Physics Informed Neural Networks (PINNs)

Elisa Riccietti and Theo Mary

LIP-ENS Lyon

The context

The problem: numerical approximation of PDE's solutions.

- *Classical approaches*: discretization methods (finite differences, finite elements) and multigrid methods (MG)
- *New advances in machine learning*: Physics Informed Neural Networks (PINNs)

Outline

Recap on classical methods and limitations

Physics Informed Neural Networks (PINNs)

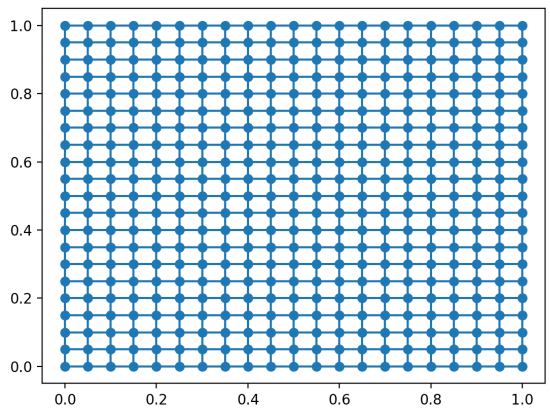
Outline

Recap on classical methods and limitations

Physics Informed Neural Networks (PINNs)

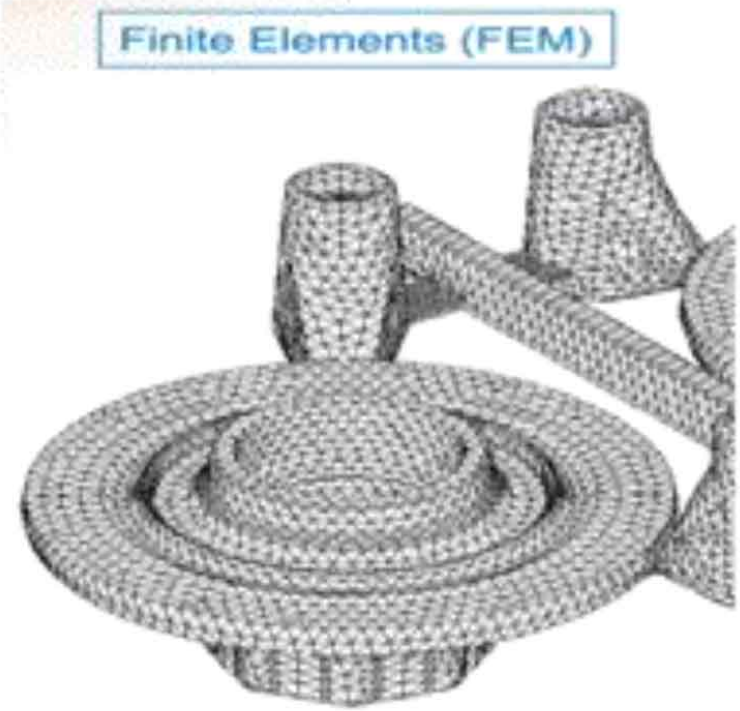
The numerical solution of PDEs: linear case

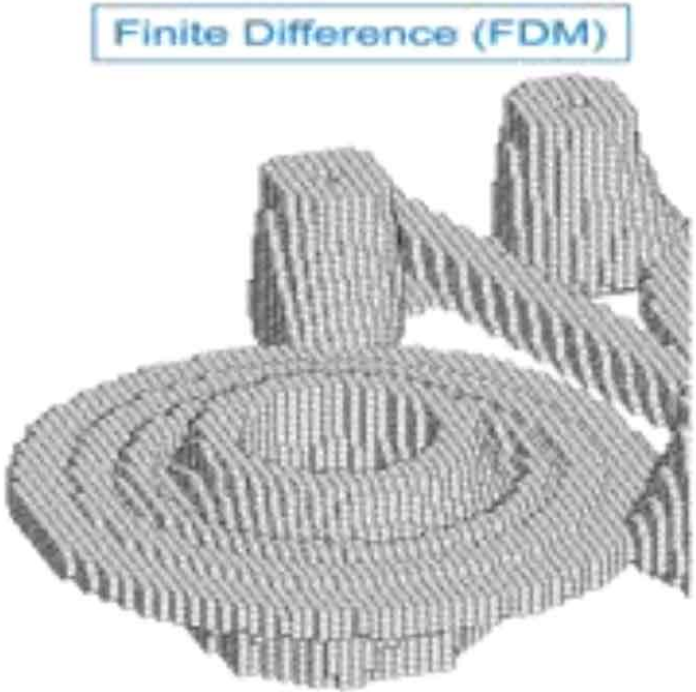
- Classically PDEs are discretized on a grid using finite differences or finite elements
- The resulting linear system $Au = f$ is solved using a fixed point iterative method (Gauss-Seidel or Jacobi)
- The size of the grids impacts the size of the system and the accuracy of the solution approximation



Limitations of classical approaches

1) May be difficult to discretize the domain





Little detour: Finite elements method (FEM)

Illustrative problems P1 and P2

P1 one-dimensional problem

$$\text{P1} \ : \begin{cases} u''(x) = f(x) \text{ in } (0,1), \\ u(0) = u(1) = 0 \end{cases}$$

P2 two-dimensional problem (Dirichlet problem)

$$\text{P2} \ : \begin{cases} u_{xx}(x,y) + u_{yy}(x,y) = f(x,y) & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

where f is given, u is unknown

FEM in two steps

1. One rephrases the original BVP in its weak form. Little to no computation is usually required for this step. The transformation is done by hand on paper.
2. Discretization: the weak form is discretized in a finite-dimensional space. This yields a large but finite-dimensional linear problem whose solution will approximately solve the original BVP.

Weak formulation for P1

If u solves P1, then for any smooth function v such that $v(0) = v(1) = 0$, we have

$$\int_0^1 f(x)v(x)\,dx = \int_0^1 u''(x)v(x)\,dx. \tag{1}$$

Conversely, if u with $u(0) = u(1) = 0$ satisfies (1) for every smooth function $v(x)$ then one may show that this u will solve P1.
We define a new operator or map $\phi(u, v)$ by using integration by parts:

$$\begin{aligned}\int_0^1 f(x)v(x)\,dx &= \int_0^1 u''(x)v(x)\,dx \\ &= u'(x)v(x)|_0^1 - \int_0^1 u'(x)v'(x)\,dx \\ &= -\int_0^1 u'(x)v'(x)\,dx \equiv -\phi(u, v),\end{aligned}$$

where we have used the assumption that $v(0) = v(1) = 0$.

The weak form of P2

If we integrate by parts using a form of Green's identities, we see that if u solves P2, then we may define $\phi(u, v)$ for any v by

$$\int_\Omega fv \, ds = -\int_\Omega \nabla u \cdot \nabla v \, ds \equiv -\phi(u, v),$$

ϕ can be turned into an inner product on a suitable space $H_0^1(\Omega)$ (Sobolev space) of once differentiable functions of Ω that are zero on $\partial\Omega$. We have also assumed that $v \in H_0^1(\Omega)$. The existence and uniqueness of the solution can also be shown.

Discretization of weak form of P1 and P2

The basic idea is to replace the infinite-dimensional linear problem:

$$\boxed{\text{Find } u \in H_0^1 \text{ such that } \forall v \in H_0^1,\ -\phi(u,v) = \int fv}$$

with a finite-dimensional version:

$$\boxed{\text{Find } u \in V \text{ such that } \forall v \in V,\ -\phi(u,v) = \int fv}$$

where V is a finite-dimensional subspace of H_0^1.

How to choose V? There are many possible choices for V, for the FEM we take a space of piecewise polynomial functions

Choice of V for P1

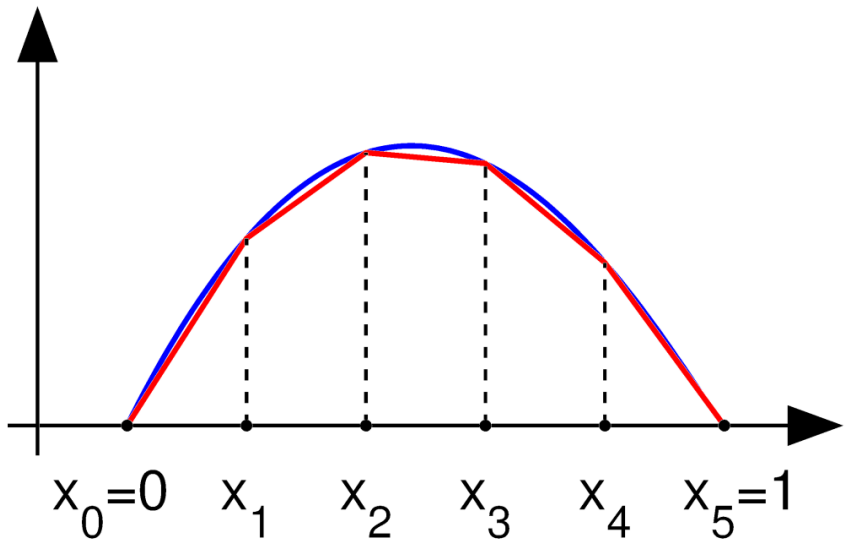
In the interval $(0, 1)$, choose n values of x with

