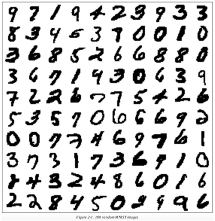
Neural Networks and numerical analysis of PDEs

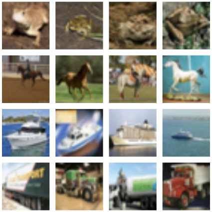
with M. Ancellin (post-doc CEA),
thanks to O. Pironneau and H. Jourdren.

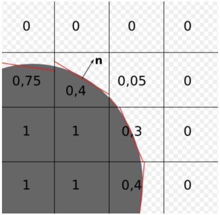
B. Després (LJLL-SU)

Neural Networks, Machine Learning, IA, ...

• Modern ML softwares : TensorFlow (Google 15'), Keras (Chollet 18'), ScikitLEarn (Inria 10'), Pytorch (Facebook), Julia, Matlab, ...







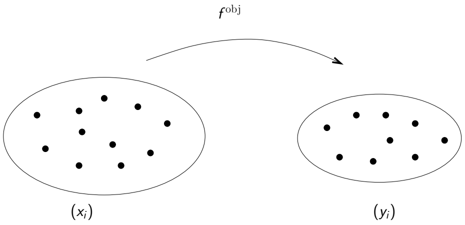
Scientific computing Hesthaven 18', Zaleski 19', D.-Jourdren 20', ...,
Numerical analysis : Yarotsky 17', Daubechies-DeVore et al 19', ...

• Main objectives of this presentation :

- presentation of a functional equation with strong contraction properties, its solution having a Neural Network/FEM interpretation.
- show some new FV schemes for $\partial_t \alpha + \mathbf{a} \cdot \nabla \alpha = 0$

Dataset

- Take a large **dataset** : $\mathcal{D} = \{(x_i, y_i),\ i = 1, \dots\} \subset \mathbb{R}^m \times \mathbb{R}^n$



Postulate : the dataset corresponds to an objective function

$$f^{\mathrm{obj}} : \mathbb{R}^m \longrightarrow \mathbb{R}^n$$

with $x_i \in \mathbb{R}^m$, $y_i = f^{\mathrm{obj}}(x_i) + \varepsilon_i \in \mathbb{R}^n$, and noise $\varepsilon_i \in \mathbb{R}^n$ (hopefully small).

- Examples :

a) **MNIST** : $x_i \in \mathbb{R}^{28 \times 28}$ and $y_i \in [0, 1]^{10}$ with $\sum_i y_i = 1$. Here $m = 784$ and $n = 10$.

b) Takagi and $f^{\mathrm{obj}}(x) = x^2$, $m = n = 1$.

c) $f^{\mathrm{obj}} \in P^r[0, 1]$ where $r \in \mathbb{N}$, $m = n = 1$.

Supervised learning of the objective function

- Take a linear function f with **weight** $W \in \mathcal{M}_{mn}(\mathbb{R})$ and **bias** $b \in \mathbb{R}^n$

$$\begin{aligned} f: \quad \mathbb{R}^m & \longrightarrow \mathbb{R}^n, \\ x & \longmapsto f(x) = Wx + b. \end{aligned} \tag{1}$$

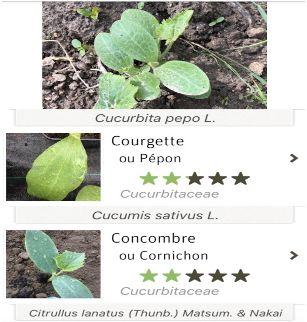
- Consider the convex cost function $J(W,b) = \frac{1}{\mathrm{card}\mathcal{D}} \sum_{(x,y)\in\mathcal{D}} |f(x) - y|^2$.
An optimal value satisfies $J(W_*, b_*) \leq J(W,b) \quad \forall (W,b)$.

Let y and $p(z)$ be discrete probabilities :
$y_i \in [0,1]$, $\sum y_j = 1$; $p_i = \frac{\exp z_i}{\sum_{j=1}^n \exp z_j} \in (0,1)$.

For **classification**, one takes **cross-entropy** (Kullback-Leibler divergence)

$$J(W,b) = - \sum_{(x,y)\in\mathcal{D}} (\log p(f(x)), y) \geq 0.$$

This cost function is convex.



