The Poisson problem: How can a simple equation be so complicated to solve efficiently ?

Clément MARADEI

Serena Team

April 9th

Outline

Simulation for the Environment

- Environmental problems are modelled by Partial Differential Equations (PDEs).
- Often, such equations can not be solved exactly.

Reliable and Efficient Numerical Algorithms

- How close is the approximate solution to the exact solution?
- How efficient is the method in terms of computational cost?

Outline

The Poisson problem

We consider the boundary value problem

Find $u : \Omega \mapsto \mathbb{R}$ such that

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

The Laplacian ($\Delta\cdot$) operator appears whenever there is diffusion.

The Poisson problem

We consider the boundary value problem

Find $u : \Omega \mapsto \mathbb{R}$ such that

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

The Laplacian ($\Delta\cdot$) operator appears whenever there is diffusion.

Introducing the weak formulation is the starting point of FEM

Find $u \in H_0^1(\Omega)$ such that

$$(\nabla u, \nabla v)_\Omega = (f, v)_\Omega \quad \forall v \in H_0^1(\Omega).$$

We do discretization from here.

Discretization

How to give numerical representation of a 2D continuous domain ?

Mesh : a Finite union of Elements.

Discretization

How to give numerical representation of a 2D continuous domain ?

Mesh : a Finite union of Elements.

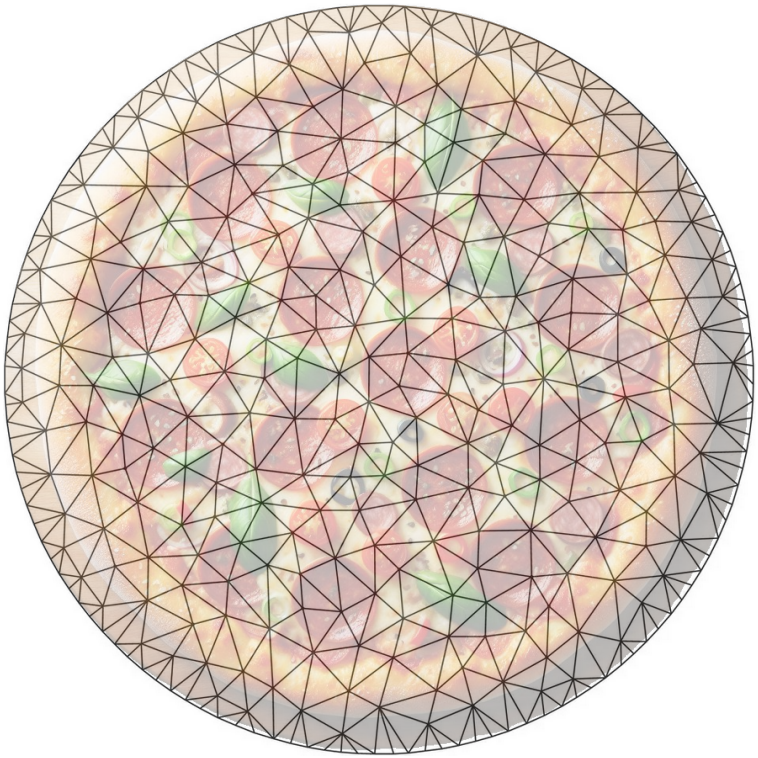
Ω_{pizza}

Discretization

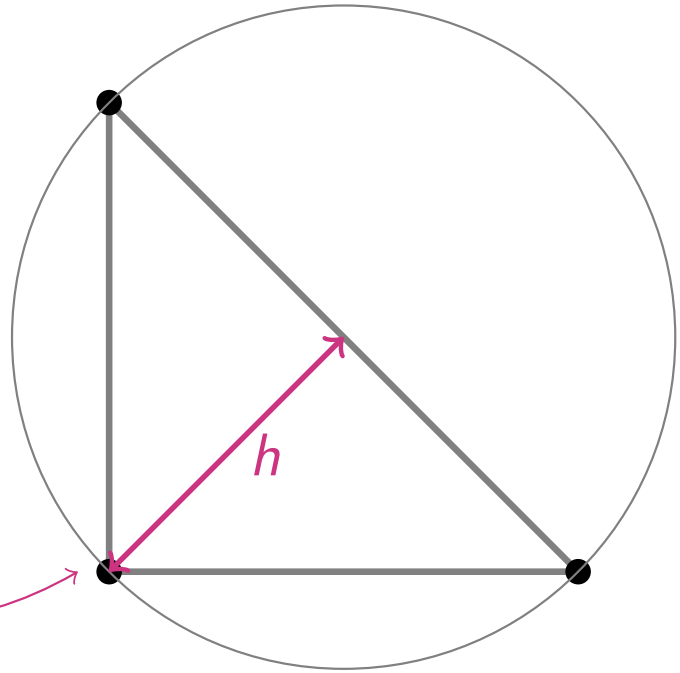
How to give numerical representation of a 2D continuous domain ?

Mesh : a Finite union of Elements.

$$\mathcal{T} = \bigcup_i K_i$$



$\mathcal{T}_{\text{pizza}}$



A mesh Element K_i

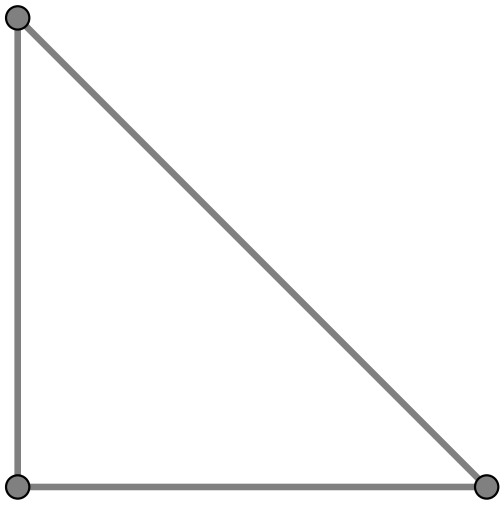
h is the mesh size, it is related to the number of element K_i in $\mathcal{T}$

Approximation

We want the value of u_h at specific points of each triangle
These function evaluations are called degrees of freedom (Dofs),
they depend on p : degree of polynomial approximation.

Approximation

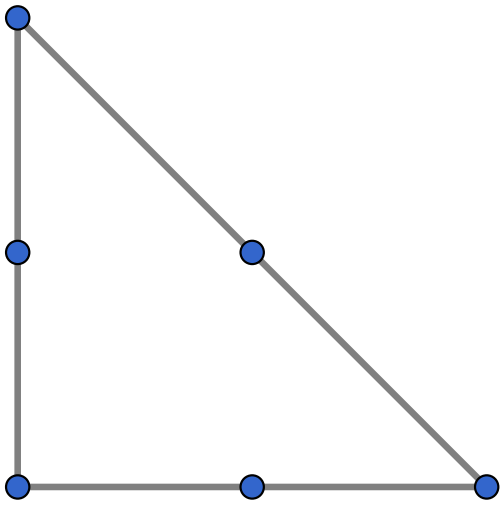
We want the value of u_h at specific points of each triangle
These function evaluations are called degrees of freedom (Dofs),
they depend on p : degree of polynomial approximation.



For $p = 1$, we have 3 Dofs on each element

Approximation

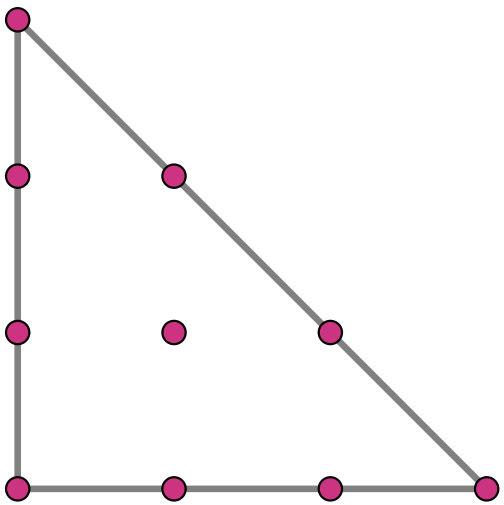
We want the value of u_h at specific points of each triangle
These function evaluations are called degrees of freedom (Dofs),
they depend on p : degree of polynomial approximation.



For $p = 1$, we have 3 Dofs on each element
For $p = 2$, we have 6 Dofs on each element

Approximation

We want the value of u_h at specific points of each triangle
These function evaluations are called degrees of freedom (Dofs),
they depend on p : degree of polynomial approximation.

