The neural semi-Lagrangian method

Solving high-dimensional advection-diffusion problems

Victor Michel-Dansac*, joint work with
Emmanuel Franck, Laurent Navoret, Vincent Vigon

EMS TAG SciML, Köln, March 11, 2026

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

1. Motivation: High-dimensional advection-diffusion equations

Motivation: High-dimensional advection-diffusion equations

High-dimensional advection-diffusion equations are ubiquitous, appearing for instance in:

1. Plasma Physics & Astrophysics
Vlasov or Vlasov-Poisson Equations

The unknown $u(t, x, v)$ represents a probability density in **phase space** ($d = 6$).
Is it critical for modeling fusion in Tokamaks, e.g. ITER, to capture dynamics where fluid models fail.

2. Parameter-dependent equations
Uncertainty Quantification & Design Optimization

Equations where u depends on physical parameters μ (e.g., advection velocity a).
The unknown $u(t, x, \mu)$ has inputs in a space of dimension $1 + d + p$.

Motivation: High-dimensional advection-diffusion equations

High-dimensional advection-diffusion equations are ubiquitous, appearing for instance in:

1. Plasma Physics & Astrophysics
Vlasov or Vlasov-Poisson Equations

The unknown $u(t, x, v)$ represents a probability density in **phase space** ($d = 6$).
Is it critical for modeling fusion in Tokamaks, e.g. ITER, to capture dynamics where fluid models fail.

2. Parameter-dependent equations
Uncertainty Quantification & Design Optimization

Equations where u depends on physical parameters μ (e.g., advection velocity a).
The unknown $u(t, x, \mu)$ has inputs in a space of dimension $1 + d + p$.

Main bottleneck: The Curse of Dimensionality
Grid-based methods scale as N^{d+p}. For $d + p = 6$, even $N = 30$ requires approx. 10^9 nodes!

Our Goal: Combine the **Semi-Lagrangian** approach (exact transport along characteristics) with **Neural Networks** (efficient low-rank manifold) to break the curse of dimensionality.

2. The advection equation and the semi-Lagrangian method

Advection at constant velocity

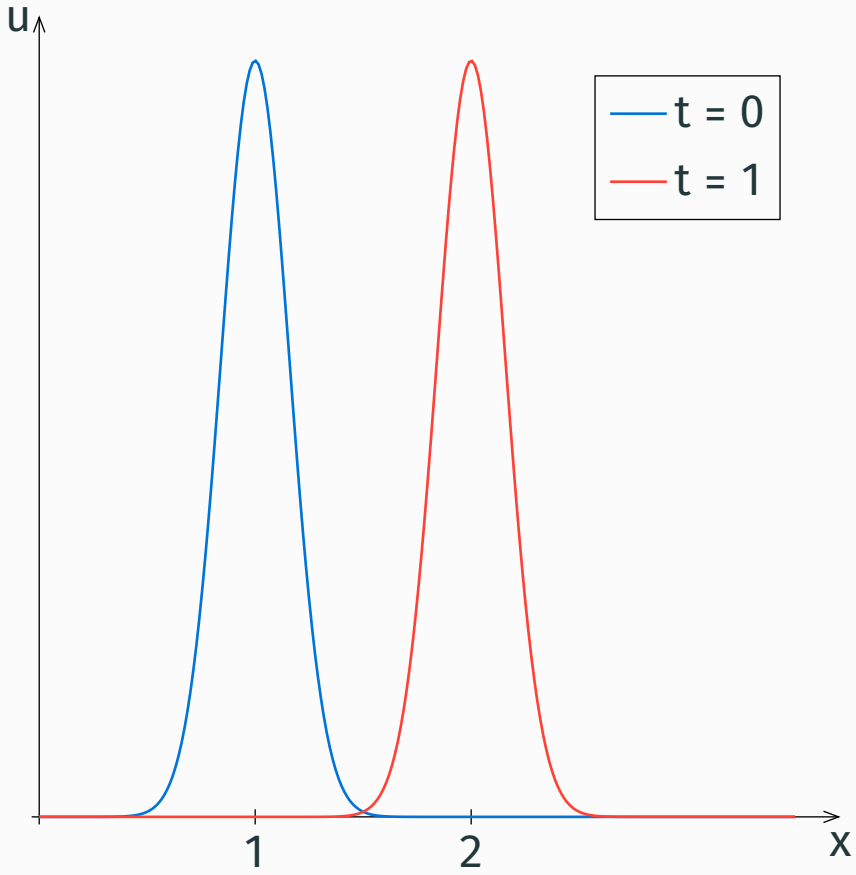
Let us consider the case of a pure advection equation:

$$\begin{cases} \partial_t u(t,x) + a(t,x) \cdot \nabla_x u(t,x) = 0 & \text{for} \quad t \in [0,T] \text{ and } x \in \Omega, \\ u(0,x) = u_0(x) & \text{for} \quad x \in \Omega. \end{cases}$$

If the advection field a is constant, the solution to this PDE is simply the transport of the initial condition u_0 at velocity a:

$$\forall t \in [0,T], \ \forall x \in \Omega, \qquad u(t,x) = u_0(x - at).$$

This is illustrated in the figure opposite, where a Gaussian function is advected at speed a = 1 in the time interval [0, 1].