Two more ideas : recursivity and non linearity

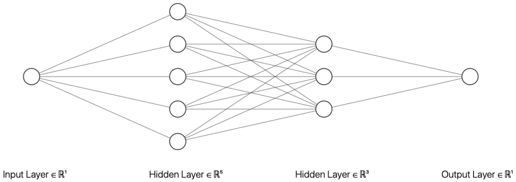
- Recursivity=composition of functions.

$a_0 = m$ is the **input layer**
$a_{p+1} = n$ is the **output layer**
$(a_1, a_2, \ldots, a_p) \in \mathbb{N}^p$ are the **(dense) hidden layers** with **neurons**

- Consider

$$\begin{aligned} f_r : \quad \mathbb{R}^{a_r} &\longrightarrow \mathbb{R}^{a_{r+1}}, \\ X_r &\longmapsto f_r(X_r) = W_r X_r + b_r \end{aligned}$$

and the function $f = f_p \circ f_{p-1} \ldots f_2 \circ f_1 \circ f_0$.



Depth $= p$ is the number of layers.
Width $= \max_r a_r$ is the maximal number of neurons per layer.

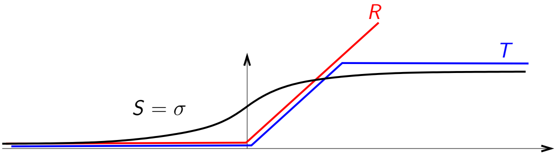
Non linearity

- Non linearity is added with an **activation function**.

Sigmoid $\in C^1(\mathbb{R})$. A sigmoid $S = \sigma$ is monotone, $0 < \sigma' < 1$, with limit value 0 at $-\infty$ and limit value 1 at $+\infty$.

ReLU $\in C^0(\mathbb{R})$. It is defined by $R(x) = \max(0, x)$.
Thresholding yields $T(x) = \min(R(x), 1)$.

Generalization component wise to activation functions $\mathbb{R}^q \to \mathbb{R}^q$.

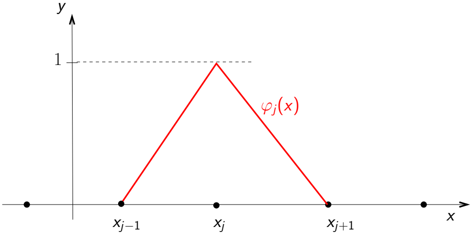


A function defined through a generic **neural network** is

$$f = f_{p+1} \circ \left\{ \begin{array}{c} R \\ S \\ T \end{array} \right\} \circ f_p \circ \cdots \circ \left\{ \begin{array}{c} R \\ S \\ T \end{array} \right\} \circ f_1 \circ \left\{ \begin{array}{c} R \\ S \\ T \end{array} \right\} \circ f_0.$$

Connection with Finite Element

Finite Element function :
hat/P_1 function



$$\begin{aligned}\varphi_j(x) &= R(R(\tfrac{x-x_{j-1}}{h}) - 2R(\tfrac{x-x_j}{h})) \\ &= T(T(\tfrac{x-x_{j-1}}{h}) + T(\tfrac{x_{j+1}-x}{h}) - 1).\end{aligned}$$

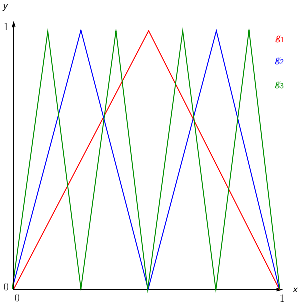
Take $f(x) = \sum_{j\in\mathbb{Z}} f_j \varphi_j(x)$ with $f_j = f^{\mathrm{obj}}(x_j)$. Then

$$\|f^{\mathrm{obj}} - f\|_{L^2_{\mathrm{loc}}} \le Ch^2 \left\| \frac{d^2}{dx^2} f^{\mathrm{obj}} \right\|_{L^2_{\mathrm{loc}}}$$

But of course, $f_j \neq f^{\mathrm{obj}}(x_j)$ in NNs.

