Multilevel methods in optimization

Elisa Riccietti

LIP-ENS Lyon

Joint work with:

- H. Calandra (TOTAL), S. Gratton - V. Mercier (IRIT), P. Toint (Univ. Namur), X. Vasseur (ISAE-SUPAERO)
- G. Lauga - N. Pustelnik - P. Gonçalves (ENS Lyon, INRIA, CNRS)

UNIVERSITÉ DE LYON ENS DE LYON

Context: large scale optimization problems

$$\min_{x} f(x) \quad \rightarrow \quad \min_{x \in \mathbb{R}^n} f(x) = \sum_{i=1}^{m} f_i(x)$$

Large scale problems

- f has a large number of unknowns: large n (ex: deep learning)
- f is the sum of a large number of terms: large m (ex: classification of large datasets)

Context: large scale optimization problems

$$\min_{x} f(x) \quad \rightarrow \quad \min_{x \in \mathbb{R}^n} f(x) = \sum_{i=1}^{m} f_i(x)$$

Large scale problems

- f has a large number of unknowns: large n (ex: deep learning) $\rightarrow$ Multilevel methods
- f is the sum of a large number of terms: large m (ex: classification of large datasets)

Outline

- **Part I**: Brief recap on multigrid methods for the solution of PDEs
- **Part II**: Their transposition to a nonlinear context: multilevel optimization methods
- **Part III**: Multilevel proximal methods for image restoration
- **Part IV**: Artificial neural networks for the solution of PDEs and multilevel methods for their training

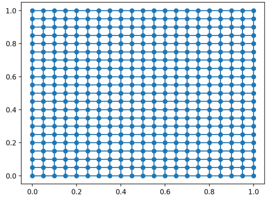
ENS DE LYON

Part I

Multigrid (MG) methods

The numerical solution of PDEs

- 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



UNIVERSITÉ DE LYON

ENS DE LYON

The intuition behind multigrid methods

- Example: $\Delta u = 0$, $v_k(j) = \sin(\frac{kj\pi}{n})$, k-th Fourier mode
- The smoothing property: hard for fixed point iterative methods to reduce the low frequency components of the error

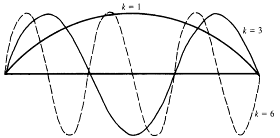
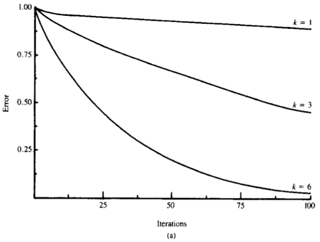


Figure 2.2: *The modes* $v_j = \sin\left(\frac{jk\pi}{n}\right)$, $0 \le j \le n$, *with wavenumbers* $k = 1, 3, 6$. *The* k*th mode consists of* $\frac{k}{2}$ *full sine waves on the interval.*



ENS DE LYON

The intuition behind multigrid methods

- How does a smooth component look like on a coarser grid?

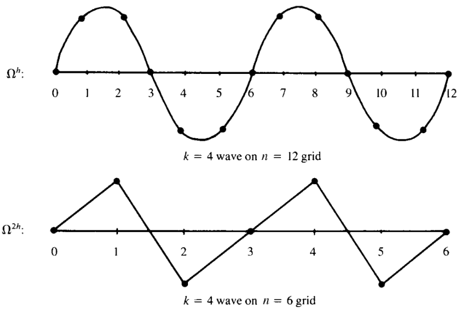
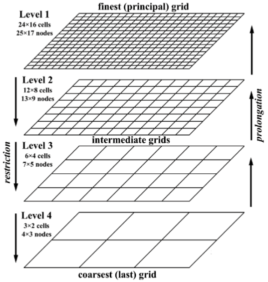


Figure 3.1: *Wave with wavenumber $k = 4$ on Ω^h ($n = 12$ points) projected onto Ω^{2h} ($n = 6$ points). The coarse grid "sees" a wave that is more oscillatory on the coarse grid than on the fine grid.*

Multigrid methods for PDEs

State-of-the-art methods for PDEs: exploit representation of the problem at different scales



- Fine scales: eliminate high frequency components of the error
- Coarse scales: eliminate low frequency components of the error

Two-level multigrid methods

Consider a linear PDE:

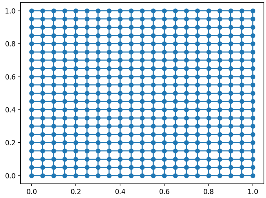
$$Au = f.$$

Consider two discretizations of the same system:

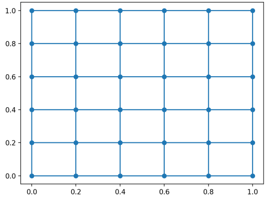
- Fine grid: $A_h u_h = f_h$
- Coarse grid: $A_H u_H = f_H$

Idea: write the solution u as the sum of a fine and a coarse term:

$$u \sim \underbrace{v_h}_{\in \mathbb{R}^h} + P(\underbrace{e_H}_{\in \mathbb{R}^H}), \; H < h.$$



