Learning, geometry and PDEs, a promising interaction ?

L. Bois[12], E. Franck[12], C. Mengel[2], L. Navoret[12], G. Steimer[12], V. Vigon[12]

ITI Seminar, Strasbourg

[1]Inria Nancy Grand Est, France
[2]IRMA, Strasbourg university, France

Outline

Deep learning for closure PDE problem

Models in plasma physics

- **Plasma**: gas at very large temperature which is electrically charged. Coupling between compressible fluid dynamic and electromagnetic.
- **Kinetic models**:

$$\partial_t f + \mathbf{v} \cdot \nabla f + (\mathbf{E} + \mathbf{v} \times \frac{\mathbf{B}}{\epsilon}) \cdot \nabla_{\mathbf{v}} f = \frac{1}{\varepsilon} Q(f, f)$$

  with $f(t, \mathbf{x}, \mathbf{v})$ the seven dimensional particles distribution and $\mathbf{E}$, $\mathbf{B}$ the electric and magnetic fields given by the Maxwell equations.
- Large dimensional multiscale PDEs with admit geometric structures to preserve and few diffusion process (less stable). So we need reduced models and adapted numerical methods.

Fluid models:

- $$\partial_t \mathbf{U} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \epsilon \nabla \cdot (\mathbf{G}(\mathbf{U}) \nabla \mathbf{U})$$

  with $\mathbf{U}(t, \mathbf{x}) \in \mathbb{R}^n$ the macroscopic quantities (density, velocity etc).
- Strongly nonlinear, multiscale PDE which admits discontinuous solutions in the low diffusion regime. So we need adapted numerical methods. Tricky point: stability.

Problem of closure

- Simple case:

$$\partial_t f + v\partial_x f + \partial_x \phi \partial_v f = \frac{1}{\varepsilon}(M(\rho, u, T) - f)$$

with

$$\Delta\phi = \rho - 1$$

and the moment $\rho = \int f dv$, $\rho u = \int v f dv$ and $p = \int (v-u)^2 f dv$.

- When ϵ tends to zero we are close to the equilibrium $M(\rho, u, T)$.
- When we take the three first moment in velocity we obtain the **Euler equation**.

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

with $E = \frac{1}{2}\rho u^2 + \frac{1}{2}p$, $q = \frac{1}{2}\int_v (v-u)^3 f dv$.

- How to compute the reduced model: we don't know q.
- **Solution**: closure:

$$q \approx \mathcal{C}(\rho(.), u(.), T(.), \epsilon)$$

- Classical closure: asymptotic analysis gives $q = 0 + O(\epsilon)$ (Euler), $q = -\epsilon\frac{3}{2}p\partial_x T + O(\epsilon^2)$. Harmett-Perkins nonlocal in space.
- How construct a closure for moderate ϵ ?

Closure by supervised learning

- Using kinetic simulation we are able to collect data:

$$D = \{\rho(.), u(.), T(.), \epsilon)_i, q(.)_i, \quad i = 1, ..., n\}$$

- We can construct closure by supervised learning where we solve

$$\min_\theta \sum_i^n \| \mathcal{C}_\theta(\rho_i(.), u_i(.), T_i(.), \epsilon_i) - q_i(.) \|_2^2$$

- The function $\mathcal{C}_\theta$ can be in large dimension (if nonlocal in space) and strongly nonlinear. For this regression problem we use a neural network.
- Neural-network:

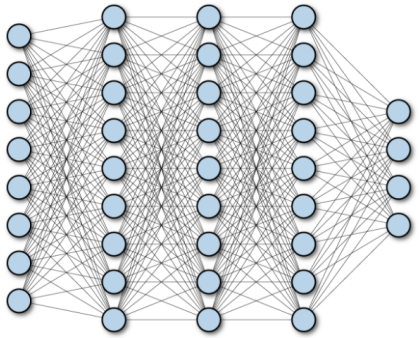
$$\mathcal{C}_\theta(x) = L_m \circ N \circ ..... \circ L_2 \circ N \circ L_1(x)$$

with $L_{\theta_i}(y) = My$ with $M \in \mathcal{M}_{n_i \times n_{i+1}}(\mathbb{R})$, $y \in \mathbb{R}^{n_{i+1}}$ and $N(y)$ a nonlinear function apply at each component of y.

- The weights θ of the neural network are the coefficients of the matrix L_i ..

Supervised learning and CNN II

- Fully connected neural network: the matrices L_i are dense.



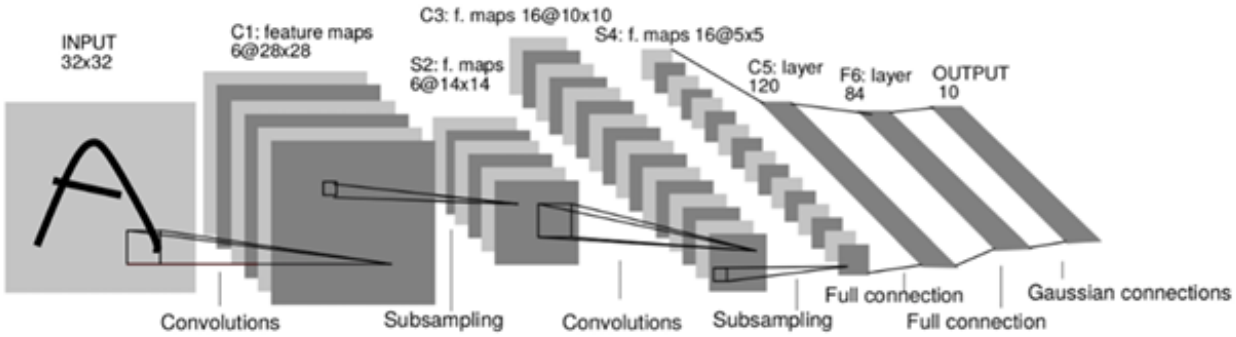
- 1D convolutional neural network: the matrices L_i are sparse Toeplitz matrices.

$$A = \begin{pmatrix} a & c & 0 & 0 \\ b & a & c & 0 \\ 0 & b & a & c \\ 0 & 0 & b & a \end{pmatrix}$$

- This is equivalent to applying a small convolution kernel to the signal 1D.
- Apply convolution kernel = apply a filter on signal (operation of the spectrum of the signal).
- Often we apply some convolution at the signal on each layer to create several new signals.