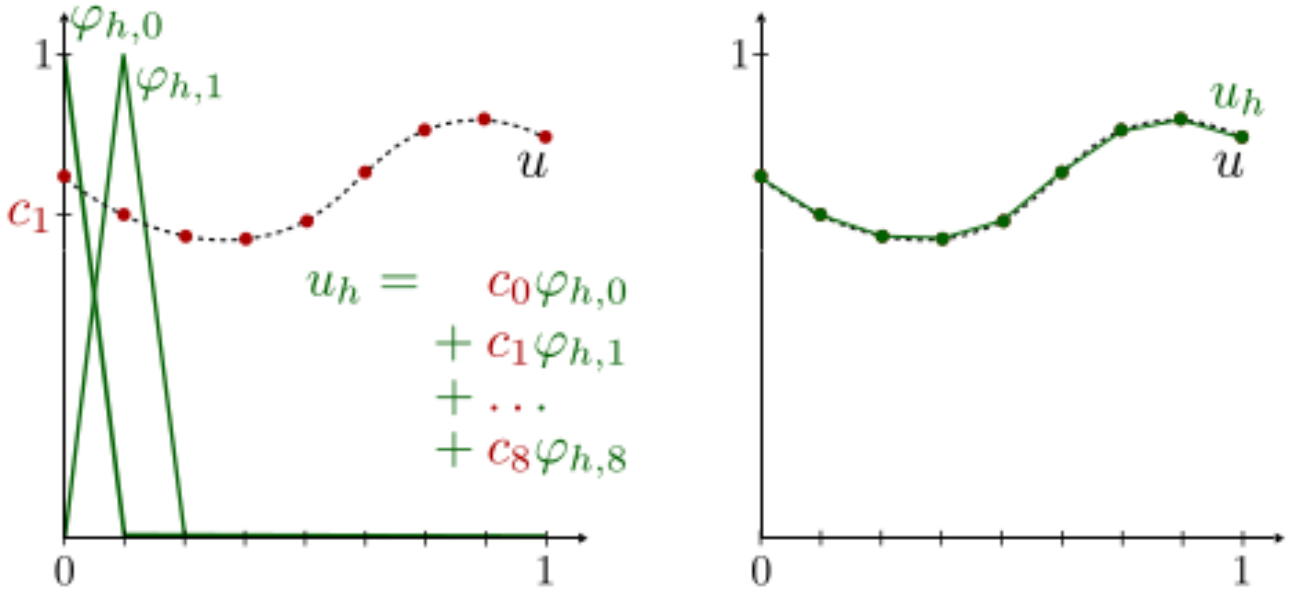


For $p = 1$, we have 3 Dofs on each element
For $p = 2$, we have 6 Dofs on each element
For $p = 3$, we have 10 Dofs on each element

Smaller h : more elements
Higher p : more Dofs per element

Finite Element space

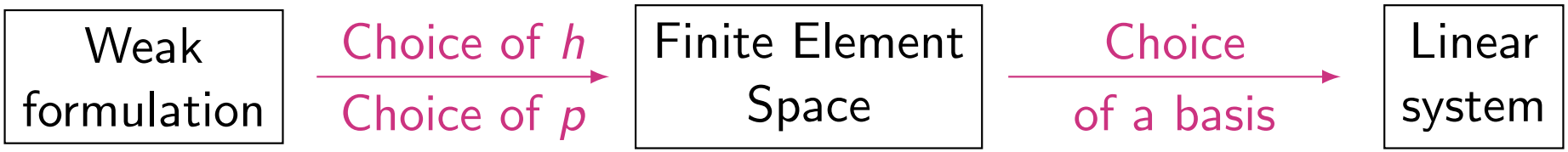
Chosing h and p allows us to build a discrete space V_h^p where the basis functions are piecewise polynomials of degree p.
We look for u_h in V_h^p and not in $H_0^1(\Omega)$.



$V_h^p = \mathrm{Span}(\phi_{h,0}, ..., \phi_{h,n})$. We are looking for $c_0, c_1, ..., c_n$.

The Finite Element approach

What is commonly done



We have to solve a linear system for $c_0, c_1, \ldots, c_n$,

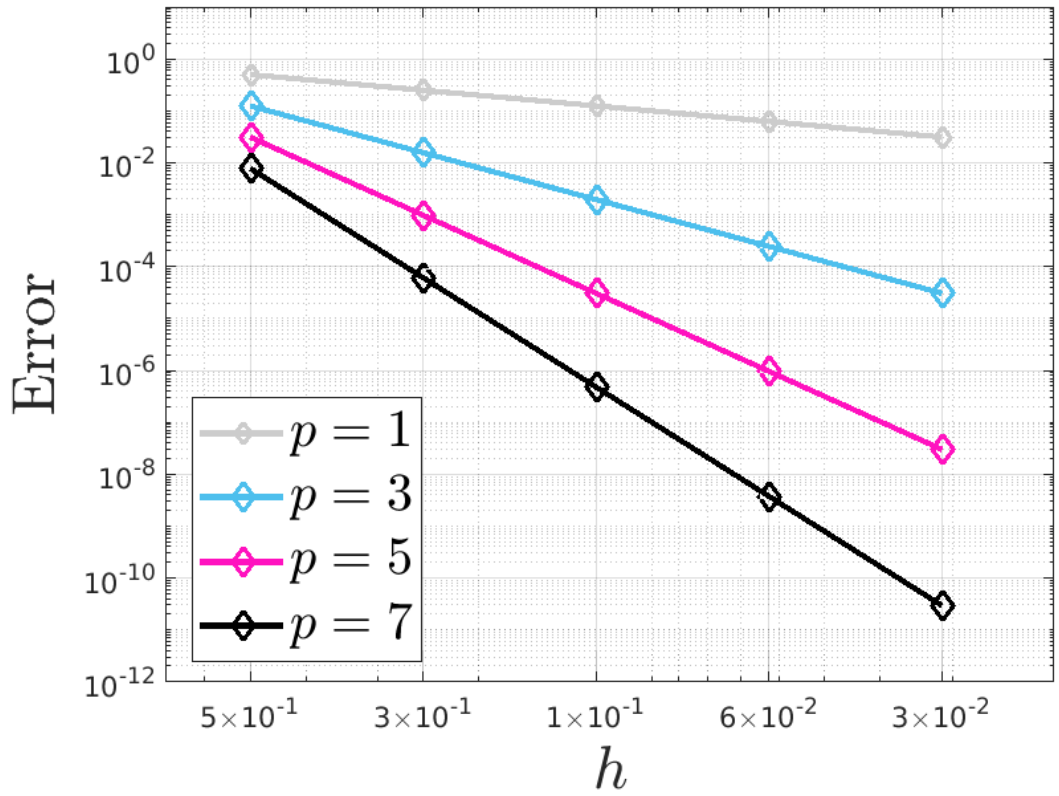
$$AU_h^p = F.$$

The size of A depends on h and p. The linear system has to be solved independently.

How to chose h and p ?

Exponential convergence of the hp FEM

$$\|\nabla(u - u_h^p)\| \leq C h^{\min\{p, s-1\}} \text{ if } u \in H^s(\Omega), s \geq 1.$$



High order FEM

How to chose h and p ?

Exponential convergence of the hp FEM

$$\|\nabla(u - u_h^p)\| \leq C h^{\min\{p, s-1\}} \text{ if } u \in H^s(\Omega), s \geq 1.$$

Challenges

- The matrix becomes less sparse for high order p.
- The matrix becomes more ill-conditioned for high order p.

The system $AU_h^p = F$ is hard to solve for high p

Iterative solvers explode in terms of iterations

Iterative solvers

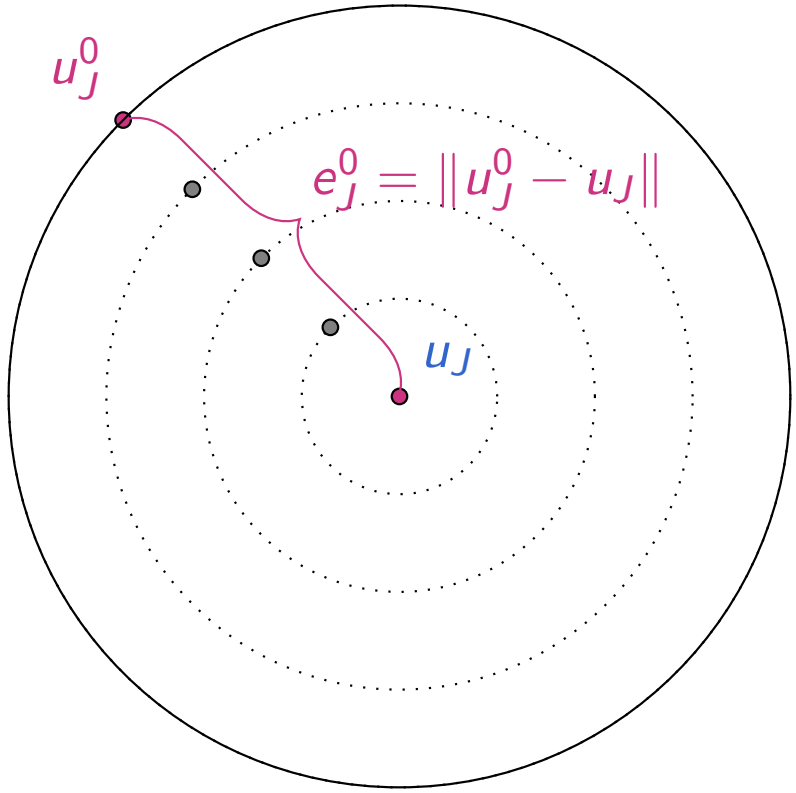
Let u_J be the exact solution of $Au_J = F$ and u_J^i the iterates of the method.