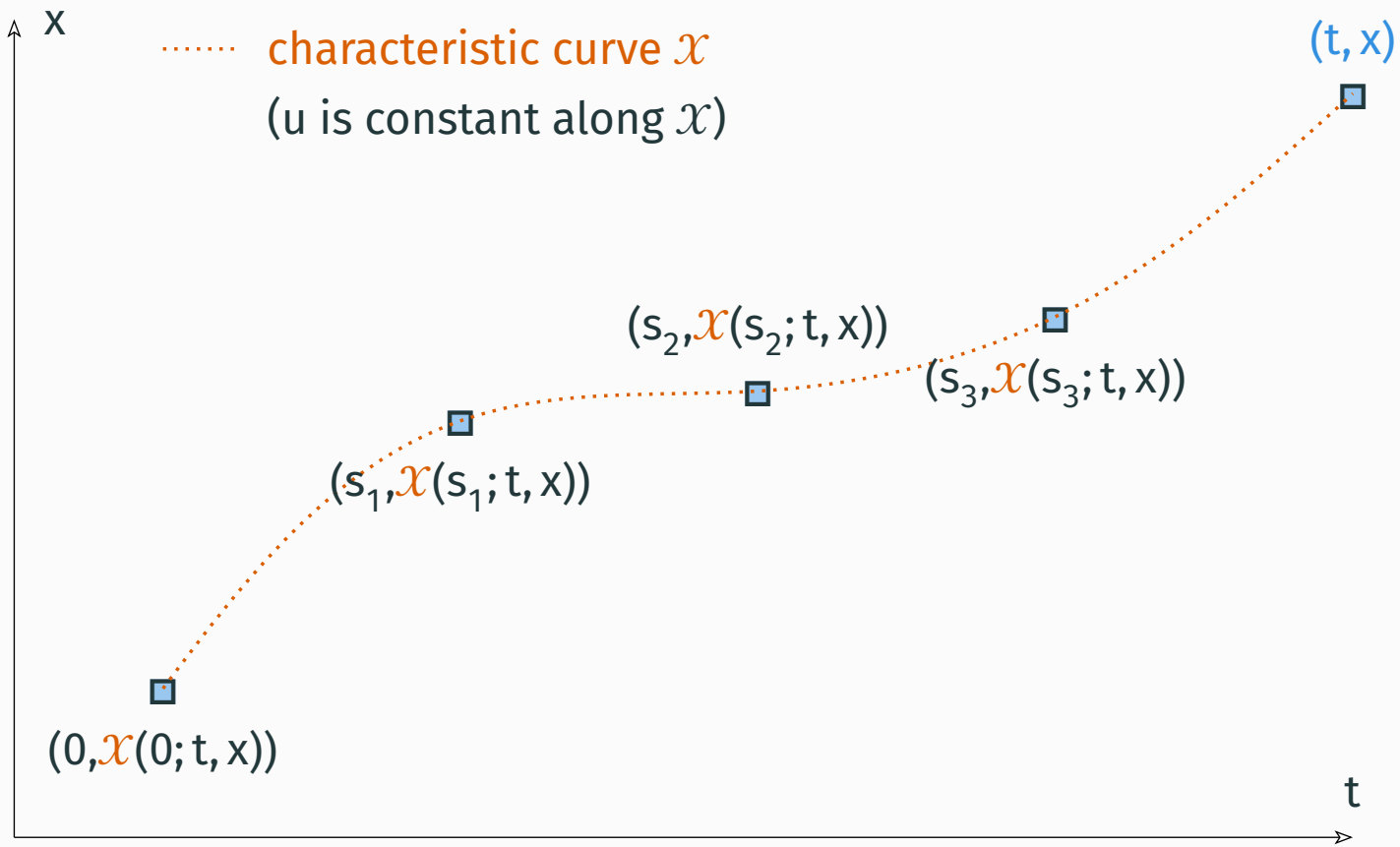
Advection at nonconstant velocity

The advection equation with variable velocity a admits the solution

$$\forall x \in \Omega, \forall t \in [0,T], \quad \forall s \in [0,t],$$
$$u(t,x) = u(s, \mathcal{X}(s;t,x)).$$

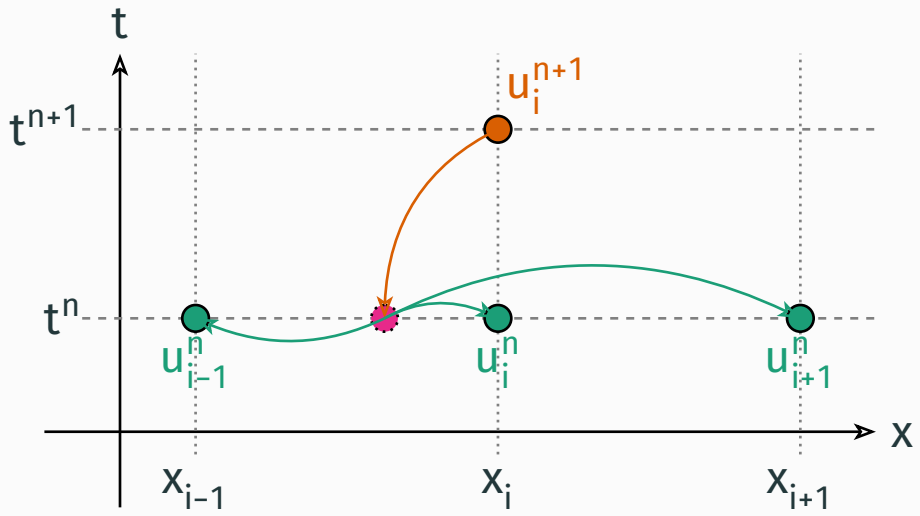
The characteristic curve $\mathcal{X}$ solves this backwards ODE for given t and x:

$$\begin{cases} \frac{d}{ds}\mathcal{X}(s;t,x) = a(s,\mathcal{X}(s;t,x)), & \forall s \in [0,t], \\ \mathcal{X}(t;t,x) = x. \end{cases}$$



The semi-Lagrangian method [Staniforth et al., 1991]

classical semi-Lagrangian scheme:
update for one mesh point



- step 1: project ● onto ● using the approximate characteristic curve
- step 2: interpolate the value at ● from the neighbors ●

This method **does not have a stability restriction** on the time step Δt!

However, it is still subject to the curse of dimensionality. Could a neural method be used here?

3. Time-discrete neural methods for PDEs

Physics-informed neural networks (PINNs)

We consider a generic partial differential equation (PDE) of the form

$$\partial_t u(t,x) + \mathcal{L}[u](t,x) = 0, \qquad \text{for} \quad t \in [0,T] \text{ and } x \in \Omega.$$

PINNs [Raissi et al., 2019] approximate the solution u by a neural network

$$u_\theta : (t,x) \mapsto \mathcal{N}(t,x,\theta),$$

whose **dofs (trainable parameters)** θ **are global constants** in time and space.

We seek θ by minimizing the **global space-time residual** of the PDE:

$$\theta \in \operatorname*{argmin}_{\vartheta \in \Theta} \int_0^T \int_\Omega \left| \partial_t u_\vartheta(t,x) + \mathcal{L}[u_\vartheta](t,x) \right|^2 \mathrm{d}x \, \mathrm{d}t.$$

This leads to a **global space-time optimization problem**, which may be difficult to solve in practice. Like in conventional numerical methods, one can also consider **time-discrete** versions of PINNs.