$R \Downarrow \qquad P \Uparrow$



Two-level multigrid methods

Update the two components in an alternate fashion:

$$u \sim v + e$$

- *Fine level*: get v_h by iterating on $A_h u = f_h$
-
-
-

Two-level multigrid methods

Update the two components in an alternate fashion:

$$u \sim v + e$$
$$r = f - Av$$

- *Fine level*: get v_h by iterating on $A_h u = f_h$
- Compute $r_h = f - Av_h$ and project $r_H = Rr_h$
-
-

UNIVERSITÉ DE LYON
ENS DE LYON

Two-level multigrid methods

Update the two components in an alternate fashion:

$$u \sim v + e$$

$$Ae = r \text{ residual equation}$$

- *Fine level*: get v_h by iterating on $A_h u = f_h$
- Compute $r_h = f - Av_h$ and project $r_H = Rr_h$
- *Coarse level*: compute correction: $A_H e_H = r_H$
-

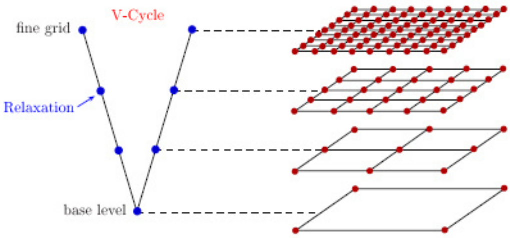
Two-level multigrid methods

Update the two components in an alternate fashion:

$$u \sim v + e$$

- *Fine level*: get v_h by iterating on $A_h u = f_h$
- Compute $r_h = f - Av_h$ and project $r_H = Rr_h$
- *Coarse level*: compute correction: $A_H e_H = r_H$
- Correct: $v_h \leftarrow v_h + P(e_H)$

General multigrid methods



W. Briggs, V. Henson, S. McCormick. A Multigrid Tutorial, SIAM, 2000.

ENS DE LYON

Part II

Multilevel optimization methods

The optimization methods

We consider large-scale nonlinear unconstrained optimization problems:

$$\min_x f(x)$$

Classical iterative optimization methods:

$$f(x_k + s) \simeq T_k(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T B_k s$$

with $T_k(x_k, s)$ Taylor model, $B_k \sim \nabla^2 f(x_k)$ or $B_k = 0$.

The optimization methods

We consider large-scale nonlinear unconstrained optimization problems:

$$\min_x f(x)$$

Classical iterative optimization methods:

$$f(x_k + s) \simeq T_k(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T B_k s$$

with $T_k(x_k, s)$ Taylor model, $B_k \sim \nabla^2 f(x_k)$ or $B_k = 0$.

At each iteration we compute a step s_k to update the iterate:

$$\min_s m_k(x_k, s) = T_k(x_k, s) + r(\lambda_k), \qquad \lambda_k > 0$$

$r(\lambda_k)$ regularization term. Set $x_{k+1} = x_k + s_k$

Classical examples

- Gradient method:

$$m_k(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{\lambda_k}{2}\|s\|^2$$

- Trust region (TR) method:

$$m_k(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{\lambda_k}{2}\|s\|^2$$

- Adaptive Cubic Regularization (ARC):

$$m_k(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s + \frac{\lambda_k}{3}\|s\|^3$$

Cubic regularization of Newton method and its global performance, Y. Nesterov and B. Polyak, 2006

Bottleneck: Subproblem solution

Solving

$$\min_s T_k(x_k, s) + r(\lambda_k)$$

represents greatest cost per iteration, which depends on the size of the problem.

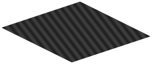
Possible solution: multilevel methods

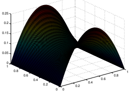
- *A multigrid approach to discretized optimization problems*, Nash, 2000
- *Recursive trust-region methods*, S. Gratton, A. Sartenaer and Ph. L. Toint, 2008
- *On high-order multilevel optimization strategies*, H. Calandra, S. Gratton, E. Riccietti, X. Vasseur, 2020

UNIVERSITÉ DE LYON

ENS DE LYON

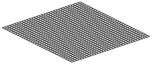
Hierarchy of problems $(n_r > n_{r-1} > \dots n_0)$

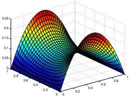




Finest Level $f_r : \mathbb{R}^{n_r} \to \mathbb{R}$

Restriction $\downarrow R_r$ $\quad$ $P_r \uparrow$ Prolongation



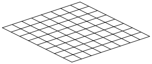


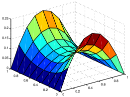
Fine Level $f_{r-1} : \mathbb{R}^{n_{r-1}} \to \mathbb{R}$

Restriction $\downarrow R_{r-1}$ $\quad$ $P_{r-1} \uparrow$ Prolongation

$\vdots$

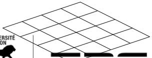
Restriction $\downarrow R_2$ $\quad$ $P_2 \uparrow$ Prolongation