Supervised learning and CNN III

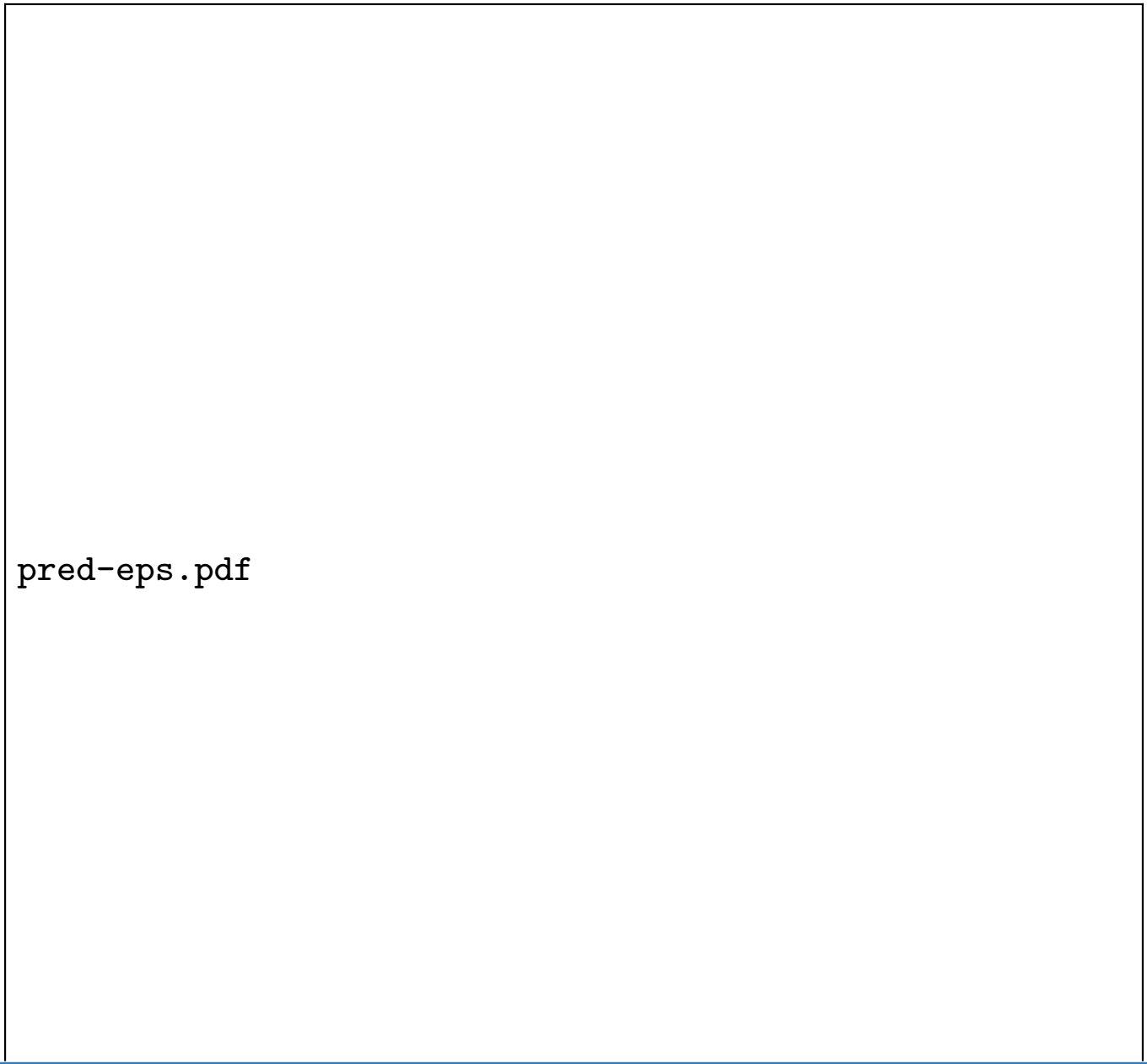
- Important properties for a convolutional layer:
  - Filters localized in space,
  - Anisotropic filter,
  - Multiple layers,
  - $O(1)$ parameters per filter (independent of input image size n),
  - $O(n)$ complexity per layer (filtering done in the spatial domain),



- The convolutional network takes into account to 2D structure of the data. Essential for good performance for signal problems.

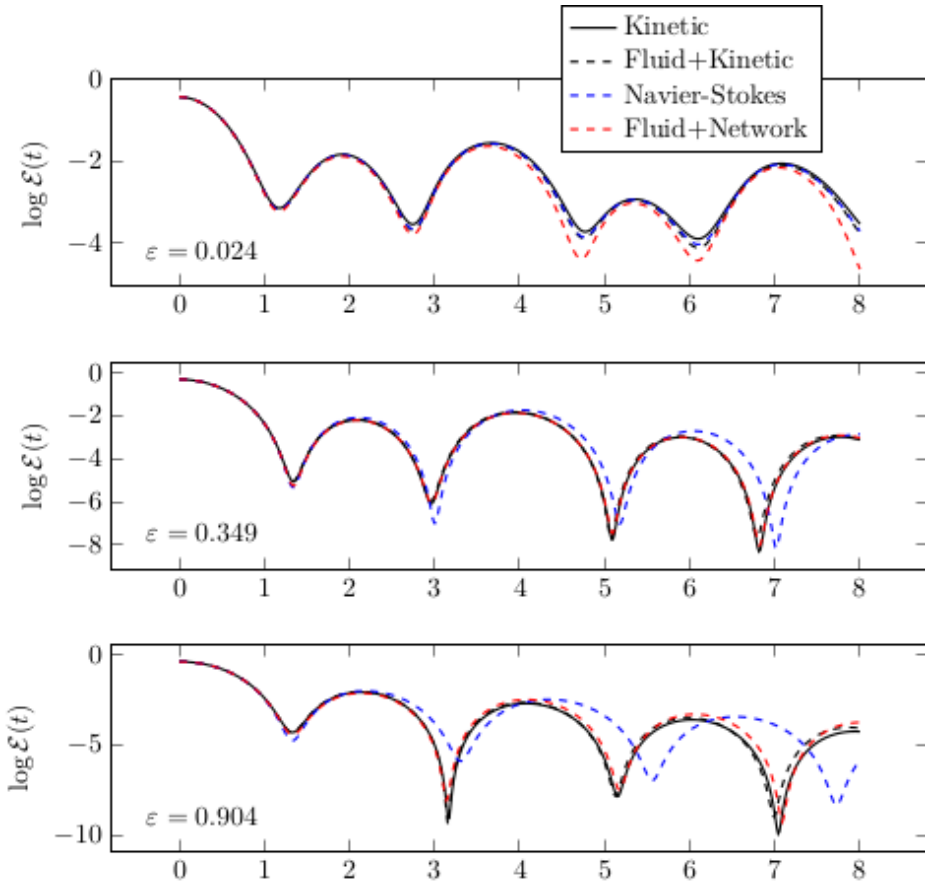
Result for closure problem I

- Average results for the prediction of q:



Result for closure problem II

- Examples of simulation:



- **Paper**: A neural network closure for the Euler-Poisson system based on kinetic simulations , L. Bois, E. Franck, L. Navoret, V. Vigon.
- ANR Milk (2022-2025) with Max-Planck institute of Plasma-Physics on reduced models with deep-learning.
- Two years of post-doc position on the closure for more complex problems.

Reduced problem for plasma physics

Vlasov and PIC code

- We consider the 1Dx1D nonlinear Vlasov-Poisson equation for plasma:

$$\partial_t f(t,x,v) + v\partial_x f + (E_{ext}(x) + E(x))\partial_v f = 0$$

with

$$-\Delta\phi(x) = \int_v f(t,x,v')dv' - 1, \quad E(x) = -\nabla\phi(x)$$

- $f(t,x,v)$ is a probability density of the particles.
- **Solver PIC** (Particle In Cells). We approximate the distribution by macro-particles

$$f(t,x,v) \approx f_N(x(t),v(t)) = \sum_{i=1}^{N} w_i \delta(x - x_i(t))\delta(v - v_i(t))$$

where

$$\begin{cases} \dfrac{dx_i}{dt} = v_i, \\ \dfrac{dv_i}{dt} = \frac{q}{m}(E + E_{ext})(x_i(t)), \end{cases} \tag{1}$$

- To compute the electric field, we compute $\int_v f(t,x,v')dv'$ on the mesh, solve the Poisson equation on the mesh, interpolate $E(x)$ at the position of macro-particles.

Reduced order modeling

- After PIC discretization, we have a large ODE (d^2N with d the dimension and N the number of particles) to solve.
- PIC method converges in $O(\frac{C(f)}{\sqrt{N}})$ so $d^2N >> 1$ (like MC methods).

ROM method

Design reduced models of size $K << N$ for a subset of initial data and parameters.
Assumption:

$$\mathbf{Y}(t) \approx A\mathbf{Z}(t)$$

with $\mathbf{Y}(t) \in \mathbb{R}^{d^2N}$, $\mathbf{Z}(t) \in \mathbb{R}^{2K}$.

