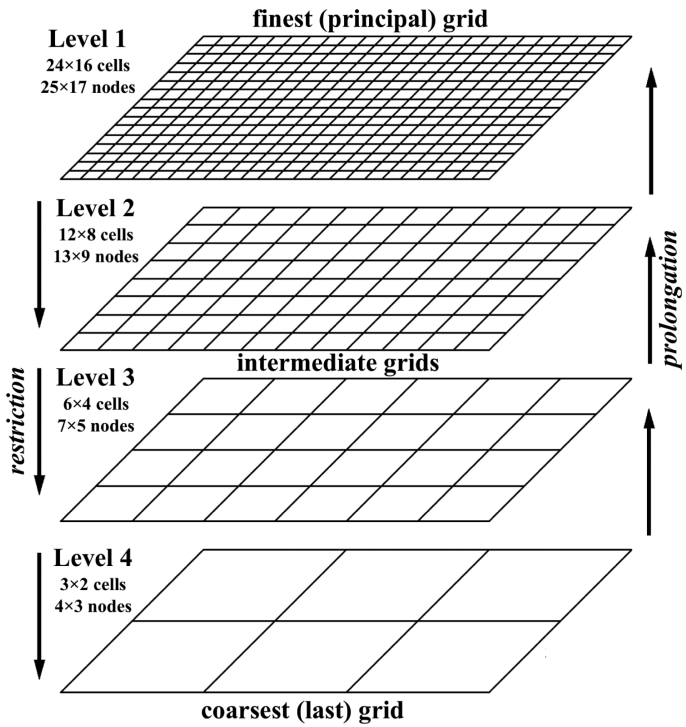
Cours 5

Multigrid and multilevel methods

Elisa Riccietti
and
Théo Mary

LIP-ENS Lyon



The concept

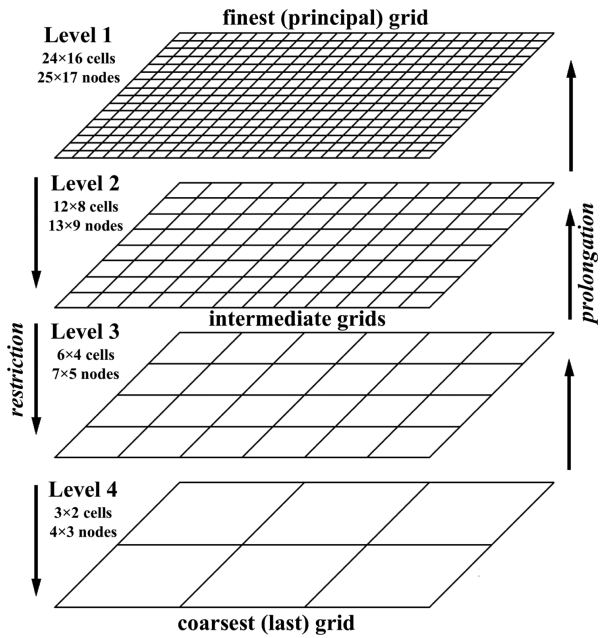
AIM? Given a problem, reduce the computational cost of the solution process

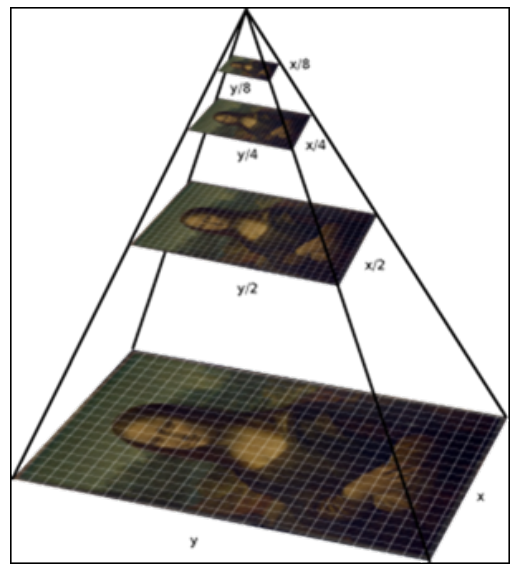
HOW? Exploit the structure to perform dimensionality reduction

The concept

AIM? Given a problem, reduce the computational cost of the solution process

HOW? Exploit the structure to perform dimensionality reduction

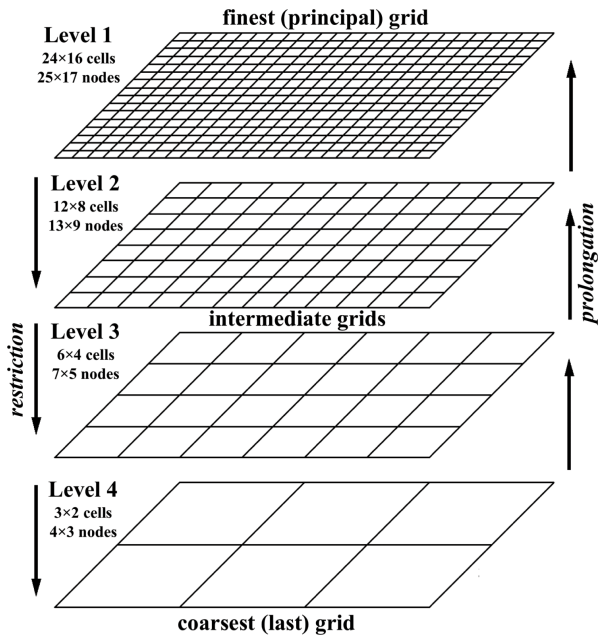


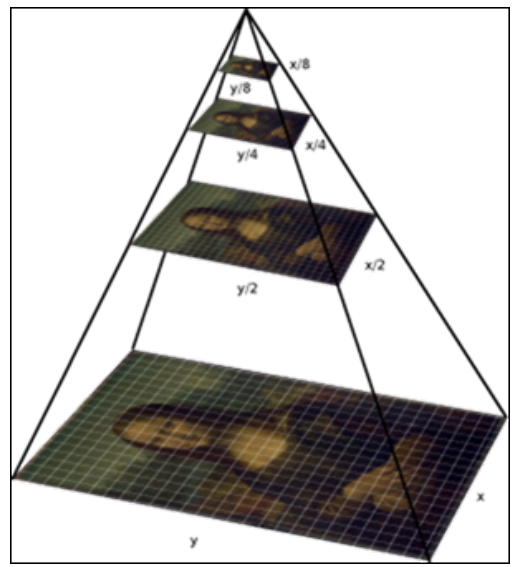


The concept

AIM? Given a problem, reduce the computational cost of the solution process

HOW? Exploit the structure to perform dimensionality reduction





Multilevel methods (ML)

Outline

The problem

Numerical solution of partial differential equations (PDEs)
Given $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$, $g : \partial\Omega \to \mathbb{R}$ and a differential operator D, find $u : \mathbb{R}^d \to \mathbb{R}$ that solves

$$D(u(x)) = f(x) \text{ in } \Omega$$
$$u(x) = g(x) \text{ in } \partial\Omega$$

Examples:

- Poisson's equation: $D(u(x)) = \Delta u(x) = \sum_{i=1}^{d} \frac{\partial^2 u(x)}{\partial x_i^2}$
- Nonlinear equation: $D(u(x)) = \sum_{i=1}^{d} a_i(x) \frac{\partial^2 u(x)}{\partial x_i^2} + \rho(x)$

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

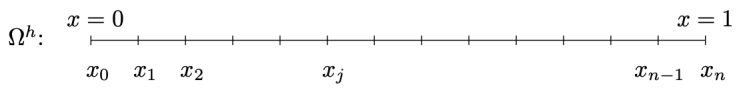
Our model test problem

A simple 1D example: **1D Poisson's problem** with Dirichlet boundary conditions.
Given $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ find $u : \mathbb{R} \to \mathbb{R}$ such that

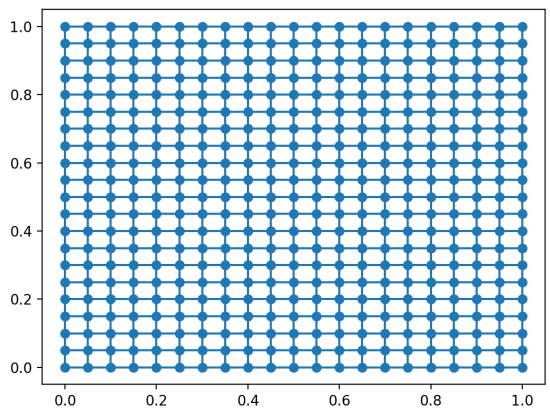
$$\begin{aligned} -u''(x) &= f(x), \ x \in (0,1) \\ u(0) &= g(0) = 0 \\ u(1) &= g(1) = 0 \end{aligned}$$

The numerical solution of PDEs: discretization

- Build a grid on Ω.
  Example: equispaced points on each axis
- Discretize D on the chosen grid and obtain a discrete problem
- The size of the grids impacts the size of the discrete problem and the accuracy of the solution approximation



1D: $h = \frac{1}{n}$, $x_j = jh$, $0 \leq j \leq n$



Example: 2D, $\Omega = [0, 1] \times [0, 1]$

Discretization of derivatives: the simple 1D case

Let $u : \mathbb{R} \to \mathbb{R}$

$$u'(x) = \lim_{h \to 0} \frac{u(x) - u(x+h)}{h}$$

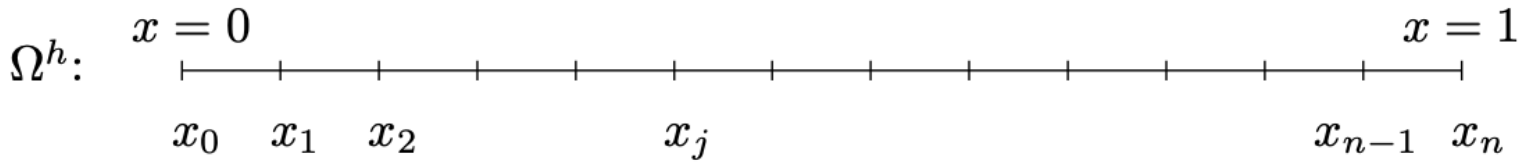
Discretization: for fixed small h

$$u'(x) \sim \frac{u(x) - u(x+h)}{h} \text{ or } \frac{u(x-h) - u(x)}{h}$$

$$u''(x) = (u')'(x) \sim \frac{u'(x) - u'(x+h)}{h} \sim \frac{\frac{u(x-h)-u(x)}{h} - \frac{u(x)-u(x+h)}{h}}{h}$$

$$= \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}$$

Discretization of the model test problem



1D: $h = \frac{1}{n}$, $x_j = jh$, $0 \le j \le n$

- The boundary conditions: $u(x_0) = g(x_0) = 0$, $u(x_n) = g(x_n) = 0$
- On the interior points of the grid: $-u''(x_j) = f(x_j)$ $j = 1, \ldots, n-1$
- Solution approximation: $(0, u_1, \ldots, u_n, 0)$
- Discretization of $u''(x_j) \sim \frac{u(x_j-h)-2u(x_j)+u(x_j+h)}{h^2} = \frac{u_{j-1}-2u_j+u_{j+1}}{h^2}$

Matrix form:

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & & \ddots & & & & \\ & & & \ddots & & & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -1 & 2 & -1 \\ & & & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_{n-1}) \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \end{bmatrix}$$

If the PDE is linear, we obtain a linear system

$$Au = f, \quad A \in \mathbb{R}^{n+1 \times n+1}$$

Fixed point methods

$Au = f$ is solved using a fixed point method.