$$0 = x_0 < x_1 < \cdots < x_n < x_{n+1} = 1$$

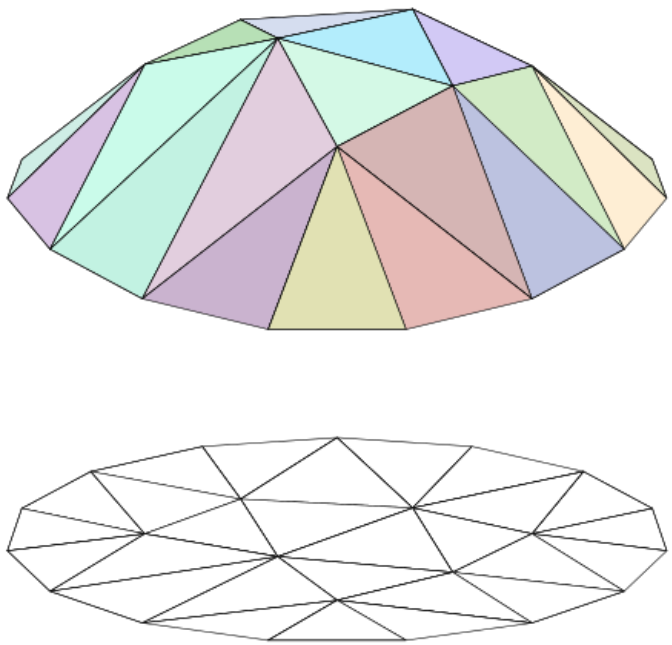
and we define V by:

$$V = \{v : [0,1] \to \mathbb{R} \ : v \text{ is continuous, } v|_{[x_k, x_{k+1}]} \text{ is linear for } k = 0, \ldots, n, \text{ and } v(0) = v(1) = 0\}$$

Observe that functions in V are not differentiable

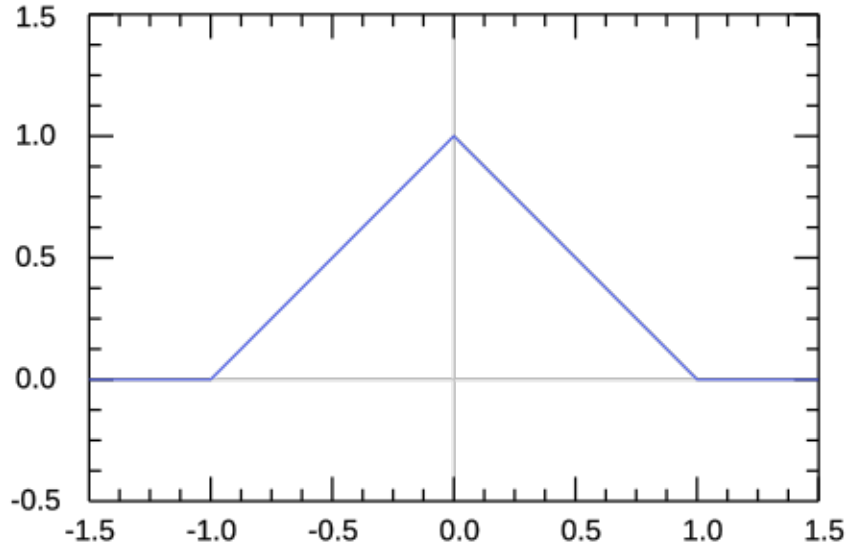


Choice of V for P2



- We consider a triangulation of Ω, V is the set of functions that are linear on each triangle.
- When the triangular mesh becomes finer and finer, the solution of the discrete problem will converge to the solution of the original P2.
- Mesh fineness: largest or average triangle size in the triangulation.

Choosing a basis of V for P1



In the one-dimensional case, for each control point x_k we will choose the piecewise linear function v_k in V whose value is 1 at x_k and zero at every x_j, $j \neq k$, i.e.,

$$v_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}} & \text{if } x \in [x_{k-1}, x_k], \\ \frac{x_{k+1} - x}{x_{k+1} - x_k} & \text{if } x \in [x_k, x_{k+1}], \\ 0 & \text{otherwise,} \end{cases}$$

for $k = 1, \ldots, n$; this basis is a shifted and scaled tent function.

Choosing a basis of V for P2

For the two-dimensional case, we choose again one basis function v_k per vertex x_k of the triangulation of the planar region Ω. The function v_k is the unique function of V whose value is 1 at x_k and zero at every x_j, $j \neq k$.

More complex FEM methods

- In case of curved domains we might replace the triangles with curvilinear elements
- Can replace "piecewise linear" with "piecewise quadratic" or even "piecewise polynomial" ("higher order element")
- The finite element method is not restricted to triangles (tetrahedra, prisms, or pyramids in 3-d or higher-order simplexes in multidimensional spaces).
- More advanced implementations (adaptive finite element methods) utilize a method to assess the quality of the results (based on error estimation theory) and modify the mesh during the solution

Matrix form of the problem

If we write

$$u(x) = \sum_{k=1}^{n} u_k v_k(x) \text{ and } f(x) = \sum_{k=1}^{n} f_k v_k(x)$$

then our problem

$$\forall v \in V,\ -\phi(u, v) = \int fv$$

taking $v(x) = v_j(x)$ for $j = 1, \ldots, n$, becomes

$$-\sum_{k=1}^{n} u_k \phi(v_k, v_j) = \sum_{k=1}^{n} f_k \int v_k v_j dx$$

for $j = 1, \ldots, n$.

Matrix form of the problem

If we denote

- $\mathrm{u} = (u_1, \ldots, u_n)^T$
- $\mathrm{f} = (f_1, \ldots, f_n)^T$
- $L = (L_{ij}) = (\phi(v_i, v_j))_{ij}$
- $M = (M_{ij}) = (\int v_i v_j dx)_{ij}$

we can write the problem in matrix form:

$$-L\mathrm{u} = M\mathrm{f}.$$

Alternatively, it is not necessary to assume $f(x) = \sum_{k=1}^{n} f_k v_k(x)$. and taking again $v(x) = v_j(x)$ for $j = 1, \ldots, n$ the problem becomes $-L\mathrm{u} = \mathrm{b}$, where $\mathrm{b} = (b_1, \ldots, b_n)^t$ and $b_j = \int f v_j dx$ for $j = 1, \ldots, n$.

Small support of the basis

The primary advantage of this choice of basis is that

$$\langle v_j, v_k \rangle = \int_0^1 v_j v_k \, dx \text{ and } \phi(v_j, v_k) = \int_0^1 v_j' v_k' \, dx$$

will be zero for almost all j, k[1].
$\rightarrow$ most of the entries of L and M are zero: sparse linear system with symmetric and positive definite matrix. A technique such as the conjugate gradient method is favored.

---

[1] In the one dimensional case, the support of v_k is the interval $[x_{k-1}, x_{k+1}]$. Hence, the integrands of $\langle v_j, v_k \rangle$ and $\phi(v_j, v_k)$ are identically zero whenever $|j - k| > 1$. Similarly, in the planar case, if x_j and x_k do not share an edge of the triangulation, then the integrals

$$\int_\Omega v_j v_k \, ds \text{ and } \int_\Omega \nabla v_j \cdot \nabla v_k \, ds$$

are both zero

Limitations of classical approaches

2) Cannot directly handle nonlinear equations

Idea: locally linearize the problem
Example:

$$F(u) = -\Delta u + e^u - f = 0$$

- Given u_0, approximate

$$F(u) \sim \hat{F}(u) := -\Delta u + e^{u_0} + e^{u_0}(u - u_0) - f$$

- Solve (approximately) $\hat{F}(u) = 0$ and get u_1.
- Repeat.

Requires to solve a sequence of linear problems and solves an approximated problem

Limitations of classical approaches

3) Curse of dimensionality: the number of samples N necessary to uniformly cover a volume grows exponentially in the dimension of the space p.

- The sampling density d is proportional to $N^{1/p}$.
- If in 1D ($p = 1$) the density is $d = 100$, this means that there is no more than $10^{-2} = 0.01$ distance between points
- To have the same density in dimension $p = 10$ we need $N = 100^{10}$ samples (i.e., an equivalent sampling of a 10-dimensional unit hypercube with a lattice that has the same spacing between adjacent points).

Outline

A new approach for PDEs

A recent development: use neural networks to approximate the solution of a PDE

M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, 2019.

Example:

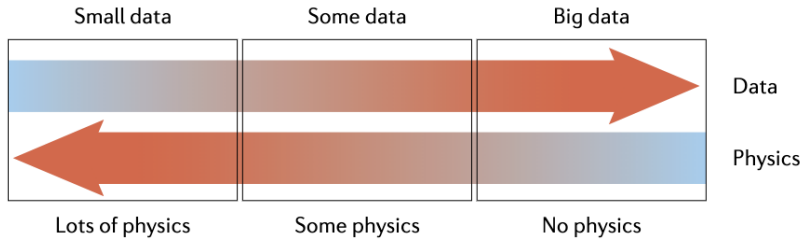
$$-\Delta u(x) = f(x)$$

write $u(x) \sim u_\theta(x) = NN(\theta, x)$

Why this approach ?