Takagi function and power of depth

Let $g : [0,1] \longrightarrow [0,1]$ be the hat function in dimension one
$g(x) = 1 - |1-2x| = 2R(x) - 4R\left(x - \frac{1}{2}\right)$ for $0 \leq x \leq 1$.
Starting from $g_1 = g$, one defines $g_{r+1} = g \circ g_r$



Set $h_r = \sqrt{3}\left(g_r - \frac{1}{2}\right)$, then

$$\int_0^1 h_r(x)h_s(x)dx = \delta_{rs}.$$

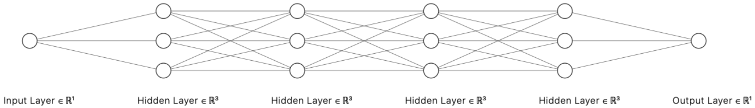
Also

$$\int_0^1 g_r'(x)g_s'(x)dx = 2^{2r}\delta_{rs}.$$

Takagi 1901' : An example of the continuous function without derivative
$f^{\mathrm{Tak}}(x) = \sum_{r\geq 1} \frac{1}{2^r} g_r(x)$. Note $f^{\mathrm{Tak}} \notin H^1(a,b)$ for all $0 \leq a < b \leq 1$.

<u>Yarostky 17' (and many others)</u> : $x^2 = x - \sum_{r\geq 1} \frac{1}{4^r} g_r(x)$. So
$\left\| x^2 - \left(x - \sum_{r=1}^{p} \frac{1}{4^r} g_r(x)\right) \right\|_{L^\infty(0,1)} \leq \sum_{p+1\leq n} \frac{1}{4^r} = \frac{1}{3\times 4^p}$.

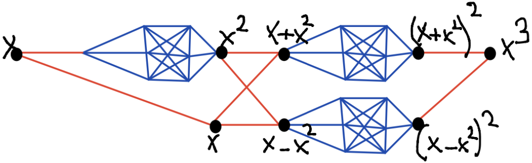
The multiplication $x \to x^2$ is obtained on a dense neural network



Width $= 3$, **depth** $= p$: accuracy $\varepsilon = O(4^{-p})$, cost $= O(p) \approx C|\log \varepsilon|$.

Polarization formula

$\Longrightarrow$ The same for $x \to x^3 = \frac{1}{4}(x + x^2)^2 - \frac{1}{4}(x - x^2)^2$ and for $x \to x^n$
$\Longrightarrow$ The same for $(x, y) \to xy = \frac{1}{4}(x + y)^2 - \frac{1}{4}(x - y)^2, \ldots$



Yarostky 17'

Set $F_{n,d} = \left\{ f \in W^{n,\infty}([0,1]^d), \|f\|_{W^{n,\infty}([0,1]^d)} \leq 1 \right\}$

Theorem

For any d, n and $\varepsilon \in (0,1)$, there is a ReLU network architecture that

- *is capable to express any function in $F_{n,d}$ with error ε*
- *has the depth at most $c\,(\log 1/\varepsilon + 1)$ and at most $c\varepsilon^{-d/n}\,(\log 1/\varepsilon + 1)$ weights and computation units, with some constant $c = c(d,n)$.*

- Yarotsky 17' : Error bounds for approximations with deep ReLU networks.
- Daubechies, DeVore et al 19' : Nonlinear App. and (Deep) ReLU Networks.
- Opschoor, Petersen, Schwab 19' : Deep relu networks and high-order finite element methods.
- He/.../Zheng 20' : Relu deep neural networks and linear finite elements.

- M. Hata, M. Yamaguti 83' :, Weierstrass's function and chaos.
- Hata, Yamaguti 84' : The Takagi Function and Its Generalization.

A functional equation for polynomials/no polarization

$I = [0,1]$ and $C^0(I)$ with norm $\|f\|_{L^\infty(I)} = \max_{i\in I} |f(x)|$.
Weierstrass theorem $\overline{\cup P^n} = C^0(I)$, where $P^n = \{\deg(p) \leq n\}$.

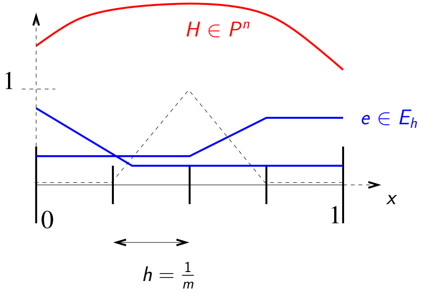


FIGURE – Objective is $f^{\mathrm{obj}} = H \in P^n$.

$V_h = \left\{u \in C^0(I),\ u_{|(x_j,x_{j+1})} \in P^1 \text{ for all } 0 \leq j \leq m-1\right\}.$
$E_h = \{e \in V_h :\ e(I) \subset I,\ e \text{ non constant on exactly one int.}\} \subset V_h.$

Functional equation

Problem : Find $(e_0, e_1, \ldots, e_r, \beta_1, \ldots, \beta_r) \in V_h \times (E_h)^r \times \mathbb{R}^r$ such that the identity below holds

$$H(x) = e_0(x) + \sum_{i=1}^{r} \beta_i H(e_i(x)), \quad x \in I,$$

with the contraction condition $K < 1$, $\qquad K = \sum_{i=1}^r |\beta_i|$.