Starting from u_J^0

n iterations

Until reaching u_J^n

With $e_J^n = \|u_J^n - u_J\| \leq \epsilon$



Any iterative solver can be seen as a strategy to decrease the algebraic error

Iterative solvers

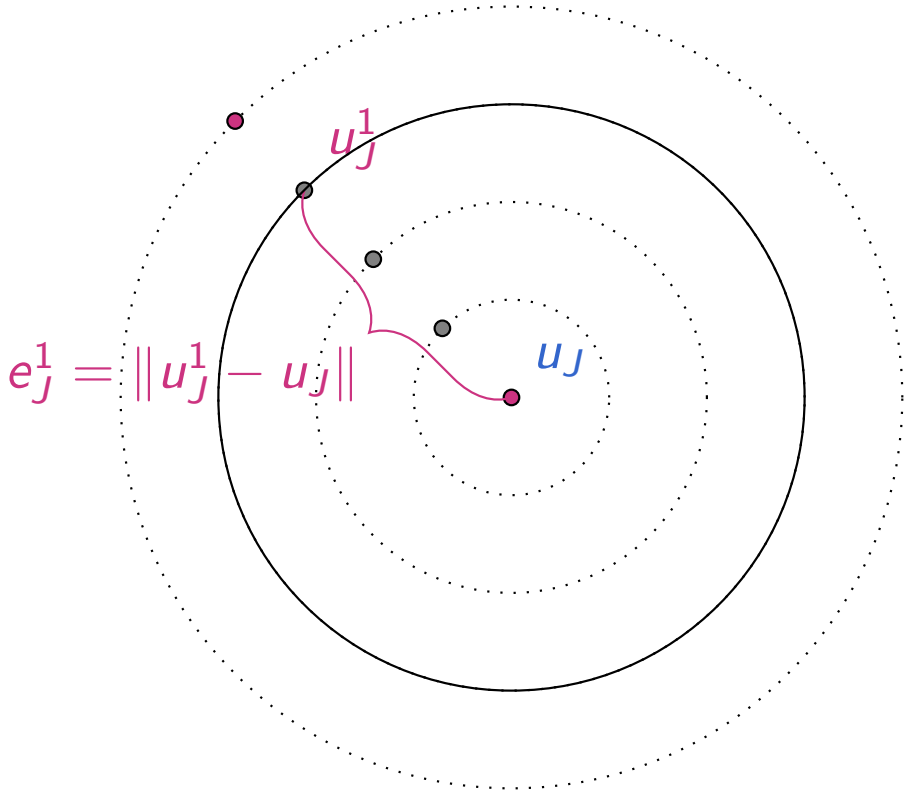
Let u_J be the exact solution of $Au_J = F$ and u_J^i the iterates of the method.

Starting from u_J^0

n iterations

Until reaching u_J^n

With $e_J^n = \|u_J^n - u_J\| \leq \epsilon$



Any iterative solver can be seen as a strategy to decrease the algebraic error

Iterative solvers

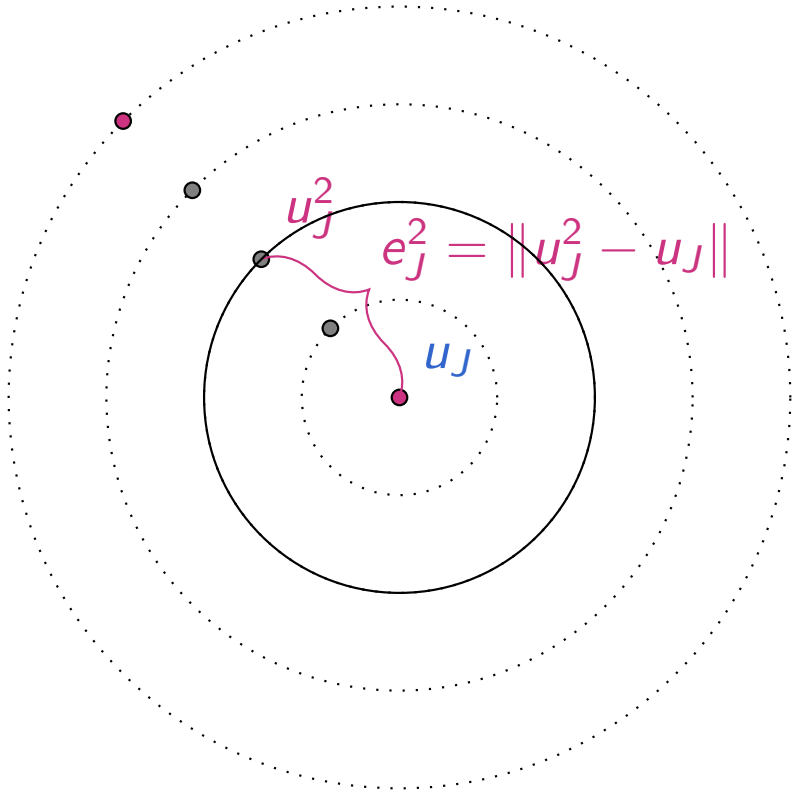
Let u_J be the exact solution of $Au_J = F$ and u_J^i the iterates of the method.

Starting from u_J^0

n iterations

Until reaching u_J^n

With $e_J^n = \|u_J^n - u_J\| \leq \epsilon$



Any iterative solver can be seen as a strategy to decrease the algebraic error

Iterative solvers

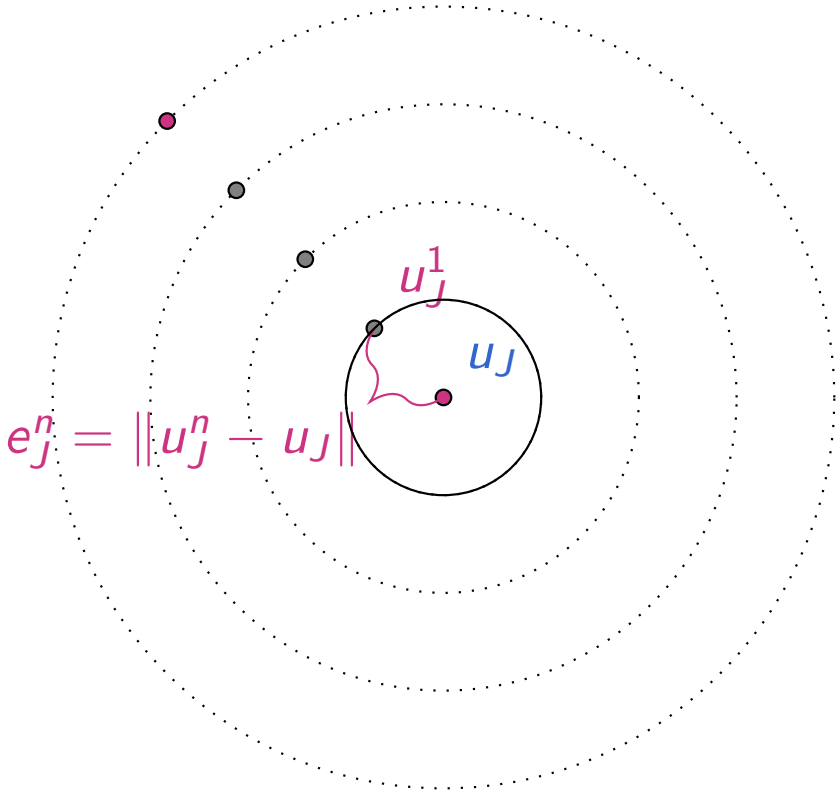
Let u_J be the exact solution of $Au_J = F$ and u_J^i the iterates of the method.