Fixed point scheme :

$$u^{(k+1)} = Ru^{(k)} + b$$

The true solution :

$$u^* = Ru^* + b$$

To define R, we decompose the matrix:

$$A = L + D + U$$

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix}}_{U}$$

Examples of fixed-point schemes

- Jacobi:

$$\mathrm{u}^{(k+1)} = \underbrace{-D^{-1}(L+U)}_{R}\,\mathrm{u}^{(k)} + \underbrace{D^{-1}\mathrm{f}}_{b}$$

- Gauss-Seidel:

$$\mathrm{u}^{(k+1)} = \underbrace{-(L+D)^{-1}U}_{R}\,\mathrm{u}^{(k)} + \underbrace{(L+D)^{-1}\mathrm{f}}_{b}$$

Fixed point: reduction of the error

Fixed point scheme :

$$u^{(k+1)} = Ru^{(k)} + b$$

The true solution :

$$u^* = Ru^* + b$$

The error :

$$e^{(k+1)} := u^{(k+1)} - u^* = Re^{(k)}$$

After M iterations:

$$e^{(M)} := u^{(M)} - u^* = Re^{(M-1)} = R^2 e^{(M-2)} = \cdots = R^M e^{(0)}$$

Convergence:

$$\lim_{M\to\infty} \|e^{(M)}\| = 0 \iff \rho(R) < 1,$$

where $\rho(R) = \max\{|\lambda_1|, \ldots, |\lambda_n|\}$ eigenvalues

The Jacobi scheme: study of the convergence

Let us consider or test problem:

$$A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & & \ddots & & & & \\ & & & \ddots & & & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -1 & 2 & -1 \\ & & & & & & -1 & 2 \end{bmatrix}$$

We can see that

$$R := -D^{-1}(L+U) = I - \frac{1}{2}A \tag{1}$$

The Jacobi scheme: study of the convergence

Eigenvalues and eigenvectors

$$\lambda_k(A) = 4\sin^2\left(\frac{k\pi}{2n}\right),\ 1 \le k \le n-1$$

$$(w_k)_j = \sin\left(\frac{k\pi j}{n}\right),\ 1 \le j \le n-1$$

Thus from (1)

$$\lambda_k(R) = 1 - \frac{1}{2}\lambda_k(A) = 1 - 2\sin^2\left(\frac{k\pi}{2n}\right),\ 1 \le k \le n-1$$

Eigenvectors are the same as those of A: $Rw_k = \lambda_k(R)w_k$

Notice that $|\lambda_k(R)| < 1$ for all k and thus the method converges, but the rate of convergence will depend on how small $|\lambda_k(R)|$ are

Reduction of the error

The eigenvectors form a basis.
Initial error:

$$e^{(0)} = \sum_{k=1}^{n-1} c_k w_k$$

After M iterations:

$$e^{(M)} := u^{(M)} - u^* = R^M e^{(0)} = \sum_{k=1}^{n-1} c_k R^M w_k = \sum_{k=1}^{n-1} c_k \lambda_k^M(R) w_k$$

- After M iterations, the kth components of the initial error (modes) w_k has been reduced by a factor of $\lambda_k^M(R) < 1$
- Modes are not mixed: the iteration can change the amplitude of a mode, but it cannot convert that mode into different modes

The Fourier modes

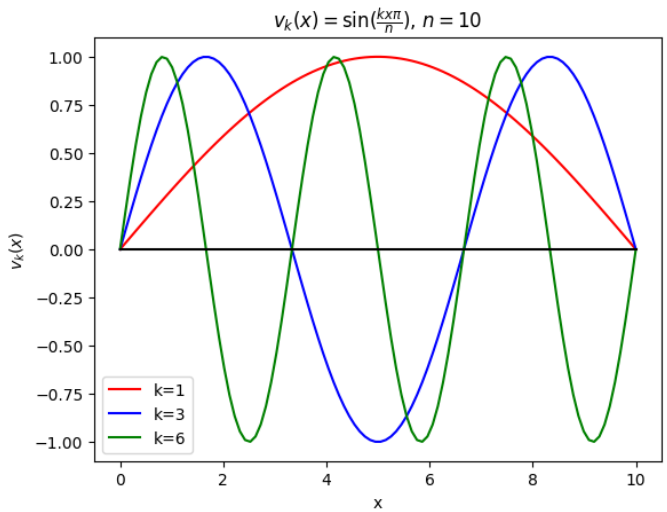
The vectors

$$(w_k)_j = \sin\left(\frac{k\pi j}{n}\right),\ 1 \leq j \leq n-1$$

are special vectors called the **Fourier modes** and k is the frequency.

On a n-point grid:

- $1 \leq k < \frac{n}{2}$ low frequencies
- $\frac{n}{2} \leq k < n-1$ high frequencies



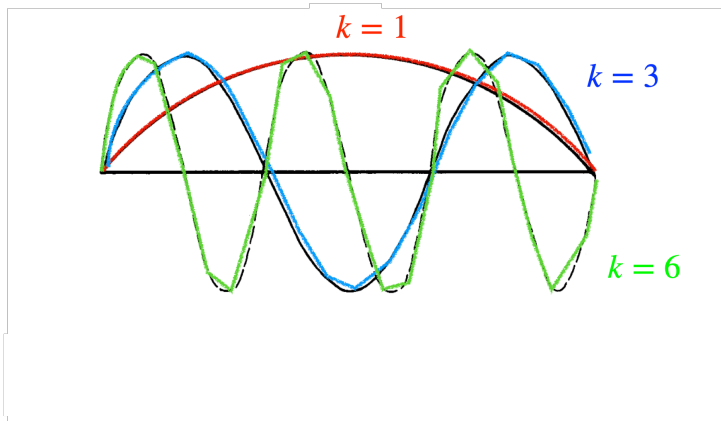
Limitation of iterative schemes: the smoothing property

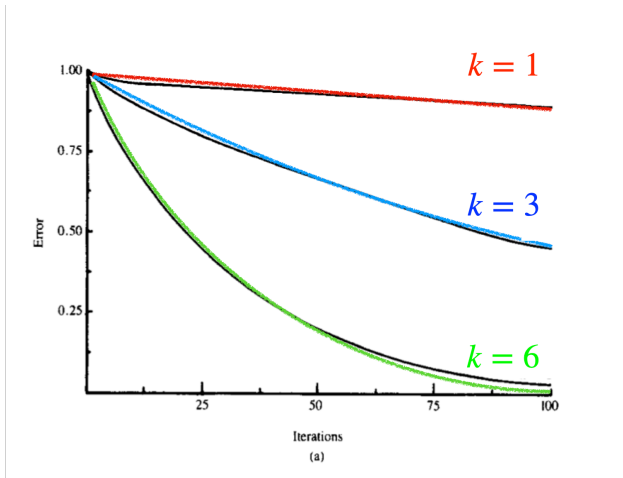
$$e^{(M)} = u^{(M)} - u^* = \sum_{k=1}^{n-1} c_k \lambda_k^M(R) w_k$$

- $\lambda_1(R) \approx 1$
- $|\lambda_k(R)| < \frac{1}{\mu}$ for all $n/2 \leq k \leq n-1$
- μ is the smoothing factor (dumping coefficient of the oscillatory components at each relaxation)

Hard to reduce the low frequency components of the error

$$v_k(x) = sin\left(\frac{xk\pi}{n}\right), 0 \leq x \leq n$$





How to make the methods efficient on all frequencies?

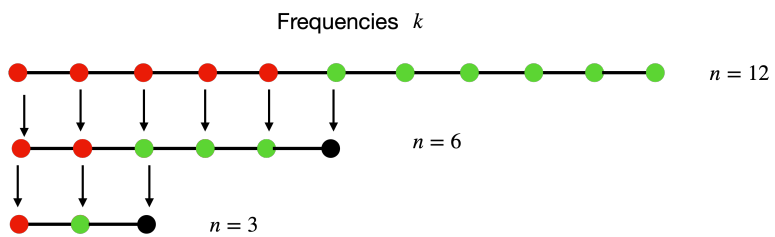
Frequency shift by coarsening!

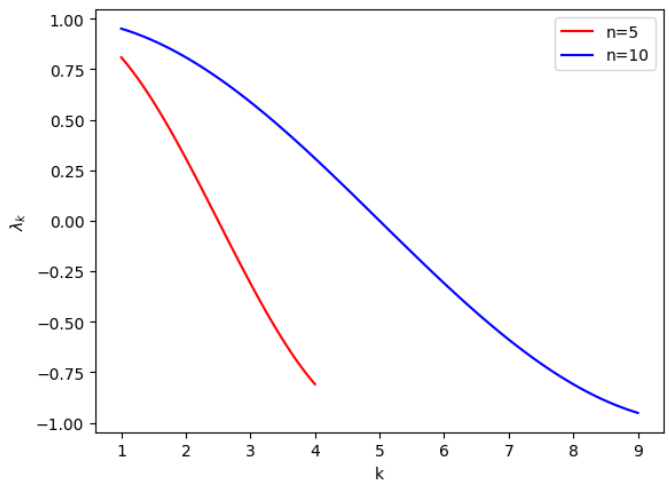
- **Fine** grid Ω^h with n points: $1 \leq k \leq n-1$
- **Coarse** grid Ω^{2h} with $n/2$ points: $1 \leq k \leq n/2$

  Grid points of Ω^{2h} are the even-numbered points of Ω^h.

Property: $(w_k^h)_{2j} = (w_k^{2h})_j$

Wavenumbers $k \leq n^h/2$ in $\Omega^h \rightarrow$ wavenumbers $k \leq n^{2h}$ in Ω^{2h}





Accelerate convergence by considering multiple grids

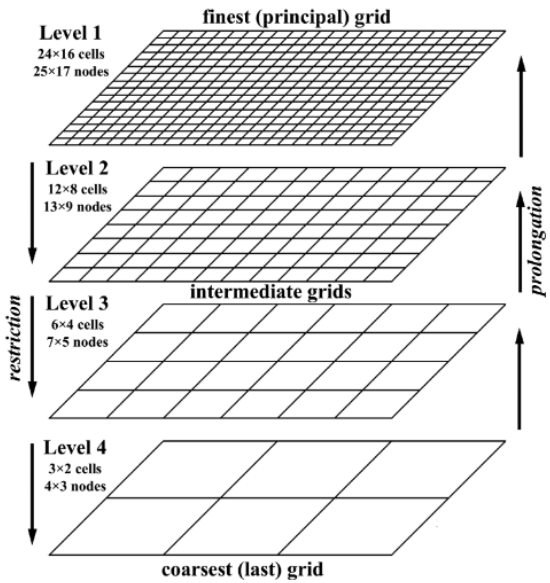
- The smoothing rate (the convergence factor for the oscillatory modes) for the standard relaxation schemes is small and independent of the grid spacing h.
- The smooth error modes, which remain after relaxation on one grid, appear more oscillatory on the coarser grids.
- Moving to successively coarser grids, all of the error components on the original fine grid eventually appear oscillatory and are reduced by relaxation.
- The overall convergence factor for a good multigrid scheme is small and independent of h.

Part I

Multigrid methods

Multigrid methods

After removing all the oscillatory components, when relaxation begins to stall, signaling the predominance of smooth error modes, it is advisable to move to a coarser grid; there, the smooth error modes appear more oscillatory and relaxation will be more effective.



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

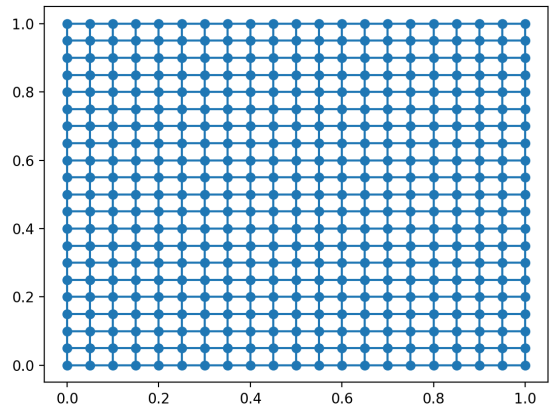
Ingredients of a two-level multigrid methods

Consider a PDE:

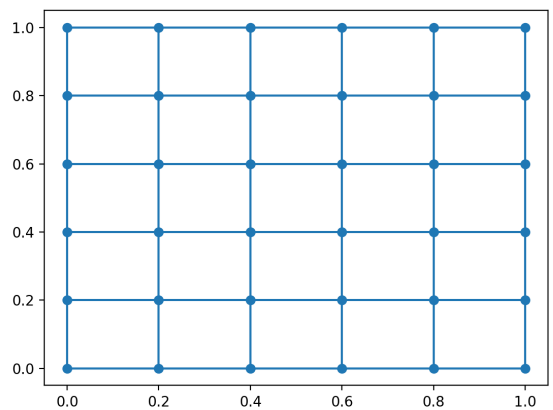
$$Au = f$$

Consider two discretizations:

- Fine grid: $A^h u^h = f^h$
- Coarse grid: $A^{2h} u^{2h} = f^{2h}$



$I_h^{2h} \Downarrow$ $I_{2h}^h \Uparrow$



Transfer operators: how do we move between two grids ?