Time-Discrete Neural Methods

In this case, the solution u is approximated by a neural network whose **dofs** θ **evolve in time**:

$$u_{\theta(t)} : x \mapsto \mathcal{N}(x, \theta(t)).$$

For each time $t \in [0, T]$, the dofs $\theta(t)$ should minimize the PDE residual

$$\int_\Omega \left| \partial_t u_{\theta(t)}(x) + \mathcal{L}[u_{\theta(t)}](x) \right|^2 dx.$$

Time-Discrete Neural Methods

In this case, the solution u is approximated by a neural network whose **dofs** θ **evolve in time**:

$$u_{\theta(t)} : x \mapsto \mathcal{N}(x, \theta(t)).$$

For each time $t \in [0, T]$, the dofs $\theta(t)$ should minimize the PDE residual

$$\int_\Omega \left| \partial_t u_{\theta(t)}(x) + \mathcal{L}[u_{\theta(t)}](x) \right|^2 dx.$$

1. Discrete PINNs [Stiasny et al., 2023]:
Discretize then Optimize

Discretize the residual in time (e.g., Explicit Euler) and **solve an optimization problem** for θ^{n+1}:

$$\theta^{n+1} \in \operatorname*{argmin}_{\vartheta \in \Theta} \int_\Omega \left| u_\vartheta(x) - u_{\theta^n}(x) + \Delta t \mathcal{L} u_{\theta^n}(x) \right|^2 dx.$$

2. Neural Galerkin [Bruna et al., 2024]:
Optimize then Discretize

Use the chain rule $\partial_t u_{\theta(t)} = \nabla_\theta u_{\theta(t)} \cdot \frac{d\theta}{dt}$, yielding a quadratic problem, and minimize the residual to find an **ODE for the dofs**. Discretizing it yields:

$$M(\theta^n)\theta^{n+1} = M(\theta^n)\theta^n + \Delta t L(\theta^n).$$

The PDE is reduced to either a **sequence of optimizations** (Discrete PINNs) or a **system of ODEs** (Neural Galerkin).

Characteristics of each method

	PINNs	discrete PINNs	neural Galerkin
nonlinear optimization	one	one PINN per time step	one at initialization
linear system	no	no	full, in $\mathbb{R}^N$ one system per step
stability restriction on Δt	N/A (not a discrete method)	yes	yes
applicable to	any PDE	any time-dependent PDE	any time-dependent PDE

Characteristics of each method

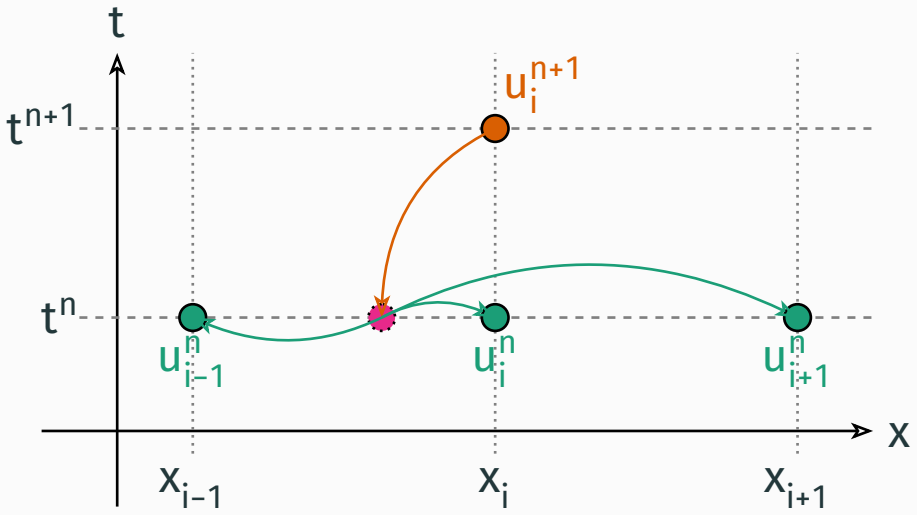
	PINNs	discrete PINNs	neural Galerkin	**neural semi-Lagrangian**
nonlinear optimization	one	one PINN per time step	one at initialization	**one function fit per time step**
linear system	no	no	full, in $\mathbb{R}^N$ one system per step	**no**
stability restriction on Δt	N/A (not a discrete method)	yes	yes	**no**
applicable to	any PDE	any time-dependent PDE	any time-dependent PDE	**advection-diffusion equations**

4. The neural semi-Lagrangian method: methodology

[Franck, Michel-Dansac, Navoret & Vigon, CMAME, 2026]

Classical Neural Semi-Lagrangian methods

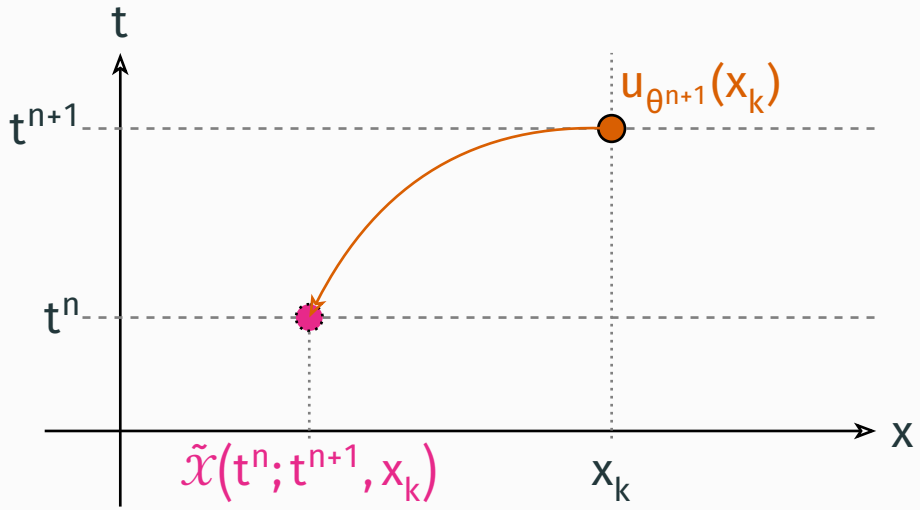
classical semi-Lagrangian scheme:
update for one mesh point



• step 1: project ● onto ● using the approximate characteristic curve

• step 2: interpolate the value at ● from the neighbors ●

neural semi-Lagrangian scheme:
update for one sampled point



• step 1: starting from ●, compute the foot ● of the approximate characteristic curve

• step 2: fit $u_{\theta^{n+1}}(x_k)$ on $u_{\theta^n}(\tilde{x}(t^n; t^{n+1}, x_k))$

Optimization problem

The exact solution of the advection equation satisfies $u(t^{n+1}, x) = u(t^n, \mathcal{X}(t^n; t^{n+1}, x))$.

Thus, we seek the dofs θ such that

$$\forall n \geq 0, \quad \forall x \in \Omega, \qquad u_{\theta^{n+1}}(x) \approx u_{\theta^n}(\mathcal{X}(t^n; t^{n+1}, x)).$$

This leads us to considering the following optimization problem, linking the dofs at time t^n and t^{n+1}:

$$\theta^{n+1} \in \operatorname*{argmin}_{\vartheta \in \Theta} \int_\Omega \left| u_\vartheta(x) - u_{\theta^n}(\mathcal{X}(t^n; t^{n+1}, x)) \right|^2 dx.$$

This nonlinear optimization problem is simply a **function approximation problem**!