Starting from u_J^0

n iterations

Until reaching u_J^n

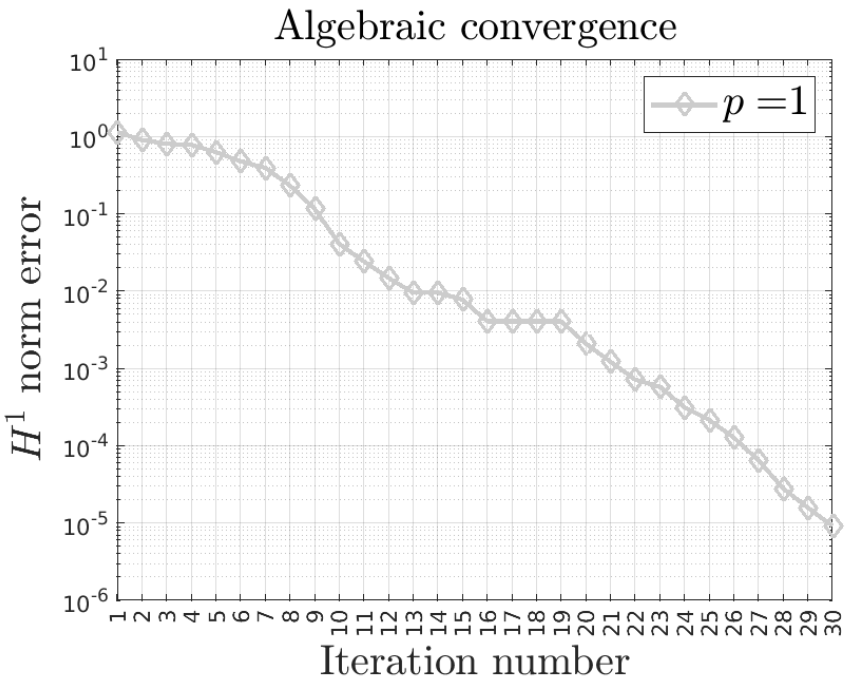
With $e_J^n = \|u_J^n - u_J\| \leq \epsilon$

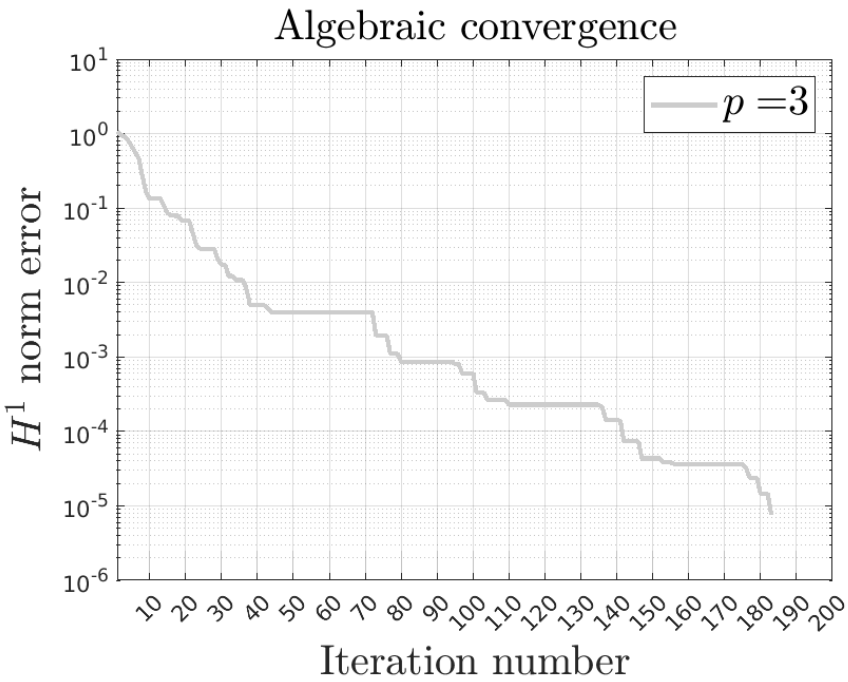


Any iterative solver can be seen as a strategy to decrease the algebraic error

Algebraic convergence of CG

Conjugate gradient for the linear system corresponding to the Poisson equation





Tolerance: $\epsilon = 10^{-5}$

From 30 iterations with $p = 1$ to 180 iterations with $p = 3$.

CG is not p-robust

Outline

A multigrid FEM solver

Ingredients for the solver:

- A hierarchy of meshes
- Domain decomposition
- Orthogonal decomposition of the error

Leading to the following properties:

- The method is parallel by design
- The solver is memory efficient
- The solver is p-robust

Where h and p are arbitrary parameters

How to decrease the algebraic error ?

The algebraic error is defined by $e_J^i = \|\nabla(u_J^i - u_J)\|$, with the H^1 semi norm.

How to decrease this quantity ?

We introduce $\rho_{J,\text{alg}}^i$ a correction to go from u_J^i to u_J^{i+1}

In the multigrid setting, $\rho_{J,\text{alg}}^i$ is computed by going through a hierarchy of meshes.

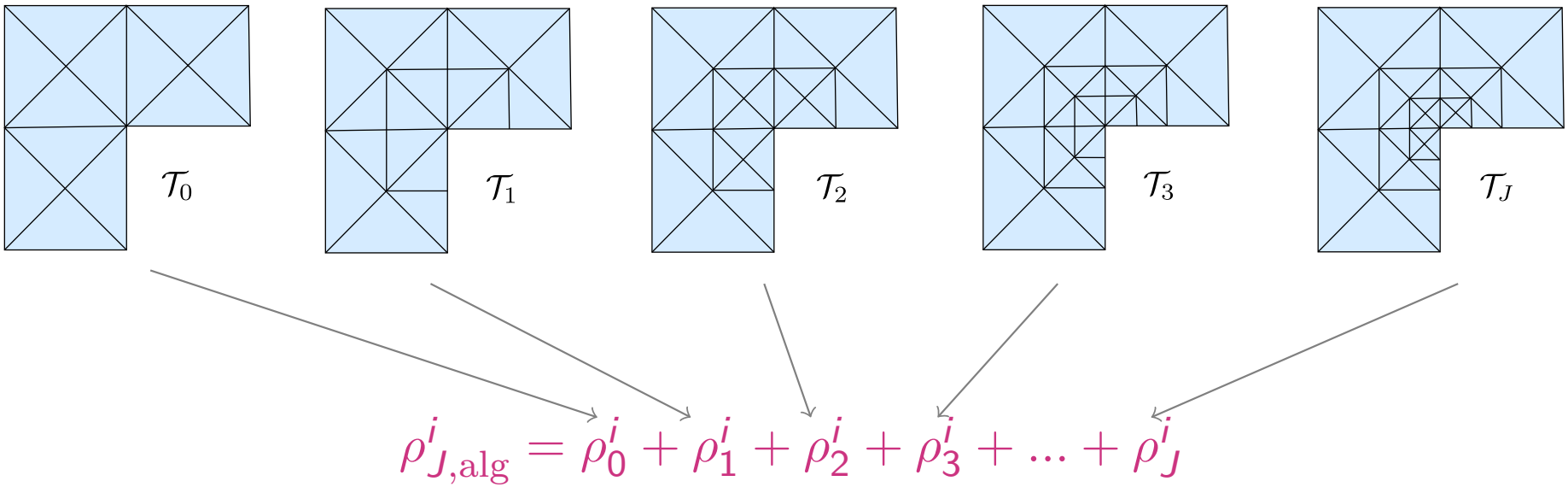
$$\rho_{J,\text{alg}}^i = \sum_{j=0}^{J} \rho_j^i$$

Where ρ_j^i are the level-wise corrections.

Hierarchy of meshes

Multigrid : Multiple Meshes

We work with an initial mesh $\mathcal{T}_0$ and its refinements $\mathcal{T}_j$.



The hierarchy of meshes does not have to be uniform

Low-cost information acquisition on coarser meshes

How to compute the correction

1. Coarse solve on $\mathcal{T}_0$ for ρ_0^i

$$(\nabla \rho_0^i, \nabla v_0) = (f, v_0) - (\nabla u_J^i, \nabla v_0) \quad \forall v \in \mathcal{P}^0 \cap H_0^1(\mathcal{T}_0)$$

2. Level Solve for ρ_j^i

For $\mathcal{T}_1$ to $\mathcal{T}_\mathcal{J}$:

- Decompose $\mathcal{T}_j$ into patches

Local solve for $\rho_{j,a}^i$

We obtain $\rho_j^i = \sum_a \rho_{j,a}^i$, the level-wise correction

- Line Search : $\rho_j^i \longrightarrow \lambda_j^i$,

$$u_j^i = u_{j-1}^i + \lambda_j^i \rho_j$$

End For

3. Update of the solution: $u_J^{i+1} = u_J^i$

Domain decomposition

Why are we doing domain decomposition?

We need to compute ρ_j^i on each mesh $\mathcal{T}_j$, to get the correction $\rho^i_{J,\text{alg}}$.

We want to go from a single global problem to multiple local problems.