- Natural approach for nonlinear equations
- Provides analytic and continuously differentiable expression of the approximate solution
- The solution is meshless, well suited for problems with complex geometries
- The training is highly parallelizable on GPU
- Allows to alleviate the effect of the curse of dimensionality

Scientific machine learning (SML)



Classical physics

- ☺ Do not require data
- ☹ Experimental data for complex physical systems is limited.
- ☹ Requires good knowledge of the physics of the system

Classical ML

- ☺ good practical performance in the big data regime
- ☹ unable to extract interpretable information and knowledge from data
- ☹ Predictions may be physically inconsistent or implausible (observational biases, noisy measurements): poor generalization

Scientific machine learning

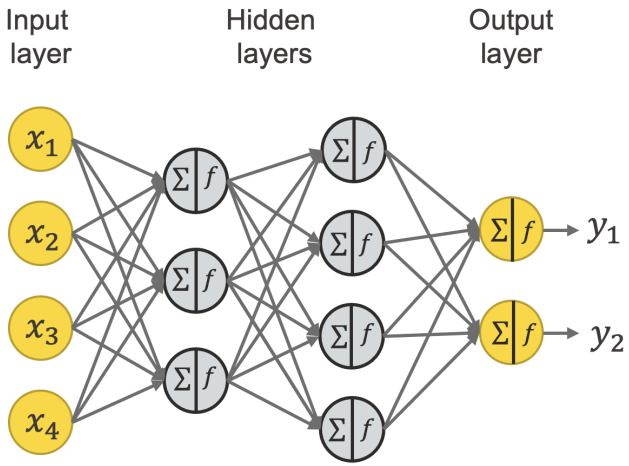
Physics-informed learning integrates (noisy) data and mathematical models, and implements them through neural networks or other kernel-based methods.

- ☺ Networks in SML can be trained from additional information obtained by enforcing the physical laws (providing 'informative priors')
- ☺ It may be possible to design specialized network architectures that automatically satisfy some of the physical invariants for better accuracy, faster training and improved generalization.
- ☺ More interpretable ML methods that remain robust in the presence of imperfect data (missing or noisy values, outliers and so on) and can provide accurate and physically consistent predictions,

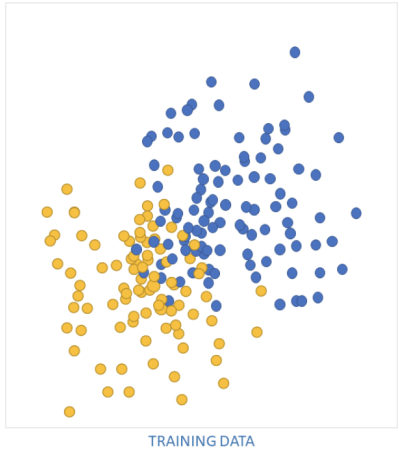
G. Karniadakis et all., Physics-informed machine learning, 2021.

General NN strategy for learning problems

Neural network



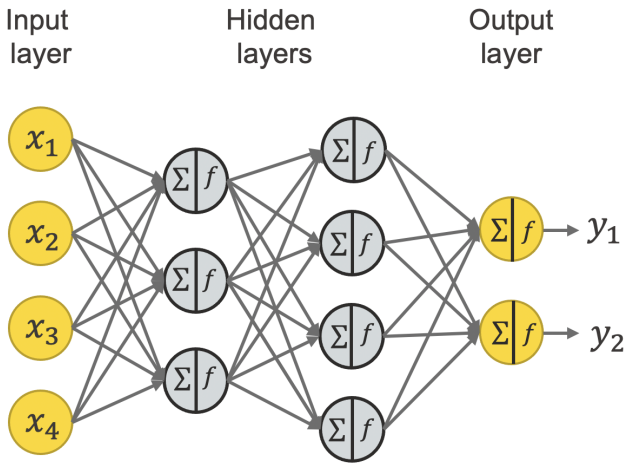
Training data



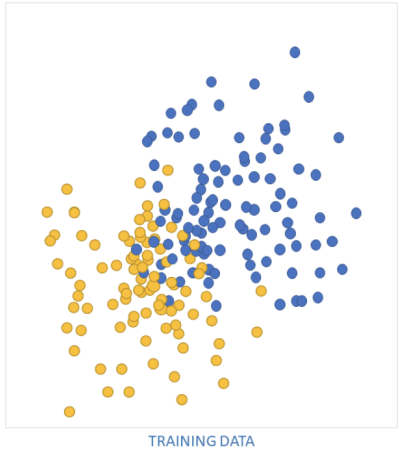
- Dataset composed of input/output couples (x_i, y_i), $i = 1, \ldots, m$.
- Compute $NN(x_i, \theta) =$ $\sigma(W_1 x_i + b_1)$

General NN strategy for learning problems

Neural network



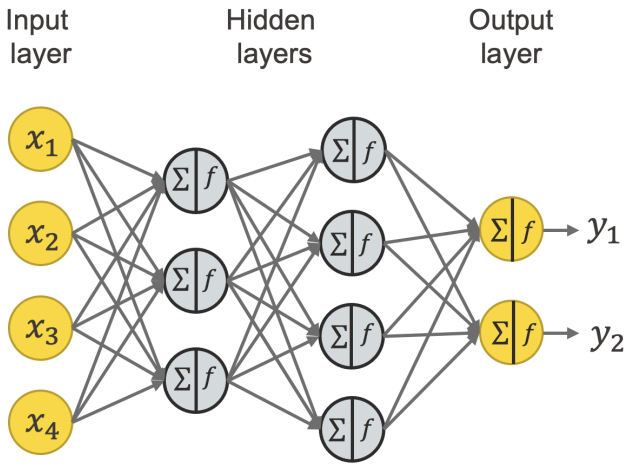
Training data



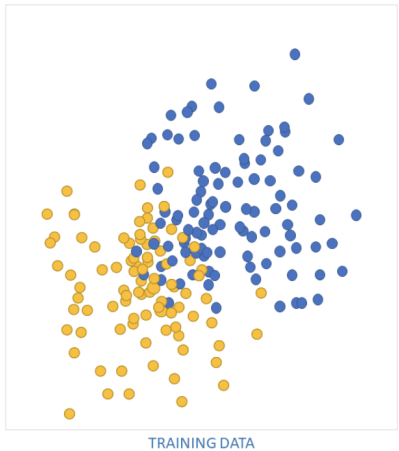
- Dataset composed of input/output couples (x_i, y_i), $i = 1, \ldots, m$.
- Compute $NN(x_i, \theta) = \qquad \sigma(W_2\sigma(W_1 x_i + b_1) + b_2)$

General NN strategy for learning problems

Neural network



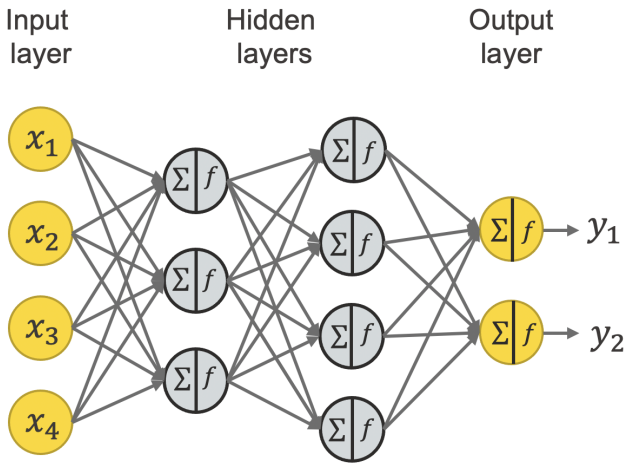
Training data



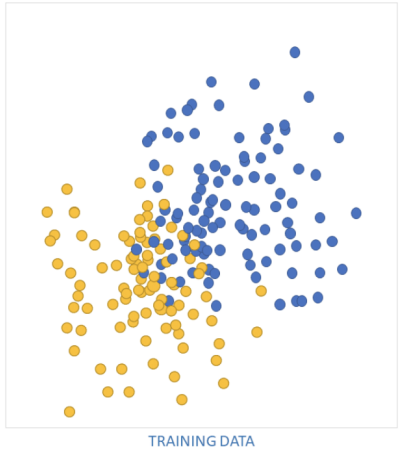
- Dataset composed of input/output couples (x_i, y_i), $i = 1, \ldots, m$.
- Compute $NN(x_i, \theta) = \sigma(W_3\sigma(W_2\sigma(W_1 x_i + b_1) + b_2) + b_3)$

General NN strategy for learning problems

Neural network

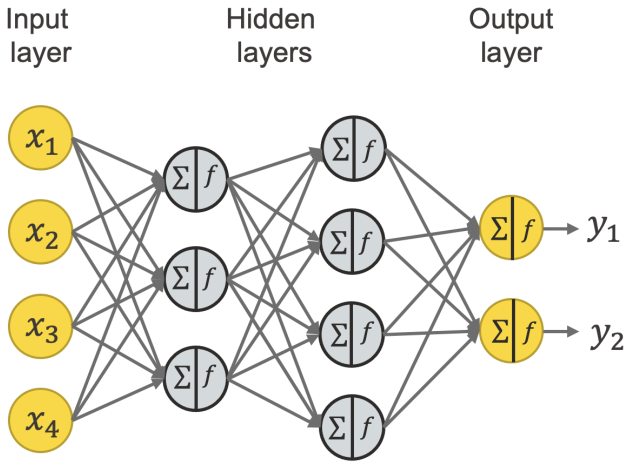


Training data

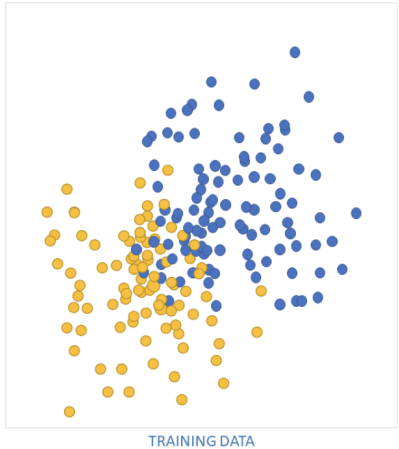


- Dataset composed of input/output couples (x_i, y_i), $i = 1, \ldots, m$.
- Compute $NN(x_i, \theta) = \sigma(W_3\sigma(W_2\sigma(W_1 x_i + b_1) + b_2) + b_3)$
- Loss function $L(\theta; x, y) = \frac{1}{m}\sum_{i=1}^{m}(NN(x_i, \theta) - y_i)^2 = MSE$
- The associated minimization problem : $\min_{\theta \in \Theta} L(\theta; x, y)$
- Optimize by stochastic gradient descent (SGD)