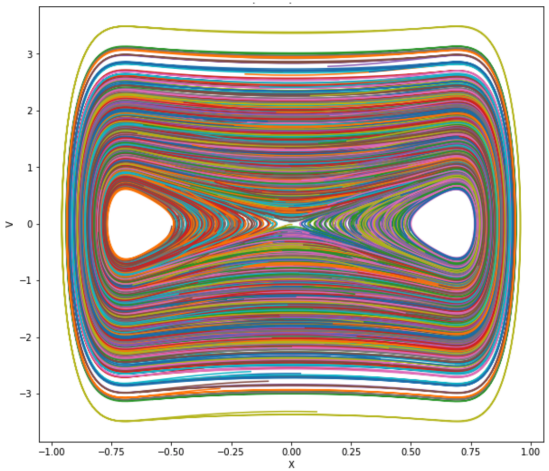
- Here we consider $f(t=0,x,v) = f_\gamma(\alpha,\beta)\mathcal{M}(v)$ with (α,β) the parameters.
- Classical approach: **POD** (see J. Hestaveen papers)
  - We collect some snapshots: $S = [(x(t_1), v(t_1)), ..., (x(t_n), v(t_n))]$.
  - By the SVD method, we compute the K dominant mods (associated to the K largest eigenvalues) and construct: $A \in \mathcal{M}_{K,N}$ (encoder) et $A^+ \in \mathcal{M}_{N,K}$ (decoder).
  - **Reduction**:

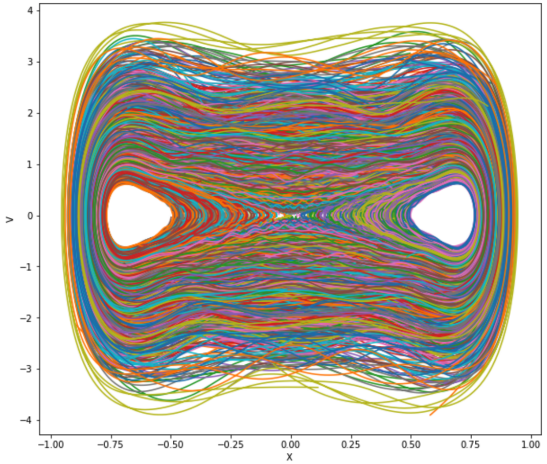
$$\partial_t \mathbf{Y}(t) = \mathbf{F}(\mathbf{Y}(t)) \rightarrow \partial_t \mathbf{Z}(t) = A\mathbf{F}(A^+\mathbf{Z}(t)), \text{ with } \mathbf{Z}(t) = A\mathbf{Y}(t)$$

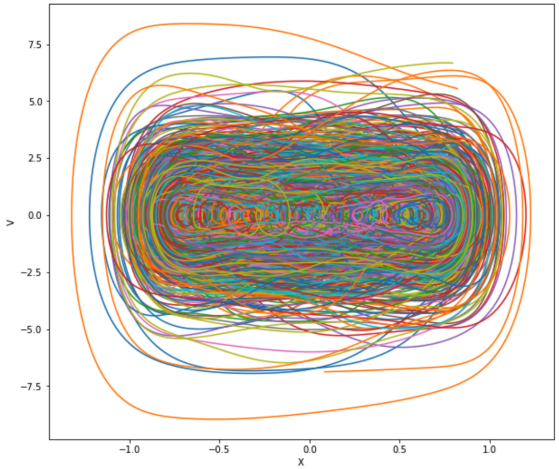
  - The term $A\mathbf{F}(A^+\mathbf{Z}(t))$ can be computed and stored in the linear case and approximated in the nonlinear case.

Nonlinear reduction and Auto-encoder

- Phase-space: PIC code (left), Reduction of PIC (middle), Reduced model (right)







- It doesn't work. Why ?

Nonlinear assumption

For nonlinear transport equation, we can assume that

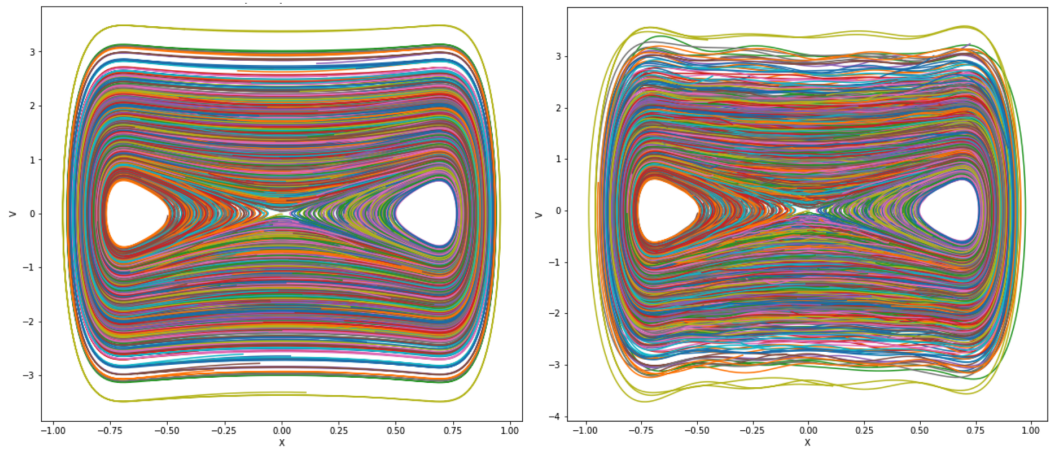
$$\mathbf{Y}(t) \approx G(\mathbf{Z}(t)), \quad \mathbf{Y}(t) \in \mathbb{R}^N, \mathbf{Z}(t) \in \mathbb{R}^K$$

- Idea: replace SVD by Deep - Auto encoder.
- In practice: Partial fully connected neural network.

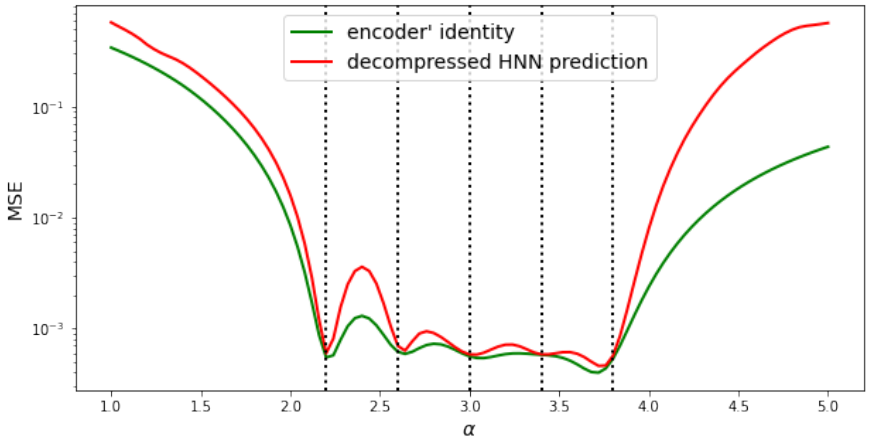
$$\begin{cases} E^x_{\theta_{ex}}(\mathbf{x}) = \bar{\mathbf{x}} \\ E^v_{\theta_{ev}}(\mathbf{v}) = \bar{\mathbf{v}} \end{cases}, \begin{cases} D^x_{\theta_{ex}}(\bar{\mathbf{x}}) = \mathbf{x} \\ D^v_{\theta_{ev}}(\bar{\mathbf{v}}) = \mathbf{v} \end{cases}$$

Results

- Results for the reduced model for one trajectory:



- Learning with different initial conditions (varying α):



- **Current step**: varying randomly β and α in the learning step.

Next: structure-preserving reduced model

- **Question**: how do you learn time models ?

$$\frac{d\mathbf{x}(t)}{dt} = F_\theta(\mathbf{x}(t))$$

- We can learn the flow F_θ using data to approximate time derivatives.
- **Physical model**: In general the physic admits specific structure (conservative, dissipative ...)
- **Example**:

$$\frac{d\mathbf{x}}{dt} = J(\mathbf{x})\nabla H(\mathbf{x}) + G(\mathbf{x})\nabla S(\mathbf{x})$$

with J skew-symmetric, G symmetric and $J(\mathbf{x})\nabla S(\mathbf{x}) = G(\mathbf{x})\nabla H(\mathbf{x}) = 0$

- We call $H(.)$ an Hamiltonian which is conserved by structure and $S(.)$ an entropy which is dissipative by structure.
- The ODE's solved in the PIC code are generated by an Hamiltonian.

