaround to the construction of nonlinear symplectic projections,
- their weaknesses are common in scientific machine learning applications:
  - neural network training = high-dimensional minimization process (non convex, local minima, no clear guarantee of global convergence),
  - hyperparameters tuning remains quite empirical,

[7] Brivio et al. 2024.

Conclusion & perspectives

- We developed generic and non-intrusive reduction methods:
  - data-driven : adapt to the structure at hand and learns tailored mappings - without manipulating governing equations,
  - neural networks : computationally intensive training, very efficient reduced dynamics prediction, highly parallelizable on GPUs,
  - joint training strategy : a workaround to the construction of nonlinear symplectic projections,
- their weaknesses are common in scientific machine learning applications:
  - neural network training = high-dimensional minimization process (non convex, local minima, no clear guarantee of global convergence),
  - hyperparameters tuning remains quite empirical,
  - no systematic way to improve accuracy, few available results on errors bounds[7].

[7] Brivio et al. 2024.

- Main improvement : systematically enhance the accuracy of the reduced model (currently: manual hyperparameter tuning) with automation, Bayesian optimization, genetic algorithm, sensitivity analysis on hyperparameters ?

[7] Brantner and Kraus 2023.

[8] Choudhary et al. 2021; Gruber and Tezaur 2023; Jin et al. 2023.

[9] Desai et al. 2021.

[10] Zhang et al. 2022.

- Main improvement : systematically enhance the accuracy of the reduced model (currently: manual hyperparameter tuning) with automation, Bayesian optimization, genetic algorithm, sensitivity analysis on hyperparameters ?

- primary limitation : AE and HNN have potentially competing objectives, design a symplectic AE[7] ?

[7] Brantner and Kraus 2023.

[8] Choudhary et al. 2021; Gruber and Tezaur 2023; Jin et al. 2023.

[9] Desai et al. 2021.

[10] Zhang et al. 2022.

- Main improvement : systematically enhance the accuracy of the reduced model (currently: manual hyperparameter tuning) with automation, Bayesian optimization, genetic algorithm, sensitivity analysis on hyperparameters ?

- primary limitation : AE and HNN have potentially competing objectives, design a symplectic AE[7] ?

- As of now, we learned canonical Hamiltonian systems: possible extensions to non-canonical Hamiltonian systems[8] or even dissipative systems with port-Hamiltonian[9] or GFINNs[10] frameworks.

[7] Brantner and Kraus 2023.

[8] Choudhary et al. 2021; Gruber and Tezaur 2023; Jin et al. 2023.

[9] Desai et al. 2021.

[10] Zhang et al. 2022.

Bibliography I

Brantner, Benedikt and Michael Kraus (2023). *Symplectic Autoencoders for Model Reduction of Hamiltonian Systems*. DOI: 10.48550/arXiv.2312.10004. arXiv: 2312.10004 [cs.LG].

Brivio, Simone, Stefania Fresca, Nicola Rares Franco, and Andrea Manzoni (2024). "Error estimates for POD-DL-ROMs: a deep learning framework for reduced order modeling of nonlinear parametrized PDEs enhanced by proper orthogonal decomposition". In: *Adv. Comput. Math.* 50.3, p. 33. ISSN: 1572-9044. DOI: 10.1007/s10444-024-10110-1.

Casas, Fernando, Nicolas Crouseilles, Erwan Faou, and Michel Mehrenberger (2017). "High-order Hamiltonian splitting for the Vlasov-Poisson equations". In: *Numer. Math.* 135.3, pp. 769–801. ISSN: 0029-599X,0945-3245. DOI: 10.1007/s00211-016-0816-z.

Choudhary, Anshul, John F. Lindner, Elliott G. Holliday, Scott T. Miller, Sudeshna Sinha, and William L. Ditto (2021). "Forecasting Hamiltonian dynamics without canonical coordinates". In: *Nonlinear Dyn.* 103.2, pp. 1553–1562. ISSN: 1573-269X. DOI: 10.1007/s11071-020-06185-2.

Côte, Raphaël, Emmanuel Franck, Laurent Navoret, G. S., and Vincent Vigon (2025). "Hamiltonian reduction using a convolutional auto-encoder coupled to a Hamiltonian neural network". In: *Commun. Comput. Phys.* 37.2, pp. 315–352. ISSN: 1815-2406,1991-7120. DOI: 10.4208/cicp.OA-2023-0300.

Desai, Shaan A., Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J. Roberts (2021). "Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems". In: *Phys. Rev. E* 104 (3), p. 034312. DOI: 10.1103/PhysRevE.104.034312.

