



Learning characteristic flows with neural networks for kinetic simulations

TSVV-G meeting 2026

Victor Fournet (victor.fournet@ipp.mpg.de)

with Martin Campos-Pinto (NMPP), Emmanuel Franck (IRMA), Philipp Krah (CEA), Victor Michel-Dansac (IRMA)

Max Planck Institute for Plasma Physics, Garching

Outline

Introduction

Learning the phase space flow from particle data: Neural δf-PIC

Learning the phase space flow from the PDE: Neural Characteristic Mapping Method

Conclusion

Outline

Introduction

Learning the phase space flow from particle data: Neural δf-PIC

Learning the phase space flow from the PDE: Neural Characteristic Mapping Method

Conclusion

Vlasov–Poisson

$$\partial_t f + v \cdot \nabla_x f - \nabla_x \phi \cdot \nabla_v f = 0, \qquad -\Delta \phi = \int f \, \mathrm{d}v - 1.$$

Characteristics:

$$\dot{X} = V, \qquad \dot{V} = -\nabla_x \phi(X).$$

Forward flow and backward flow:

$$\Phi_{s,t} : (X(s), V(s)) \mapsto (X(t), V(t)), \qquad \Psi_{s,t} = \Phi_{s,t}^{-1}.$$

Pullback and semigroup: for $0 = t_0 < \cdots < t_n = t$,

$$f(t, z) = f_{\mathrm{init}}(\Psi_{0,t}(z)), \qquad \Psi_{0,t} = \Psi_{t_0,t_1} \circ \Psi_{t_1,t_2} \circ \ldots \circ \Psi_{t_{n-1},t_n}.$$

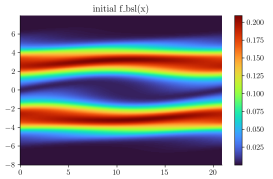
Family of Flow based numerical methods

Backward Semi-Lagrangian method (BSL)

Approximates the **backward** flow $\Psi_h \approx \Phi^{-1}$:

$$f(t,x,v) \approx f_{init}(\Psi_h(0,t;x,v))$$

- High-order accurate
- But needs a 6D phase-space grid: $O(N^6)$, grid points



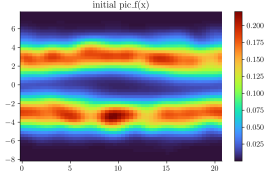
250K grid points

Particle-In-Cell (PIC) method

Push markers along the **forward** flow $\Phi_h \approx \Phi$:

$$f(t^n,x,v) \approx \sum_{k=1}^{N_p} w_k S_\varepsilon(x - x_k^n)\delta(v - v_k^n)$$

- No grid: cheap in high dimension
- But **noisy**: error $\approx \frac{\sigma}{\sqrt{N_p}}$ (Monte Carlo)



10K particles

Particle denoising (Monte Carlo variance reduction)

- δf **approach**:

$$f^n = f^m_{\text{bulk}} + \delta f^n \quad \text{with} \quad \begin{cases} f^m_{\text{bulk}} : & \text{smooth, easy to evaluate and to integrate (goal: } \approx f^n) \\ \delta f^n : & \text{PIC solution (goal: small } \Rightarrow \text{ low noise)} \end{cases}$$

  $m = \left\lfloor \frac{n}{N_\Psi} \right\rfloor$, N_Ψ period of bulk update.

- **common choice**: static f_{bulk}, e.g. Maxwellian

$$f_{\text{bulk}}(x, v) = \frac{n_0(x)}{\sqrt{2\pi T_0(x)}} \exp\Big(- \frac{|v - u_0(x)|^2}{2 T_0(x)} \Big)$$

Particle denoising (Monte Carlo variance reduction)

- δf **approach**:

$$f^n = f^m_{\text{bulk}} + \delta f^n \quad \text{with} \quad \begin{cases} f^m_{\text{bulk}}: & \text{smooth, easy to evaluate and to integrate (goal: } \approx f^n) \\ \delta f^n: & \text{PIC solution (goal: small } \Rightarrow \text{ low noise)} \end{cases}$$

  $m = \left\lfloor \frac{n}{N_\Psi} \right\rfloor$, N_Ψ period of bulk update.

- **common choice**: static f_{bulk}, e.g. Maxwellian

$$f_{\text{bulk}}(x,v) = \frac{n_0(x)}{\sqrt{2\pi T_0(x)}} \exp\Big(- \frac{|v - u_0(x)|^2}{2T_0(x)} \Big)$$

- **if large deviations**: this is not enough, a dynamic bulk is needed, e.g.:
  - ▷ evolving Maxwellian, moments computed from PIC density (Murugappan et al '24)
  - ▷ coarse spline projection of the PIC density (Allfrey Hatzky '03)
  - ▷ slowly evolving orbit-averaged solutions from PIC density (Bottino '25)

Particle denoising (Monte Carlo variance reduction)

- δf **approach**:

$$f^n = f^m_{\text{bulk}} + \delta f^n \quad \text{with} \quad \begin{cases} f^m_{\text{bulk}}: & \text{smooth, easy to evaluate and to integrate (goal: } \approx f^n) \\ \delta f^n: & \text{PIC solution (goal: small } \Rightarrow \text{ low noise)} \end{cases}$$

  $m = \left\lfloor \frac{n}{N_\Psi} \right\rfloor$, N_Ψ period of bulk update.