Next step

- Keep an Hamiltonian structure on the reduced model learning the Hamiltonian and not the ODE flow.
- **Advantage**: Hamiltonian structure + specific time integrator (symplectic scheme) allows to assure some stability for the reduced model.

Deep learning and geometry

Unstructured data and non-euclidian domain

- Many possible applications of deep-learning for PDE's:
  - **Modeling some terms**: closure, turbulence models, reduced models,
  - **Solve PDE**,
  - **Compute coefficients of numerical schemes**: stabilization viscosity, flux limiter, dynamic splitting coefficient
  - **Adaptive solving**: shocks/interface detection, error prediction, meshes refinement.
- For these applications we need neural networks on unstructured data (cloud point, meshes ...) and non euclidean geometry (surface of Torus or sphere ....)

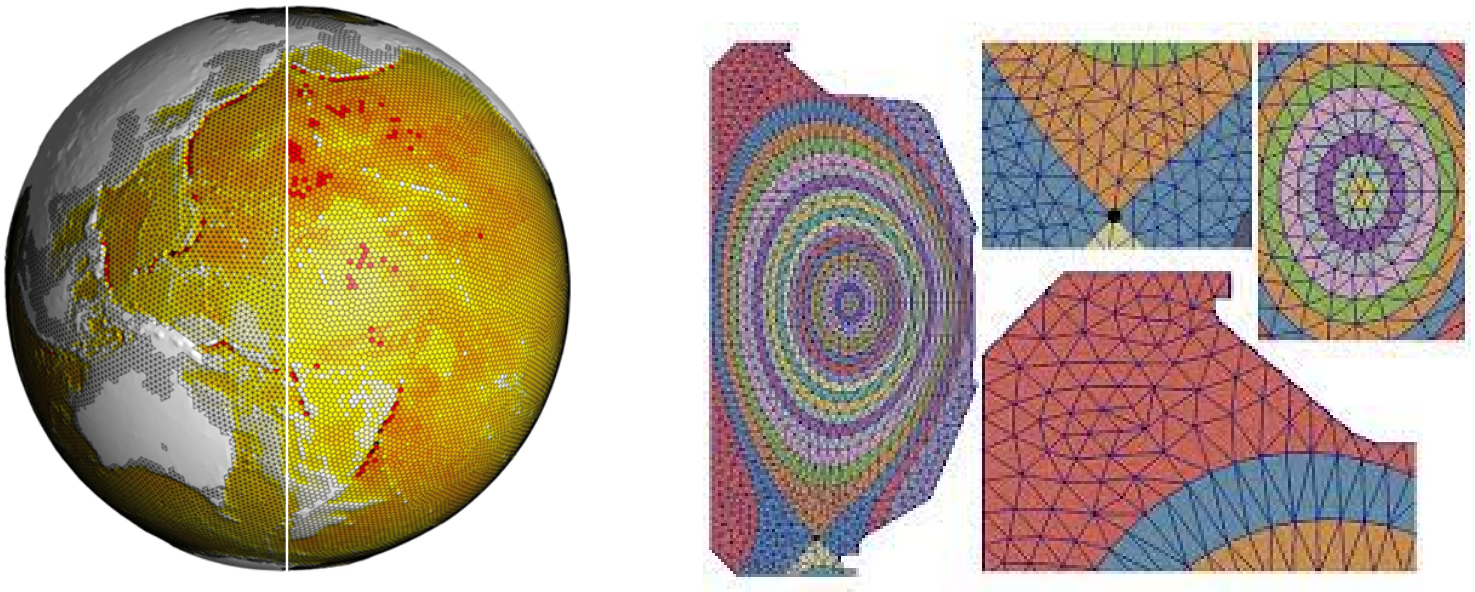


Figure: Picture from google image.

Unstructured data and non-euclidian domain II

- We consider a Graph (Mesh are include inside)
  - Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  - Vertices $\mathcal{V} = \{1, \dots, n\}$
  - Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
  - Edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$
  - Vertex fields $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}^h\}$, represented as $\mathbf{f} = (f_1, \dots, f_n)$
- How to define/generalize the convolution on graph/mesh ?
- No natural answer.

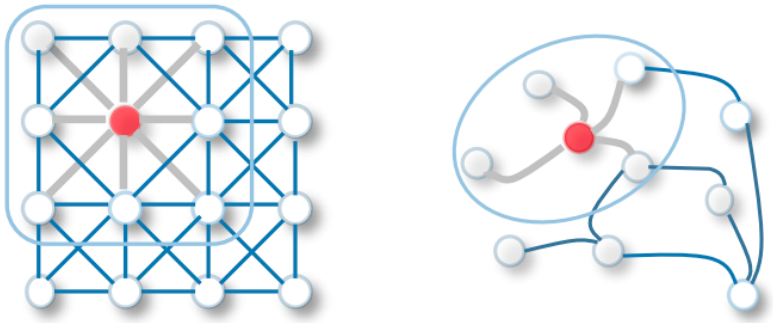


Figure: Picture from google.

Convolution and Fourier transform

- **Spatial definition of convolution**: Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f\left(x'\right) g\left(x - x'\right) dx'$$

- **Spectral definition of convolution**:

$$(f \star g)(x) = \mathcal{F}^{-1}(\mathcal{F}g * \mathcal{F}f)$$

with $\mathcal{F}$ the Fourier transform.

- **Discrete case**:

$$(f \star g)(x) = \Phi(\Phi^t \mathbf{g} \circ \Phi^t \mathbf{f})$$

with Φ the fourier matrix. Each column corresponds to one eigenvector of the Laplacian.

- To apply filter/define convolution we use a Fourier basis (basis function with increasing frequency).
- **Example**: how to apply this on a function/signal defined on the sphere ?

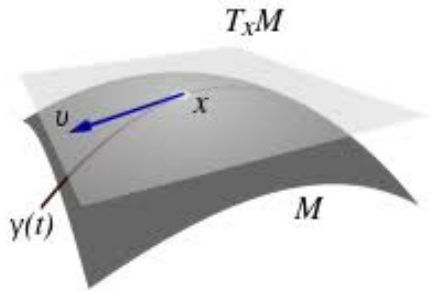
Differential geometry and Laplace Beltrami

Differential geometry

- Application of differential calculus tools to the geometry.
- allows to study differential function defined on curve/surface/manifold etc ...

Riemannian geometry

- differential geometry with the notion of distance and angle.



- Important tools: Laplace-Beltrami

$$\Delta f = \frac{1}{\sqrt{|g|}} \partial_i \left(\sqrt{|g|} g^{ij} \partial_j f \right)$$

with g the Riemanian metric tensor (manifold-dependent).

- **Key point**: the eigenvectors of Δ give a basis of function defined on the manifold with increasing frequency. Fourier basis on each Riemannian manifold.

Example: manifold harmonics

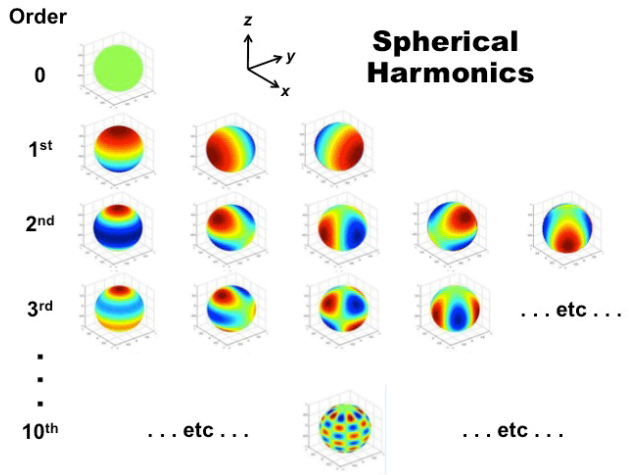


Figure: Google images

Figure: Issue to B. Vallet and B. Lévy. Spectral Geometry Processing with Manifold Harmonics

Graph Laplacian

Question

Can we construct the same theory for graph/mesh to design specific Fourier basis and filters ? yes