Neural network



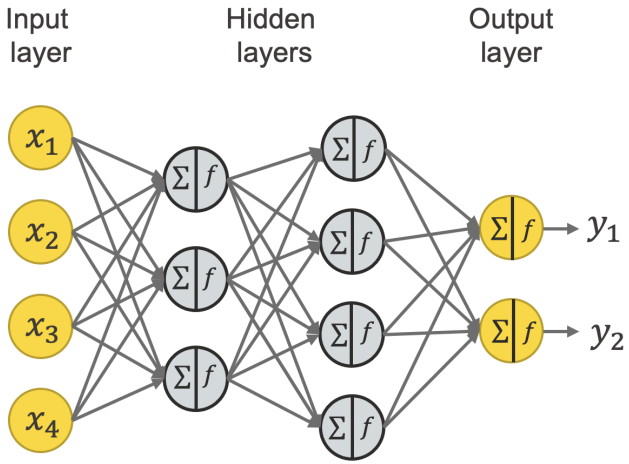
Training data



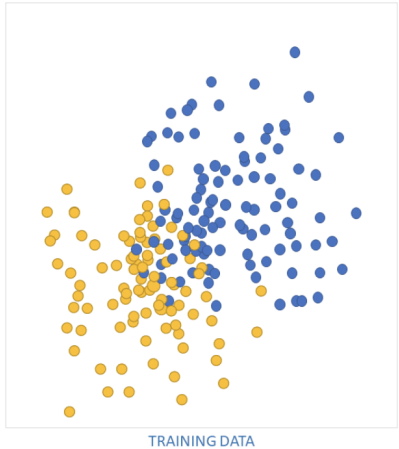
Training problem:

$$\min_{\theta \in \Theta} L(\theta) = \frac{1}{m} \sum_{i=1}^{m} (NN(\theta, x_i) - y_i)^2$$

Neural network



Training data



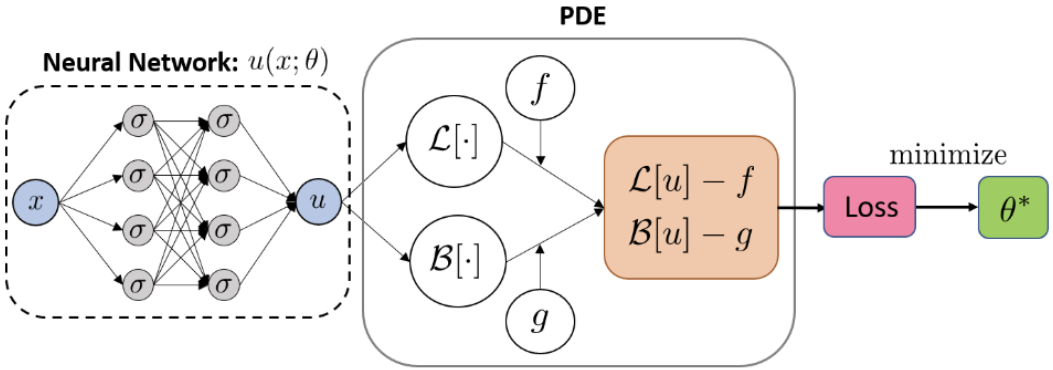
Training problem:

$$\min_{\theta \in \Theta} L(\theta) = \frac{1}{m} \sum_{i=1}^{m} (NN(\theta, x_i) - y_i)^2$$

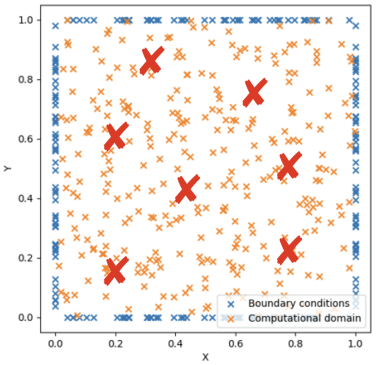
How to integrate the physical knowledge in the model?

Physics Informed Neural Networks (PINNs)

Neural network $NN(\theta, x) \approx u(x)$



Training data



Training problem: $\min_{\theta \in \Theta} L(\theta) = L_{OBS}(\theta) + L_{PDE}(\theta)$

$$L_{OBS}(\theta) = \frac{1}{m_1} \sum_{x_i \in \Omega \cup \partial\Omega} (NN(\theta, x_i) - y_i)^2,$$

$$L_{PDE}(\theta) = \frac{1}{m_{2,i}} \sum_{x_i \in \Omega} (\mathcal{L}(NN(\theta, x_i)) - f(x_i))^2)$$

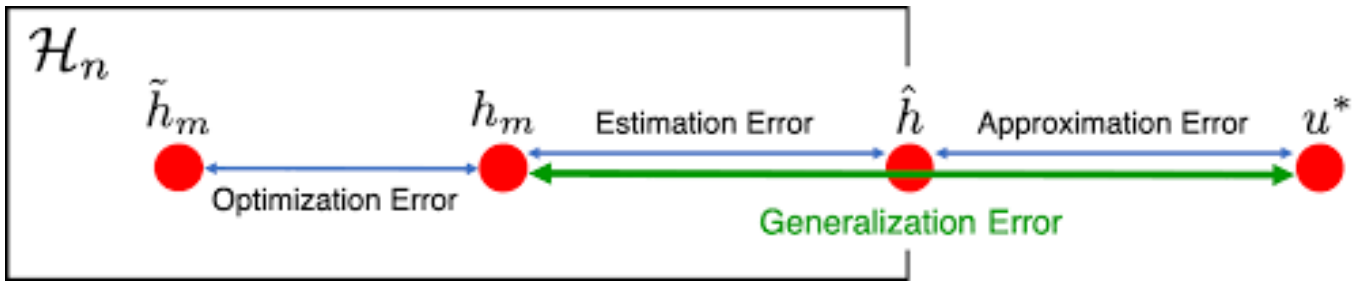
$$+ \frac{1}{m_{2,b}} \sum_{x_i \in \partial\Omega} (\mathcal{B}(NN(\theta, x_i)) - g(x_i))^2)$$

1D and 2D example

On the blackboard !

Convergence theory [Shin, Darbon, Karniadakis, 2020]

- the universality property of NN (approximation error)
- statistical sampling,
- ability of numerical optimizers (ADAM,SGD,...) to reach an approximate global optimum of nonconvex function



- $\tilde{h}_m$ our network,
- h_m a perfectly trained network on the dataset,
- $\hat{h}$ function minimizing the problem with infinitely many data,
- u^* the solution of the underlying PDE

Convergence theory [Shin, Darbon, Karniadakis, 2020]

- L_{PINN} expected loss
- L_m empirical loss over m samples
- α Holder constant of the loss
- d dimension of the space
- HP: the derivation is based on the probabilistic space filling arguments, assume that training data distributions cover the interior and the boundary

With high probability

$$L_{PINN}(h) \leqslant L_m(h) + C(m^{\alpha/d})$$

and

$$L_{PINN}(h_m) \leqslant C(m^{\alpha/d})$$

with $h_m \in H_n$ minimizer of L_m If PDE is linear (elliptic or parabolic)

$$\lim_{m\to\infty} h_m = u^* \text{ in } C^0$$

(seq of minimizers conv uniformly to PDE sol in infinite data regime)

Convergence analysis

Most recent advances: regularized PINNs [Doumeche, Biau, Boyer, 2024]

$$L_{reg}(u) = L_{PINN}(u) + \lambda_t \|u\|^2_{H^m}$$

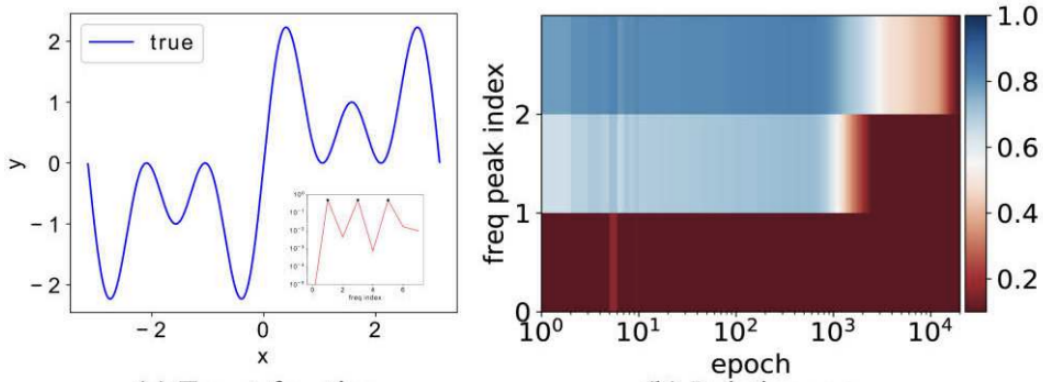
Assume u^* unique solution in H^m, then almost surely

$$\lim_{\lambda_t \to 0, D \to \infty, n \to \infty, k \to \infty} \|u_{\hat{\theta}(k,n,D,\lambda_t)} - u^*\|_{H^m} = 0$$

D: width of the neural network
n: number of points
k: iterations

The optimization error

An important ingredient: the F-principle



(a) Target function (b) Relative error

Figure 1: 1d input. (a) $f(x)$. Inset : $|\hat{f}(k)|$. (b) $\Delta_F(k)$ of three important frequencies (indicated by black dots in the inset of (a)) against different training epochs. The parameters of the DNN is initialized by a Gaussian distribution with mean 0 and standard deviation 0.1. We use a tanh-DNN with widths 1-8000-1 with full batch training. The learning rate is 0.0002. The DNN is trained by Adam optimizer [20] with the MSE loss function.