Algorithm for the parametric problem

Algorithm: Neural semi-Lagrangian method for a **parametric advection equation**

$$\partial_t u(t,x,\mu) + a(t,x,\mu)\cdot\nabla u(t,x,\mu) = 0 \quad \text{with initial condition} \quad u(0,x,\mu) = u_0(x,\mu).$$

1. **initialization**: compute the initial dofs θ^0 by solving

$$\theta^0 \in \operatorname*{argmin}_{\vartheta\in\Theta} \int_{\mathbb{P}}\int_{\Omega} \left|u_\vartheta(x,\mu) - u_0(x,\mu)\right|^2 \mathrm{d}x\,\mathrm{d}\mu$$

2. **for** $n \in \{0,\dots,n_T-1\}$:
   i. approximately solve the characteristic ODE to find $\tilde{x}(t^n; t^{n+1}, x, \mu)$
   ii. compute the dofs θ^{n+1} by solving

$$\theta^{n+1} \in \operatorname*{argmin}_{\vartheta\in\Theta} \int_{\mathbb{P}}\int_{\Omega} \left|u_\vartheta(x,\mu) - u_{\theta^n}\big(\tilde{x}(t^n; t^{n+1}, x, \mu)\big)\right|^2 \mathrm{d}x\,\mathrm{d}\mu$$

Rough error estimate

Theorem: Let u be the exact solution of the pure transport equation on a cuboid Ω with periodic boundary conditions, and let $\tilde{u}^n$ be the approximate solution at time t^n. Then, we have, for all n such that $t^n < T$,

$$\|\tilde{u}^n - u(t^n, \cdot)\|_{L^2(\Omega)} \leqslant \sum_{m=0}^{n} \varepsilon_{\text{int}}^m + \sum_{m=0}^{n} \varepsilon_{\text{opt}}^m + \sum_{m=0}^{n} \varepsilon_{\text{approx}}^m + \frac{CT}{n_\tau}\left(\frac{\Delta t}{n_\tau}\right)^p,$$

where $C = \text{cst}$ is independent of Δt, p and n_τ are the order and number of sub-time steps of the characteristic curve solver, and $\varepsilon_{\text{opt}}^m$, $\varepsilon_{\text{int}}^m$, and $\varepsilon_{\text{approx}}^m$ are, respectively, upper bounds of the optimization, integration and approximation errors.

Sketch of proof: With u^{n+1} the true solution of the optimization problem, the proof splits the error:

$$\begin{aligned}\left\|\tilde{u}^{n+1} - u(t^{n+1}, \cdot)\right\|_{L^2(\Omega)} \leqslant \left\|\tilde{u}^{n+1} - u^{n+1}\right\|_{L^2(\Omega)} + \left\|u^{n+1} - u_{\theta^n} \circ \tilde{\mathcal{X}}^{n+1}\right\|_{L^2(\Omega)} + \\ \left\|u_{\theta^n} \circ \tilde{\mathcal{X}}^{n+1} - u_{\theta^n} \circ \mathcal{X}^{n+1}\right\|_{L^2(\Omega)} + \left\|u_{\theta^n} \circ \mathcal{X}^{n+1} - u(t^n, \cdot) \circ \mathcal{X}^{n+1}\right\|_{L^2(\Omega)}.\end{aligned}$$

Each term is then identified with the corresponding error, which completes the proof. □

5. The neural semi-Lagrangian method: numerical results

[Franck, Michel-Dansac, Navoret & Vigon, CMAME, 2026]

The Scimba library

Scimba is a `Python` + `pytorch` **library** containing SciML methods for PDEs. A `jax` version is in development.

The current version of the code solves **parametric PDEs** using nonlinear approximation spaces such as neural networks.

To achieve this, the code provides **several optimization strategies**, including Adam and **natural gradient** methods.

More information is available at:

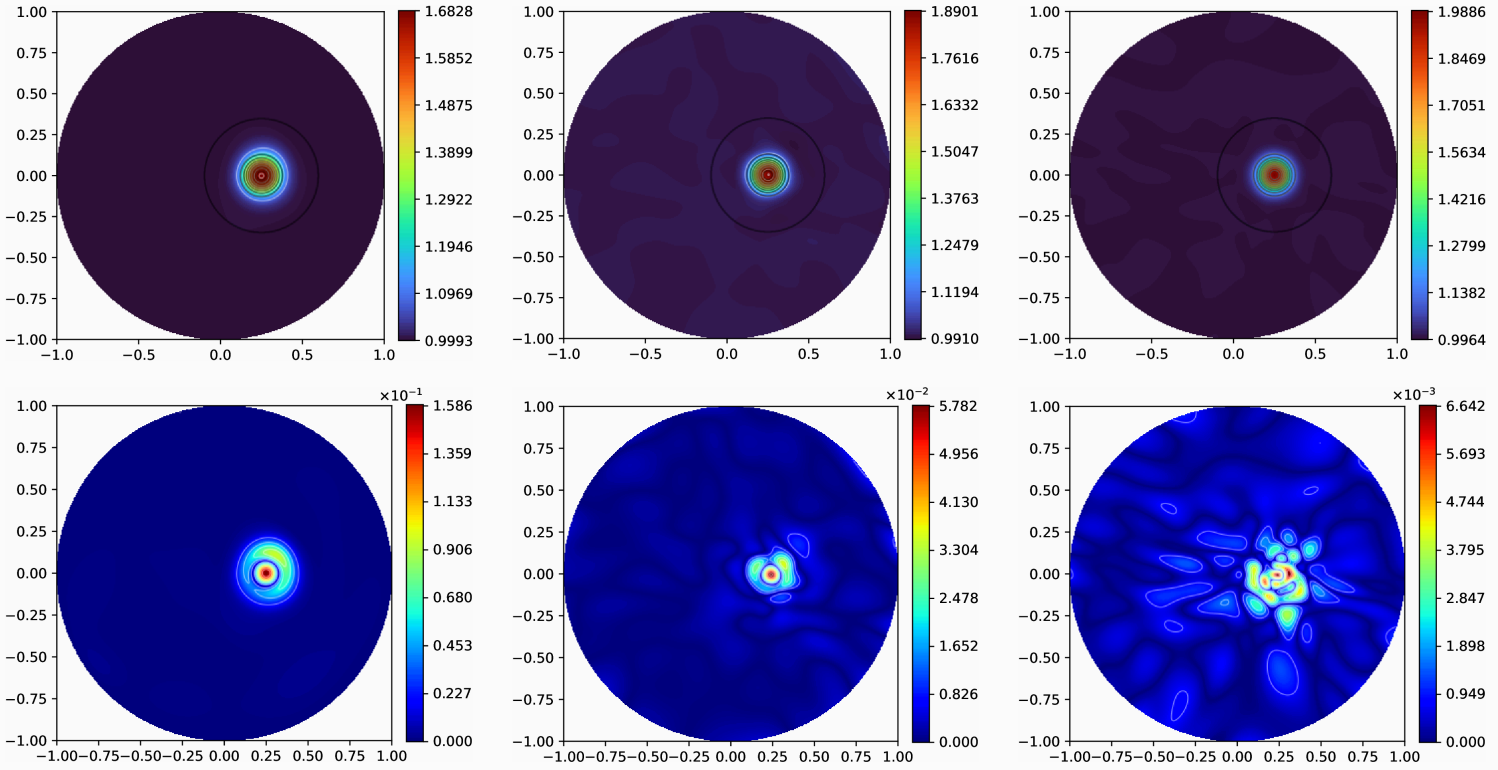
https://www.scimba.org

Rotating pulse in 2D (space) + 2D (parameters)

$$u_{ex}(t, x, y; c, v) = 1 + \exp\left(-\frac{1}{2v^2}\left((x - c\cos(2\pi t))^2 + (y - c\sin(2\pi t))^2\right)\right), \qquad t \in [0, 1],\ x \in \mathbb{D}^1,\ c \in [0.2, 0.4],\ v \in [0.05, 0.1]$$

prediction at t = 1

error at t = 1



PINN | neural Galerkin | neural semi-Lagrangian

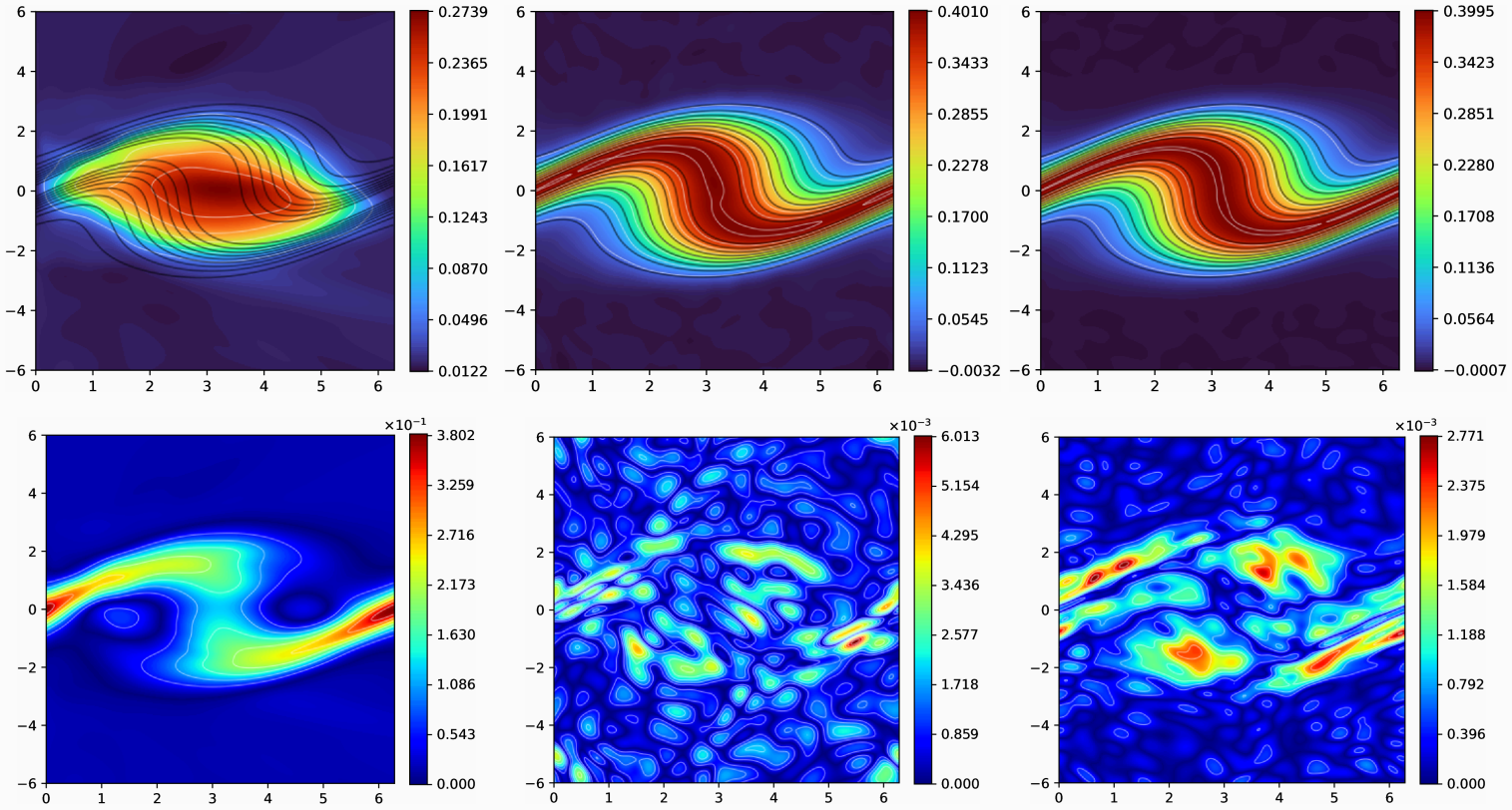
	PINN	NG	NSL
N	13500	6800	6800
Δt	N/A	0.05	0.2
GPU time	5 min	30 sec	30 sec
error $\Omega \times \mathcal{P}$	1.5×10^{-1}	5.7×10^{-2}	6.6×10^{-3}

Vlasov equation in 1D (space) + 1D (velocity)