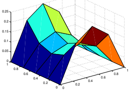




Coarse Level $f_2 : \mathbb{R}^{n_2} \to \mathbb{R}$

Restriction $\downarrow R_1$ $\quad$ $P_1 \uparrow$ Prolongation





Coarsest Level $f_0 : \mathbb{R}^{n_0} \to \mathbb{R}$

Multilevel strategy

Hierarchy of problems

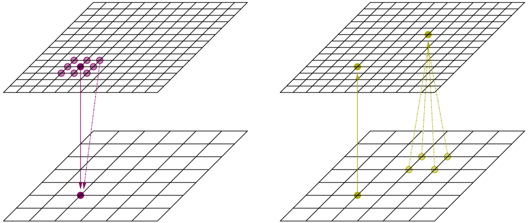
- $\{f^\ell(x^\ell)\}$, $x^\ell \in \mathbb{R}^{n_\ell}$
- $n_{\ell-1} < n_\ell \rightarrow f^{\ell-1}$ is cheaper to optimize compared with f^ℓ
- $\mu^{\ell-1}$ model for $f^{\ell-1}$

$$x_k^\ell \xrightarrow{R^\ell} x_{0,k}^{\ell-1} := R^\ell x_k^\ell \xrightarrow{\min_x \mu^{\ell-1}(x)} x_{*,k}^{\ell-1} \xrightarrow{s_k^\ell = P^\ell(x_{*,k}^{\ell-1} - x_{0,k}^{\ell-1})} x_{k+1}^\ell = x_k^\ell + s_k^\ell$$

What do we need to use such a method ?

Transfer operators

- $R = I_h^H \in \mathbb{R}^{N_H \times N_h}$: from fine to coarse ($N_H < N_h$).
- $P = I_H^h \in \mathbb{R}^{N_h \times N_H}$: from coarse to fine.
- Relation between the operators : $I_H^h = (I_h^H)^T$.



I_h^H $\qquad$ I_H^h

What do we need to use such a method ?

Lower level model

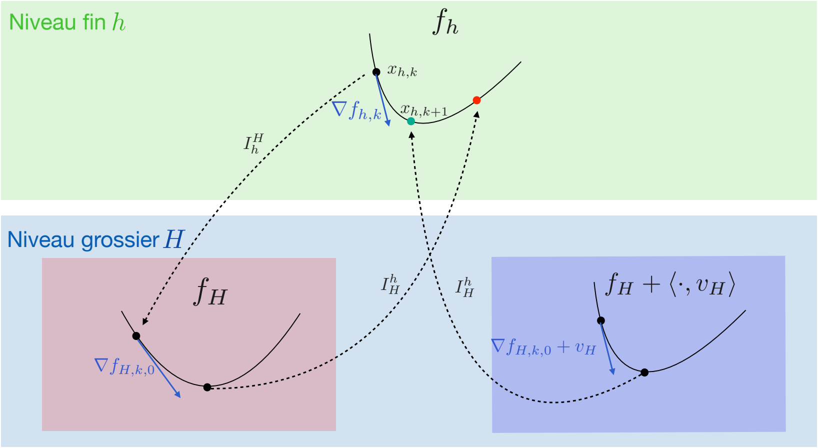
When to use the lower level model?

- Choose lower level model $\mu^{\ell-1}$ if
  - if $\|\nabla\mu_k^{\ell-1}(x_{0,k}^{\ell-1})\| = \|R^\ell \nabla f^\ell(x_k^\ell)\| \geq \kappa \|\nabla f^\ell(x_k^\ell)\|$, $\kappa > 0$
  - if $\|\nabla\mu_k^{\ell-1}(x_{0,k}^{\ell-1})\| > \epsilon^\ell$
- Minimize regularized Taylor model otherwise.

How to define the lower level model?

Modify $f^{\ell-1}$ to ensure coherence among levels

First order coherence



Coherence between levels: first order

Let $x_{0,k}^{\ell-1} = Rx_k^\ell$. Model with first order correction:

$$\mu_{1,k}^{\ell-1}(x_{0,k}^{\ell-1}, s^{\ell-1}) = f^{\ell-1}(x_{0,k}^{\ell-1} + s^{\ell-1}) + < v^{\ell-1}, s^{\ell-1} >$$
$$v^{\ell-1} = R^\ell \nabla f^\ell(x_k^l) - \nabla f^{\ell-1}(x_k^{\ell-1})$$

This ensures that

$$\nabla \mu_{1,k}^{\ell-1}(x_{0,k}^{\ell-1}) = R^\ell \nabla f^\ell(x_k^\ell)$$

$\rightarrow$ first-order behaviours of f^ℓ and $\mu^{\ell-1}$ are coherent around x_k^ℓ. If $s^\ell = P^\ell s^{\ell-1}$

$$0 > < \nabla \mu_{1,k}^{\ell-1}(x_{0,k}^{\ell-1}), s^{\ell-1} > = < \nabla f^\ell(x_k^\ell), P^\ell s^{\ell-1} > = < \nabla f^\ell(x_k^\ell), s^\ell >$$

Numerical results for 4 levels ARC

$$\begin{cases} -\Delta u(z) + e^{u(z)} = g(z) & \text{in } \Omega \subset \mathbb{R}^2, \\ u(z) = 0 & \text{on } \partial\Omega, \end{cases}$$

The following nonlinear minimization problem is then solved:

$$\min_{u \in \mathbb{R}^{n^d}} \frac{1}{2} u^T A u + \|e^{u/2}\|^2 - g^T u,$$

which is equivalent to the system $Au + e^u = g$.