$\Rightarrow$ PINNs are not effective in approximating highly oscillatory solutions

On the Spectral Bias of Neural Networks

Nasim Rahaman [*1 2] Aristide Baratin [*1] Devansh Arpit [1] Felix Draxler [2] Min Lin [1] Fred A. Hamprecht [2] Yoshua Bengio [1] Aaron Courville [1]

WHEN AND WHY PINNS FAIL TO TRAIN: A NEURAL TANGENT KERNEL PERSPECTIVE

A PREPRINT

Sifan Wang
Graduate Group in Applied Mathematics and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
sifanw@sas.upenn.edu

Xinling Yu
Graduate Group in Applied Mathematics and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
xlyu@sas.upenn.edu

Paris Perdikaris
Department of Mechanichal Engineering and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
pgp@seas.upenn.edu

Can we exploit MG for the training of PINNs?

How to transpose the ingredients of success of MG Basic idea of MG: Exploiting "complementarity" between problems involved

Classical MG vs Neural networks

- Consider a minimization method and a class of problems for which this method is efficient

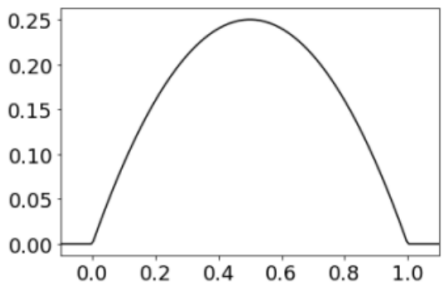
  smoothing (GS orJ) first-order (GD, SGD)

  high-frequency low-frequency
- Split the problem depending of its frequency content
- Shift the frequencies

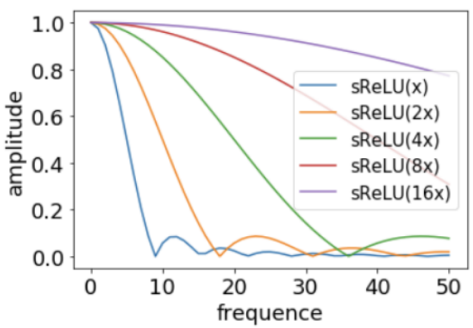
  Coarser discretizations Specialized architectures (Mscale networks)

Specialized architectures

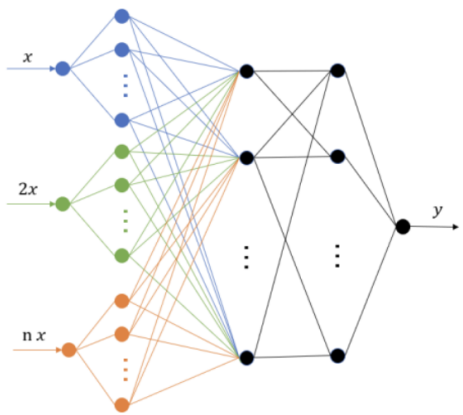
- **Mscale networks**: [Liu, Cai and Xu, (2020)] frequency-selective subnetworks + wavelet-inspired and frequency-located activation functions