- The derivative operator $d : \mathcal{H}(V) = (\mathbb{R}^{|V|}, <,>_V) \rightarrow (\mathbb{R}^{|E|}, <,>_E) = \mathcal{H}(E)$ can ben defined

$$\forall e_{ij} \in \mathcal{E}, \quad d(e_{ij}) = \gamma(w_{ij})(f(j) - f(i))$$

with $f(i), f(j)$ the value of the function f at the vertex i and j and

$$< f, g >_V = \frac{1}{|V|} \sum_{i=1}^{|V|} f_i g_i \chi_i, \quad < f, g >_E = \frac{1}{2|V|} \sum_{i,j}^{|V|} f_{ij} g_{ij} w_{ij}$$

- We can define the dual operator with the relation

$$< df, g >_V = < f, d^* g >_E, \quad \forall f \in \mathcal{H}(V), g \in \mathcal{H}(E)$$

- As in the exterior calculuss theory (differential geometry), the graph Laplacian $\Delta : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ is given by

$$\Delta(i) = d^* d$$

Graph Laplacian II

- Explicit formula.

$$\Delta(i) = \left[f(i)\frac{1}{n}\sum_{j=1}^{n}\frac{w_{ij}}{d_i} - \frac{1}{n}\sum_{j=1}^{n} f(j)\frac{w_{ij}}{d_i} \right]$$

- for specific choice of $\gamma(.)$ and $\chi(.)$.

Question

Can we link this graph Laplacian to Laplace-Beltrami operator ? yes

Result

We take $\mathcal{M}$ a Riemanian manifold of dimension m in $\mathbb{R}^d$. $X = (x_1, ...x_n)$ are points of the manifold distributed with a probability law $p(x)$ on $\mathcal{M}$.
We construct a graph of nearest neighbors (close to mesh). On some technical assumption we have:

$$\Delta \rightarrow_{n\rightarrow\infty} C\frac{1}{p^s}\nabla\cdot(p^s\nabla f)$$

If $s = 0$ and $p(x)$ an uniform law, we obtain the Laplace-Beltrami operator.

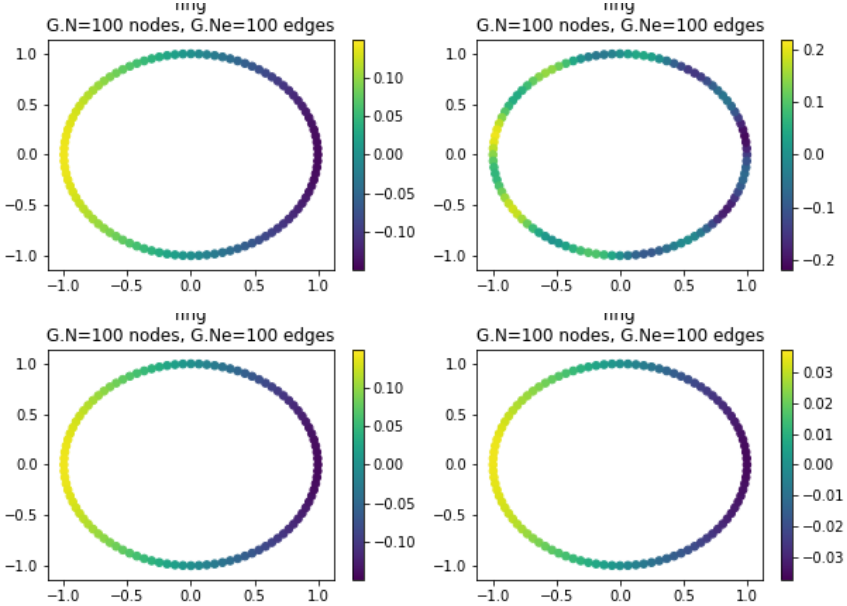
- So, if the graph is "a discrete version of a manifold", the graph Laplacian can be an approximation of the Laplace-Beltrami operator.

Spectral convolution and Graph-signal process

- We consider a Graph G and the Graph Laplacian matrix L.
- We call Φ the eigenvectors of L.
- We obtain a natural definition of convolution on graph:

$$\mathbf{f} \star \mathbf{g} = \Phi(\underbrace{\Phi^t \mathbf{g}}_{K \in D_{n,n}(\mathbf{R})} \circ \Phi^t \mathbf{f})$$

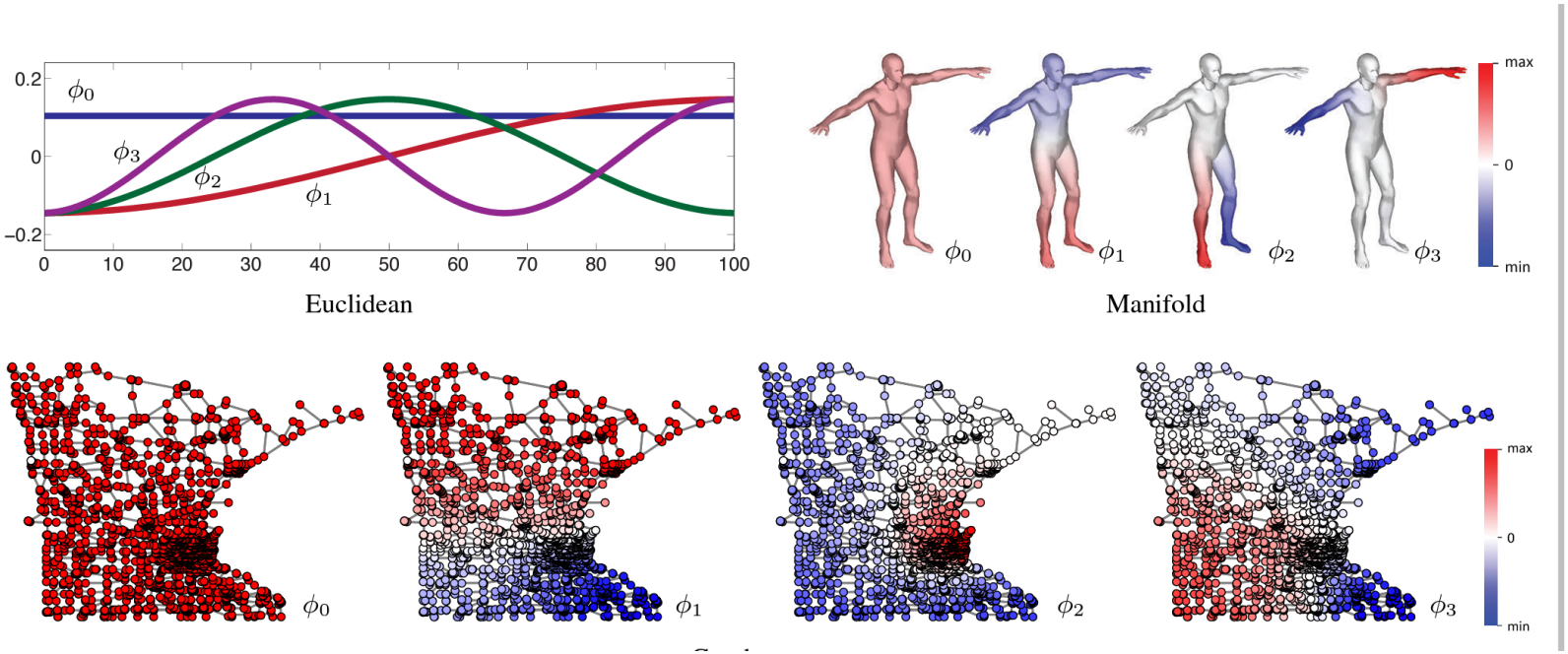
with n signal size and K the filter



- Defaults for Convolution layer of neural-network:
  - No guarantee of spatial localization of filters,
  - $O(n)$ parameters per layer,
  - $O(n^2)$ computation of forward and inverse Fourier transforms,
  - Filters are basis-dependent do not generalize across graphs.