Vlasov equation: $\partial_t f(t,x,v) + v\,\partial_x f(t,x,v) + \sin(x)\,\partial_v f(t,x,v) = 0, \quad t \in [0,4.5],\ x \in [0,2\pi],\ v \in [-6,6]$

prediction at t = 1.5

error at t = 1.5



PINN neural Galerkin neural semi-Lagrangian

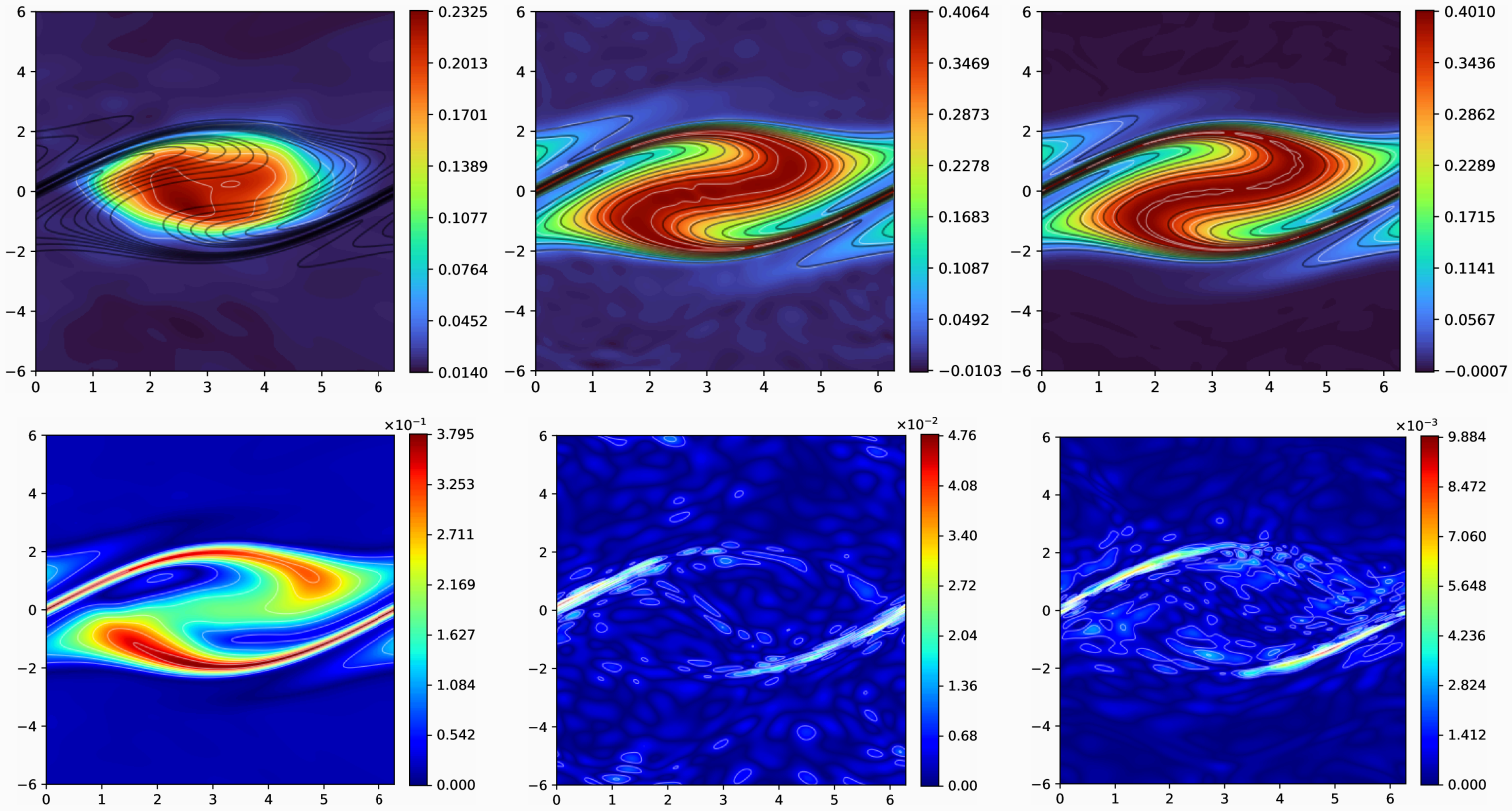
	PINN	NG	NSL
N	20000	6800	6800
Δt	N/A	0.01	1.5
GPU time	5 min	4.5 min	2 min
error $\Omega \times \mathcal{P}$	3.8×10^{-1}	6.0×10^{-3}	2.7×10^{-3}

Vlasov equation in 1D (space) + 1D (velocity)

Vlasov equation: $\partial_t f(t,x,v) + v\,\partial_x f(t,x,v) + \sin(x)\,\partial_v f(t,x,v) = 0, \quad t \in [0,4.5],\ x \in [0,2\pi],\ v \in [-6,6]$

prediction at t = 3

error at t = 3



PINN | neural Galerkin | neural semi-Lagrangian

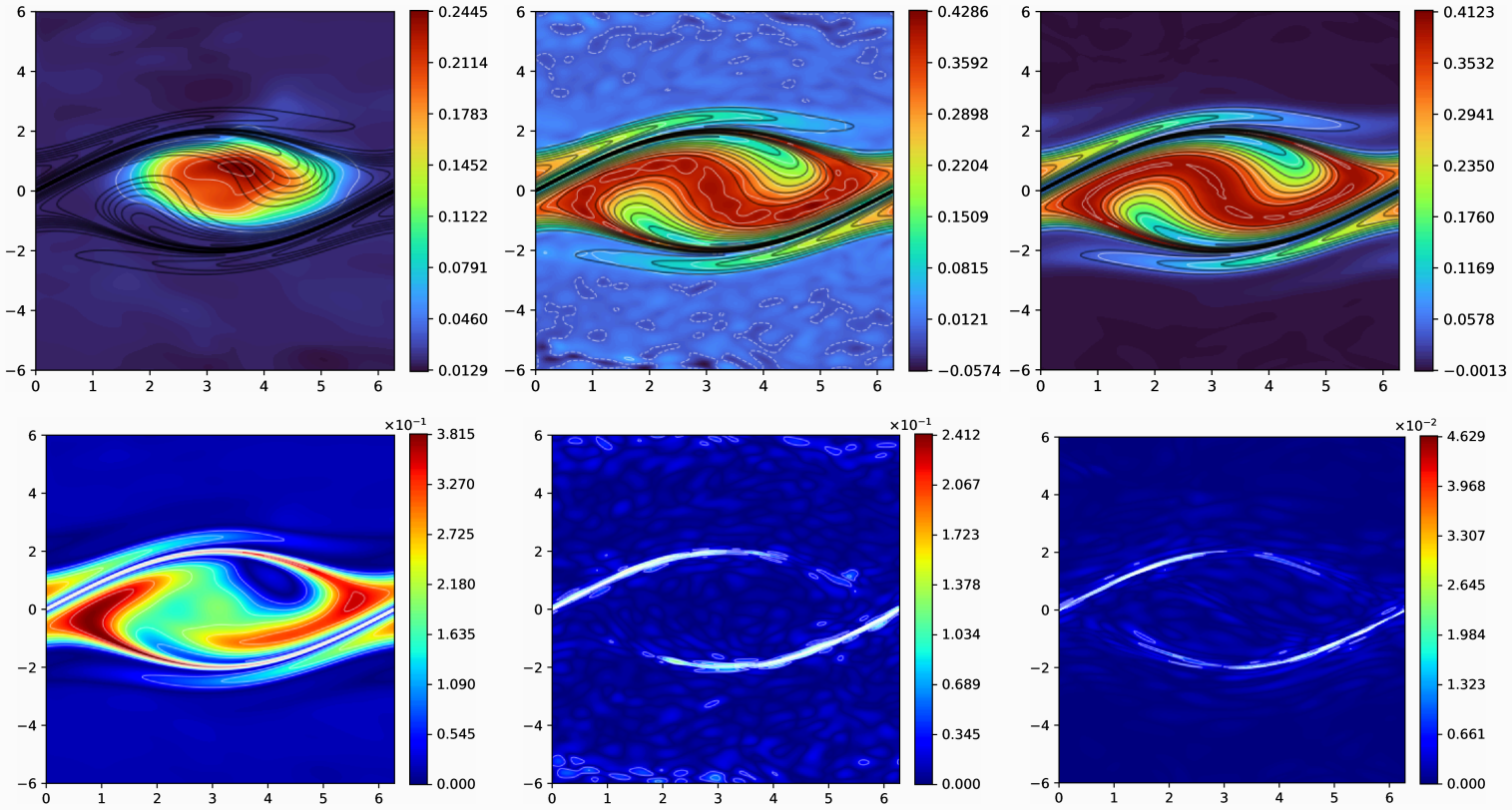
	PINN	NG	NSL
N	20000	6800	6800
Δt	N/A	0.01	1.5
GPU time	5 min	4.5 min	2 min
error $\Omega \times \mathcal{P}$	3.8×10^{-1}	4.8×10^{-2}	9.8×10^{-3}