- Coarse approximations: coarser discretization of the problem

		$n = 1024$		$n = 4096$	
		ARC	MARC 4	ARC	MARC 4
$\bar{u}_1$	it_T/it_f	11/11	7/2	23/23	15/4
	save		**2.2**		**4.1**
$\bar{u}_2$	it_T/it_f	27/27	13/4	56/56	22/6
	save		**3.9**		**6.1**

Part III

Multilevel proximal methods

What if the objective function is non-smooth?

$$\min_{x \in \mathbb{R}^N} F(x) := f(x) + g(x)$$

with f differentiable but g may not be differentiable

Example: image restoration

Find an image $\hat{x}$ close to $\bar{x}$ from a degraded image z.

$\bar{x}$

z

$\hat{x}$

Problem formulation

Standard degradation model : $z = \mathrm{A}\bar{x} + \epsilon$

- $\mathrm{A} \in \mathbb{R}^N \times \mathbb{R}^N$ models the linear degradation
- $\epsilon \in \mathbb{R}^N$ is a random noise

•Ill-posed problem :

$$\widehat{x} \in \underset{x \in \mathbb{R}^N}{\mathrm{Argmin}} \frac{1}{2} \|\mathrm{A}x - z\|_2^2 + \lambda \|\mathrm{W}x\|_1$$

- N size of the image
- $\mathrm{W} \in \mathbb{R}^{N \times N}$ linear transform enforcing sparsity in x.
- λ regularization parameter

Classical solution methods

$$x_{k+1} = x_k - D_k$$

•If f and g are differentiable : gradient method

$$D_k = \tau_k(\nabla f(x_k) + \nabla g(x_k))$$

Classical solution methods

$$x_{k+1} = x_k - D_k$$

•If f and g are differentiable : gradient method

$$D_k = \tau_k(\nabla f(x_k) + \nabla g(x_k))$$

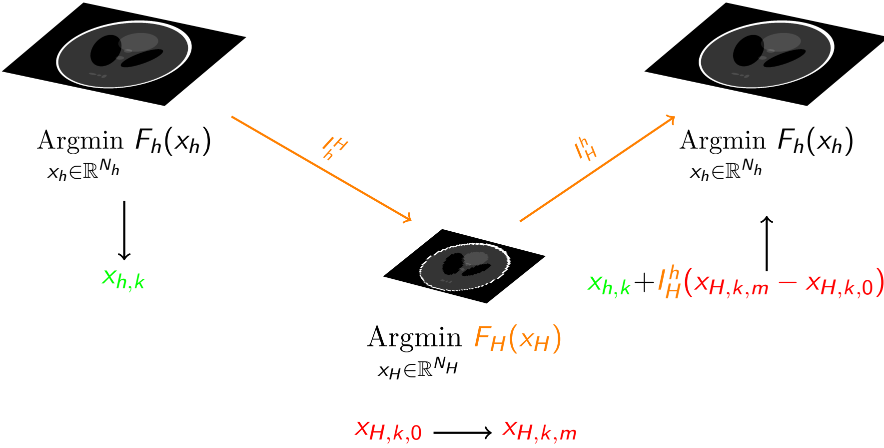
•If g is not differentiable : proximal gradient method

$$D_k = x_k - \text{prox}_{\tau_k g}\left(x_k - \tau_k \nabla f(x_k)\right)$$

Remark : $\text{prox}_{\tau g}(\cdot) = \underset{x \in \mathbb{R}^N}{\text{argmin}} \frac{1}{2}\|x - \cdot\|_2^2 + \tau g(x)$ has an explicit form for many choices of g (cf. *Prox Repository*).

ENS DE LYON

Multilevel method for non-smooth functions



What do we need to use such a method ?

- Function F_H at coarse level :

$$F_H = f_H + g_H$$

In image restoration :

$$\forall x \in \mathbb{R}^{N_h} \quad f_h(x) = \frac{1}{2}\|\mathrm{A}x - z\|_2^2 \qquad g_h(x) = \lambda\|\mathrm{W}x\|_1$$

$$\forall x \in \mathbb{R}^{N_H} \quad f_H(x) = \frac{1}{2}\|I_h^H \mathrm{A} I_H^h x - I_h^H z\|_2^2 \quad g_H(x) = \lambda\|\mathrm{W}x\|_1.$$

where :

- $I_h^H \mathrm{A} I_H^h$: Galerkin reduction of A.
- W: wavelets decomposition

UNIVERSITÉ DE LYON ENS DE LYON

What do we need to use such a method ?

- Lower level model
  - non-smooth approximation $f_H(x_H) + g_H(x_H) \rightarrow$ proximal method
  - smooth approximation $f_H(x_H) + {}^{\gamma}g_H(x_H) \rightarrow$ gradient method

ENS DE LYON

What do we need to use such a method ?