- **common choice**: static f_{bulk}, e.g. Maxwellian

$$f_{\text{bulk}}(x,v) = \frac{n_0(x)}{\sqrt{2\pi T_0(x)}} \exp\Big(- \frac{|v - u_0(x)|^2}{2 T_0(x)} \Big)$$

- **if large deviations**: this is not enough, a dynamic bulk is needed, e.g.:
  - ▷ evolving Maxwellian, moments computed from PIC density (Murugappan et al '24)
  - ▷ coarse spline projection of the PIC density (Allfrey Hatzky '03)
  - ▷ slowly evolving orbit-averaged solutions from PIC density (Bottino '25)
- These approaches use the PIC density δf and require many particles for strong noise reduction

Outline

Introduction

Learning the phase space flow from particle data: Neural δf-PIC

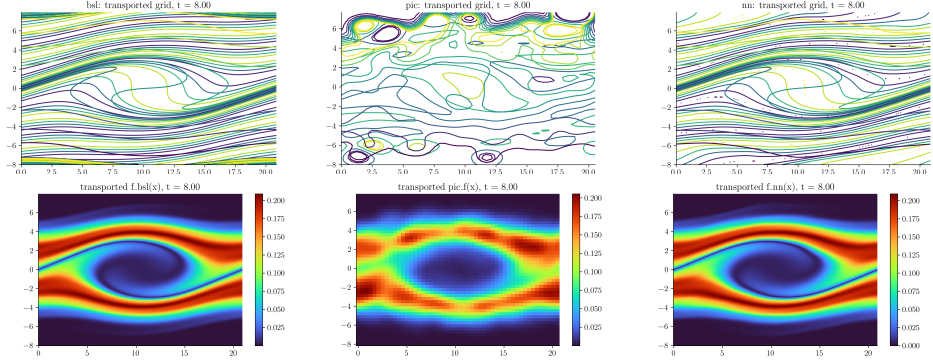
Learning the phase space flow from the PDE: Neural Characteristic Mapping Method

Conclusion

Leaning the flow from particle data

Backward flow learned from PIC marker pairs, $\min_\theta \frac{1}{N_p} \sum_k |\Psi_\theta(z_k^n) - z_k^{n-N_\Psi}|^2$, then $f \approx f_{\text{init}}(\Psi_\theta)$.

Semi-Lagrangian (reference)	Particle-In-Cell	Neural Network
512×512 grid	10K particles	10K particles + 1K parameters

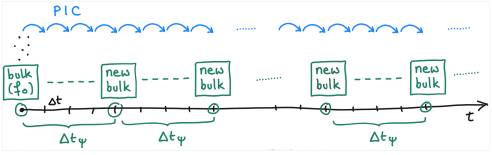


Neural δf PIC scheme

- δf **approach**:

$$f^n = f^m_{\text{bulk}} + \delta f^n \quad \text{with} \quad \begin{cases} f^m_{\text{bulk}} : & \text{smooth, easy to evaluate and to integrate (goal: } \approx f^n) \\ \delta f^n : & \text{PIC solution (goal: small } \Rightarrow \text{ low noise)} \end{cases}$$

- **here**:
  - PIC step: solve for δf, $z_k^n \to z_k^{n+N_\Psi}$, update the weights δw_k^n
  - Every N_Ψ timesteps, $m = \lfloor n/N_\Psi \rfloor$: bulk update
    - Neural flow (fine): $f^m_{\text{NN}} = f_{\text{init}}(\Psi_{\theta^1} \circ \ldots \circ \Psi_{\theta^m})$
      Loss : $\min_\theta \frac{1}{N_p} \sum_k |\Psi_\theta(z_k^n) - z_k^{n-N_\Psi}|^2$

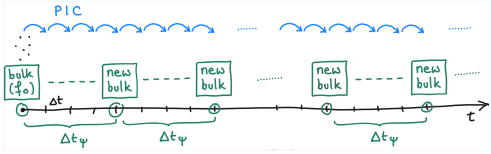


Neural δf PIC scheme

- δf **approach**:

$$f^n = f^m_{\text{bulk}} + \delta f^n \quad \text{with} \quad \begin{cases} f^m_{\text{bulk}} : & \text{smooth, easy to evaluate and to integrate (goal: } \approx f^n) \\ \delta f^n : & \text{PIC solution (goal: small } \Rightarrow \text{ low noise)} \end{cases}$$