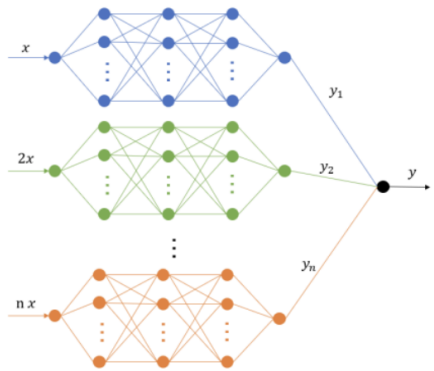
(b) sReLU



(a) sReLU

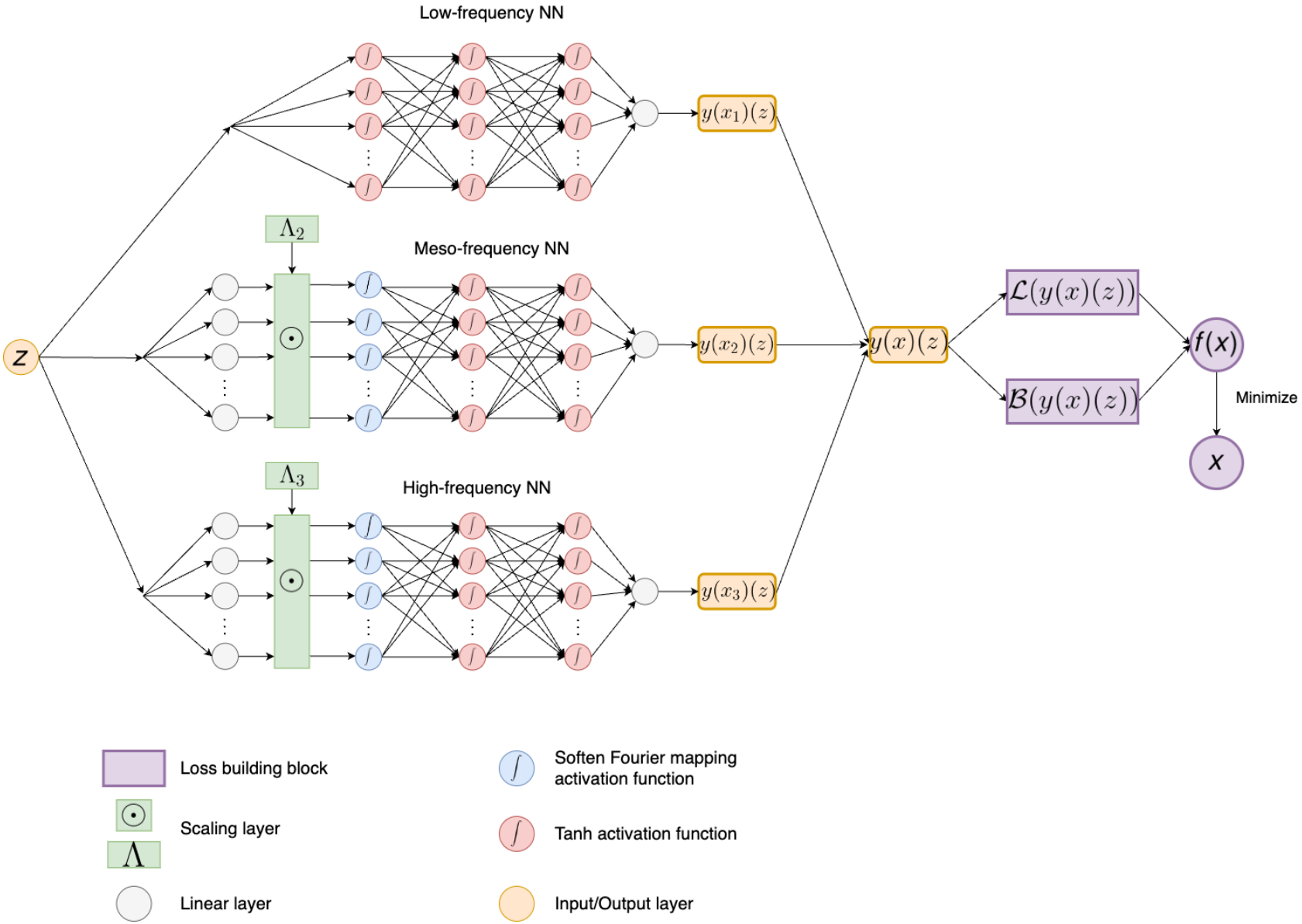


(a) MscaleDNN-1



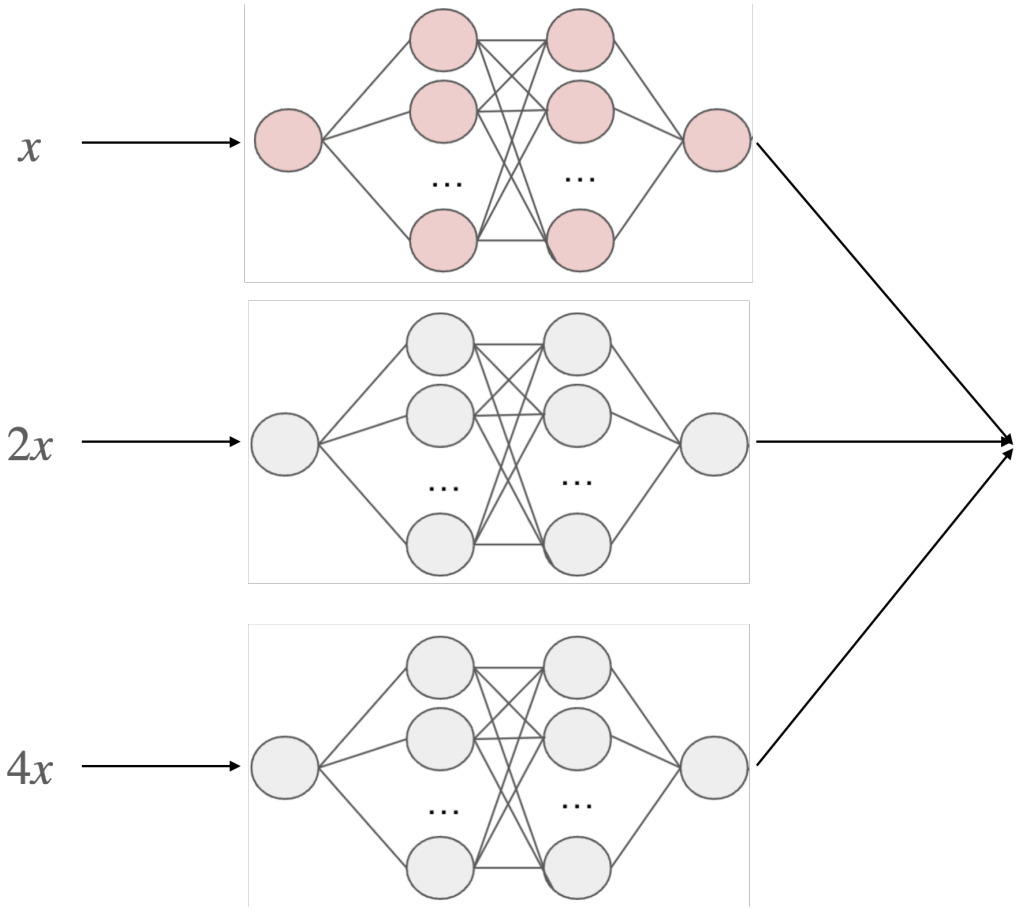
(b) MscaleDNN-2

Our architecture



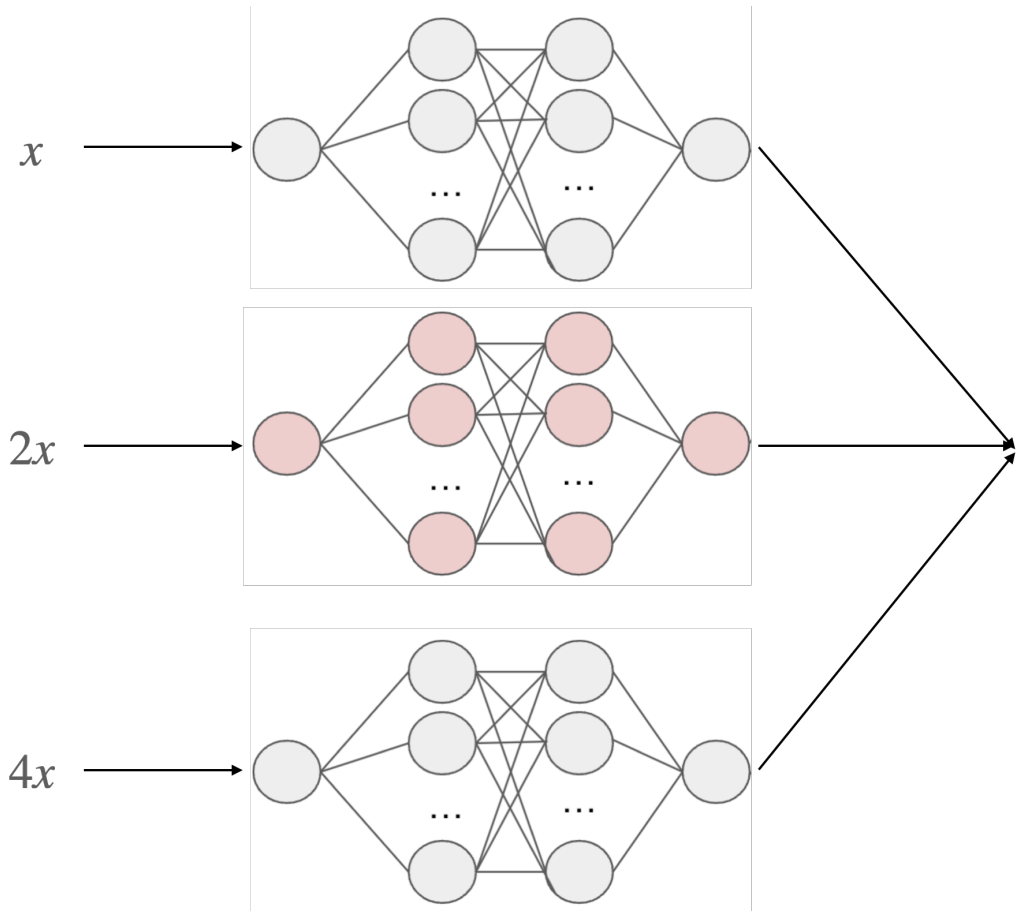
Multilevel PINNs: the training

From simultaneous training to BCD training!



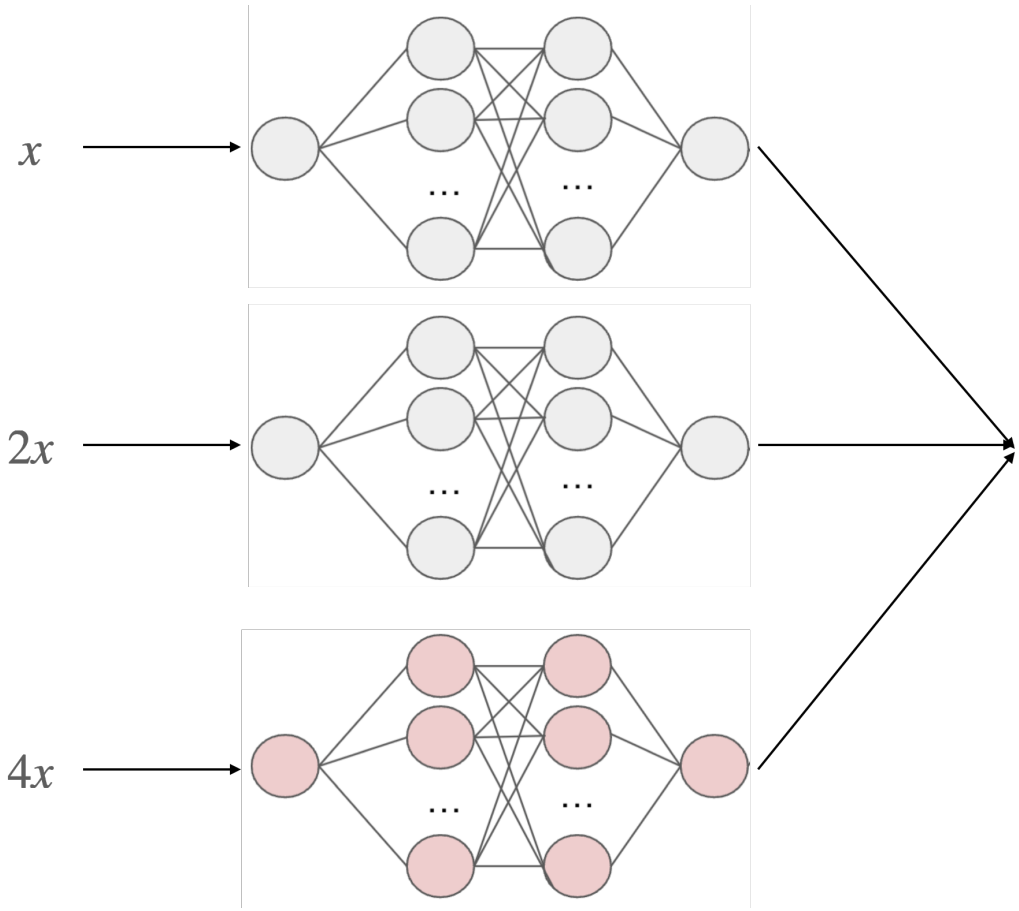
Multilevel PINNs: the training

From simultaneous training to BCD training!



Multilevel PINNs: the training

From simultaneous training to BCD training!



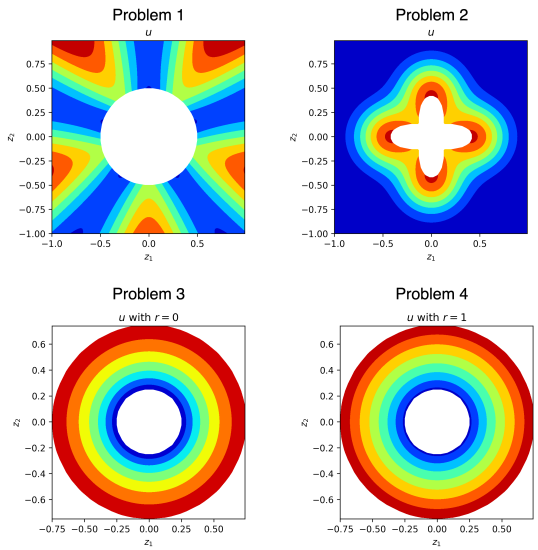
How to select the blocks?