Theorem (D.+Ancellin 20')

Existence of solution for well chosen $e_i \in E_h$ and m large enough.

Example : set $e_1(x) = \min(2x, 1)$ and $e_2(x) = \min(2(1-x), 1)$ with $e_1, e_2 \in E_h$ for $h = 1/2$. Then $H_1(x) = x(1-x)$ satisfies

$$H_1(x) \quad = \quad \tfrac{1}{4}(g(x) - 1) + \tfrac{1}{4} H_1(e_1(x)) + \tfrac{1}{4} H_1(e_2(x)), \quad e_1, e_2 \in E_h.$$

Once $(e_0, e_1, \ldots, e_r, \beta_1, \ldots, \beta_r) \in V_h \times (E_h)^r \times \mathbb{R}^r$ are determined,
it is a functional equation with H as a solution.

Structure of the proof 1/3

Generically one has $e_i(x) = a_i + (b_i - a_i)\frac{x-x_j}{h}$ for $x_j \le x \le x_{j+1}$.
The global problem is equivalent to local problems where $p = H'' \in P^{n-2}$.

> For all subintervals (index is $j = 0, 1 \ldots, m-1$) :
> find triples $(a_i, b_i, \gamma_i)_{i=1}^s \in (I \times I \times \mathbb{R})^s$ such that $b_i - a_i \neq 0$ for all i and
>
> $$p(x_j + hy) = \sum_{i=1}^s \gamma_i p(a_i + (b_i - a_i)y), \qquad 0 \le y \le 1.$$

$\Longrightarrow$: The correspondance is $\gamma_i = \beta_i \left(\frac{b_i - a_i}{h}\right)^2$.
$\Longleftarrow$: For $b_{i,j} - a_{i,j} \neq 0$ define

$$e_{i,j}(x) = \begin{cases} a_{i,j} & \text{for } 0 \le x \le x_j, \\ a_{i,j} + \frac{b_{i,j} - a_{i,j}}{h}(x - x_j) & \text{for } x_j \le x \le x_{j+1} = x_j + h, \\ b_{i,j} & \text{for } x_{j+1} \le x \le 1, \end{cases}$$

$\beta_{i,j} = \frac{h^2}{(b_{i,j} - a_{i,j})^2}\gamma_{i,j}$ and $e_0(x) = H(x) - \sum_j \sum_i \beta_{i,j} H(e_{i,j}(x))$.
Then $e_0 \in V_h$.

By differentiation, the problem is equivalent to the square linear system

$$M_j X_j = b_j, \qquad 0 \le j \le m-1. \tag{2}$$

The square $(n-1)\times(n-1)$ matrix is

$$M_j = \begin{pmatrix} p(a_{1,j}) & p(a_{2,j}) & \dots & p(a_{n-1,j}) \\ c_{1,j}p'(a_{1,j}) & c_{2,j}p'(a_{2,j}) & \dots & c_{n-1,j}p'(a_{n-1,j}) \\ \dots & \dots & \dots & \dots \\ c_{1,j}^{n-2}p^{(n-2)}(a_{1,j}) & c_{2,j}^{n-2}p^{(n-2)}(a_{2,j}) & \dots & c_{n-1,j}^{n-2}p^{(n-2)}(a_{n-1,j}) \end{pmatrix} \tag{3}$$

where $c_{i,j} = b_{i,j} - a_{i,j}$.
The unknown of the linear system is $X_j = (\gamma_{1,j}, \gamma_{2,j}, \dots, \gamma_{n-1,j})^T \in \mathbb{R}^{n-1}$.