- **here**:
  - PIC step: solve for δf, $z_k^n \to z_k^{n+N_\Psi}$, update the weights δw_k^n
  - Every N_Ψ timesteps, $m = \lfloor n/N_\Psi \rfloor$: bulk update
    - Neural flow (fine): $f^m_{\text{NN}} = f_{\text{init}}(\Psi_{\theta^1} \circ \ldots \circ \Psi_{\theta^m})$
      Loss : $\min_\theta \frac{1}{N_p} \sum_k |\Psi_\theta(z_k^n) - z_k^{n-N_\Psi}|^2$



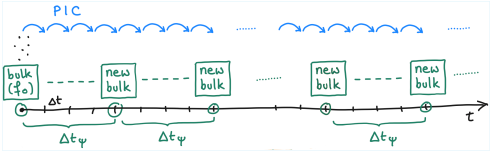
$$\rho^n = \int f^m_{\text{bulk}}\, \mathrm{d}v + \sum_k \delta w_k^n\, S_\varepsilon(x - x_k^n), \qquad \delta w_k^n = \frac{f_{\text{init}}(z_k^0) - f^m_{\text{bulk}}(z_k^n)}{N_p\, g(z_k^0)}$$

Neural δf PIC scheme

- δf **approach**:

$$f^n = f^m_{\text{bulk}} + \delta f^n \quad \text{with} \quad \begin{cases} f^m_{\text{bulk}} : & \text{smooth, easy to evaluate and to integrate (goal: } \approx f^n) \\ \delta f^n : & \text{PIC solution (goal: small } \Rightarrow \text{ low noise)} \end{cases}$$

- **here**:
  - PIC step: solve for δf, $z^n_k \to z^{n+N_\Psi}_k$, update the weights δw^n_k
  - Every N_Ψ timesteps, $m = \lfloor n/N_\Psi \rfloor$: bulk update
    - Neural flow (fine): $f^m_{\text{NN}} = f_{\text{init}}(\Psi_{\theta^1} \circ \ldots \circ \Psi_{\theta^m})$
      Loss : $\min_\theta \frac{1}{N_p} \sum_k |\Psi_\theta(z^n_k) - z^{n-N_\Psi}_k|^2$
    - Coarse spline grid (cheap, exact $\int \cdot \, \mathrm{d}v$):
      $f^m_{\text{bulk}} = \text{SplineApprox}(f^m_{\text{NN}})$



$$\rho^n = \int f^m_{\text{bulk}} \, \mathrm{d}v + \sum_k \delta w^n_k \, S_\varepsilon(x - x^n_k), \qquad \delta w^n_k = \frac{f_{\text{init}}(z^0_k) - f^m_{\text{bulk}}(z^n_k)}{N_p \, g(z^0_k)}$$

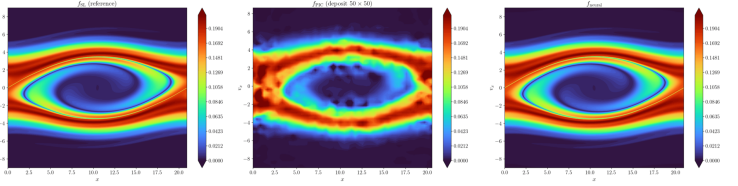
Control variate: the network error only changes the variance, error $\approx \frac{\sigma\left(f^n - f^m_{\text{bulk}}\right)}{\sqrt{N_p}}$.

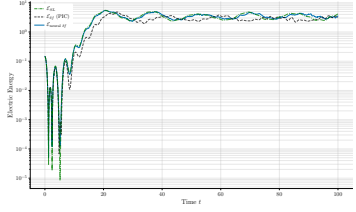
1D1V two-stream instability

$f_{\text{init}} = (1 + \varepsilon \cos kx)\frac{1}{2\sqrt{2\pi}}(e^{-(v-v_0)^2/2} + e^{-(v+v_0)^2/2})$, $\varepsilon = 0.05$, $k = 0.3$, $v_0 = 3$; $N_p = 10^4$, $N_\Psi = 20$, $t \in [0, 99]$.

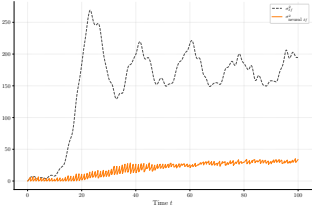
Fine bulk at $t = 99$: 40 SympNets $\times$ 480 parameters $\approx 1.9 \times 10^4$, vs SL grid $1024^2 \approx 1.0 \times 10^6$ values.

BSL δf Nδf





Electric energy: BSL, δf, Nδf



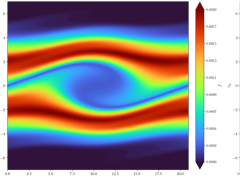
Weight variance: δf vs Nδf

3D3V four-stream instability: a proof of concept

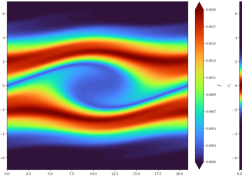
$N_p = 10^8$, single GPU.