Franck, Emmanuel, Laurent Navoret, Vincent Vigon, Raphaël Côte, and G. S. (June 2025). "Reduced Particle in Cell method for the Vlasov-Poisson system using auto-encoder and Hamiltonian neural". working paper or preprint. URL: https://hal.science/hal-05116555.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. URL: http://www.deeplearningbook.org.

Greydanus, Sam, Misko Dzamba, and Jason Yosinski (2019). "Hamiltonian neural networks". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates Inc. DOI: 10.48550/arXiv.1906.01563.

Gruber, Anthony and Irina Tezaur (2023). "Canonical and noncanonical Hamiltonian operator inference". In: *Comput. Methods Appl. Mech. Engrg.* 416, Paper No. 116334, 45. ISSN: 0045-7825,1879-2138. DOI: 10.1016/j.cma.2023.116334.

Hairer, Ernst, Christian Lubich, and Gerhard Wanner (2006). *Geometric numerical integration*. Second. Vol. 31. Springer Series in Computational Mathematics. Structure-preserving algorithms for ordinary differential equations. Springer-Verlag, Berlin, pp. xviii+644. ISBN: 978-3-540-30663-4.

Hesthaven, Jan S., Cecilia Pagliantini, and Nicolò Ripamonti (2024). "Adaptive symplectic model order reduction of parametric particle-based Vlasov-Poisson equation". In: *Math. Comp.* 93.347, pp. 1153–1202. ISSN: 0025-5718,1088-6842. DOI: 10.1090/mcom/3885.

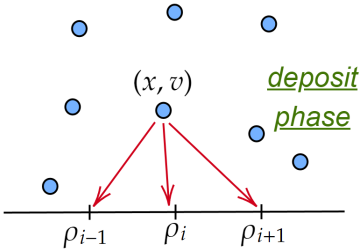

Jin, Pengzhan, Zhen Zhang, Ioannis G. Kevrekidis, and George Em Karniadakis (2023). "Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks". In: *IEEE Trans. Neural Netw. Learn. Syst.* 34.11, pp. 8271–8283. ISSN: 2162-237X,2162-2388. DOI: 10.1109/tnnls.2022.3148734.


Kraus, Michael, Katharina Kormann, Philip J. Morrison, and Eric Sonnendrücker (2017). "GEMPIC: geometric electromagnetic particle-in-cell methods". In: *J. Plasma Phys.* 83.4. ISSN: 1469-7807. DOI: 10.1017/s002237781700040x.


Lange, Kenneth (2010). "Singular Value Decomposition". In: *Numerical analysis for statisticians*. Second. Statistics and Computing. Springer, New York, pp. 129–142. ISBN: 978-1-4419-5945-4. DOI: 10.1007/978-1-4419-5945-4_9.

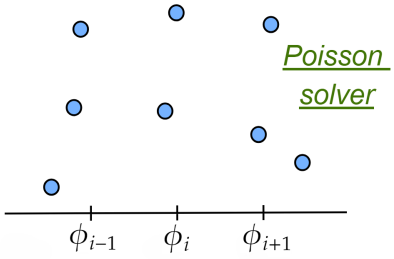

Peng, Liqian and Kamran Mohseni (2016). "Symplectic model reduction of Hamiltonian systems". In: *SIAM J. Sci. Comput.* 38.1, A1–A27. ISSN: 1064-8275,1095-7197. DOI: 10.1137/140978922.

 Zhang, Zhen, Yeonjong Shin, and George Em Karniadakis (2022). "GFINNs: GENERIC formalism informed neural networks for deterministic and stochastic dynamical systems". In: *Philos. Trans. Roy. Soc. A* 380.2229, Paper No. 20210207, 21. ISSN: 1364-503X,1471-2962. DOI: 10.1098/rsta.2021.0207.



- Compute electric density on the grid

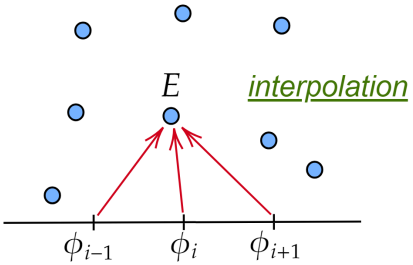
$$\rho^n_{\text{grid}} = \Lambda^0(x(t;\mu))^T \mathbb{1}_N.$$



- Solve Poisson equation

$$\phi^n_{\text{grid}} = L^{-1}\rho^n_{\text{grid}} = L^{-1}\Lambda^0(x(t;\mu))^T \mathbb{1}_N$$

- $L \in \mathcal{M}_{n_x}(\mathbb{R})$ is a discrete Laplace operator.



- Interpolate electric field at particles position

$$E^n = \nabla\Lambda^0(x)\phi^n_{\text{grid}} = \nabla\Lambda^0(x)L^{-1}\Lambda^0(x(t;\mu))^T\mathbb{1}_N.$$