The RHS of the linear system is $b_j = \left(p(x_j), hp'(x_j), \dots, h^{n-1}p^{(n-1)}(x_j)\right)^T$ which is bounded **uniformly with respect to** m

$$\|b_j\|_\infty \le C_1.$$

End of the proof 3/3

Take $\mu = \frac{1}{2(n-1)}$. For $1 \leq i \leq n-1$, set

$$a_{i,j} = i\mu \text{ and } b_{i,j} = (i+1)\mu.$$

Then

- the RHS is bounded $\|b_j\|_\infty \leq C_1(p)$,
- the matrix M_j is independent of the sub-interval index j,
- the matrix M_j is Vandermonde and $\|M_j^{-1}\| \leq C_2$,
- the solution is bounded $\|\gamma_j\|_\infty \leq C_3 = C_1 C_2$,
- the constant is

$$K = \sum_{j=0}^{m-1} \sum_{i=1}^{n-1} |\beta_{i,j}| \leq \sum_{j=0}^{m-1} \sum_{i=1}^{n-1} \frac{h^2 |\gamma_{i,j}|}{(b_i - a_i)^2}$$

$$\leq \sum_{j=0}^{m-1} \sum_{i=1}^{n-1} \frac{h^2 C_3}{\mu^2} \leq m(n-1) \frac{h^2 C_3}{\mu^2} = \frac{C_4}{m}.$$

Therefore the contraction property is satisfied for $m \geq C_4 = C_4(H)$.

Interpretation

Solution of a fixed point equation $\Longrightarrow$ Neural Network with ReLU

$$\begin{cases} H_0 = 0, \\ H_{k+1} = e_0 + \sum_{1\leq i\leq r} \beta_i H_k \circ e_i. \end{cases}$$

$$H_k = e_0 + \sum_{p=1}^{k-1} \left(\sum_{1\leq i_1,\dots,i_p\leq r} \left(\beta_{i_1} \dots \beta_{i_p}\right) e_0 \circ e_{i_1} \circ \dots \circ e_{i_p} \right), \quad k \geq 1.$$

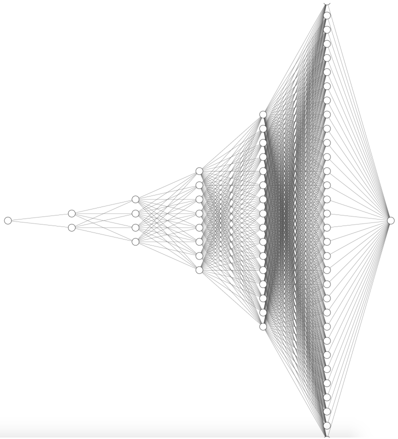
One has

$$\|H_k - H\|_{L^\infty(I)} \leq K^k \|H\|_{L^\infty(I)}, \qquad K = \sum_{1\leq i\leq r} |\beta_i| < 1. \qquad (4)$$

- D.-Ancellin, CRAS 2020, hal preprint https://hal.sorbonne-universite.fr/hal-02959678

Exponential structure of the NN

Here $W \approx r^p$ with $p = k$ the layer index, equal to the depth of the NN.



This NN is fully non standard with respect to the literature (Goodfellow et al, Deep Learning, 2016).

The Network can be flattened.

Training tests

Training= find the best weights/coefficients to fit the objective function.

$$W := \text{all degrees of freedom}$$

$$W \mapsto J(W) = \|H_k(W) - H\|_{L^2(I)}.$$

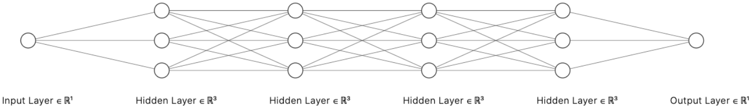
Keras-Tensorflow Degrees of Freedom= all weights W_r, b_r

Julia
Degrees of Freedom= β_i for $1 \leq i \leq r$ and $e_0 \in V_h$.

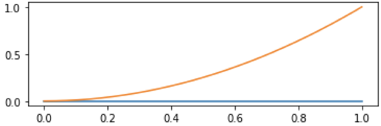
The basis functions $e_i \in E_h$ for $i \geq 1$ chosen in advance
do not belong to the degrees of freedom.

$$f^{\mathrm{obj}}(x) = x^2$$

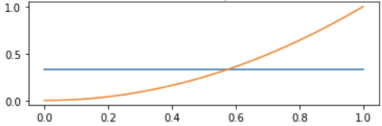
Tensorflow/training+testing/initialization $W^0 = 0 \Rightarrow$ huge problems .



$W^0 = 0$ and $f^0(x) = 0$



$f^k(x) \approx \frac{1}{3}$



Lemma : Take $p \geq 1$ and $(W_r^0, b_r^0) = 0$ for all r. Then $\nabla_{W_r^0} f = 0$ for all r and $\nabla_{b_r^0} f = 0$ for all $r \geq 1$. (proof for $p = 1$ comes from $f(x) = W_1 R(W_0 x + b_0) + b_1$.)