Fine bulk: 12 SympNets $\times$ 960 parameters $\approx 1.2 \times 10^4$, vs SL grid $64^2 \times 32 \times 63^2 \times 31 \approx 1.6 \times 10^{10}$ values.

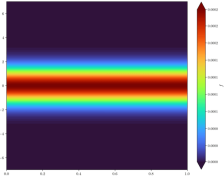
(x, v_x) plane

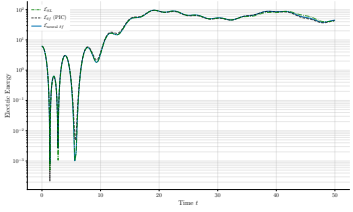


(y, v_y) plane

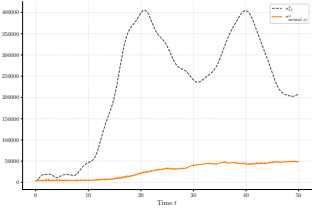


(z, v_z) plane





Electric energy: SL, δf, Nδf



Weight variance: δf vs Nδf

Outline

Introduction

Learning the phase space flow from particle data: Neural δf-PIC

Learning the phase space flow from the PDE: Neural Characteristic Mapping Method

Conclusion

Learning the flow from the PDE: Eulerian vs Lagrangian PINNs

$$\partial_t f + \mathbf{a} \cdot \nabla_z f = 0, \qquad \nabla_z \cdot \mathbf{a} = 0 \qquad \text{(Vlasov: } \mathbf{a} = (v, -\nabla\phi)\text{)}$$

PINN: a network minimises the PDE residual at random collocation points, no mesh.

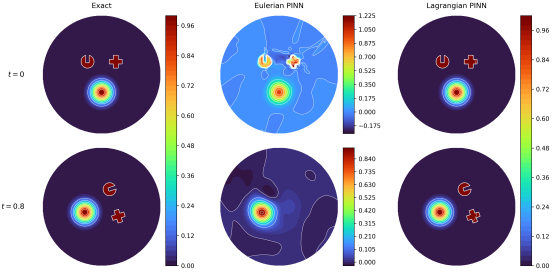
Eulerian PINN (Raissi et al. '19):
$f_\theta(t,z) \approx f(t,z)$

$$\min_\theta \|\partial_t f_\theta + \mathbf{a} \cdot \nabla_z f_\theta\|^2 + \lambda \|f_\theta(0,\cdot) - f_{\text{init}}\|^2$$

Lagrangian PINN:
$\Psi_\theta(t,z) \approx \Psi_{0,t}(z)$, then $f_\theta = f_{\text{init}} \circ \Psi_\theta$

$$\min_\theta \|\partial_t \Psi_\theta + (\mathbf{a} \cdot \nabla_z)\Psi_\theta\|^2 + \lambda \|\Psi_\theta(0,\cdot) - \text{Id}\|^2$$

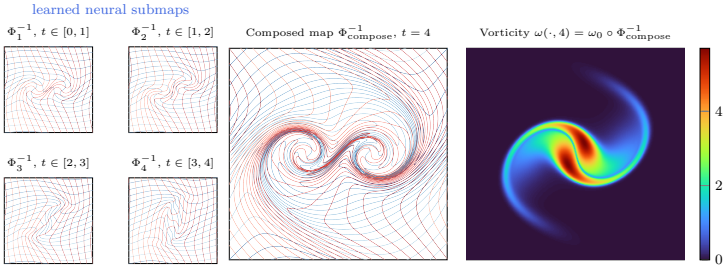
Difference with Neural δf-PIC:
- trained from the PDE residual, not from PIC data;
- continuous in time, $\Psi_\theta(t,z)$ for all t, instead of a discrete map $z^n \mapsto z^{n-N_\Psi}$.



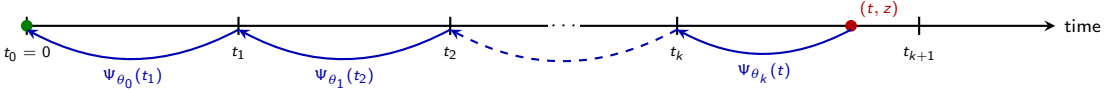
Rigid rotation $\mathbf{a} = 2\pi(-x_2, x_1)$, $t = 0$ and $t = 0.8$ (Eulerian PINN: own color scale).

The network approximates the phase space characteristic flow.