From fine to coarse: I_h^{2h} (restriction)

- Injection:

$$v_j^{2h} = v_{2j}^h$$

- Full weighting

$$v_j^{2h} = \frac{1}{4}(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h)$$

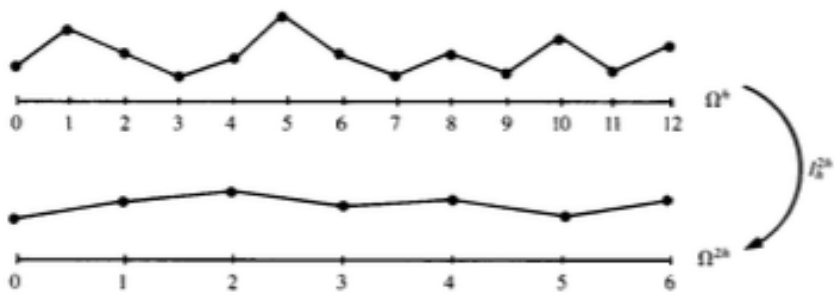


Figure 3.4: *Restriction by full weighting of a fine-grid vector to the coarse grid.*

Transfer operators: how do we move between two grids ?

From coarse to fine: I_{2h}^{h} (prolongation)

- Interpolation

$$v_{2j}^{h} = v_{j}^{2h}$$
$$v_{2j+1}^{h} = \frac{1}{2}(v_{j}^{2h} + v_{j+1}^{2h})$$

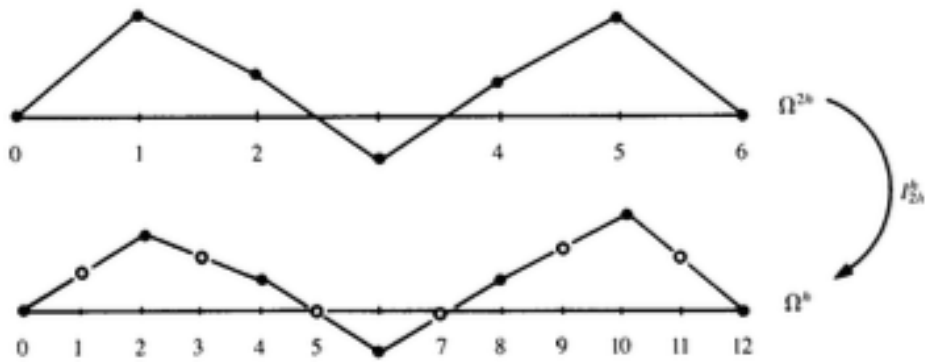


Figure 3.2: *Interpolation of a vector on coarse grid Ω^{2h} to fine grid Ω^{h}.*

How to define the coarse level operator A^{2h}?

- Galerkin approximation:

$$A^{2h} = I_h^{2h} A^h I_{2h}^h$$

- Discretization of differential operator on the coarse grid

Two-level multigrid methods

Consider a PDE:

$$Au = f$$

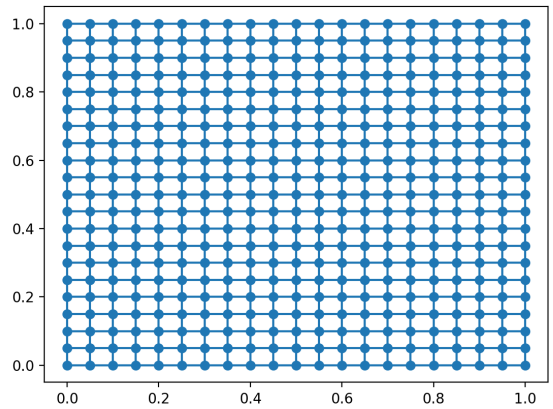
Consider two discretizations:

- Fine grid: $A^h u^h = f^h$
- Coarse grid: $A^{2h} u^{2h} = f^{2h}$

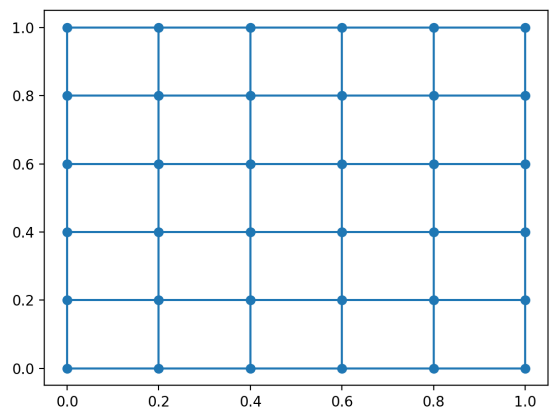
Idea: write the solution u as the sum of a fine and a coarse term:

$$u \sim \underbrace{v^h}_{\in \mathbb{R}^h} + P(\underbrace{e^{2h}}_{\in \mathbb{R}^{2h}}).$$

and update the two components in an alternate fashion.



$I_h^{2h} \Downarrow$ $I_{2h}^h \Uparrow$



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 $r^{2h} = Rr^h$
-
-

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 $r^{2h} = Rr^h$
- *Coarse level*: compute correction: $A^{2h} e^{2h} = r^{2h}$
-

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 $r^{2h} = Rr^h$
- *Coarse level*: compute correction: $A^{2h} e^{2h} = r^{2h}$
- Correct: $v^h \leftarrow v^h + P(e^{2h})$

Link to iterative refinement, see

General multigrid methods - V-cycle

V-Cycle Scheme

$$\mathbf{v}^h \leftarrow V^h(\mathbf{v}^h, \mathbf{f}^h)$$

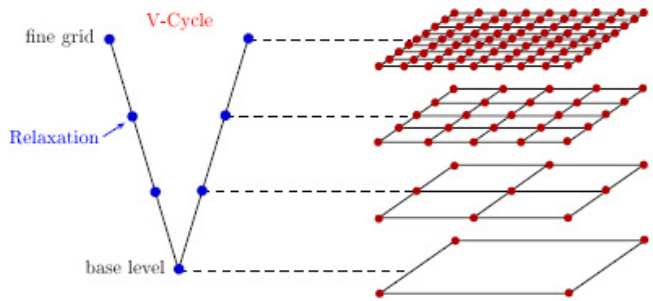
- Relax on $A^h\mathbf{u}^h = \mathbf{f}^h$ ν_1 times with initial guess $\mathbf{v}^h$.
- Compute $\mathbf{f}^{2h} = I_h^{2h}\mathbf{r}^h$.
  - Relax on $A^{2h}\mathbf{u}^{2h} = \mathbf{f}^{2h}$ ν_1 times with initial guess $\mathbf{v}^{2h} = \mathbf{0}$.
  - Compute $\mathbf{f}^{4h} = I_{2h}^{4h}\mathbf{r}^{2h}$.
    - Relax on $A^{4h}\mathbf{u}^{4h} = \mathbf{f}^{4h}$ ν_1 times with initial guess $\mathbf{v}^{4h} = \mathbf{0}$.
    - Compute $\mathbf{f}^{8h} = I_{4h}^{8h}\mathbf{r}^{4h}$.

      .
      .
      .

      - Solve $A^{Lh}\mathbf{u}^{Lh} = \mathbf{f}^{Lh}$.

      .
      .
      .

    - Correct $\mathbf{v}^{4h} \leftarrow \mathbf{v}^{4h} + I_{8h}^{4h}\mathbf{v}^{8h}$.
    - Relax on $A^{4h}\mathbf{u}^{4h} = \mathbf{f}^{4h}$ ν_2 times with initial guess $\mathbf{v}^{4h}$.
  - Correct $\mathbf{v}^{2h} \leftarrow \mathbf{v}^{2h} + I_{4h}^{2h}\mathbf{v}^{4h}$.
  - Relax on $A^{2h}\mathbf{u}^{2h} = \mathbf{f}^{2h}$ ν_2 times with initial guess $\mathbf{v}^{2h}$.
- Correct $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$.
- Relax on $A^h\mathbf{u}^h = \mathbf{f}^h$ ν_2 times with initial guess $\mathbf{v}^h$.



General multigrid methods - V-cycle

V-Cycle Scheme (Recursive Definition)

$$\mathbf{v}^h \leftarrow V^h(\mathbf{v}^h, \mathbf{f}^h).$$

1. Relax ν_1 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ with a given initial guess $\mathbf{v}^h$.

2. If Ω^h = coarsest grid, then go to step 4.

   Else

   $$\mathbf{f}^{2h} \leftarrow I_h^{2h}(\mathbf{f}^h - A^h\mathbf{v}^h),$$
   $$\mathbf{v}^{2h} \leftarrow \mathbf{0},$$
   $$\mathbf{v}^{2h} \leftarrow V^{2h}(\mathbf{v}^{2h}, \mathbf{f}^{2h}).$$

3. Correct $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$.

4. Relax ν_2 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ with initial guess $\mathbf{v}^h$.

General multigrid methods - other cycles

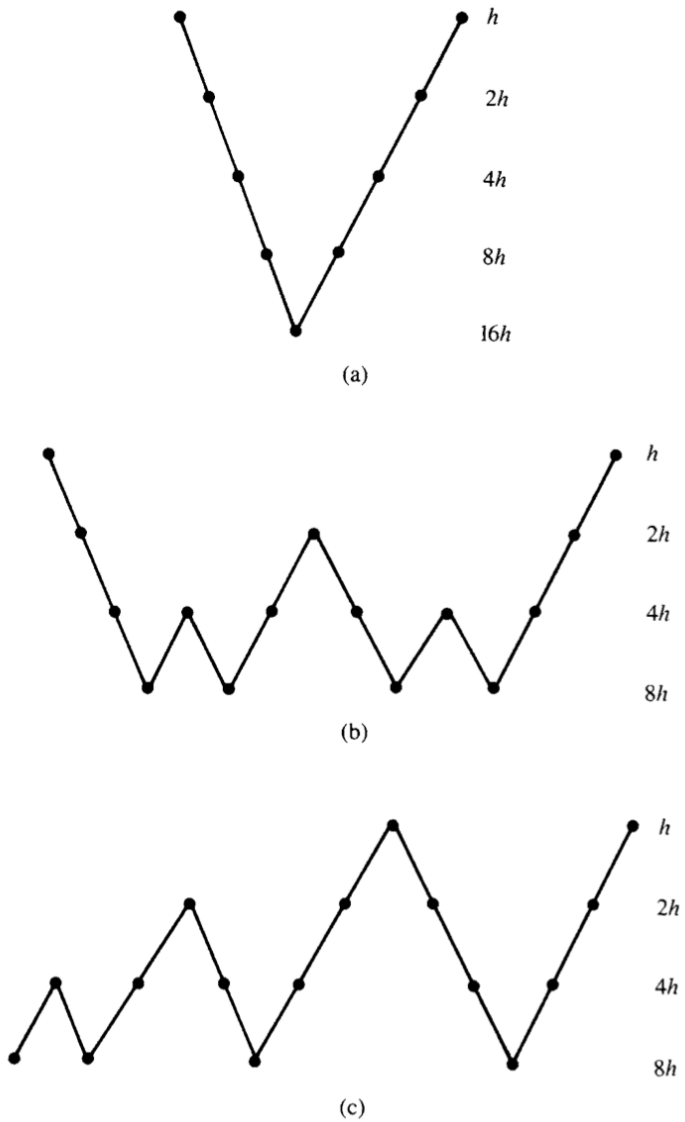


Figure 3.6: *Schedule of grids for* (a) *V-cycle,* (b) *W-cycle, and* (c) *FMG scheme, all on four levels.*