Julia/$\beta^0 = 0 \Rightarrow$ convergence

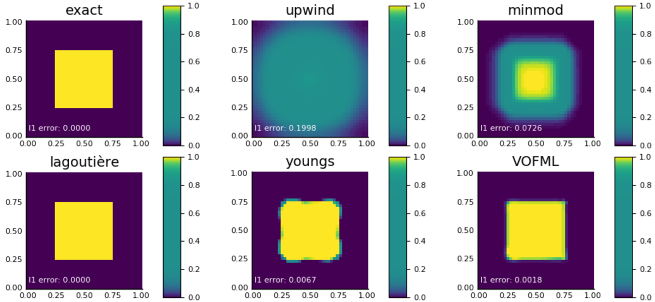
k	1	2	3	4	5	6	7	8
$E(k)$	2.5e-2	6.2e-3	1.5e-3	3.9e-4	9.6e-5	2.4e-5	6.0e-6	1.5e-6
$K(k)$	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Essential for fluid-structure interaction, bubble flows, multiphase flows, ...
(inspired by Hesthaven 18', Zaleski 19')
Model equation is advection

$$\partial_t \alpha + \mathbf{a} \cdot \nabla \alpha = \partial_t \alpha + \nabla \cdot (\mathbf{a}\alpha) = 0, \quad \mathbf{a} = (1, 1).$$

Needs : a) maximum principle, b) conservative, c) super-efficient for

$$\alpha_0(x) = \mathbf{I}_\omega(x) \text{ an indicatrix function} \in BV.$$

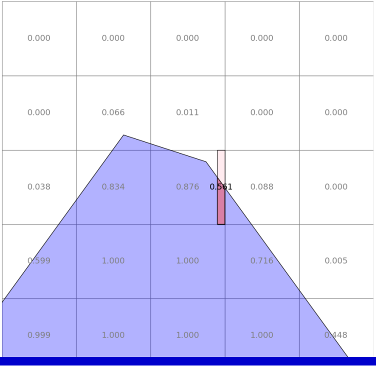


Idea of VOF

Horizontal direction $\partial_t \alpha + \partial_x \alpha = 0$ with $a = 1$

- $\Omega_{ij} = [(i-1)\Delta x, i\Delta x] \times [(j-1)\Delta x, j\Delta x]$
- Swept $:= [(1-\beta)\Delta x < x < \Delta x]$
- $\beta = a\frac{\Delta t}{\Delta x}$ the horizontal measure of the Swept region

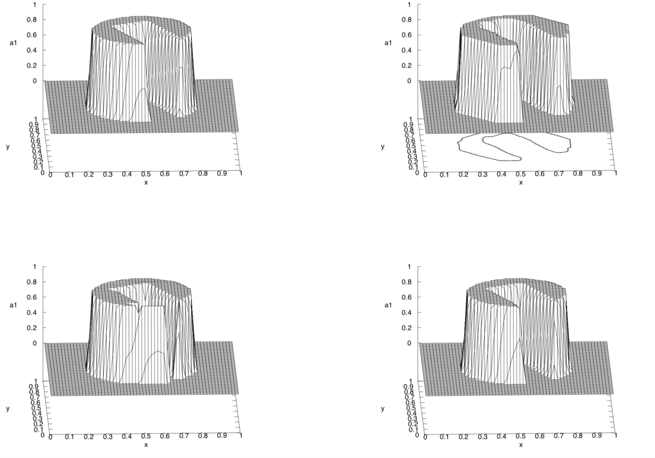
$$\alpha_{ij} = \frac{|\Omega_{ij} \bigcap \omega|}{|\Omega_{ij}|} \in [0,1], \qquad \alpha_* = \frac{|\Omega_{00} \bigcap \text{Swept} \bigcap \omega|}{|\Omega_{00} \bigcap \text{Swept}|} \in [0,1]$$



Consider $f^{\text{obj}} : \mathbb{R}^m \to \mathbb{R}^n$ with $m = 5^2 + 1$ and $n = 1$

$$f^{\text{obj}}(\alpha_1, \dots, \alpha_{25}, \beta) = \alpha_*.$$

Advection of Zalezak profile



- D.+Jourdren : Machine Learning design of Volume of Fluid schemes for compressible flows, JCP 20'.

Conclusion

- Numerical methods for PDEs (FEM, FV, ...)
  offer a natural avenue to understand NNs
- A new functional equation explains that for univariate polynomials, NNs=fixed point iterations.
- Production of new very non linear numerical methods (FV so far).