- Lower level model
  - non-smooth approximation $f_H(x_H) + g_H(x_H) \to$ proximal method
  - smooth approximation $f_H(x_H) + {}^{\gamma}g_H(x_H) \to$ gradient method

In both cases we need the correction term v_H.
Idea: use a smoothing and do as in the smooth case :

$$v_H = I_h^H D_{h,k} - \nabla f_H(x_{H,k,0}) - \nabla g_{h,\gamma}(x_{H,k,0})$$

where

$$D_{h,k} = -\nabla F_h(x_{h,k}) \text{ in the smooth case}$$
$$D_{h,k} = -\nabla f_h(x_{h,k}) - \nabla^{\gamma} g_h(x_{h,k}) + \gamma \nabla^{\gamma} g_h(\nabla f(x_{h,k})) \text{ otherwise}$$

It holds: $F_h(x_{h,k} + I_H^h(x_{H,k,m} - x_{H,k,0})) \leq F_h(x_{h,k}) + \beta\gamma$

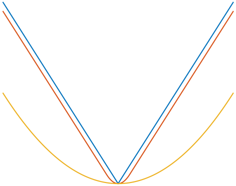
Multilevel method for non-smooth functions

Smooth approximation of F_H :

- $F_H \Rightarrow F_{H,\gamma} = f_H + {}^{\gamma}g_H$
- Use the Moreau envelope : ${}^{\gamma}g_H = \inf_{y \in \mathcal{H}} g_H(y) + \frac{1}{2\gamma}\| \cdot - y\|^2$

Properties of the Moreau envelope :

- $\nabla {}^{\gamma}g_H = \gamma^{-1}(\mathrm{Id} - \mathrm{prox}_{\gamma g_H})$
- $\nabla {}^{\gamma}g_H$ γ^{-1} - Lipschitz



Moreau envelope of l_1 for $\gamma = 0.1$ and $\gamma = 1$

Multilevel algorithm: k-th iteration

if *descent condition* **then**

Coarse level:

Initialisation : $x_{H,k,0} = I_h^H x_{h,k}$

First order coherence $v_{H,k}$

Minimization of F_H : $x_{H,k,m} = \underbrace{\Phi_H \circ \ldots \circ \Phi_H(x_{H,k,0})}_{m \text{ times}}$

Fine level correction : $I_H^h(x_{H,k,m} - x_{H,k,0})$

else

Fine model :

Iterations on F_h : $x_{h,k+1} = \Phi_h(x_{h,k})$.

end

Example for a 2048×2048 image

$\bar{x}$

z

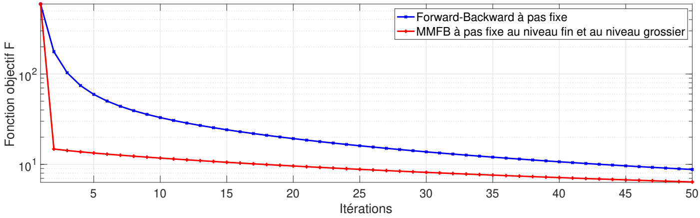
$\widehat{x}$

Image : Lunch atop a Skyscraper. $\widehat{x}$ after 50 iterations of MMFB.

UNIVERSITÉ DE LYON

ENS DE LYON

Evolution of F_h



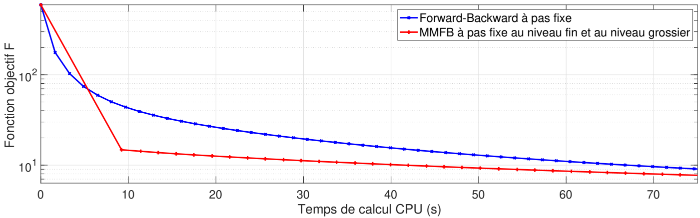
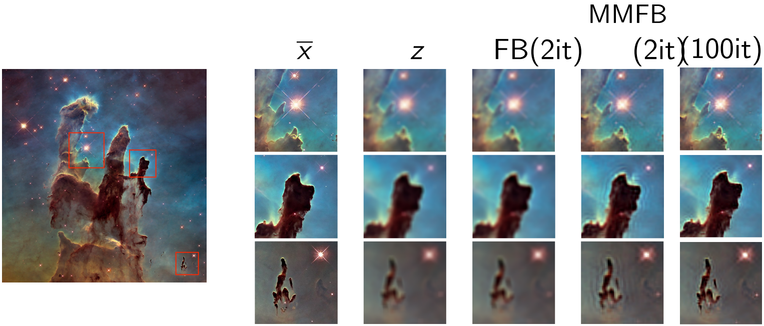
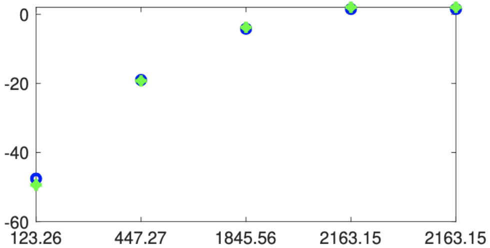


Image reconstruction



From left to right : original image, zoom on : original image, degraded image, restoration by FB after 2 iterations, restoration by MMFB after 2 and 100 iterations. Pillars of Creation. Credit : NASA, ESA/Hubble.