Schemes for the cycles

Full Multigrid V-Cycle (Recursive Form)

$$\mathbf{v}^h \leftarrow FMG^h(\mathbf{f}^h).$$

1. If Ω^h = coarsest grid, set $\mathbf{v}^h \leftarrow \mathbf{0}$ and go to step 3.

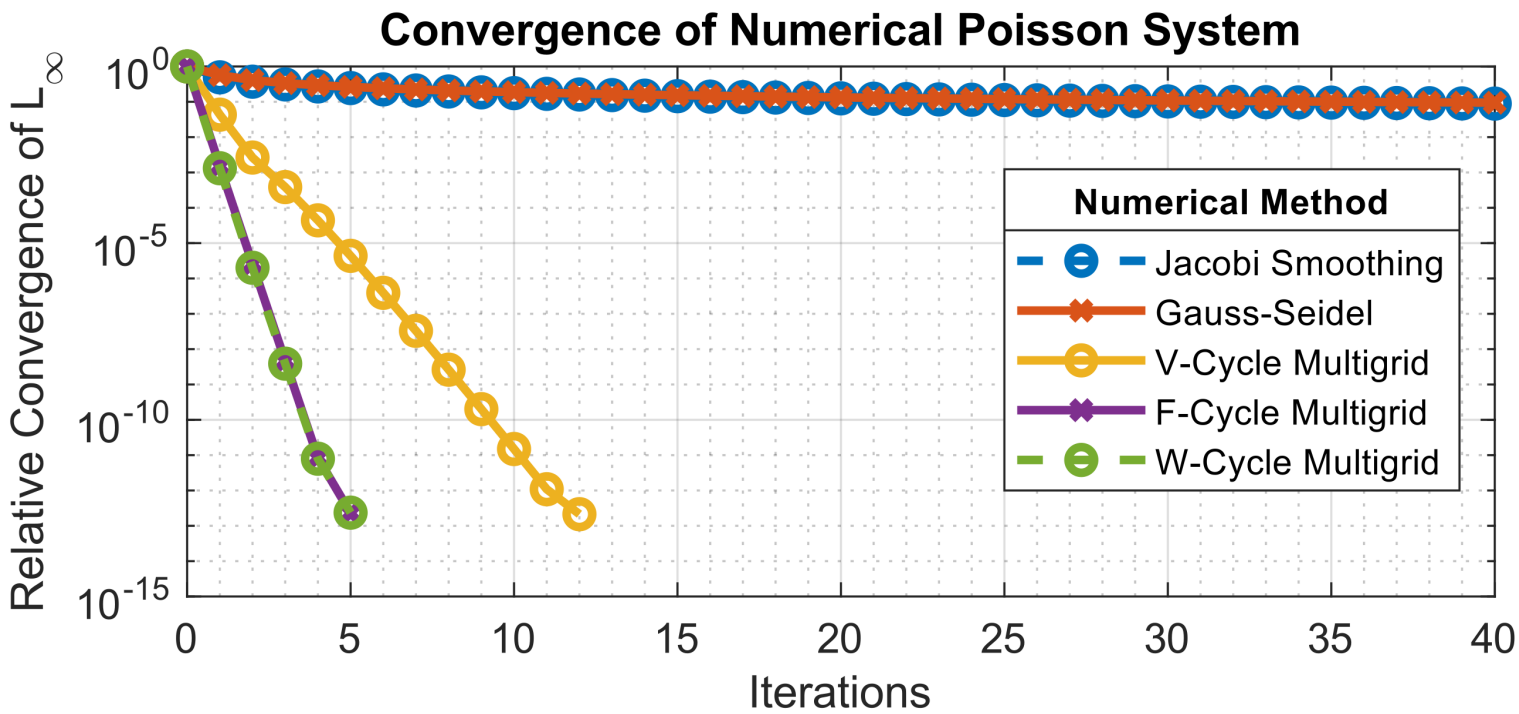
Else

$$\mathbf{f}^{2h} \leftarrow I_h^{2h}(\mathbf{f}^h),$$
$$\mathbf{v}^{2h} \leftarrow FMG^{2h}(\mathbf{f}^{2h}).$$

2. Correct $\mathbf{v}^h \leftarrow I_{2h}^h \mathbf{v}^{2h}$.

3. $\mathbf{v}^h \leftarrow V^h(\mathbf{v}^h, \mathbf{f}^h)$ ν_0 times.

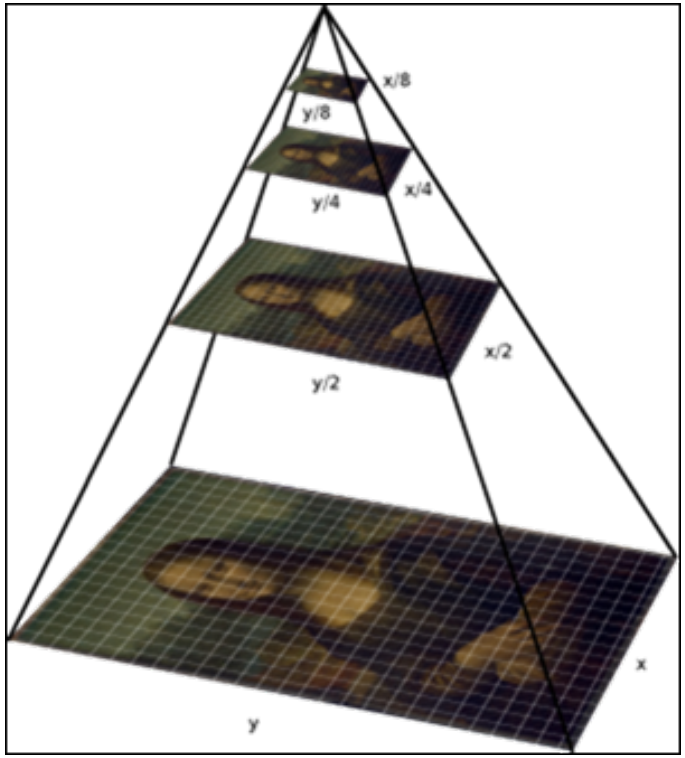
Numerical example



From linear to nonlinear: multilevel methods

Multigrid methods work for linear systems: $Ax = b$
An extension exists for nonlinear equations: $\mathcal{A}(x) = b$ FAS (Full Approximation Scheme)

But what happens if we want to tackle a more general optimization problem with a hierarchical structure ?



Part II

Multilevel optimization methods

Classical iterative optimization methods

Large-scale nonlinear unconstrained optimization problems:

$$\min_x f(x)$$

- Build a model ($B_k \sim \nabla^2 f(x_k)$ or $B_k = 0$):

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

Classical iterative optimization methods

Large-scale nonlinear unconstrained optimization problems:

$$\min_x f(x)$$

- Build a model ($B_k \sim \nabla^2 f(x_k)$ or $B_k = 0$):

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

- Compute a step s_k ($r(\lambda_k)$ regularization term):

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

Classical iterative optimization methods

Large-scale nonlinear unconstrained optimization problems:

$$\min_x f(x)$$

- Build a model ($B_k \sim \nabla^2 f(x_k)$ or $B_k = 0$):

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

- Compute a step s_k ($r(\lambda_k)$ regularization term):

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

- Update the iterate $x_{k+1} = x_k + s_k$

Classical examples

- Gradient method:

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

- Adaptive second order method:

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

- Adaptive Cubic Regularization (ARC):

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

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

MG/OPT: *A multigrid approach to discretized optimization problems*, Nash, 2000[1]

RMTR: *Recursive trust-region methods*, S. Gratton, A. Sartenaer and Ph. L. Toint, 2008[2]

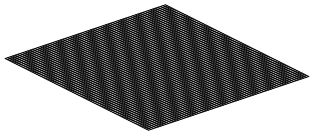
[1] https://optimization-online.org/wp-content/uploads/2012/04/3447.pdf

[2] https://www.cerfacs.fr/algor/reports/2007/TR_PA_07_42.pdf

Multilevel spirit

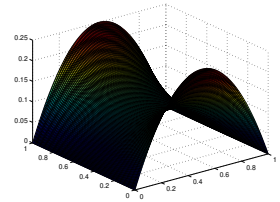
- Exploit structure of the problem to build a hierarchical representation of the problem
- Exploit lower levels to compute a cheap (but useful) step
- Simplify the landscape in a nonconvex setting

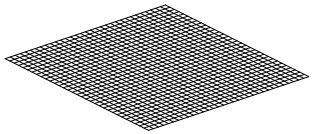
Hierarchy of problems ($n_r > n_{r-1} > \ldots n_0$)



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

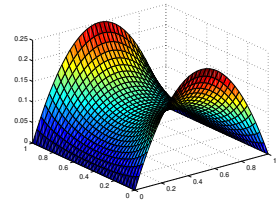
Restriction $\downarrow R_r$ $P_r \uparrow$ Prolongation





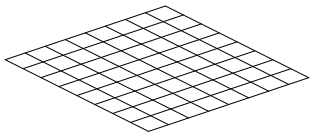
Fine Level $f_{r-1} : \mathbb{R}^{n_{r-1}} \to \mathbb{R}$

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



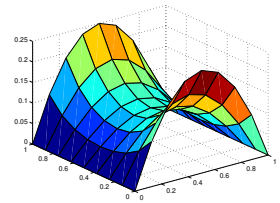
$\vdots$

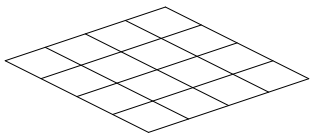
Restriction $\downarrow R_2$ $P_2 \uparrow$ Prolongation



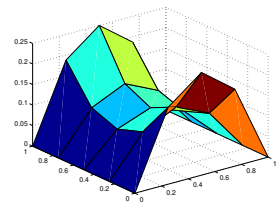
Coarse Level $f_2 : \mathbb{R}^{n_2} \to \mathbb{R}$

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





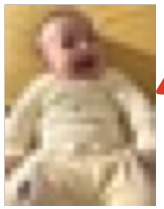
Coarsest Level $f_0 : \mathbb{R}^{n_0} \to \mathbb{R}$