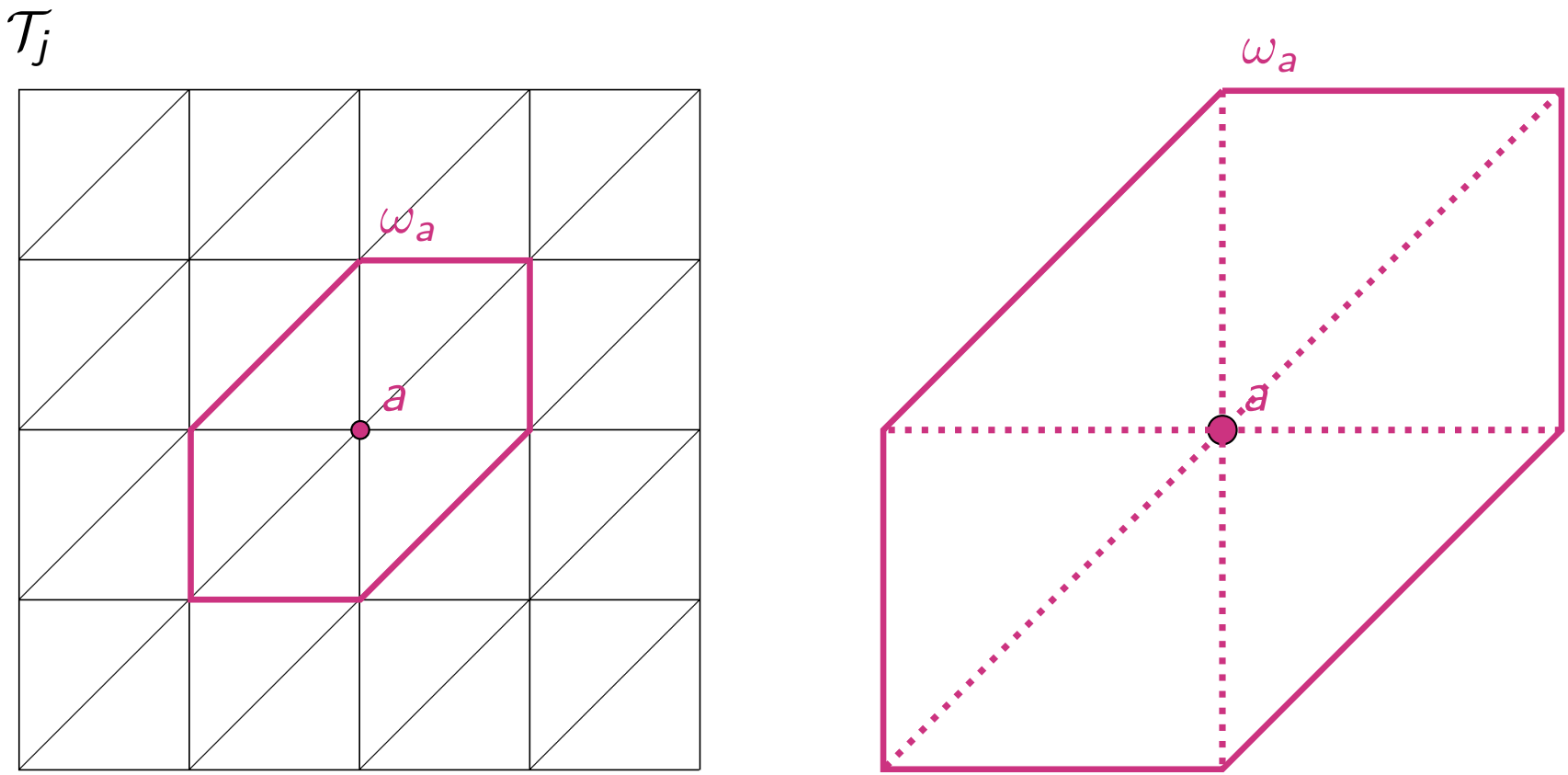
The reasons :

- More memory efficient
- More time efficient
- Parallelizable!

It is possible from theory using the so-called *Partition of unity*.

But... What is a patch ?

Example of a patch at a mesh level j.



a is the vertex patch

ω_a is the patch subdomain

Global solve too costly

Solving for ρ_j^i ?

$$\forall v \in V_j^p, \quad \overbrace{(\nabla \rho_j^i \, , \nabla v)_{\mathcal{T}_j}}^{\text{Unknown}} = \overbrace{(f, v)_{\mathcal{T}_j} - (\nabla u_{j-1}^i \, , \nabla v)_{\mathcal{T}_j}}^{\text{Known}}$$

Choice of basis

$$K_j x_j = F_j^1 - F_j^2$$

The vector x_j contains the coefficients of ρ_j^i in the discrete space.

The matrix K_j is assembled on the mesh $\mathcal{T}_j$... It is global!

Solving this linear system is really costly, especially for the mesh $\mathcal{T}_J$.

Patch problems

We have the patch decomposition $\rho_j^i = \sum_a \rho_{j,a}^i$.

In order to get $\rho_{j,a}^i$, we solve :

$$\forall v \in V_a^p, \quad \overbrace{(\nabla \rho_{j,a}^i, \nabla v)_{\omega_a}}^{\text{Unknown}} = \overbrace{(f, v)_{\omega_a} - (\nabla u_{j-1}^i, \nabla v)_{\omega_a}}^{\text{Known}}$$

Choise of basis

$$K_a x_a = F_a^1 - F_a^2$$

The vector x_a contains the coefficients of ρ_a^i in the discrete space.

The matrix K_a is assembled on the mesh ω_a... It is local!

This results in

- $\texttt{size}(K_a) << \texttt{size}(K_j)$
- The computation of $\rho_{j,a}^i$can be parallelized

Line Search

How can we further improve the convergence ?

At each level ρ_j^i can be seen as the descent direction.

In order to get closer to u_J at each level step j, we can introduce λ_j^i solution of:

$$\lambda_j^i := \text{argmin}_{\lambda \in \mathbb{R}} \|\nabla(u_J - (u_{J,j-1} + \lambda \rho_j^i))\|^2.$$

$$\text{Minimization of } \lambda \mapsto f(\lambda) \;\Big|\; f'(\lambda) = 0$$

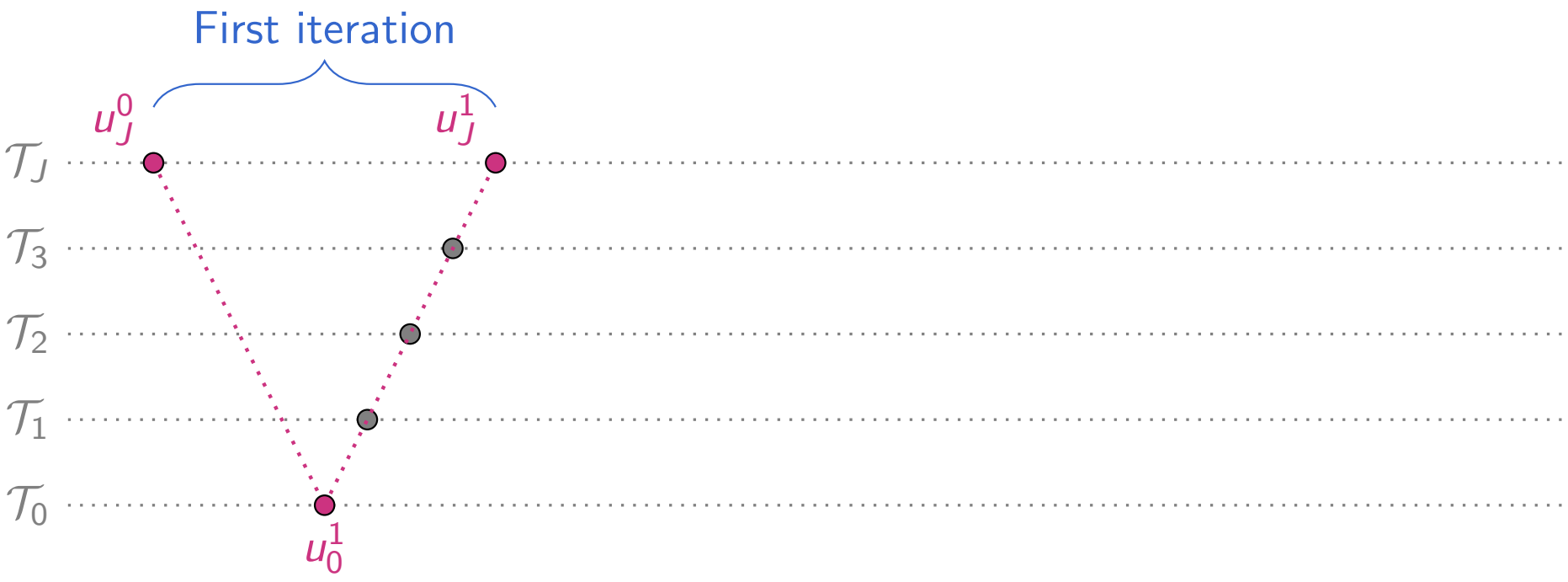
$$\lambda_j^i = \frac{(f, \rho_j^i) - (\nabla u_{J,j-1}, \nabla \rho_j^i)}{\|\nabla \rho_j^i\|^2}$$

We update on level j :