Spectral convolution and Graph-signal process II

- **Key reference**: *Geometric deep learning: going beyond Euclidean data*. M. Bronstein an al (2016).
- Example (picture come from to the reference):



First GCNN: Chebnet

- **First problem**: $O(n)$ parameters per layer. Solution:

$$\mathbf{f} \star \mathbf{g} \approx \Phi_k(\underbrace{\Phi_k^t \mathbf{g}}_{K \in D_{k,k}(\mathbf{R})} \circ \Phi_k^t \mathbf{f}))$$

  with $k << n$ a cut-off frequency. So Φ_k is the k first eigenvectors. K contains the trainable parameters.
- **Second problem**: the filter can be non-local in space. **Remark**: duality smoothness-locality in fourier/spatial domain. Solution:

$$K_{ii} = \sum_{j=1}^{k} \theta_j B_s(\lambda_i)$$

  with B_s a smooth B-Splines function and λ_i eigenvalues of graph Laplacian.
- Using this we obtain a localized spatial filter with r parameters.
- **Third problem**: Computation of ϕ with complexity $O(n)$. **Idea**:

$$K_{ii} = \sum_{j=1}^{k} \theta_j \lambda^i = P_\theta(\Lambda)$$

- We remark

$$\Phi^t P_\theta(\Lambda) \Phi^t \mathbf{f} = P_\theta(L)\mathbf{f}$$

  So a polynomial of graph Laplacian gives a good candidate for convolutional layer.

Second GCNN: GCN

ChebNet (X. Bresson and al 2016)

We define the normalized graph laplacian $\tilde{L} = 2\frac{L}{\lambda_{max}} - I_d$ and propose

$$P_\theta(\tilde{L})\mathbf{f} = \sum_{j=1}^{k} \theta_j T_j(\tilde{L})\mathbf{f}$$

with T_j a recurrent Tchebychev polynomial. So

$$T_j(\tilde{L})\mathbf{f} = 2\tilde{L}T_{j-1}(\tilde{L}) - T_{j-2}(\tilde{L}), \text{ with } T_0 = \mathbf{f} \text{ and } T_1 = \tilde{L}\mathbf{f}$$

- Simple convolution layer:
  - Using another definition of graph laplacian and choosing $k = 2$:

    $$P_\theta = \theta_0 + \theta_1(L - I_d) = \theta_0 - \theta_1 D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

    with D degree matrix and a W weight matrix. We choose $\theta = \theta_0 = -\theta_1$.
  - Eigenvalues are in $[0, 2]$. Lot of composition can generate instabilities. To finish we apply a transformation to have a spectrum in $[-1, 1]$.

Vanilla GCN (Kipf, Welling 2016)

$$G = \theta(I_d + \tilde{D}^{-\frac{1}{2}} \tilde{W} \tilde{D}^{-\frac{1}{2}})$$

avec $\tilde{W} = W + I_d$ et $\tilde{D} = diag(\sum_{i \neq j} \tilde{w}_{ij})$.

Recents GCNN

- Lot of difference convolutional layers based on spectral approach.
- Some spatial approachs have also been designed:
  - Geodesic CNN (G. Maskic and al 2015): based on local coordinate on tangent space. Mainly for mesh of manifold.
  - Monet (F. Monti and al 2017): generalization of GCNN based on local coordinate. Mainly for mesh of manifold.
  - GRAND (M. Bronstein and al 2022): the convolution layer is a spatial and time discretization of:

$$\partial_t \rho = \nabla \cdot (A(x,t)\nabla\rho)$$

    with A a matrix. Use the theory which link ANN and ODE discretization (Neural ODE R. T. Cheng 2018).
  - Beltrami-flow(M. Bronstein and al 2021): the convolution layer is a spatial and time discretization of:

$$\partial_t \rho = \frac{1}{\sqrt{DetG(x,t)}} \nabla \cdot \left(\frac{\nabla\rho}{DetG(x,t)} \nabla\rho \right)$$

    with the Riemannaian metric $G = I_d + \alpha(\nabla_x \rho)^t (\nabla_x \rho)$. Low diffusion in direction of high gradient.

Application to mesh and PDE I

- Train a neural network to make some task on varying meshes. **Difficulty**: Chebnet/GCN valid when we change the mesh. Not possible for general graph.
- Refinement/moving mesh cannot change the topology of the mesh and the manifold associated.
- We can hope that networks based on a spectral approach of convolution will continue to work.
- Examples: number detected on meshes.