Neural Characteristic Mapping Method (NCMM): use the semigroup property



2D Euler, vortex merger: four learned backward submaps, their composition, $\omega(\cdot,4) = \omega_0 \circ$ composition.



$$f(t,z) = f_{\text{init}}(\Psi_{\theta_0}(t_1) \circ \Psi_{\theta_1}(t_2) \circ \cdots \circ \Psi_{\theta_k}(t)\,(z))$$

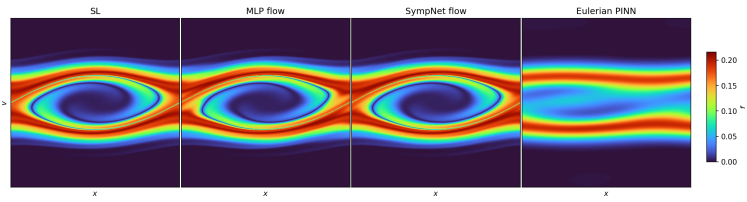
each network trained on its own sub-interval (identity at t_k), then frozen

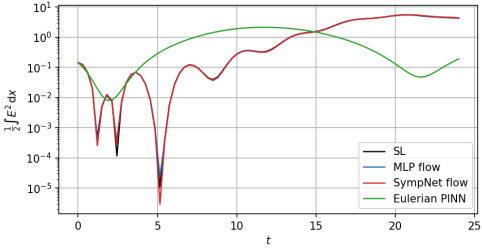
Neural counterpart of characteristic mapping methods (Mercier, Yin, Nave, Krah, Schneider, ...).

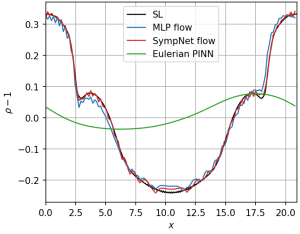
Vlasov–Poisson 1D1V: two-stream instability

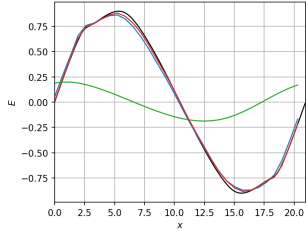
$N_t = 6$ submaps of $\sim$1100 parameters, Poisson on 32 points; Eulerian PINN with $8\times$ more parameters. Snapshots at $T = 24$.

$$\partial_t f + v\,\partial_x f + E\,\partial_v f = 0,$$
$$E = -\partial_x \phi, \quad -\partial_{xx}\phi = \int f\,\mathrm{d}v - 1,$$
$$f_0 = (1 + \varepsilon \cos kx)\,\frac{e^{-(v-v_0)^2/2} + e^{-(v+v_0)^2/2}}{2\sqrt{2\pi}},$$
$$\varepsilon = 0.05,\ k = 0.3,\ v_0 = 3.$$

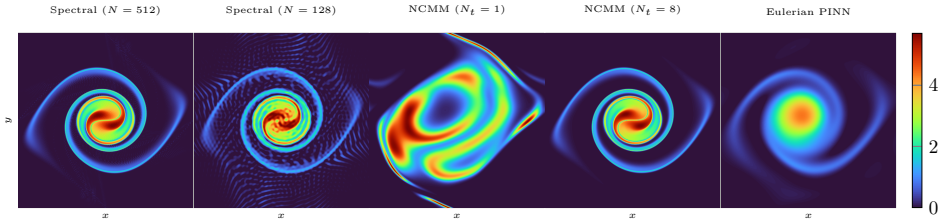


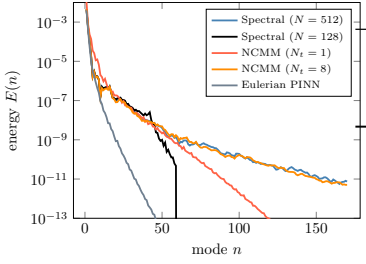






2D Euler, vortex merger: sub-grid resolution





$t = 8$	Enstrophy err.	Energy err.
Spectral ($N = 512$)	-7.3×10^{-8}	-9.1×10^{-14}
NCMM ($N_t = 1$)	2.1	1.3×10^{-2}
NCMM ($N_t = 8$)	-6.6×10^{-3}	7.5×10^{-6}
Eulerian PINN	-2.8×10^{-1}	-1.2×10^{-3}

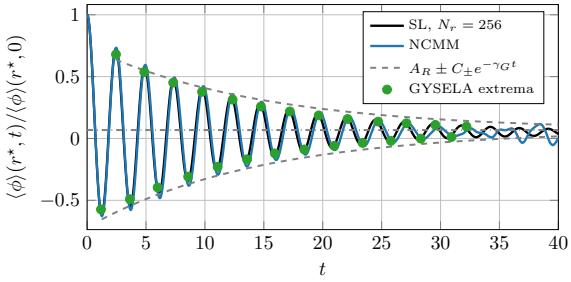
$\partial_t \omega + \mathbf{u} \cdot \nabla \omega = 0,$

$\Delta \psi = \omega, \quad \mathbf{u} = \nabla^\perp \psi,$

ω_0: two Gaussian vortices, $\sigma = 0.07$, distance 0.3.