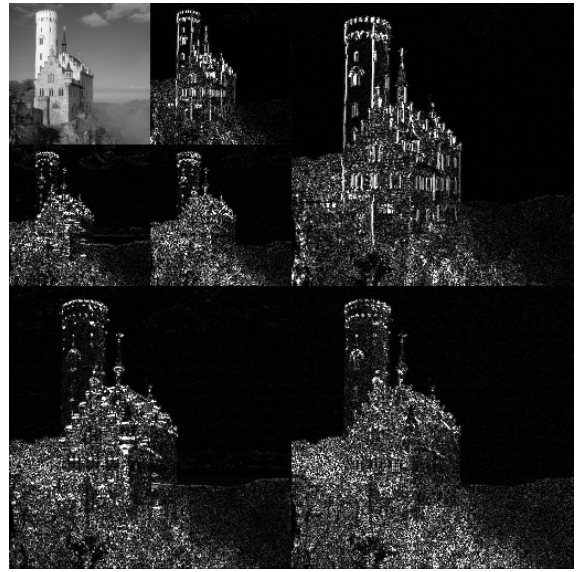


Example 1: image restoration

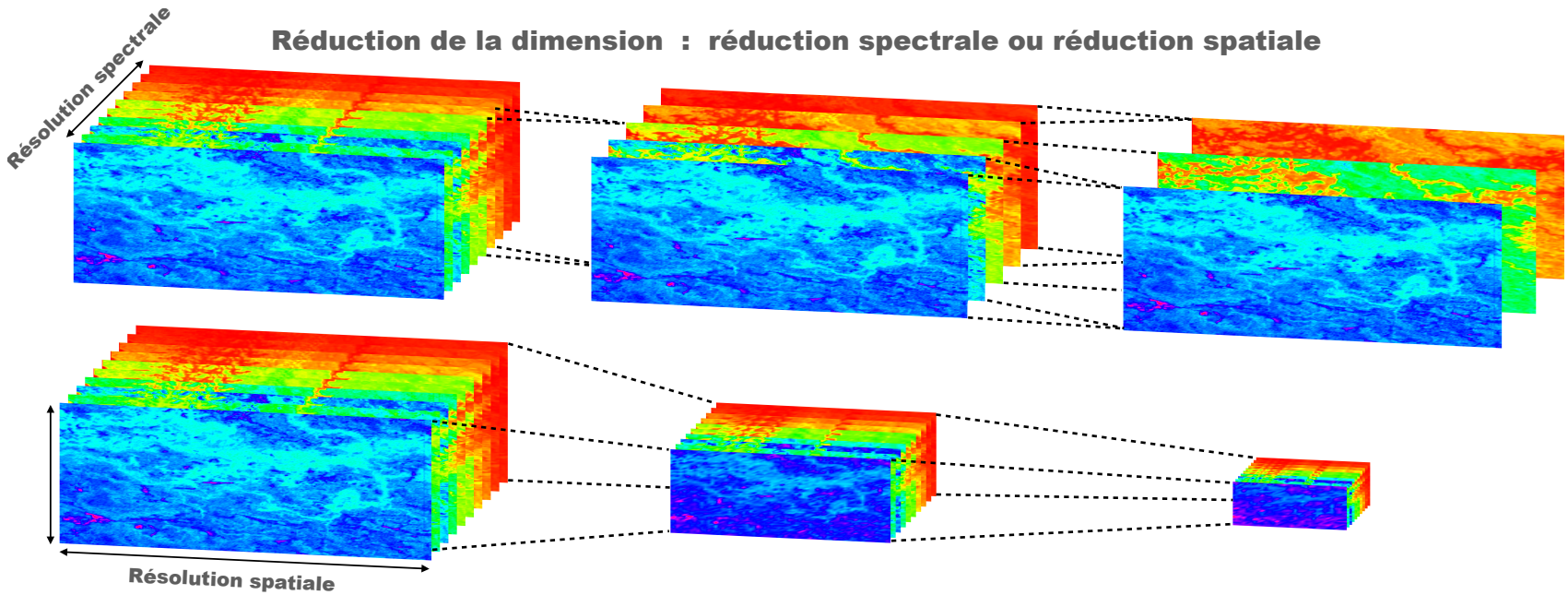
Decimation

Decimation

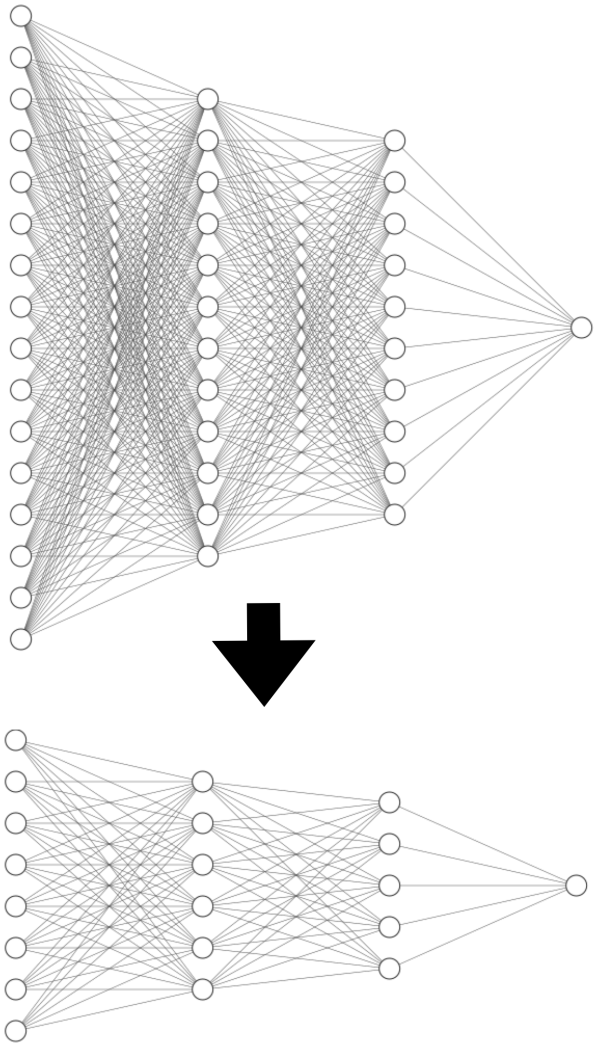


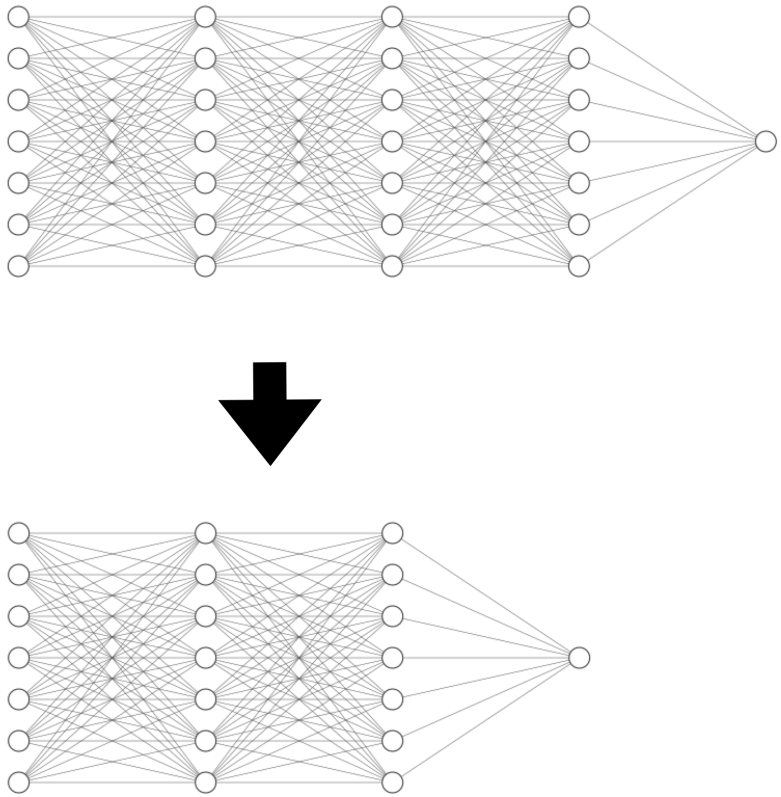


Example 2: hyperspectral imaging



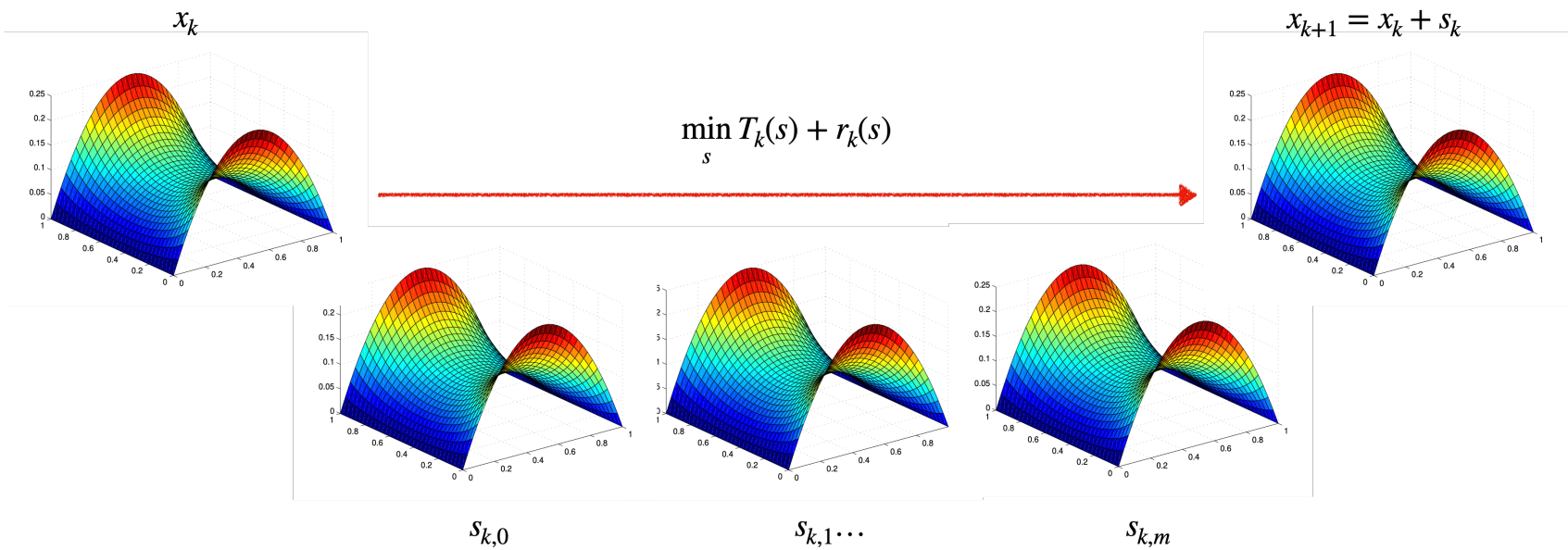
Example 3: neural networks





Classical one-level method

$$\min_x f(x)$$



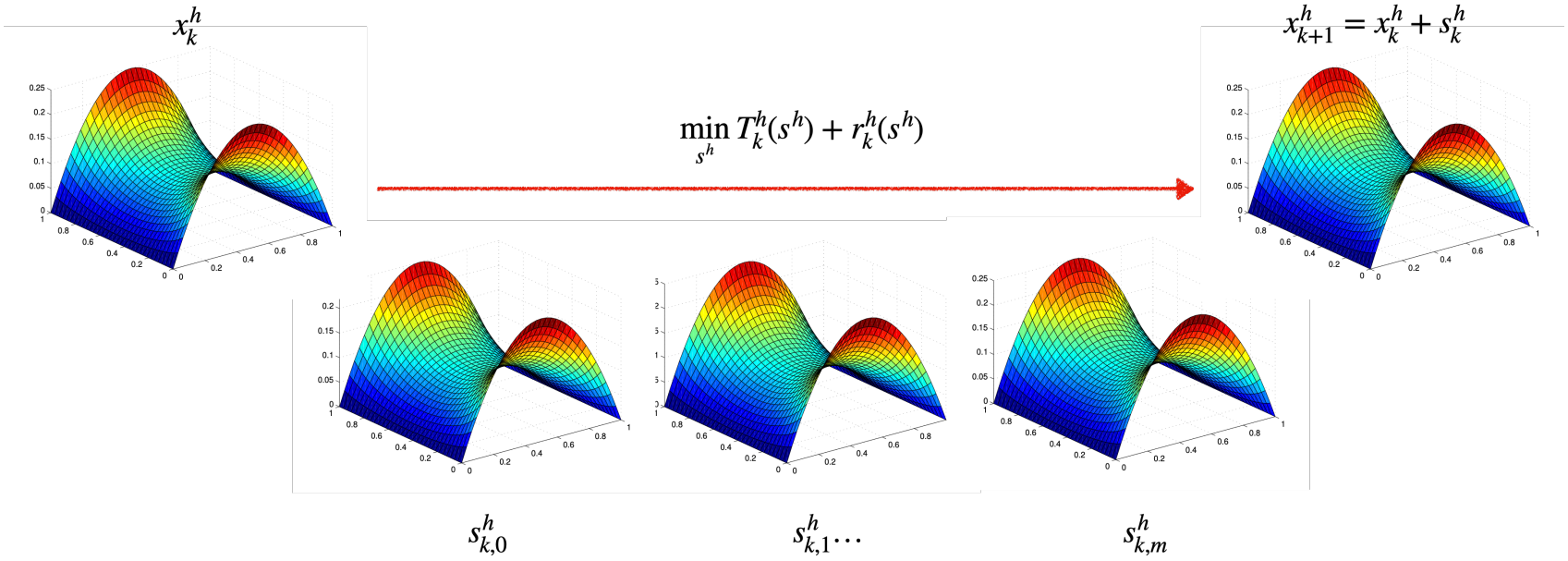
Multilevel strategy: step computation

Two choices:

1. Classical fine step
2. Coarse step

Fine step

$$\min_{x^h} f^h(x^h)$$



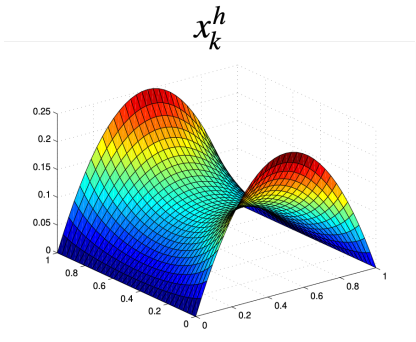
Multilevel strategy: step computation

Two choices:

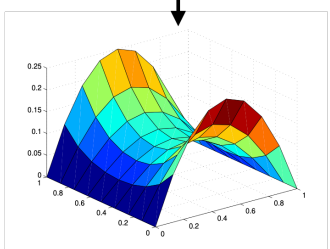
1. Classical fine step
2. Coarse step

Coarse step

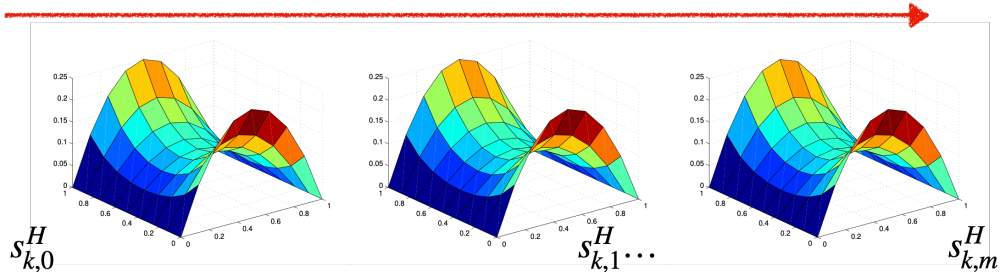
$$\min_{x^h} f^h(x^h) \sim \min_{x^H} f^H(x^H)$$

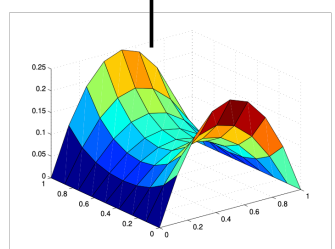


I_h^H

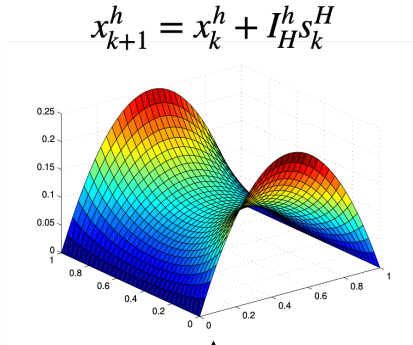


$$\min_{s^H} \mu_k^H(s^H)$$





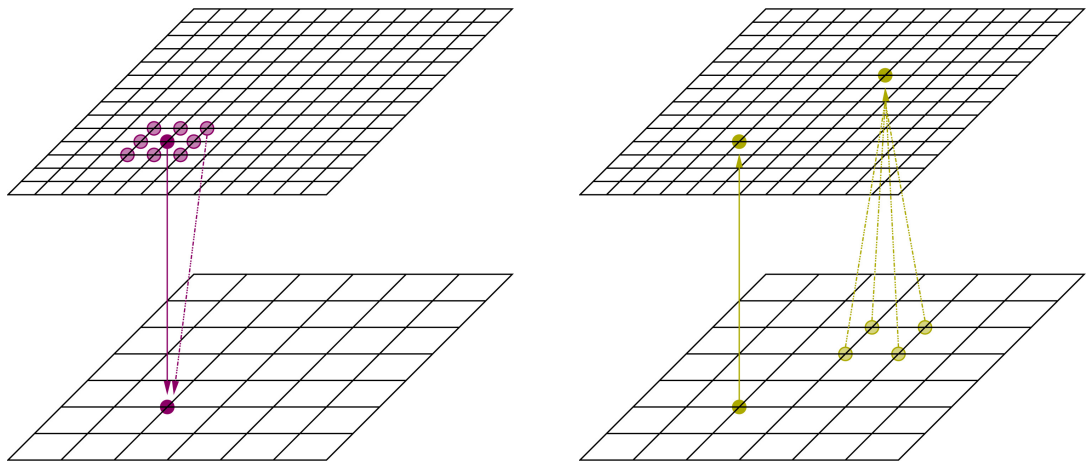
I_H^h



What do we need to use such a method ?