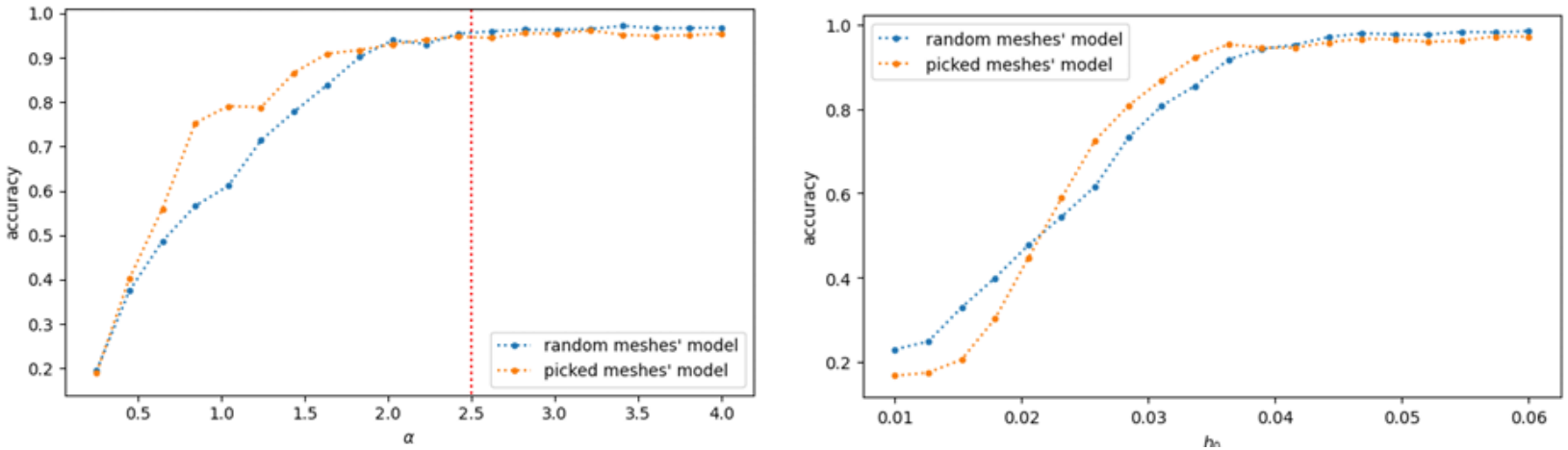


Figure: accuracy results compare to the mesh deformation or mesh refinement

Application to mesh and PDE II

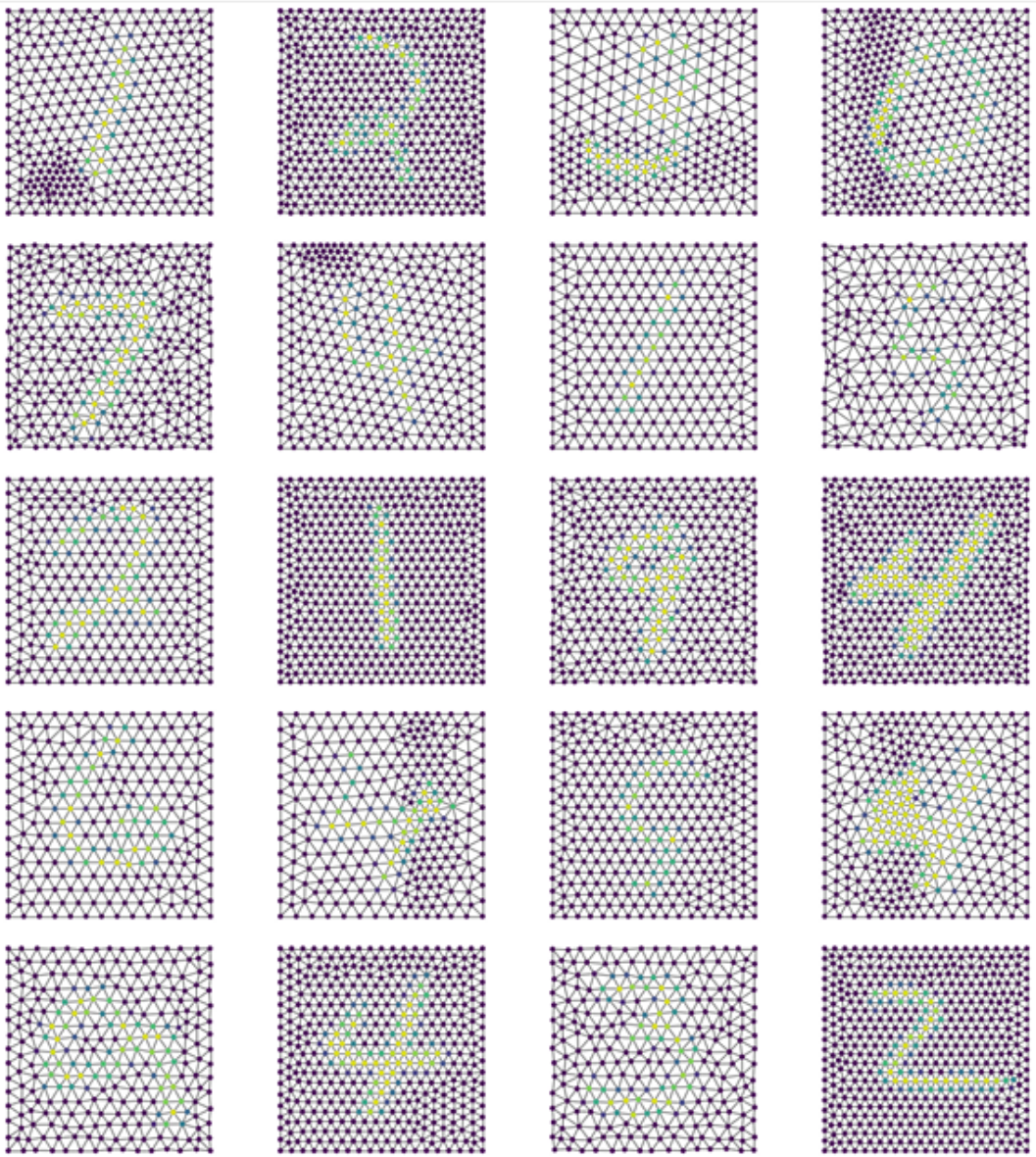
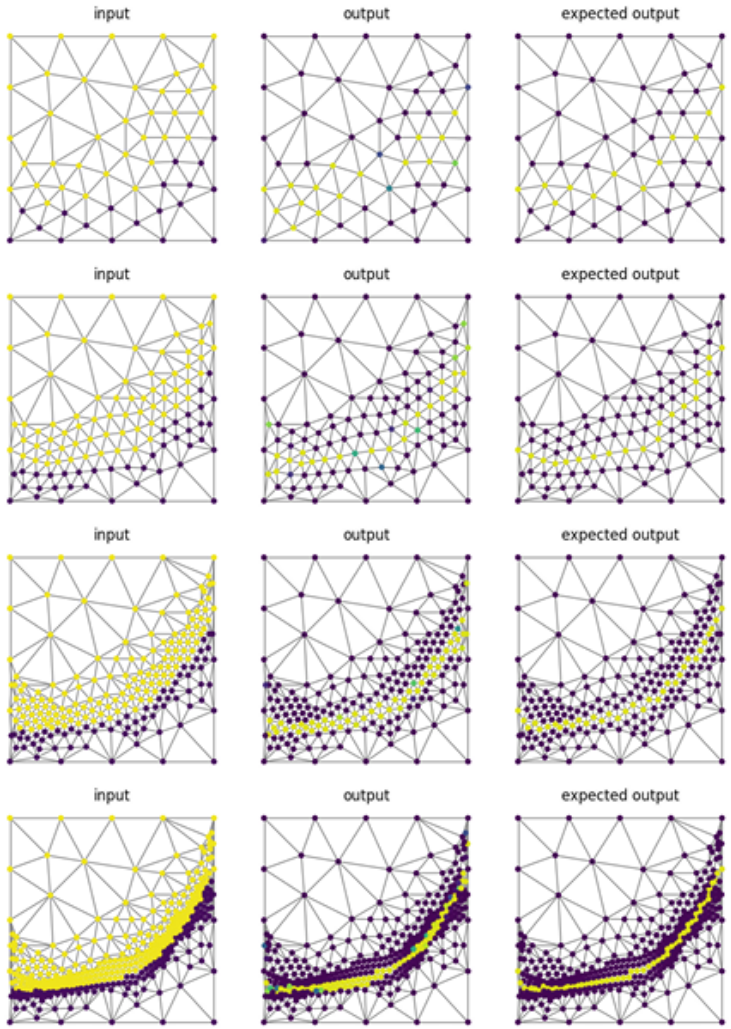


Figure 12: The 20 random meshes.

Application to mesh and PDE III

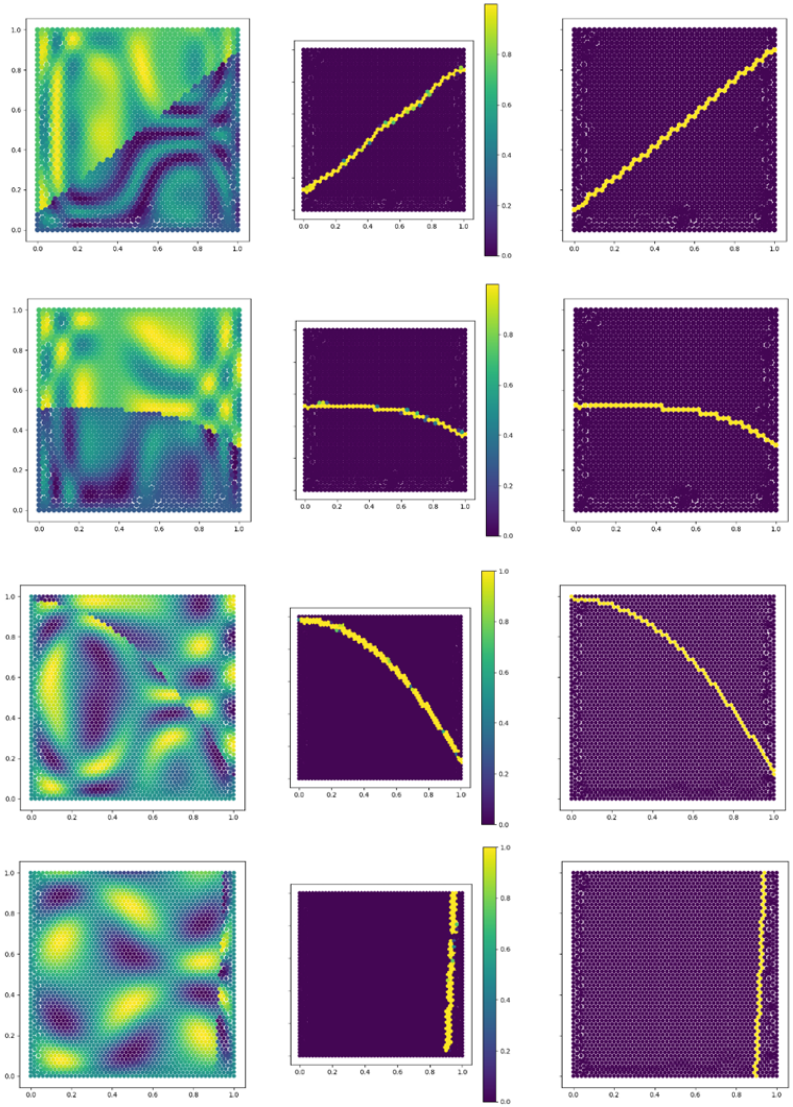
A first application: discontinuity/shock detection and mesh refinement.