Vlasov equation in 1D (space) + 1D (velocity)

Vlasov equation: $\partial_t f(t,x,v) + v\,\partial_x f(t,x,v) + \sin(x)\,\partial_v f(t,x,v) = 0, \quad t \in [0,4.5],\ x \in [0,2\pi],\ v \in [-6,6]$

prediction at t = 4.5

error at t = 4.5



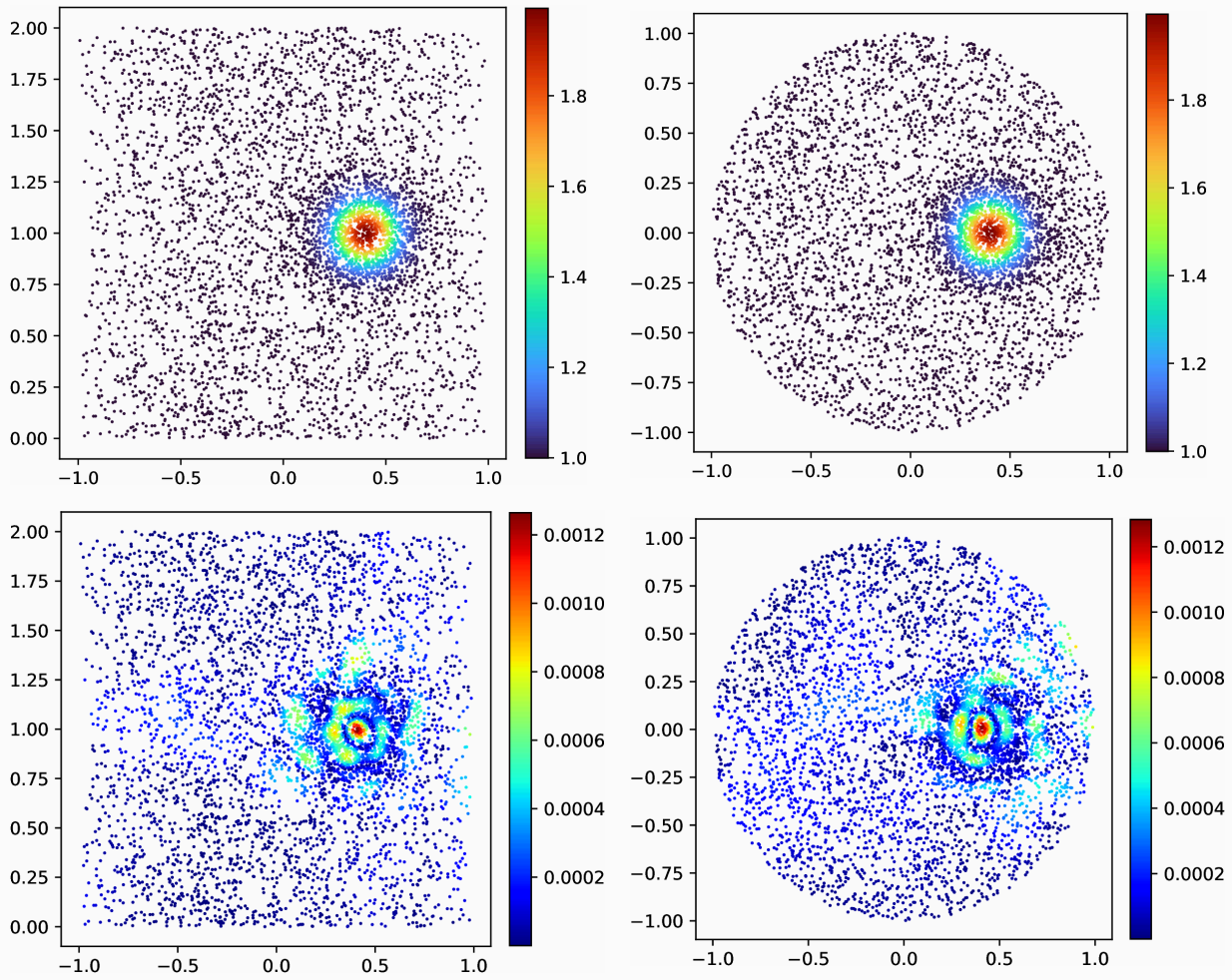
PINN | neural Galerkin | neural semi-Lagrangian

	PINN	NG	NSL
N	20000	6800	6800
Δt	N/A	0.01	1.5
GPU time	5 min	4.5 min	2 min
error $\Omega \times \mathcal{P}$	3.8×10^{-1}	2.4×10^{-1}	4.6×10^{-2}