Transfer operators

- $I_h^H \in \mathbb{R}^{N_H \times N_h}$: from fine to coarse ($N_H < N_h$).
- $I_H^h \in \mathbb{R}^{N_h \times N_H}$: from coarse to fine.
- Relation between the operators : $I_H^h = \alpha (I_h^H)^T$, $\alpha > 0$.



I_h^H I_H^h

What do we need to use such a method ?

Lower level model

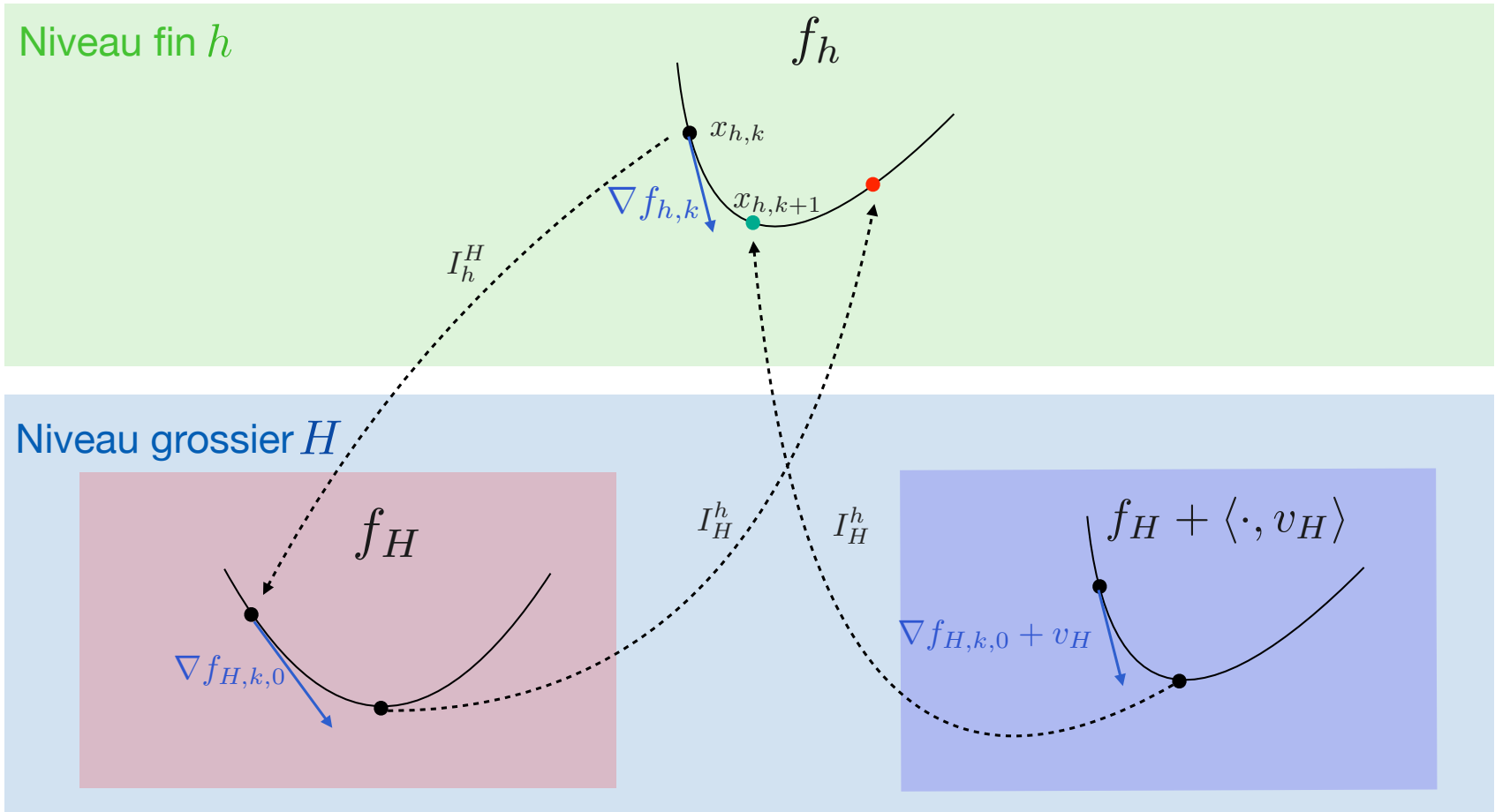
When to use the lower level model?

- Choose lower level model μ^H if
  - if $\|I_h^{2h} \nabla f^h(x_k^h)\| \geq \kappa \|\nabla f^h(x_k^h)\|$, $\kappa > 0$
  - if $\|\nabla \mu_k^H(x_k^H)\| > \epsilon^H$
- Minimize regularized Taylor model otherwise.

How to define the lower level model?

Modify f^H to ensure coherence among levels

First order coherence



Coherence between levels: first order

Let $x_k^H = I_h^H x_k^h$. Model with first order correction:

$$\mu_k^H(s^H) = f^H(x_k^H + s^H) + (v^H)^T s^H$$
$$v^H = I_h^H \nabla f^h(x_k^h) - \nabla f^H(x_k^H)$$

This ensures that

$$\nabla \mu_k^H(0^H) = I_h^H \nabla f^h(x_k^h)$$

- first-order behaviours of f^h and μ_k^H are coherent around x_k^h.
- If s^H descent direction for μ_k^H and $s^h = P^h s^H$, then s^h is a descent direction for f^h

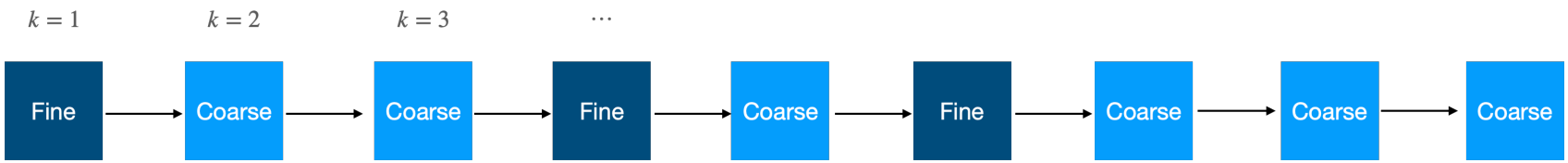
$$0 > \nabla \mu_k^H(0^H)^T s^H = (\nabla f^h(x_k^h))^T P^h s^H = (\nabla f^h(x_k^h))^T s^h$$

Multilevel algorithm: k-th iteration

Given x_k^h

1. If descent condition, decide to go to step 2 or step 3. Else go to step 2.
2. Fine iteration: iteration on f_h.
   - 2.1 Minimize $T_k^h(s)$ obtaining s_k^h.
   - 2.2 Set $x_{k+1}^h = x_k^h + s_k^h$.
3. Coarse iteration:
   - 3.1 Initialisation : $x_k^H = I_h^H x_k^h$
   - 3.2 Compute first order coherence v^H
   - 3.3 Minimization (possibly approximate) of $\mu_k^H(s^H) = f^H(x_k^H + s^H)$: $s_{m,k}^H = \underbrace{\Phi^H \circ \ldots \circ \Phi^H(0^H)}_{m \text{ times}}$
   - 3.4 Fine level update: $x_{k+1}^h = x_k^h + I_H^h s_{m,k}^H$

A possible iterative scheme



Parameters of the coarse iterations

- Φ_H: any minimization scheme, can be different from fine level one, can use second order
- How many iterations m?
  - m large enough to compensate projection cost
  - m not too large to keep first order coherence

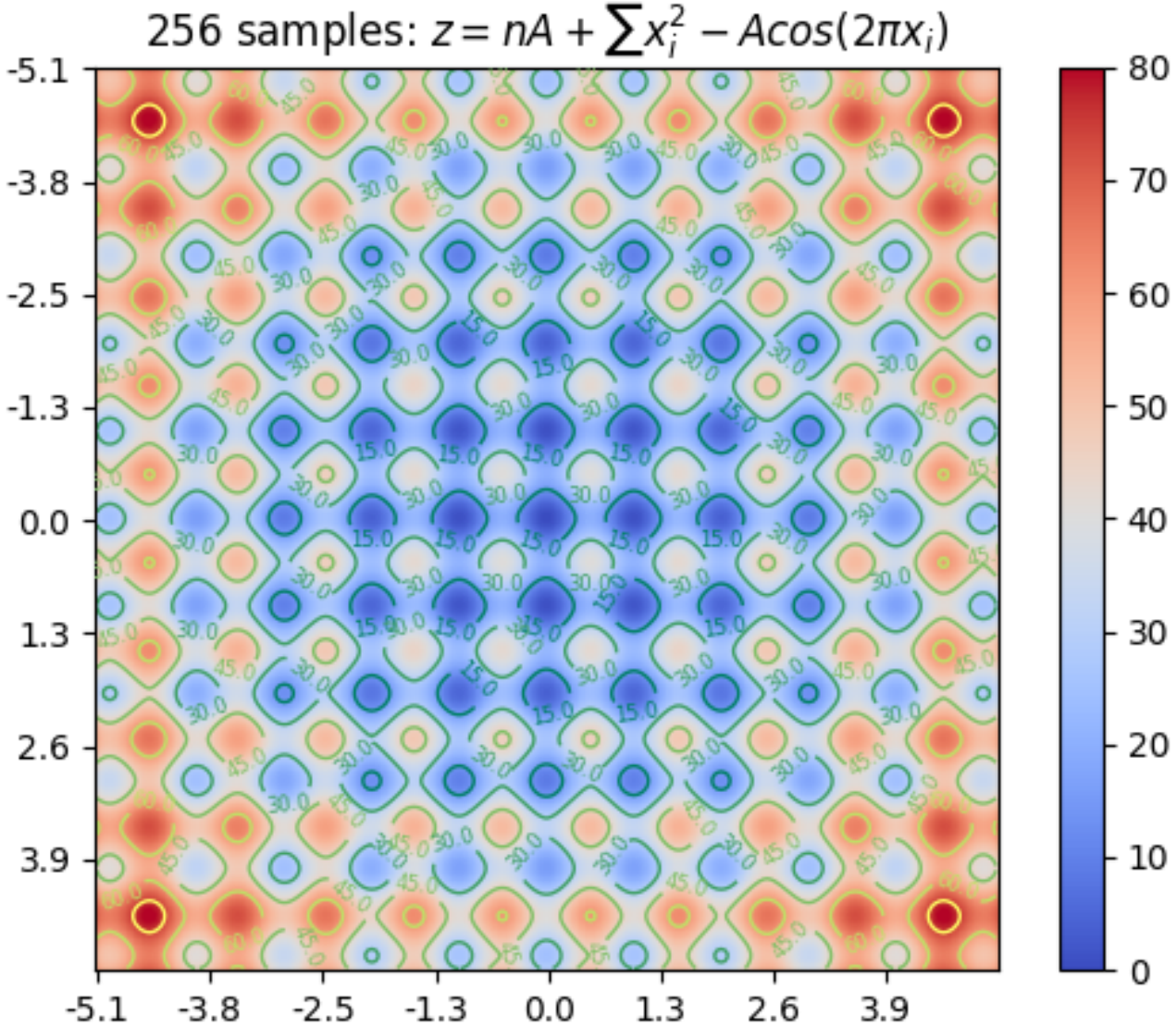
Effect of coarse projection in a nonlinear setting