A quick approximation



Blue: smooth F_H. Green: non-smooth F_H

- y axis: gain of MMFB over FB in CPU time
- x axis: time needed for FB to reach a percentage threshold (5, 2, 1, 0.1and 0.01 %) of $\|F(x_0) - F(x^*)\|$

Perspectives

- More systematic characterisation of the performance of the method
- Extension to others first order approaches
- More specialized transfer operators
- Refinement of convergence guarantees : over smoothable functions
  - Convergence of the iterates: done
  - Global convergence : $F_h(x_{h,k} + I_H^h(x_{H,k,m} - x_{H,k,0})) \leq F_h(x_{h,k}) + \beta\gamma$

Part IV

Artificial neural networks for PDEs and multilevel training

A new approach for PDEs

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

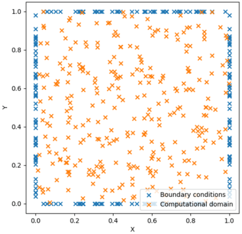
M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, 2019.

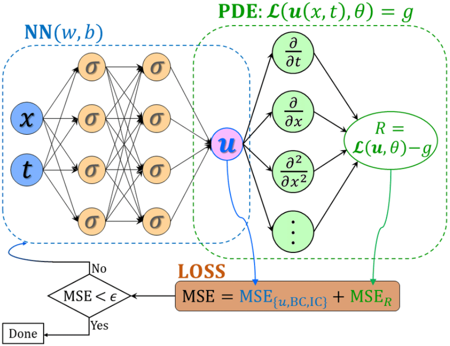
Why this approach ?

- 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

Physics Informed Neural Networks (PINNs)

$\mathcal{L}(u(x,t),\theta) = g(x,t)$
$+$ BC $+$ IC
in $\Omega \times [0, T]$



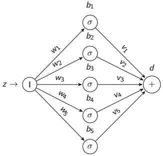


Given x_i, t_i sampled randomly, and $y_i = u^*(x_i, t_i)$,

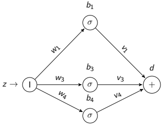
$$MSE_{\{u,BC,IC\}} = \frac{1}{N_m} \sum_{x_i \in \Omega \cup \partial\Omega,\, t_i \in [0,T]} (NN(x_i,t_i) - y_i)^2,$$

$$MSE_R = \frac{1}{N_r} \sum_{x_i \in \Omega,\, t_i \in [0,T]} R(x_i,t_i)^2$$

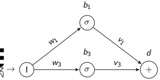
How to exploit multilevel method for training of ANNs?



$R_1 \Downarrow P_1 \Uparrow$



$R_2 \Downarrow P_2 \Uparrow$



Large-scale problem

- How to build the hierarchy of problems? The variables to be optimized are the network's weights: NO evident geometrical structure to exploit!

ENS DE LYON

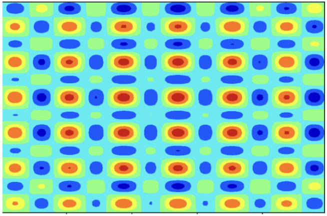
How do we select the hierarchy of variables?

- First attempt: exploit the algebraic structure: split the variables based on an Algebraic multigrid (AMG) C/F splitting
- Works, but it requires the calculation of the Hessian of the loss → too expensive for large problems

On a multilevel Levenberg-Marquardt method for the training of artificial neural networks and its application to the solution of partial differential equations, H. Calandra, S. Gratton, E. Riccietti, X. Vasseur, 2020

How to overcome this problem? → Second attempt: Go back to MG methods!

On the spectral bias of neural networks



On the Spectral Bias of Neural Networks

Nasim Rahaman [*12] 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

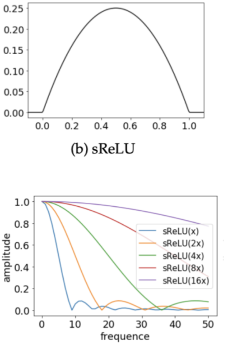
Paris Perdikaris
Department of Mechanichal Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
pgp@seas.upenn.edu

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

Mscale networks

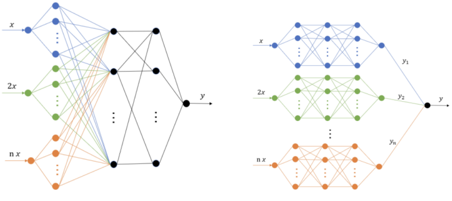
Z.Q. Liu, W. Cai, and Z.Q. John Xu, Multi-scale Deep Neural Network (MscaleDNN) forSolving Poisson-Boltzmann Equation in Complex Domains, 2020

Idea: simultaneous training of frequency-selective subnetworks



(b) sReLU

(a) sReLU

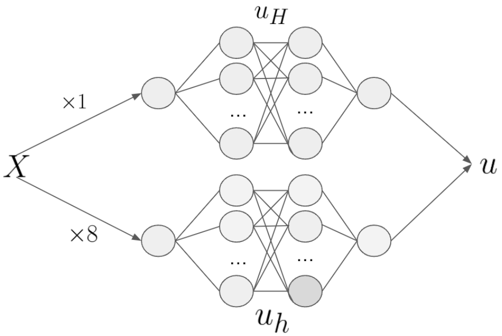


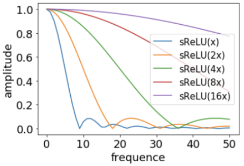
(a) MscaleDNN-1

(b) MscaleDNN-2

Multilevel PINNs: the architecture

Idea: $u_{sol}(x) \sim u_h(\theta_h, x) + u_H(\theta_H, x)$
Also exploit frequency-selective subnetworks