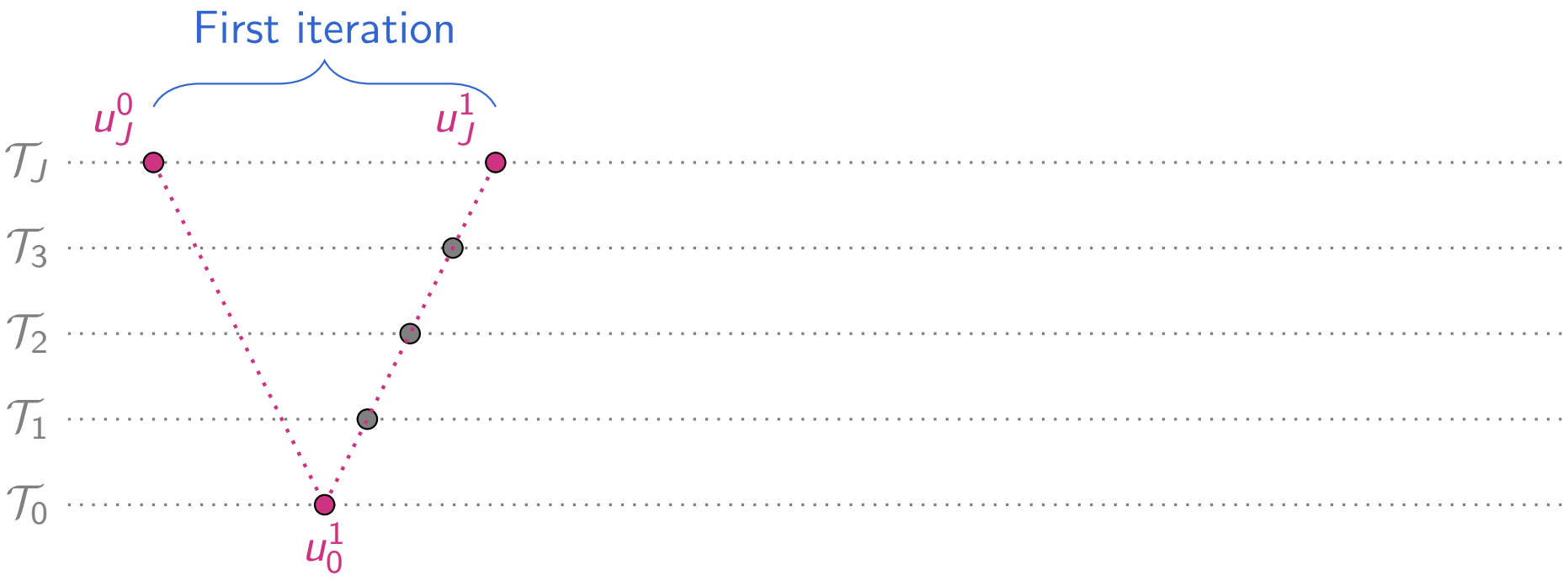
$$u_j = u_{j-1}^i + \lambda_j^i \rho_j^i$$

Outline

The Incredible V-cycle



The Incredible V-cycle

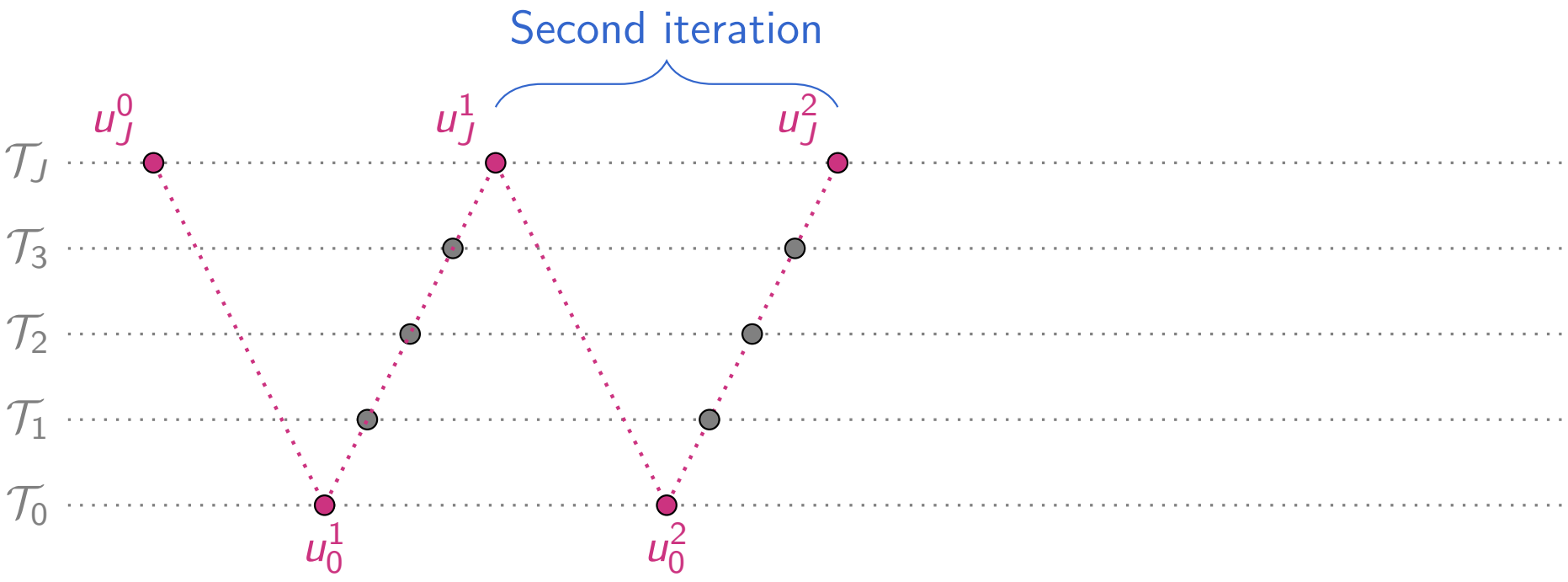


- On $\mathcal{T}_0$: global coarse solve for ρ_0^1
- On $\mathcal{T}_1$: Local patch solves to construct $\rho_1^1 = \sum_a \rho_{1,a}^1$

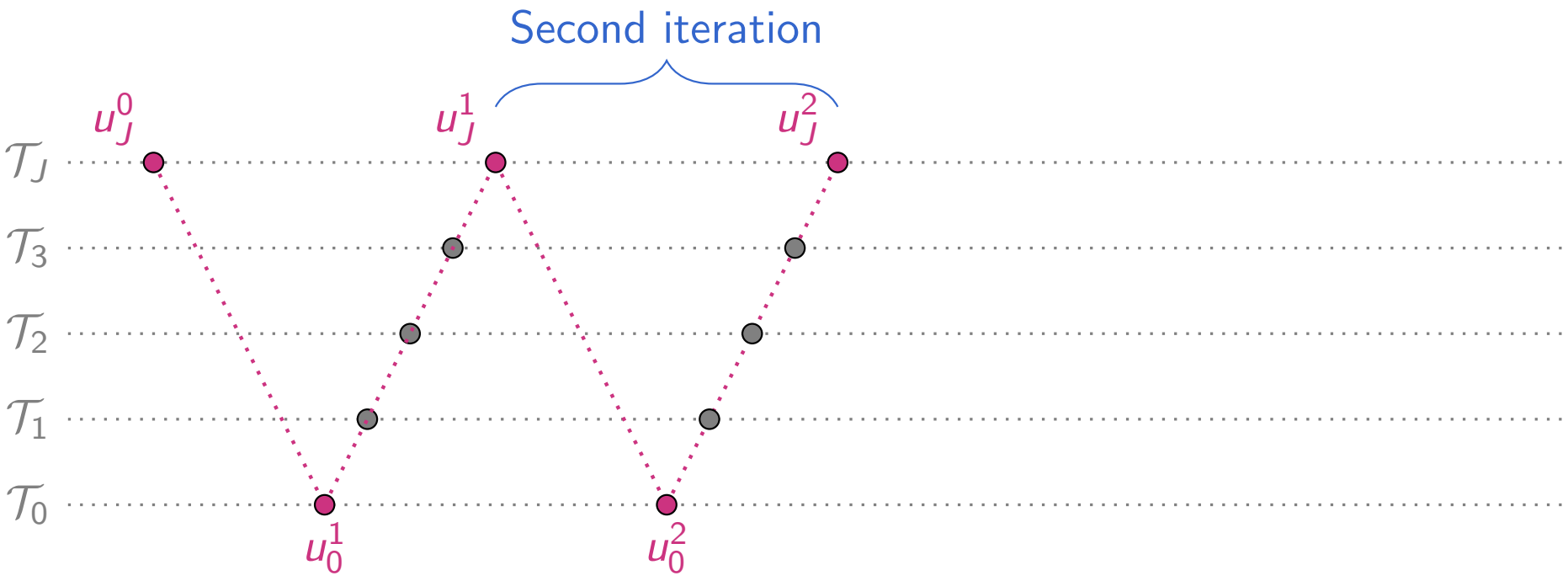
  ⋮
- On $\mathcal{T}_J$: Local patch solves to construct $\rho_J^1 = \sum_a \rho_{J,a}^1$

If $e_J^1 \leq \epsilon$, we continue.