Enstrophy $\mathcal{Z}(t) = \int_\Omega \omega^2 \, \mathrm{d}x$, energy $\mathcal{E}(t) = \int_\Omega |\mathbf{u}|^2 \, \mathrm{d}x$, both conserved by the Euler flow.
Errors: $\mathcal{Z}(8) - \mathcal{Z}(0)$ and $\mathcal{E}(8) - \mathcal{E}(0)$.

Towards gyrokinetics: axisymmetric drift-kinetic Rosenbluth–Hinton test



$$\partial_t f + \dot{r}\,\partial_r f + \dot{\theta}\,\partial_\theta f + \dot{v}_\parallel\,\partial_{v_\parallel} f = 0,$$

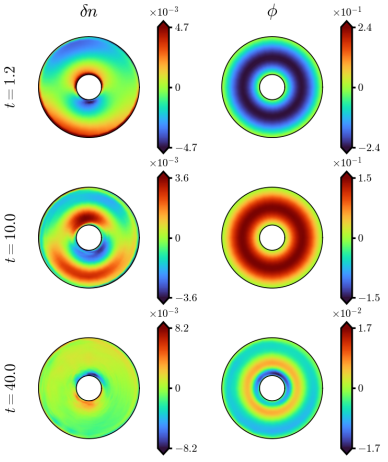
$$-\rho_*^2 a^2 \Delta_\perp \phi + \phi - \langle\phi\rangle = \delta n,$$

$$\delta n = \iint 2\pi B\,(f - f_{\mathrm{eq}})\,\mathrm{d}v_\parallel\,\mathrm{d}\mu,$$

$$f_0 = f_{\mathrm{eq}}(1 + \varepsilon \sin(\pi \frac{r - r_{\min}}{r_{\max} - r_{\min}})), \quad \varepsilon = 10^{-3}.$$

	ω_G	γ_G	A_R
RH theory	–	–	0.068
GYSELA	2.53	0.0685	–
SL (fine)	2.57	0.102	0.061
NCMM	2.54	0.099	0.052

$(r, \theta, v_\parallel)$, μ parameter; Q1 FEM 32^2; $N_t = 80$ MLP submaps, $T = 40$.



δn and ϕ at $t = 1.2$, 10, 40

Outline

Introduction

Learning the phase space flow from particle data: Neural δf-PIC

Learning the phase space flow from the PDE: Neural Characteristic Mapping Method

Conclusion

Comparison and perspectives

Summary: two ways to learn the characteristic flow

From particle data: Neural δf-PIC

- Training data: marker trajectories computed by the PIC scheme.
- Loss: $\mathcal{L}(\theta) = \sum_k |\Psi_\theta(z_k^n) - z_k^{n-N_\Psi}|^2$.
- The network builds a control variate: $f = f_{\text{bulk}}(\Psi_\theta) + \delta f$.

From the PDE: Neural Characteristic Mapping Method

- Physics-informed network for the flow map.
- Loss: $\mathcal{L}(\theta) = \|\partial_t \Psi_\theta + (\mathbf{a} \cdot \nabla)\Psi_\theta\|^2$.
- The network builds the solution: $f = f_0(\Psi_\theta)$.

Shared ingredients: Approximate the phase space flow with neural network, composition of sub-flows.

Next steps

- Neural δf: drift-kinetic and gyrokinetic models (with Alberto Bottino and Thomas Hayward-Schneider), parallel implementation.
- NCMM: adaptive time partition, parametric / many-query problems, 5D gyrokinetics, sources and collisions.
- Bridge both: train the flow on particle data and physics residual together.

NCMM: V. Fournet, P. Krah, M. Campos-Pinto, E. Franck, V. Michel-Dansac, *A neural characteristic mapping method: Lagrangian PINNs based on flow maps for transport-dominated problems*, arXiv:2610.12395.
Neural delta-f: V. Fournet,M. Campos-Pinto, E. Franck, V. Michel-Dansac, *Non-linear control variate in δf particle-in-cell methods using symplectic neural networks* ,arXiv:2606.30622.

Backup: NCMM, coupling to the field equation and architectures

Picard iteration (velocity frozen in each optimisation step), for every step on interval $[t_k, t_{k+1}]$:

1. Field update: evaluate $\rho_\Theta(\tau_j, \cdot)$ on a coarse grid at N_τ times by composing the submaps (velocity moments by quadrature for kinetic models); solve Poisson / Biot–Savart / quasi-neutrality.
2. Flow update: interpolate the stored field at the collocation points (no gradient through it), one optimiser step (natural gradient) on the submap loss.

Submap architectures (same interface: evaluation, Jacobian determinant):

- **Perturbation of identity**: $\Psi_\theta(t, \mathbf{x}) = \mathbf{x} + h\mathcal{N}_\theta(t, \mathbf{x})$, MLP with periodic embedding.
- **Periodic SympNet** (time-conditioned), same as in Neural δf-PIC: $\det \nabla \Psi_\theta \equiv 1$.