(a) sReLU

Multilevel PINNs: the loss

Problem definition

$$\begin{aligned} \mathcal{L}(u(x)) &= g(x),\ x \in \Omega, \\ u_{sol}(x) &\sim u_h(\theta_h, x) + u_H(\theta_H, x) \end{aligned}$$

Fine problem

$$MSE_h(\theta_h) = MSE_{R,h}(\theta_h) + MSE_{B,h}(\theta_h)$$

$$MSE_{R,h}(\theta_h) = ||\mathcal{L}(\hat{u}_h(\theta_h) + u_H) - g||^2$$

$$MSE_{B,h}(\theta_h) = ||\hat{u}_h(\theta_h) + u_H - u||^2$$

Computed on z_h the fine sampling

Coarse problem

$$MSE_H(\theta_H) = MSE_{R,H}(\theta_H) + MSE_{B,H}(\theta_H)$$

$$MSE_{R,H}(\theta_H) = ||\mathcal{L}(\hat{u}_H(\theta_H) + u_h) - g||^2$$

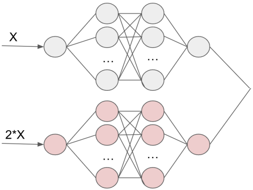
$$MSE_{B,H}(\theta_H) = ||\hat{u}_H(\theta_H) + u_h - u||^2$$

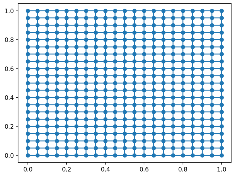
Computed on z_H the coarse sampling

Multilevel PINNs: the training

Algorithm 1 2-level training of PINNs

1: Freeze coarse-network parameters, unfreeze fine-network parameters
for *i=1,2,...* **do**
2:
end
Perform ν_1 epochs for the minimization of the fine problem
3: Freeze fine-network parameters, unfreeze coarse-network parameters
4: Perform ν_2 epochs for the minimization of the coarse problem
5:
6: Return : $u_H + u_h$



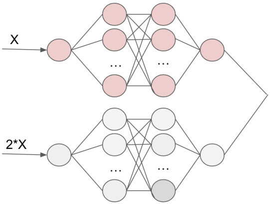


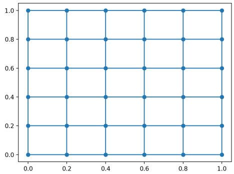
Multilevel PINNs: the training

Algorithm 2 2-level training of PINNs

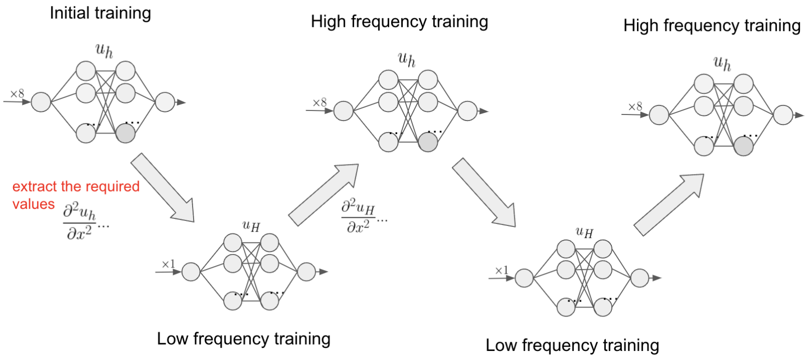
Freeze coarse-network parameters, unfreeze fine-network parameters
for *i=1,2,...* **do**
    Perform ν_1 epochs for the minimization of the fine problem
end
Freeze fine-network parameters, unfreeze coarse-network parameters
Perform ν_2 epochs for the minimization of the coarse problem

Return : $u_H + u_h$



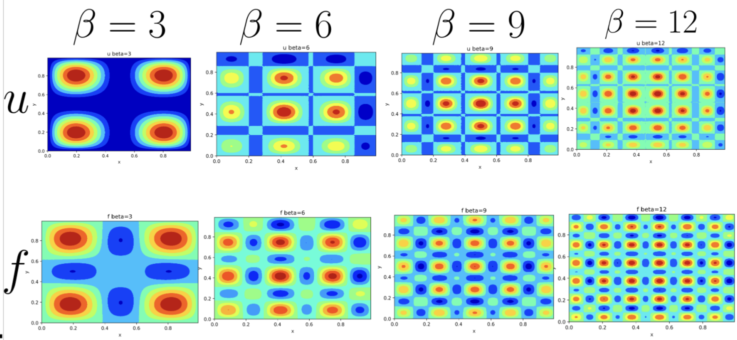


Multilevel PINNs: V-cycles



A simple Poisson problem

- $\Omega = [0,1] \times [0,1]$
- $\Delta u = f \ \forall x \in \Omega$
- $u = 0 \ \forall x \in \partial\Omega$
- $u(x,y) = (sin(\pi x) + sin(\beta \pi x)) * (sin(\pi y) + sin(\beta \pi y))$