The Incredible V-cycle



The Incredible V-cycle

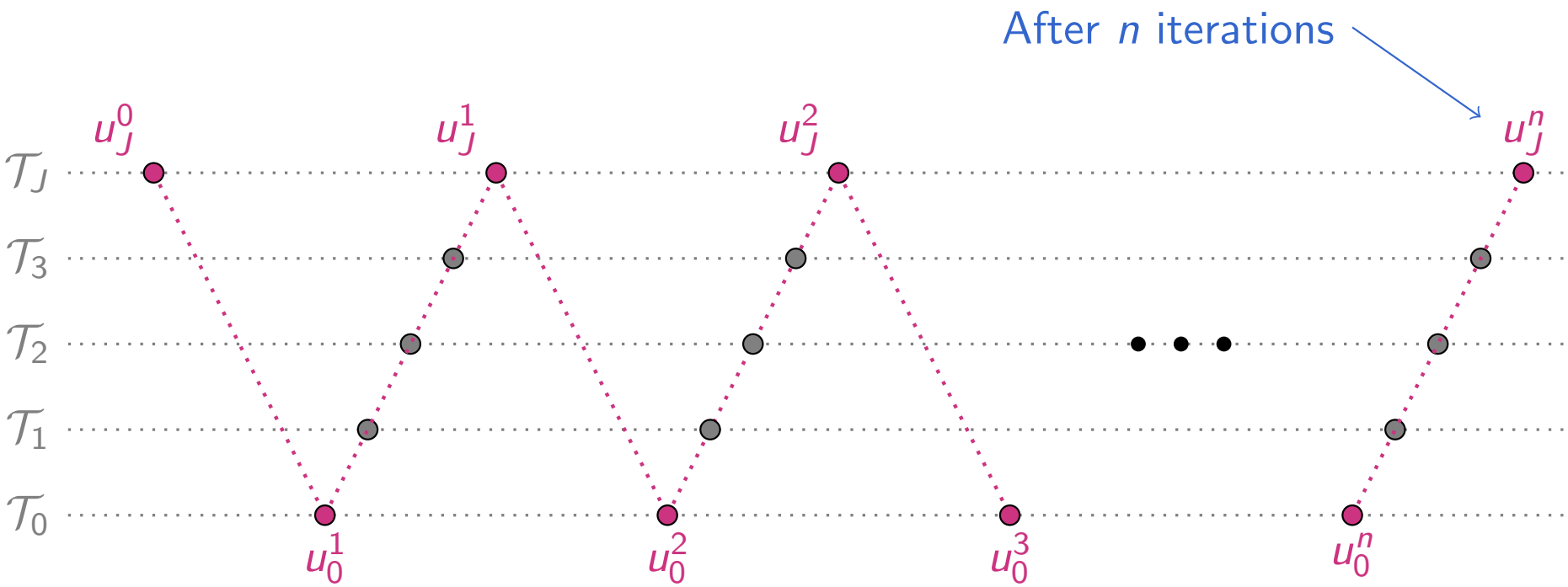


- On $\mathcal{T}_0$: global coarse solve for ρ_0^2
- On $\mathcal{T}_1$: Local patch solves to construct $\rho_1^2 = \sum_a \rho_{1,a}^2$

  ⋮
- On $\mathcal{T}_J$: Local patch solves to construct $\rho_J^2 = \sum_a \rho_{J,a}^2$

If $e_J^2 \leq \epsilon$, we still continue...

The Incredible V-cycle



We have performed nJ correction on u_J^0

When the desired tolerance is reached, the solver stops.

We obtain u_J^n the final approximation of u_J

Properties

We have the following orthogonal decomposition :

$$\left\|\nabla\left(u_J - u_J^{i+1}\right)\right\|^2 = \left\|\nabla\left(u_J - u_J^{i}\right)\right\|^2 - \sum_{j=0}^{J}\left(\lambda_j^i \left\|\nabla \rho_j^i\right\|\right)^2$$

The error at iteration $i+1$ is the error at iteration i minus a positive and computable quantity

Properties

We have the following orthogonal decomposition :

$$\left\|\nabla\left(u_J - u_J^{i+1}\right)\right\|^2 = \left\|\nabla\left(u_J - u_J^{i}\right)\right\|^2 - \sum_{j=0}^{J}\left(\lambda_j^i \left\|\nabla \rho_j^i\right\|\right)^2$$

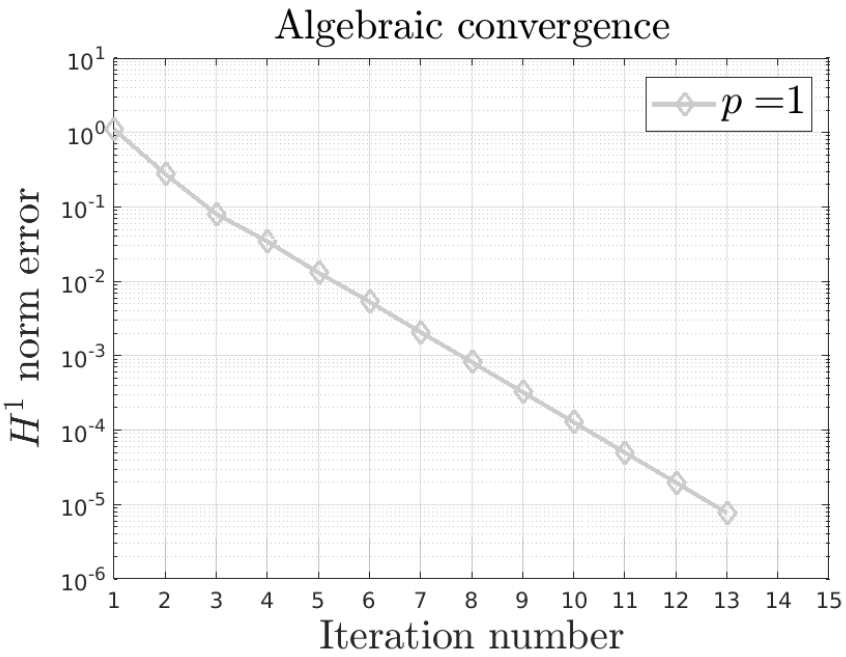
The error at iteration $i+1$ is the error at iteration i minus a positive and computable quantity