1. Cost reduction: usually a factor 4 from one level to another in dimension 2
   - Level h: $f^h(u), u \in \mathbb{R}^{n^2}$
   - Level H: $f^H(u), u \in \mathbb{R}^{n^2/4}$

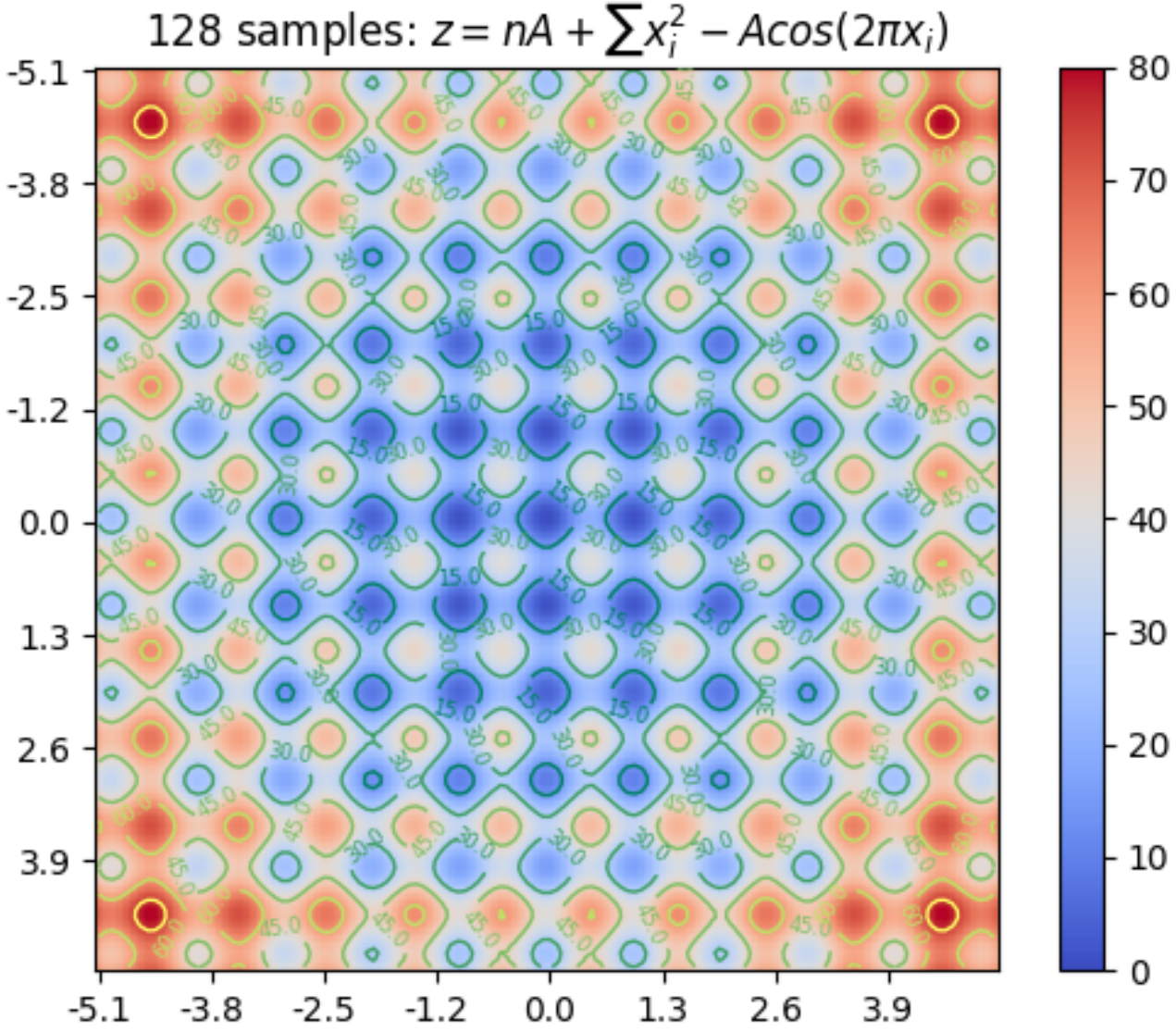
   $\rightarrow$ for the same cost, can do at coarse level 4 times the number of iterations at fine level