Rotated and advected pulse in 3D (space) + 2D (parameters)

prediction at t = 2

error at t = 2



y = 0 plane

z = 1 plane

- $\Omega = \mathbb{D}^1 \times [0, 2]$ is a cylinder
- initial pulse of amplitude 1, rotated in the (x, y)-plane and advected along the z axis
- improvements in the NSL method:
  - adaptive sampling
  - **optimization with Gauss-Newton**
- configuration of the NSL method:
  - $\Delta t = 0.5$
  - $N \approx 7700$
- GPU time:
  - 13 min (initialization)
  - 21 min (time-stepping)
  - total: 34 min
- L^∞ error in $\Omega \times \mathcal{P}$: 1.2×10^{-3}

Advection-diffusion equation

d	classical SL			neural SL		
	dofs	CPU time	L^2 error	dofs	CPU time	L^2 error
1	36	0.96	1.57e-3	127	29.84	1.56e-3
2	1444	2.19	1.63e-3	463	33.07	1.54e-3
3	64 000	3.31	1.56e-3	1009	39.45	1.54e-3
4	2.56×10^6	17.09	1.49e-3	1765	50.29	1.55e-3
5	2.43×10^7	50.64	9.11e-3	2731	76.76	1.52e-3
6	4.70×10^7	110.62	3.91e-3	3907	115.22	1.53e-3
7	6.27×10^7	158.15	1.00e-2	5293	172.53	1.54e-3
8	1.00×10^8	271.33	1.66e-2	6889	254.38	1.57e-3

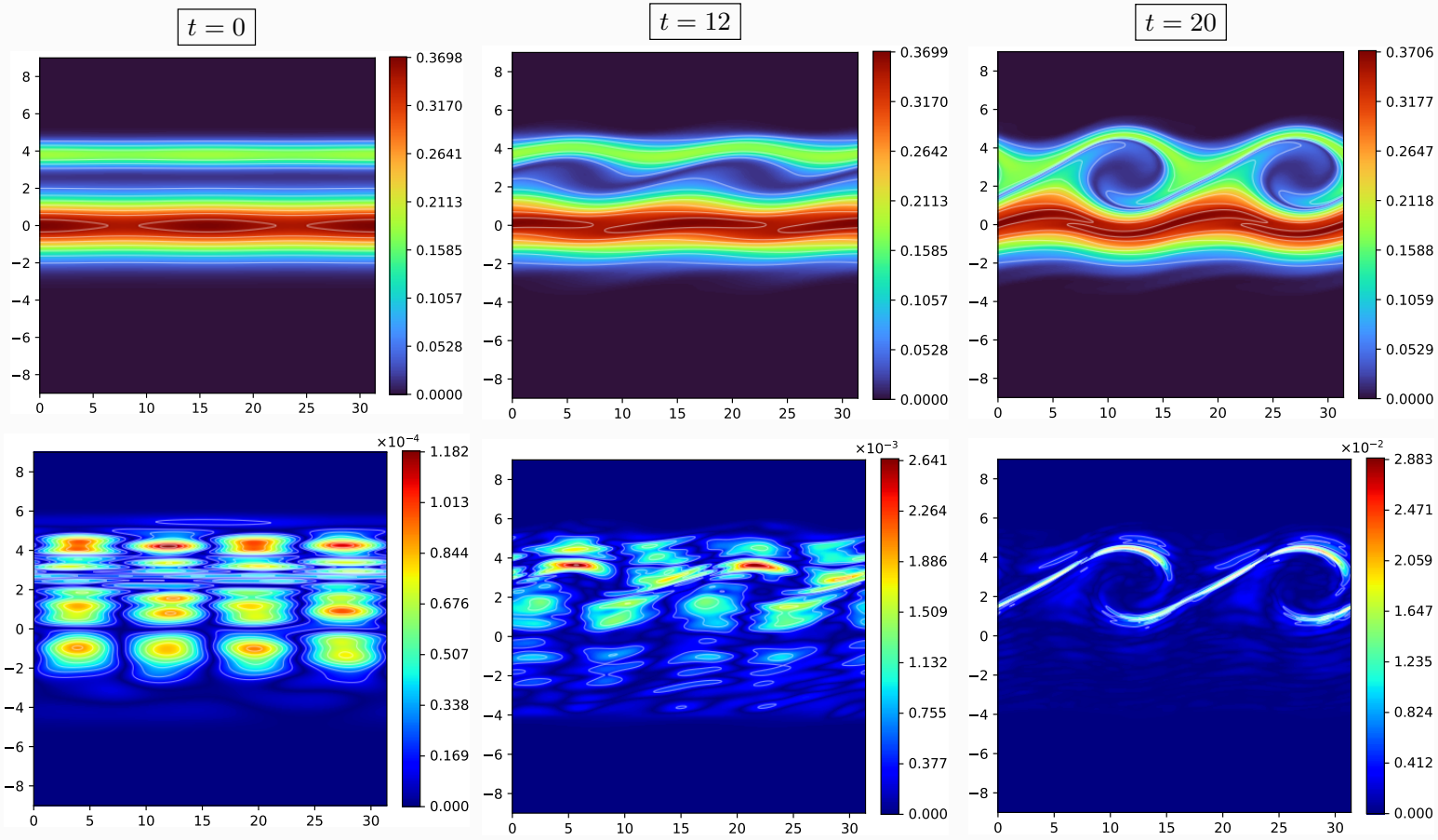
Extension to the Vlasov-Poisson equation: the bump-on-tail instability

Vlasov-Poisson equation: $\partial_t f + v\,\partial_x f + E(t,x)\,\partial_v f = 0$, where $E(t,x) = \partial_x \Psi(t,x)$ with $\partial_{xx}\Psi(t,x) = \int f(t,x,v)\,dv$

It is a nonlinear PDE! f is solved with the neural SL method, and E with a PINN.

prediction

error



6. Conclusion and outlook

Conclusion and outlook

Conclusion:

Summary of the neural SL method:
- fully meshless
- stable without a time step condition
- able to treat high dimensional advection-diffusion problems

Perspectives and follow-up work:
- nonlinear PDEs (Vlasov-Poisson works, Navier-Stokes is ongoing)
- application to kinetic relaxation
- rigorous error analysis

Related work in the MACARON team includes strongly hybridizing classical and neural methods, e.g.:
- modifying the **discontinuous Galerkin basis functions** to make any scheme **well-balanced** [Franck et al, 2024];
- learning an **initial guess** for **Newton's method** when none is available [Aghili et al, 2025];
- physics-informed learning for **volume-preserving shape optimization** [Bélières-Frendo et al, 2025];
- enriching **finite element approximation spaces** to improve the solution of **elliptic PDEs** [Barucq et al, 2026].

We are organizing a Workshop on **JAX for Scientific Computing** in **Strasbourg** in **June 2026**:
https://majsc2026.pages.math.unistra.fr/

Thank you for your attention!