Implemented in the scimba library.

Backup: unbiasedness of the estimator (for *any* bulk)

Moment estimator for a test function α:

$$\widehat{I}(\alpha) = \underbrace{\int \alpha\, f^m_{\text{bulk}}\, \mathrm{d}z}_{\text{analytic}} + \sum_k \delta w^n_k\, \alpha(z^n_k), \qquad \delta w^n_k = \frac{f_{\text{init}}(z^0_k) - f^m_{\text{bulk}}(z^n_k)}{N_p\, g(z^0_k)}.$$

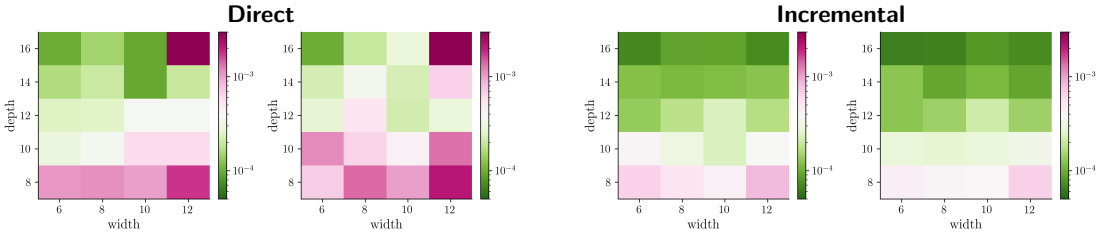
Markers $z^0_k \sim g$ i.i.d., pushed $z^n_k = \Phi(z^0_k)$ by the particle pusher (not the network), with $\det \nabla\Phi = 1$ and $f_{\text{init}}(z_0) = f(t^n, \Phi(z_0))$. Then

$$\mathbb{E}\Big[\sum_k \delta w^n_k\, \alpha(z^n_k)\Big] = \int \frac{f_{\text{init}}(z_0) - f^m_{\text{bulk}}(\Phi(z_0))}{g(z_0)}\, \alpha(\Phi(z_0))\, g(z_0)\, \mathrm{d}z_0 \overset{z=\Phi(z_0)}{=} \int (f - f^m_{\text{bulk}})\, \alpha\, \mathrm{d}z.$$

Same bulk in the analytic moment and in the weights: $\mathbb{E}[\widehat{I}(\alpha)] = \int \alpha\, f\, \mathrm{d}z$ for any bulk; network and spline errors enter only the variance.

- Symplecticity of the network is *not* needed here; it only improves the bulk, i.e. the variance.
- Holds for a given flow and bulk (as for any PIC): the network adds no bias with respect to full-f PIC.
- A quadrature of the fine (filamented) bulk instead of the exact spline moment would turn the quadrature error into a bias that does not decrease with N_p.

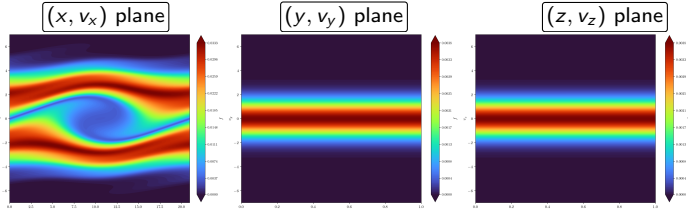
Backup: flow learning, direct vs incremental (train & test)



Left = train error, right = test error (600 Adam / 900 NG). Incremental: train ≈ test (no overfitting).

Backup: 3D3V two-stream instability ($N_p = 10^6$)

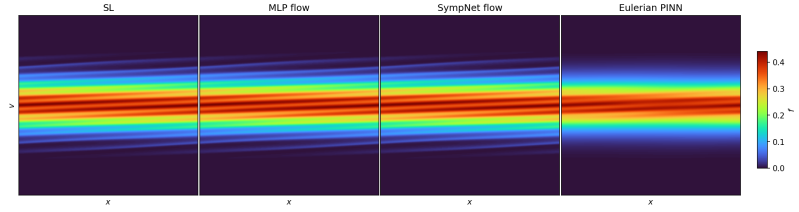
$N_p = 10^6$; SympNets with 960 parameters, 13 composed networks for 99 bulk updates; fine bulk f^m_{NN} of Nδf shown; single GPU, non-parallel code.
Reference: BSL6D on $64^3 \times 63^3$ (Schild et al., CPC 2024), electric energy courtesy of N. Schild. Four-stream ($N_p = 10^8$): main talk.