2. Smoothing of the landscape: escape local minima$\rightarrow$ better solution

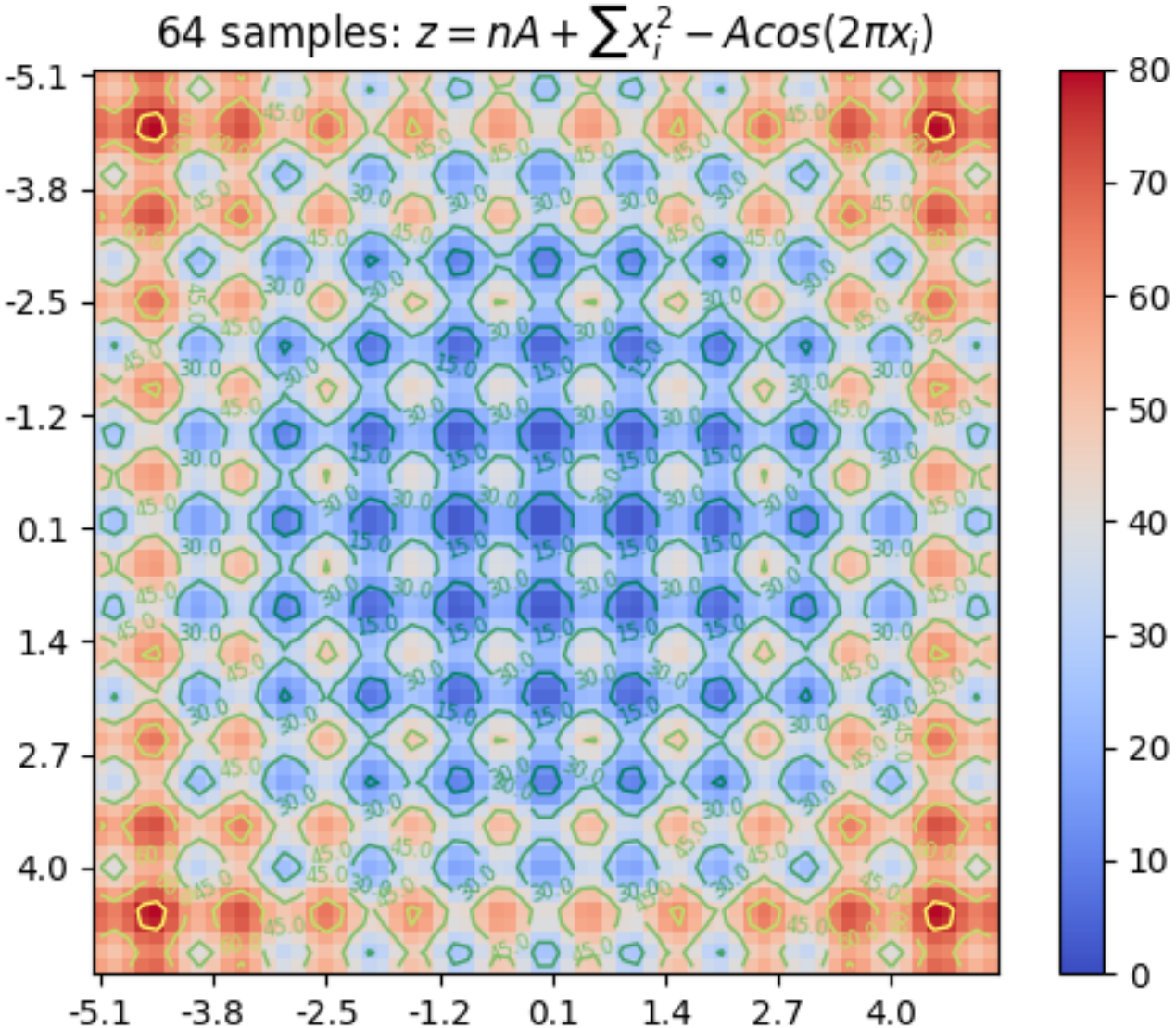
Smoothing of the landscape



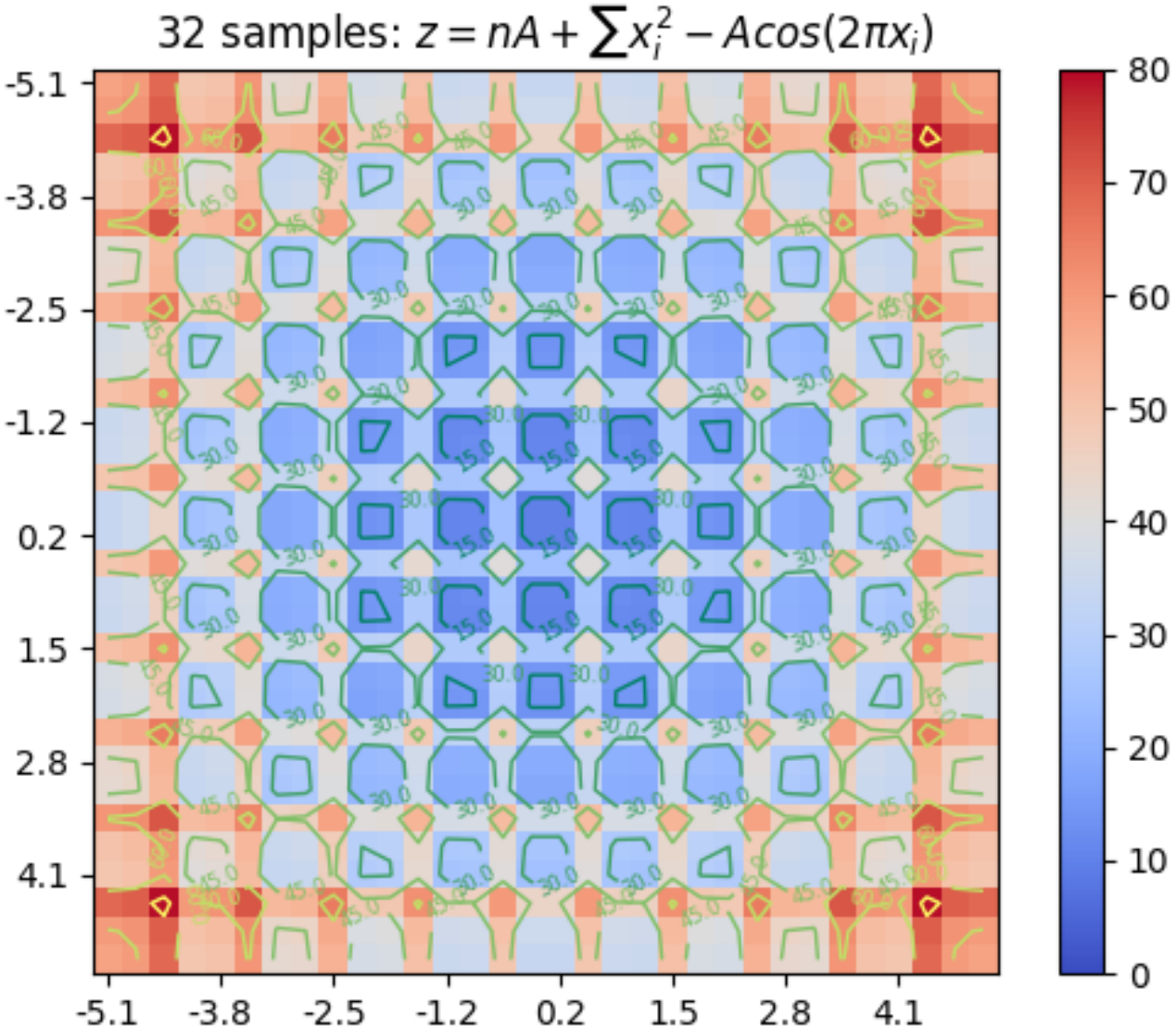
Smoothing of the landscape



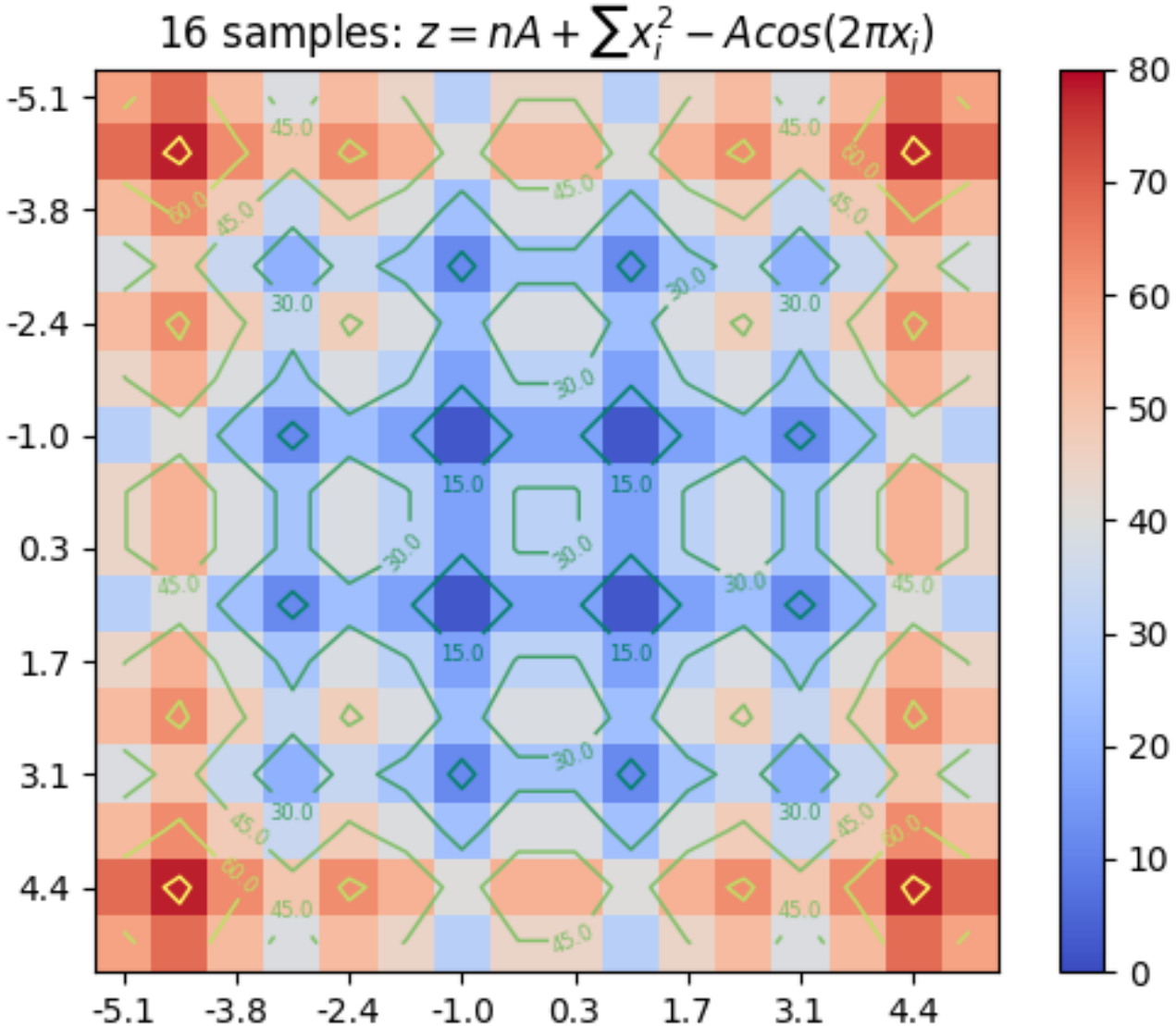
Smoothing of the landscape



Smoothing of the landscape



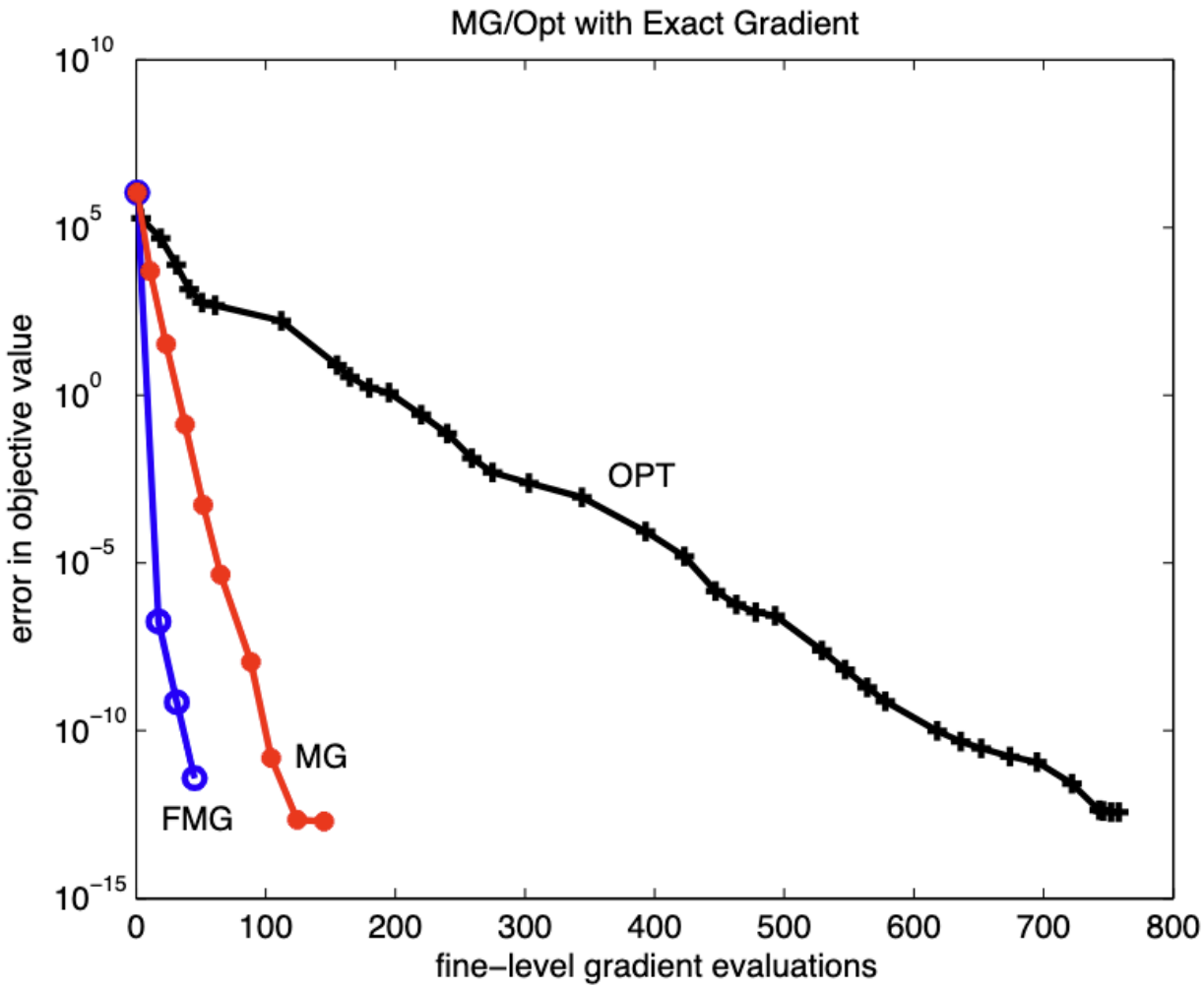
Smoothing of the landscape



Numerical results for 5 levels MG/OPT with line-search

Problem:

$$\min_a F(a) = \frac{1}{2}\int_0^\pi \left(\frac{\partial u}{\partial x_2}(x_1, 0) - \phi(x_1)\right)^2 dx_1,$$



Numerical results for 5 levels RMTR

Performance profile[3] on 17 infinite-dimensional problems involving differential operators

- AF=one-level Newton
- MR=from down to up
- MF=RMTR from up to down
- FM= RMTR from down to up

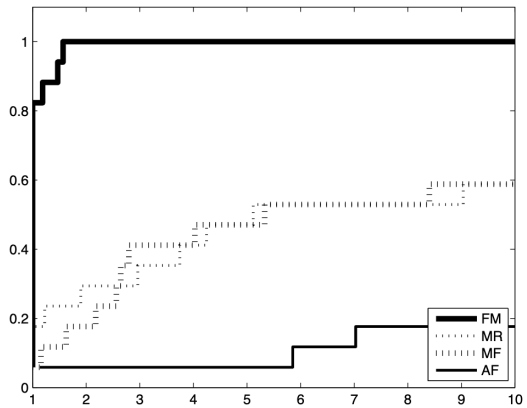


Figure 4.7: Performance profile for CPU time with variants AF, MF, MR and FM (17 test problems).

Recursive trust-region methods, S. Gratton, A. Sartenaer and Ph. L. Toint, 2008

Benchmarking Optimization Software with Performance Profiles , E. D. Dolan, J. J. Moré, 2002

[3] Performance profile $p_A(\tau)$ of an algorithm A at point τ shows the fraction of the test set for which the algorithm is able to solve within a factor of τ of the best algorithm for the given measure

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:

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

which is equivalent to the nonlinear 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**

Extension to q-order models

On high-order multilevel optimization strategies, H. Calandra, S. Gratton, E. R., X. Vasseur, 2020

We define

$$\mu^H_{q,k}(s^H) = f^H(x^H_{0,k} + s^H) + \text{corr}$$

$$\text{corr} = \sum_{i=1}^{q} \frac{1}{i!} [\mathcal{R}(\nabla^i f^h(x_k)) - \nabla^i f^H(x^H_k)] \underbrace{(s^H, \ldots, s^H)}_{i \text{ times}},$$

Example

For $q = 2$

$$\begin{aligned}\text{corr} =& (I^H_h \nabla f^h(x^h_k) - \nabla f^H(x^H_k))^T x^H_k \\ &+ \frac{1}{2}(x^H_k)^T (I^H_h \nabla^2 f^h(x^h_k) I^h_H - \nabla^2 f^H(x^H_k)) x^H_k\end{aligned}$$

Theoretical results: Assumptions

Assumption 1

Let us assume that for all levels the q-th derivative tensors of f^h are Lipschitz continuous.

Assumption 2

There exist strictly positive scalars $\kappa_{EB}, \rho > 0$ such that

$$\operatorname{dist}(x, \mathcal{X}) \leq \kappa_{EB} \|\nabla_x f(x)\|, \quad \forall x \in \mathcal{N}(\mathcal{X}, \rho),$$

where $\mathcal{X}$ is the set of second-order critical points of f, $\operatorname{dist}(x, \mathcal{X})$ denotes the distance of x to $\mathcal{X}$ and $\mathcal{N}(\mathcal{X}, \rho) = \{x \mid \operatorname{dist}(x, \mathcal{X}) \leq \rho\}$.

On the Quadratic Convergence of the Cubic Regularization Method under a Local Error Bound Condition, Yue, M.C. and Zhou, Z. and So, A.M.C., 2018: generalized to higher-order methods

Theoretical results: 1) global convergence

Theorem

Let Assumption 1 hold. Then, the sequence of iterates generated by the algorithm converges globally to a first-order stationary point.

E. G. Birgin, J. L. Gardenghi, J. M. Martinez, S. A. Santos and Ph. L. Toint, 2017: generalized to multilevel framework

Theoretical results: 2) complexity

Theorem

Let Assumption 1 hold. Let f_{low} be a lower bound on f. Then, the method requires at most

$$O(\epsilon^{-\frac{q+1}{q}})$$

iterations to achieve an iterate x_k such that $\|\nabla f(x_k)\| \leq \epsilon$

E. G. Birgin, J. L. Gardenghi, J. M. Martinez, S. A. Santos and Ph. L. Toint, 2017: $k = O(\epsilon^{-\frac{q+1}{q}})$ Complexity of standard method is maintained

Theoretical result: 3) local convergence

Theorem

Let Assumptions 1 and 2 hold. Assume that $\mathcal{L}(f(x_k))$ is bounded for some $k \geq 0$ and that it exists an accumulation point x^* such that $x^* \in \mathcal{X}$. Then, the whole sequence $\{x_k\}$ converges to x^* and it exist strictly positive constants $c \in \mathbb{R}$ and $\bar{k} \in \mathbb{N}$ such that:

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^q} \leq c, \quad \forall k \geq \bar{k}.$$