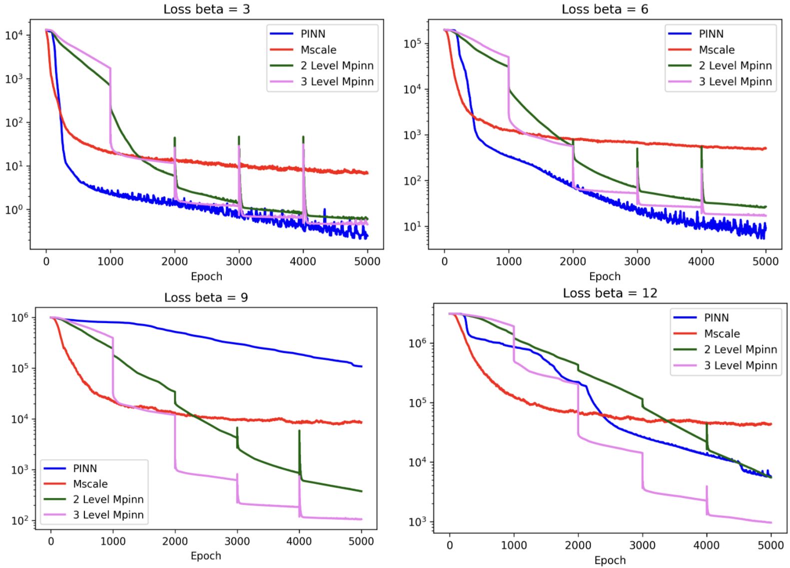


Experimental settings

In what follows:

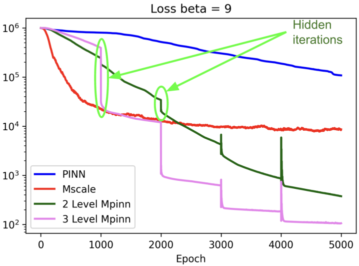
- The PINNs have two hidden layers of 300 neurons each.
- The Mscale have four subnetworks of two hidden layers of 150 neurons each, the input scaling used are 1,2,4 and 8.
- The two-level MPINN is composed of two networks of two hidden layers of 210 neurons each and trained in a V-cycle with 1 and 8 input scalings ($\nu_1 = \nu_2 = 1000$).
- The three-level MPINN is composed of three networks of two hidden layers of 150 neurons each and trained in a V-cycle with 1,4 and 8 input scalings ($\nu_1 = \nu_2 = \nu_3 = 1000$).
- The input of all networks is a regular grid sample of 80×80 points
- In all cases, we plot the median for ten random runs.

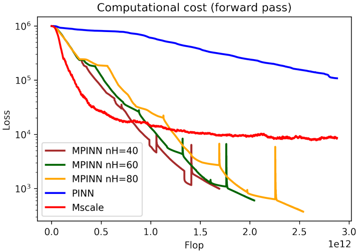
Varying β (the frequency content)



Hidden iterations - Computational cost

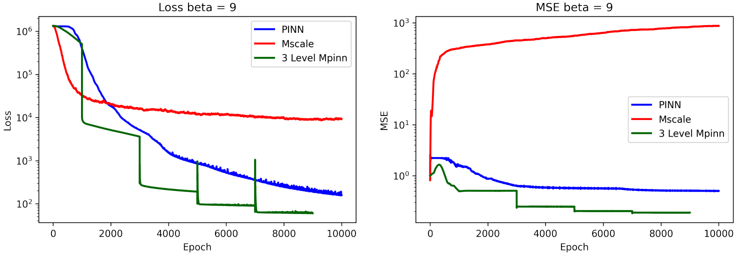
. . . as a function of coarse grid size (nH)





An elliptic nonlinear problem

- $-\Delta u + 100u^2 = f \ \forall x \in \Omega$
- $u(x, y) = (sin(\pi x) + sin(\beta\pi x)) * (sin(\pi y) + sin(\beta\pi y))$



Perspectives

For the multigrid training method:

- Perform further extensive testing, including more complex problems
- Pursue the sensitivity analysis for the relative sizes of the grids
- Investigate theoretical aspects:
  - convergence of the iterates from an optimization point of view
  - convergence to the solution in functional space

Conclusion

- Multilevel methods: a versatile framework
- Can be used in many different contexts to improve speed of convergence and computational time
- A new perspective: derivative-free multilevel methods (ongoing work)

Thank you for your attention!

Slides and papers available here

`bit.ly/elisaIRIT`

- *On high-order multilevel optimization strategies*, H. Calandra, S. Gratton, E. Riccietti, X. Vasseur, 2020
- *On a multilevel Levenberg-Marquardt method for the training of ANNs and its application to the solution of PDEs*, H. Calandra, S. Gratton, E. Riccietti, X. Vasseur, 2020
- *Multilevel physics informed neural networks (MPINNs)*, E. Riccietti, V. Mercier, S. Gratton, P. Boudier, 2022
- *Méthodes proximales multi-niveaux pour la restauration d'images*, G. Lauga, E. Riccietti, N. Pustelnik, P. Gonçalves, 2022