Criterion:

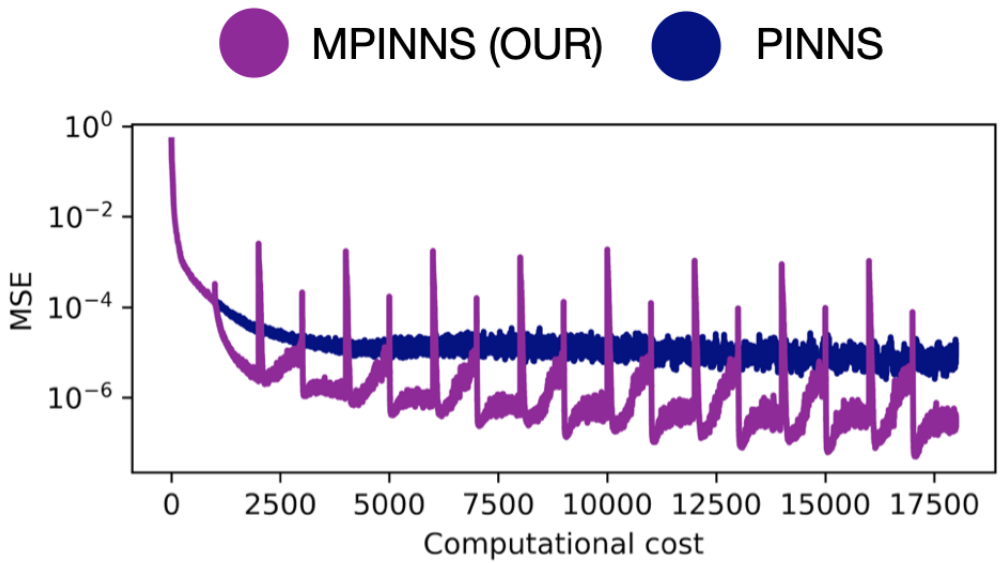
$\|\nabla_i f(x)\| \geq \tau \|\nabla f(x)\|$, $\tau \in (0, 1)$

Numerical results: MSE vs iterations

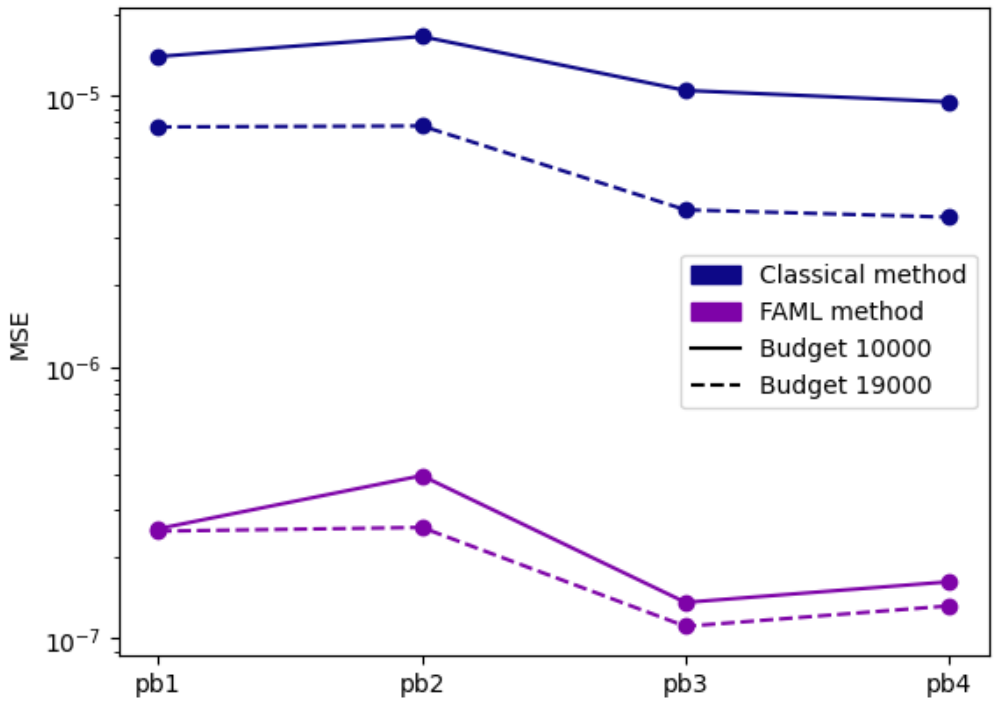
Problem: Navier Stockes equation on Ω



Ω



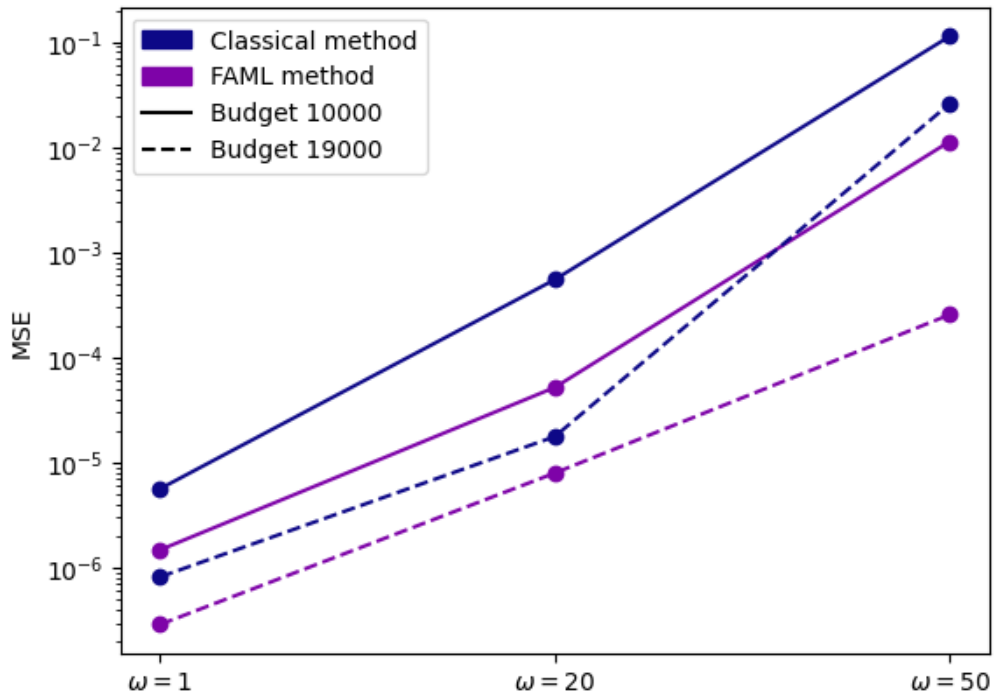
Numerical results: final MSE on average (10 runs)



Numerical results: MSE vs iterations

Problem: Heat equation

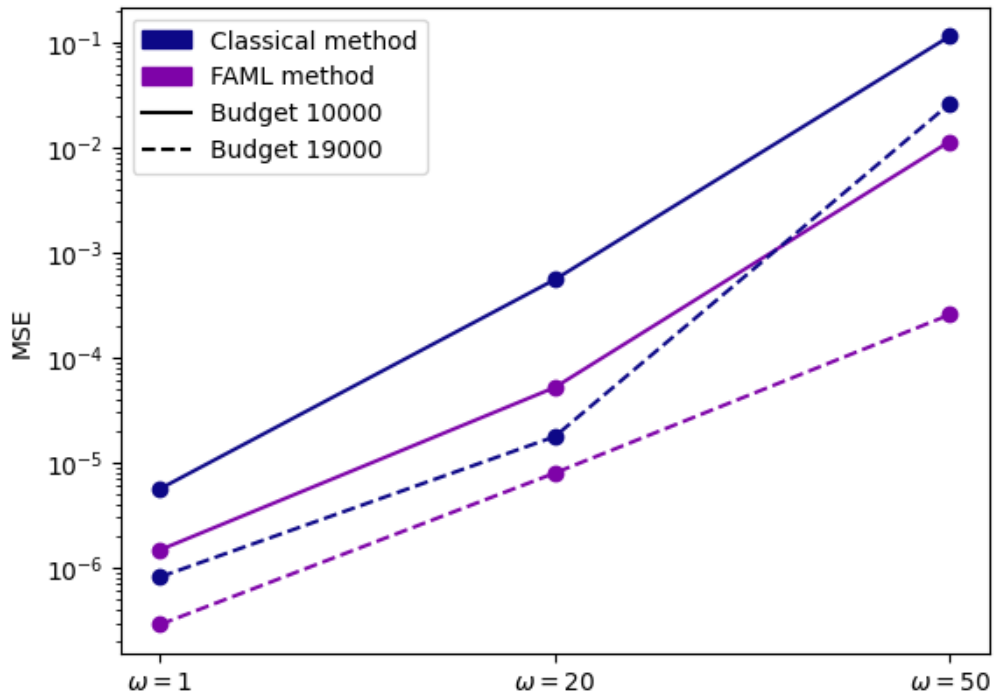
$$\begin{cases} \dfrac{\partial u(z,t)}{\partial t} = \dfrac{1}{(\omega\pi)^2} \dfrac{\partial^2 u(z,t)}{\partial z^2}, \\ u(z,0) = \sin(\omega\pi z), \ z \in [0,1], \\ u(0,t) = u(1,t) = 0, \ t \in [0,1], \end{cases}$$



Numerical results: MSE vs iterations

Problem: Heat equation

$$\begin{cases} \dfrac{\partial u(z,t)}{\partial t} = \dfrac{1}{(\omega\pi)^2}\dfrac{\partial^2 u(z,t)}{\partial z^2}, \\ u(z,0) = \sin(\omega\pi z), \ z \in [0,1], \\ u(0,t) = u(1,t) = 0, \ t \in [0,1], \end{cases}$$



Good numerical results, but ... does the method converge?