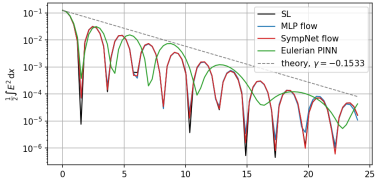


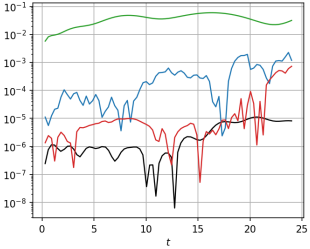
Electric energy: BSL6D, δf, Nδf

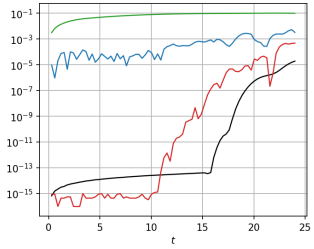
Weight variance: δf vs Nδf (reduced by $\approx$10)

Backup: NCMM, Landau damping and VP invariants









Landau damping ($T = 24$) and two-stream drift of total energy and L^2 norm. Training on A100: $\sim$1 h (MLP), $\sim$3.5 h (SympNet), cost of submap k grows linearly with k.

Shared building block: (periodic) symplectic neural networks

Definition 1: SympNet (Jin et al. '20)

Shear maps with gradient modules $\widehat{\sigma}_{K,b}(y) = K^\top \sigma(Ky + b) \approx \nabla V(y)$:

$$\psi_{\mathrm{up}}(x, v) = \begin{pmatrix} x + \widehat{\sigma}_{K,b}(v) \\ v \end{pmatrix}, \quad \psi_{\mathrm{down}}(x, v) = \begin{pmatrix} x \\ v + \widehat{\sigma}_{K,b}(x) \end{pmatrix}, \quad \Psi_\theta = \psi^\ell_{\mathrm{up}} \circ \psi^\ell_{\mathrm{down}} \circ \cdots \circ \psi^1_{\mathrm{up}} \circ \psi^1_{\mathrm{down}}.$$

- Symplectic for every θ, hence $\det \nabla \Psi_\theta = 1$ and all L^p norms of f are conserved; universal approximation within symplectic maps.
- $\psi_{\mathrm{up}} \circ \psi_{\mathrm{down}}$ with $\nabla V = \Delta t \, \nabla \phi$, $\nabla T = \Delta t \, v$ is the symplectic Euler scheme.
- **Periodic extension** (this work): trigonometric embedding of the potential

$$\tilde{V}(x) = \mathbf{1}^\top \Sigma\Big(K_1 \cos \frac{2\pi x}{L} + b_1\Big) + \mathbf{1}^\top \Sigma\Big(K_2 \sin \frac{2\pi x}{L} + b_2\Big), \qquad \tilde{\psi}_{\mathrm{down}}(x, v) = (x,\ v + \nabla \tilde{V}(x)),$$

The resulting map is symplectic and periodic on $L\mathbb{T}^d \times \mathbb{R}^d$, with $3w\ell(d+1)$ parameters ($\sim 10^3$ here).

SympNets: P. Jin, Z. Zhang, A. Zhu, Y. Tang, G. E. Karniadakis, Neural Networks (2020).

Shared building block: (periodic) symplectic neural networks

Definition 2: SympNet (Jin et al. '20)

Shear maps with gradient modules $\widehat{\sigma}_{K,b}(y) = K^\top \sigma(Ky + b) \approx \nabla V(y)$:

$$\psi_{\mathrm{up}}(x, v) = \begin{pmatrix} x + \widehat{\sigma}_{K,b}(v) \\ v \end{pmatrix}, \quad \psi_{\mathrm{down}}(x, v) = \begin{pmatrix} x \\ v + \widehat{\sigma}_{K,b}(x) \end{pmatrix}, \quad \Psi_\theta = \psi^\ell_{\mathrm{up}} \circ \psi^\ell_{\mathrm{down}} \circ \cdots \circ \psi^1_{\mathrm{up}} \circ \psi^1_{\mathrm{down}}.$$

- Symplectic for every θ, hence $\det \nabla \Psi_\theta = 1$ and all L^p norms of f are conserved; universal approximation within symplectic maps.
- $\psi_{\mathrm{up}} \circ \psi_{\mathrm{down}}$ with $\nabla V = \Delta t \, \nabla \phi$, $\nabla T = \Delta t \, v$ is the symplectic Euler scheme.
- **Periodic extension** (this work): trigonometric embedding of the potential

$$\tilde{V}(x) = \mathbf{1}^\top \Sigma\Big(K_1 \cos \frac{2\pi x}{L} + b_1\Big) + \mathbf{1}^\top \Sigma\Big(K_2 \sin \frac{2\pi x}{L} + b_2\Big), \qquad \tilde{\psi}_{\mathrm{down}}(x, v) = (x,\ v + \nabla \tilde{V}(x)),$$

The resulting map is symplectic and periodic on $L\mathbb{T}^d \times \mathbb{R}^d$, with $3w\ell(d+1)$ parameters ($\sim 10^3$ here).

SympNets: P. Jin, Z. Zhang, A. Zhu, Y. Tang, G. E. Karniadakis, Neural Networks (2020).