Case 1: constant by part function. Localization of discontinuity by GCN and iterative refinement.



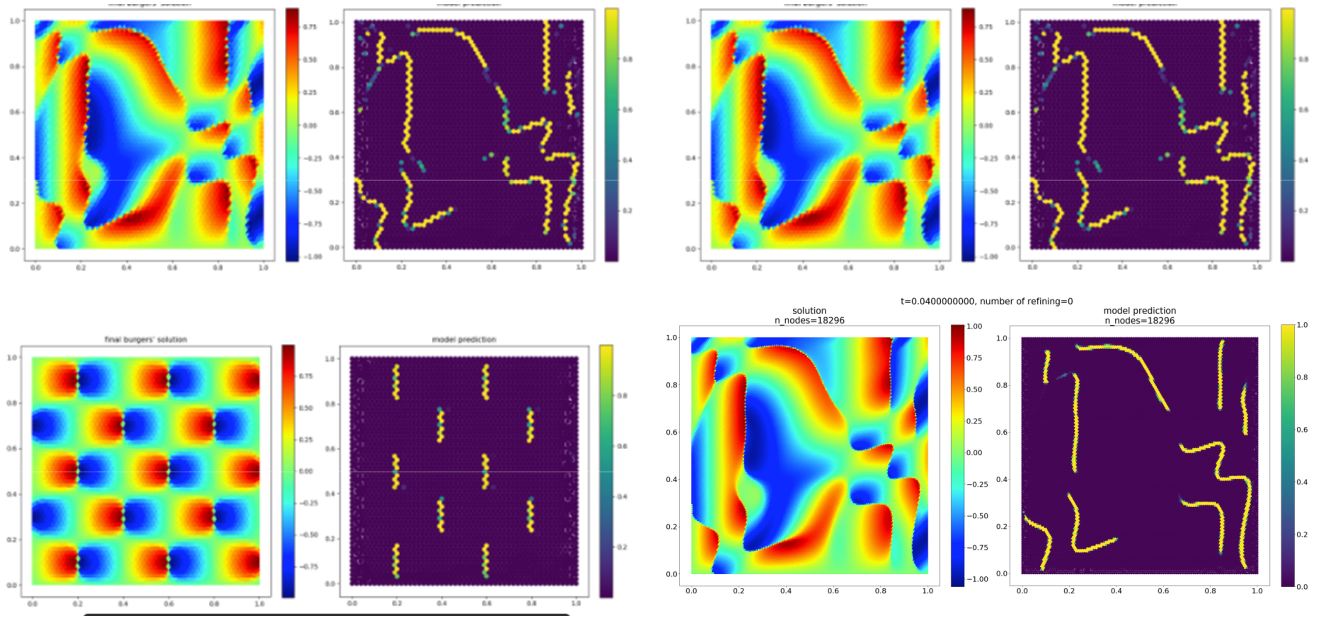
Application to mesh and PDE IV

- **Case 2**: Detection of discontinuity for non piecewise constant function. Unet architecture with Chebnet and geometric pooling layers.
- Training on a single mesh (for informatic reason)

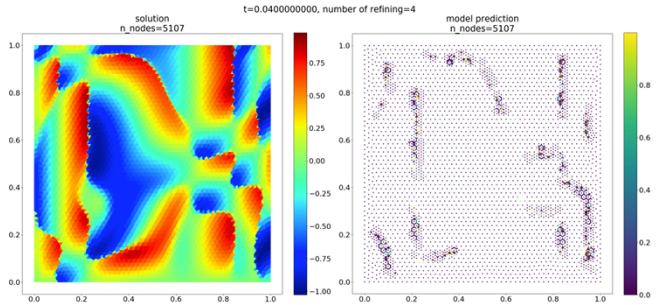


Application to mesh and PDE V

- **Case 3**: Detection of discontinuity in Burgers simulation using previous training



- A first refinement approach



- After refinement the discontinuity is always detected. Less after four refinements but training on a single mesh.

Theory of geometric deep learning

Neural Network architecture and symmetry I

Curse of dimensionality

The space generated by the neural-network function with RELU activation is dense in the $C_0(\Omega)$ functional space for the $L^\infty(\Omega)$ norm with $\Omega \subset \mathbb{R}^d$. If $f \in H^s(\Omega)$ then

$$\sup_{x\in\Omega} \mid f(x) - ANN(x) \mid \approx O(n^{-\frac{s}{d}})$$

with n the number of neurals.

- How to explain the good performance of neural networks for some problems in high dimension (vision, translation problem).

Possible answer

- For some problems on structure data the neural network "encodes" some priors on the problems which allows it to break the curse of dimensionality.
- Mallat 2016, Bronstein and al, 2020

- Which structured data ? which prior. **Theory of Geometric Deep Learning** (Bronstein and al, 2020).

Neural Network architecture and symmetry II

Which data

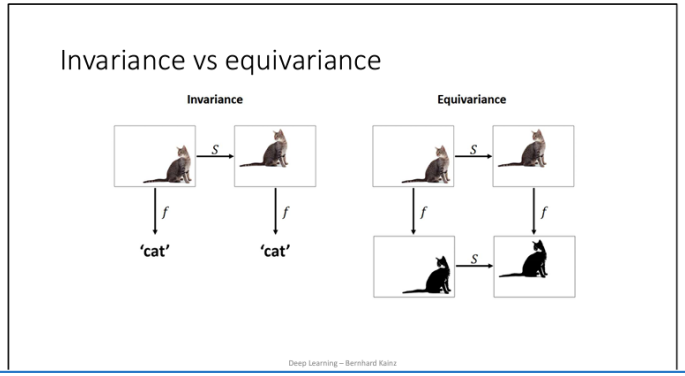
- Grids and sequence (time series and vision problem)
- Homogeneous non-euclidean space (vision problem on sphere etc)
- General manifold and meshes (vision problem, form analysis, ...)
- Graphs and set/point cloud (many applications)

Which prior I

$g \in G$ wth G a group of symetry, $x \in X(\Omega)$ a signal on Ω. f is the target function on Ω.

- G-**Invariance**:
$$f(g.x) = f(x), \quad \forall g \in G, \forall x \in X(\Omega)$$
- G-**Equivariance**:
$$f(g.x) = \rho(g)f(x), \quad \forall g \in G, \forall x \in X(\Omega)$$

with ρ the representation of g (group representation theory).



Neural Network architecture and symmetry III

Which priors II

- Stability compared to signal local deformation . We consider $f(x)$ the target function with x the signal. It is equivalent to say that f is locally invariant compared to the group of diffeomorphism G:

$$\| f(g.x) - f(x) \| \leq c(g) \| x \|$$

with $g.x$ an action group.

- Stability compared to domain local deformation. We consider $f(x, \Omega)$ the target function with x signal and Ω the domain. It is equivalent to say that f is locally invariant compared to the group of isomorphism G:

$$\| f(x, \Omega) - f(x_2, \Omega_2) \| \leq d_D(\Omega, \Omega_2) \| x \|$$

with $x_2 = \eta^{-1} \circ x$ and η invertible mapping between the domains.

Which priors III

- Scale separation. We define a coarsering operator: $P : x \in X(\Omega) \rightarrow x_2 \in X(\Omega_2)$ with x_2 the signal on the coarse domain (less degrees of freedom). We assume that

$$f \approx f_2 \circ P$$

with f the target function, and f_2 the target function on the coarse domain.

Neural Network architecture and symmetry IV

Geometric prior II

- **Grids/sequence**: translation invariance/equivariance,
- **Homogeneous non-euclidean space**: associated symmetry group. Rotation for sphere,
- **Graph and set**: permutation invariance/equivariance,
- **Manifold and mesh**: isometry or local gauge invariance/equivariance.

Geometric Deep Learning blueprint

A "good" neural network chain:

- Convolutional layers which encode G-equivariance and stability to local deformation,
- local Pooling/un-pooling layers which encode scale separation,
- global pooling layer for classification which encodes G-invariance.