ML methods: abstraction from the PDE context

Problem: $\mathcal{F}$ space of continuous functions parametrized by x

$$\min_{y \in \mathcal{F}} f(y)$$

Approach: we look for y as the sum of two terms

$$y(x) = y_1(x_1) + y_2(x_2).$$

This yields the optimization problem

$$\min_{(x_1,x_2) \in \mathbb{R}^n} f(y_1(x_1) + y_2(x_2)),$$

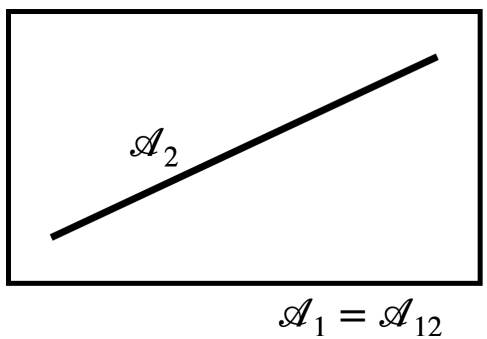
where $n = n_1 + n_2$.

ML methods: approximation spaces

$$\mathcal{A}_{12} = \{y \in \mathcal{F} \mid y(x) = y_1(x_1) + y_2(x_2) \text{ for some } (x_1, x_2) \in \mathbb{R}^n\}$$
$$\mathcal{A}_i = \{y \in \mathcal{F} \mid y(x) = y_i(x_i) \text{ for some } x_i \in \mathbb{R}^{n_i}\} \quad (i = 1, 2).$$

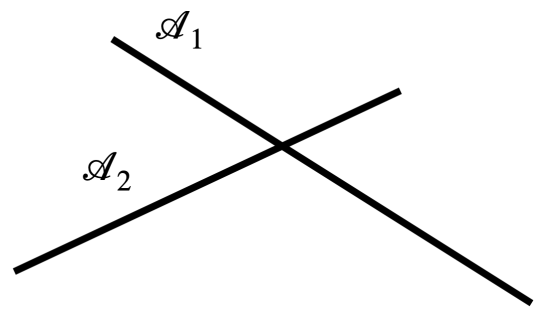
Hierarchical context

$$\mathcal{A}_2 \subset \mathcal{A}_1 = \mathcal{A}_{12}$$



Distributed context

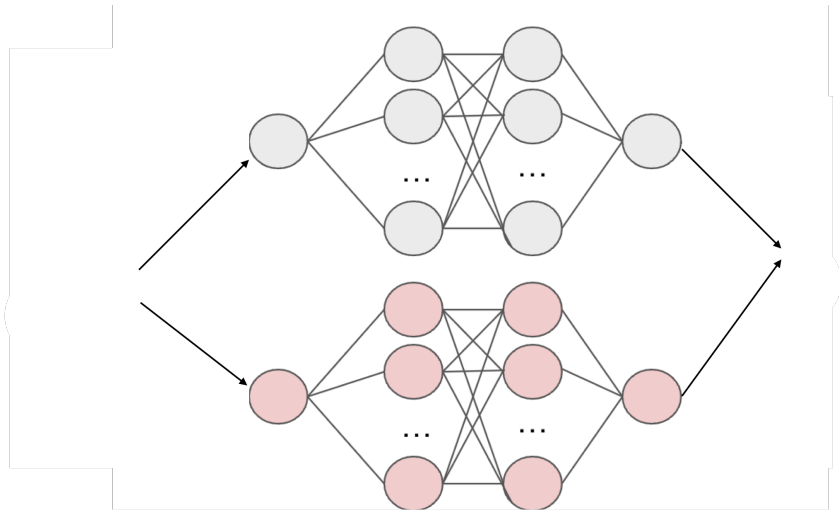
$$\mathcal{A}_1, \mathcal{A}_2 \subset \mathcal{A}_{12}$$



The distributed context

Example: neural networks

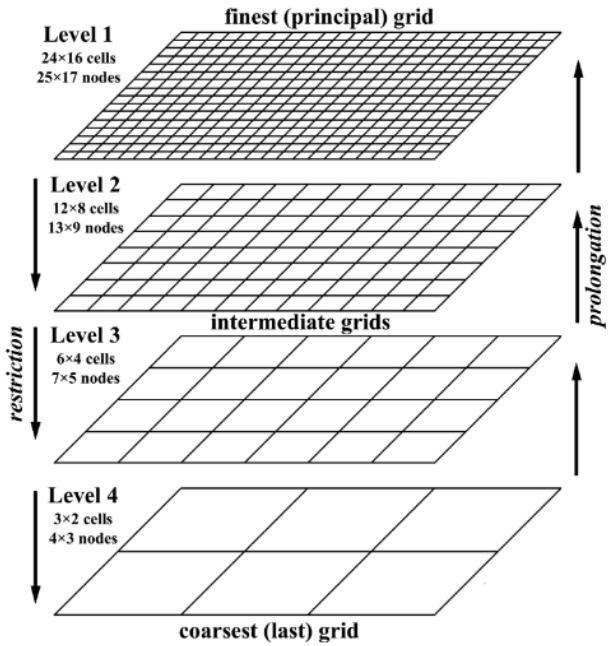
$$f(x) = \text{loss}$$
$$y_1(x_1) = NN_1(x_1)$$
$$y_2(x_2) = NN_2(x_2)$$



The hierarchical context

Example: classical MG

$$f(x) = \frac{1}{2}x^T Ax + x^T b$$



A block coordinate descent (BCD) perspective on ML

The hierarchical context

Alternate:

$$\min_{(x_1, x_2) \in \mathbb{R}^{n_1+n_2}} f(y_1(x_1)+y_2(x_2))$$

and

$$\min_{\substack{x_2 \in \mathbb{R}^{n_2} \\ x_1 \text{ fixed}}} f(y_1(x_1) + y_2(x_2))$$

The distributed context

Alternate:

$$\min_{\substack{x_2 \in \mathbb{R}^{n_1} \\ x_1 \text{ fixed}}} f(y_1(x_1) + y_2(x_2))$$

and

$$\min_{\substack{x_1 \in \mathbb{R}^{n_1} \\ x_2 \text{ fixed}}} f(y_1(x_1) + y_2(x_2)).$$

A BCD-ML algorithm: an iteration

How to update x?

1 Partition x in blocks: $(x_1, \ldots, x_n)$

1) $x =$ 

A BCD-ML algorithm: an iteration

How to update x?

2 Select a block i $(x_1, \ldots, x_i, \ldots, x_n)$

- Criterion: $\|\nabla_i f(x)\| \geq \tau \|\nabla f(x)\|$, $\tau \in (0, 1)$

2)

A BCD-ML algorithm: an iteration

How to update x?

3 Update the block:

- p_k iterations of a first-order method (possibly *stochastic*)

$$\min_{x_i} f(x_1, \ldots, x_i, \ldots, x_n) \rightarrow x_i^{new}$$

- $x_i \leftarrow x_i^{new}$

3)

$$\min_{x_3} f(x_1, x_2, x_3, x_4)$$

$$x_3 \leftarrow x_3^{new}$$

BCD theory for nonconvex problems

- Powell (1973): cyclic BCD may fail on nonconvex continuously differentiable functions.
- Bertsekas (1999): convergence of cyclic BCD if minimizer along any coordinate direction from any point is unique
- Attouch et all. (2010) + Bolte et all (2014), proximal alternating methods under Kurdyka-Lojasiewicz (KL) property convergence of sequence to stationary points
- Amaral et all. (2022) high (p)-order BCD smooth nonconvex for Lipschitz continuous $\nabla f(x_k)$ + regularized models $\rightarrow O(\epsilon^{-(p+1)})$

A BCD-ML algorithm: convergence theory

Theorem (Gratton, Mercier, R., Toint, 2023)

If f has L-Lipschitz continuos gradient and step-size $\alpha_k = \alpha < 1/L$

- **Deterministic**

$$\|\nabla f(x^{(K)})\| \leq \epsilon \rightarrow K = O\left(\frac{1}{\epsilon^2 p}\right)$$

- **Stochastic**

$$\mathbb{E}\left(\sum_{k=1}^{K} \|\nabla f(x^{(k)})\|^2\right) \leq C_1(\sigma^2) + O\left(\frac{1}{K}\right) - C_2(\sigma^2)p$$

p coarse iterations, σ^2 variance of stochastic gradient

A few references

- S. Gratton, A. Sartenaer, Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization, SIAM J. Opt., 19:414–444, 2008
- W. Briggs, V. Henson, S. McCormick. A Multigrid Tutorial, SIAM, 2000
- H. Calandra, S. Gratton, E. Riccietti, X. Vasseur. On high-order multilevel optimization strategies. SIAM J. Opt., 31.1: 307-330, 2021.
- S. Wang, H. Wang, P. Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. CMAME, 384, 2021
- Z. Liu, W. Cai, Z. Xu. Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains. arXiv:2007.11207, 2020
- E. Riccietti, V. Mercier, S. Gratton, P. Boudier, Multilevel physics informed neural networks (MPINNs), technical report, 2021
- V. Mercier, S. Gratton, P. Boudier. A coarse space acceleration of deep-DDM, arxiv:2112.03732, 2022
- H. Calandra, S. Gratton, E. Riccietti, X. Vasseur. On a multilevel Levenberg-Marquardt method for the training of artificial neural networks and its application to the solution of partial differential equations, OMS, 2020