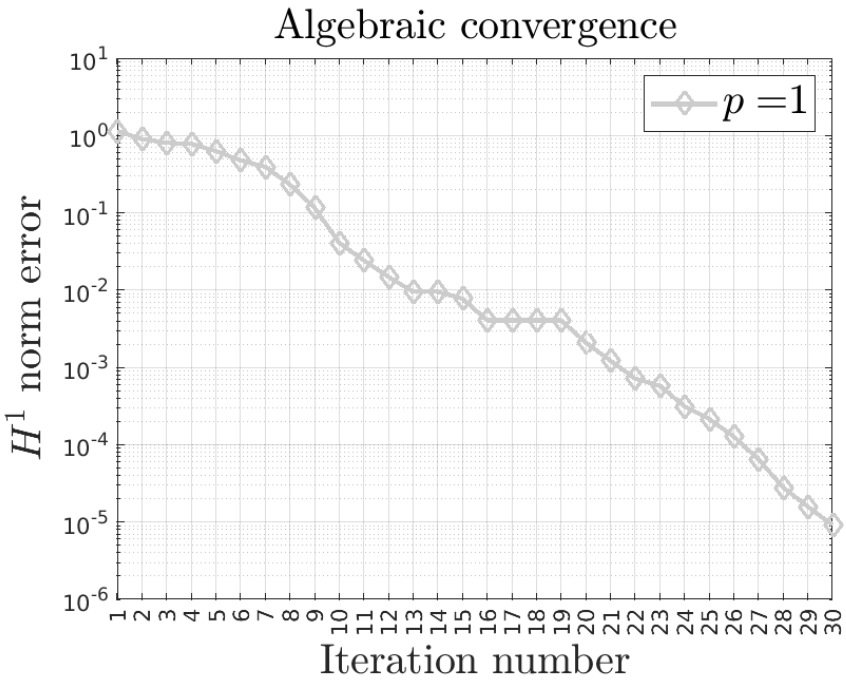
And the $p-$robustness property

$$\|\nabla(u_J - u_J^{i+1})\| \leq \alpha \|\nabla\left(u_J - u_J^{i}\right)\|$$

The contraction factor is given by $\alpha \in]0,1[$. It doesn't depend on p

p-robustness

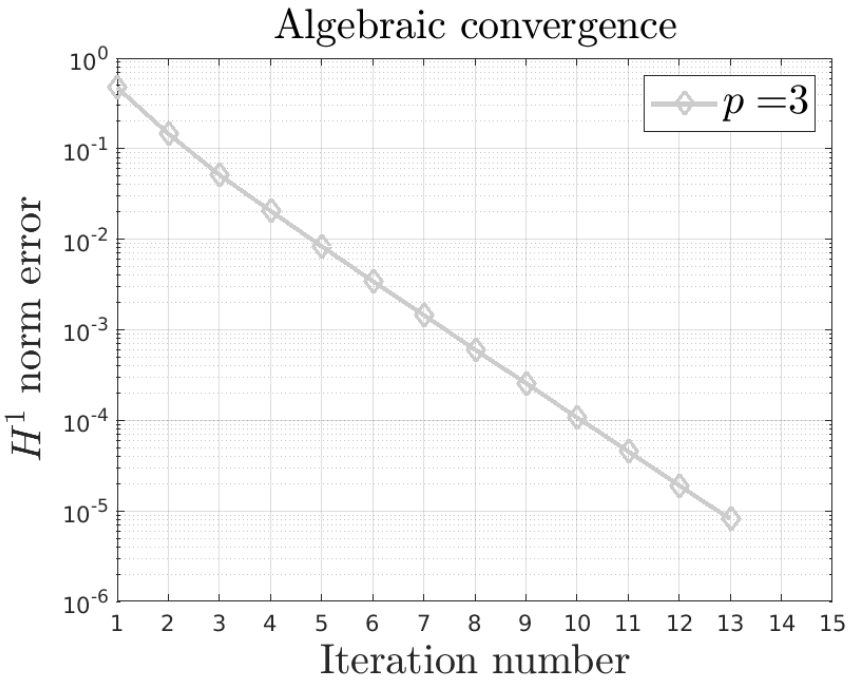


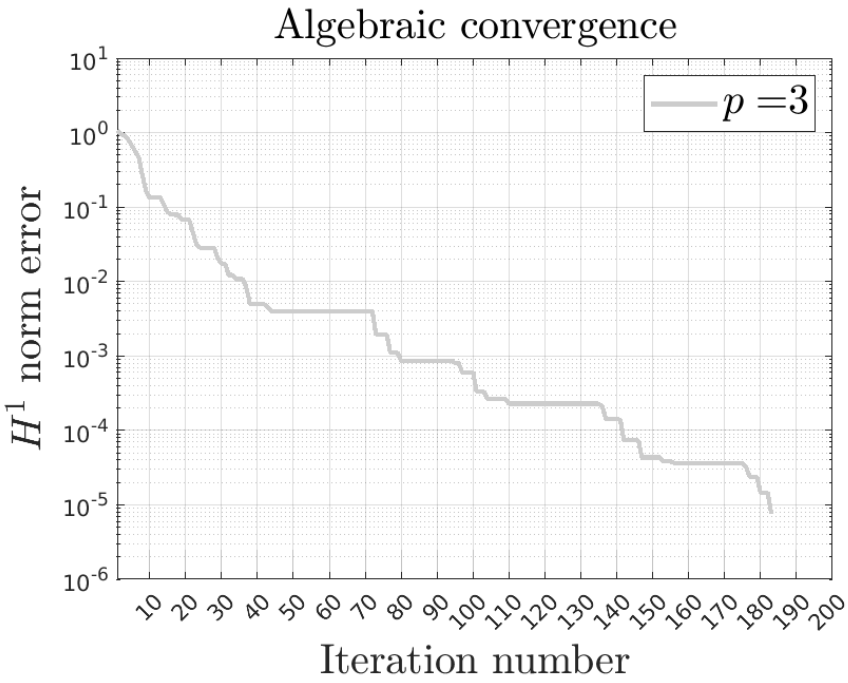


Multigrid : 13 iterations and **CG : 30 iterations**

Tolerance: $\epsilon = 10^{-5}$

p-robustness

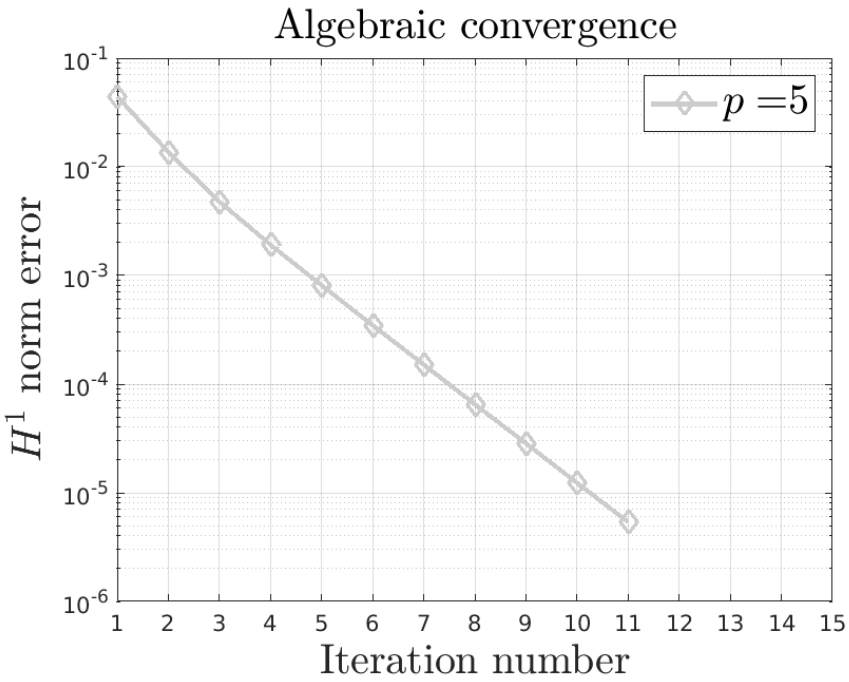


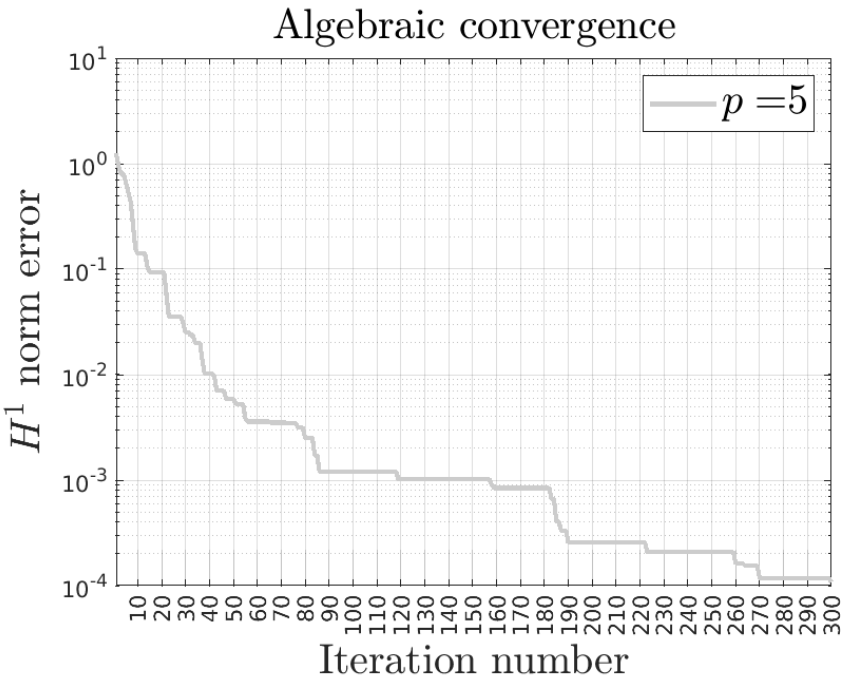


Multigrid : 13 iterations and **CG : 182 iterations**

Tolerance: $\epsilon = 10^{-5}$

p-robustness

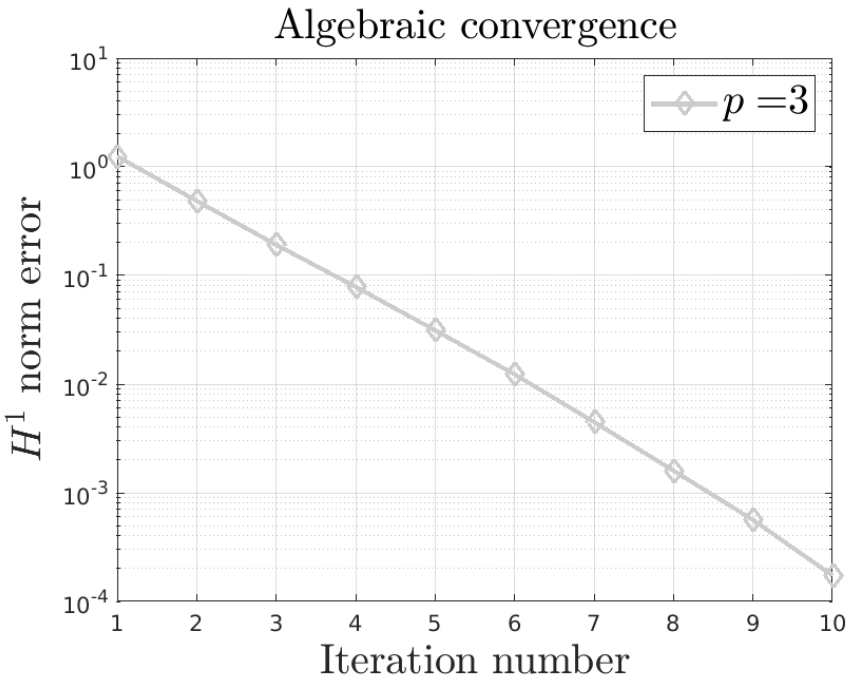


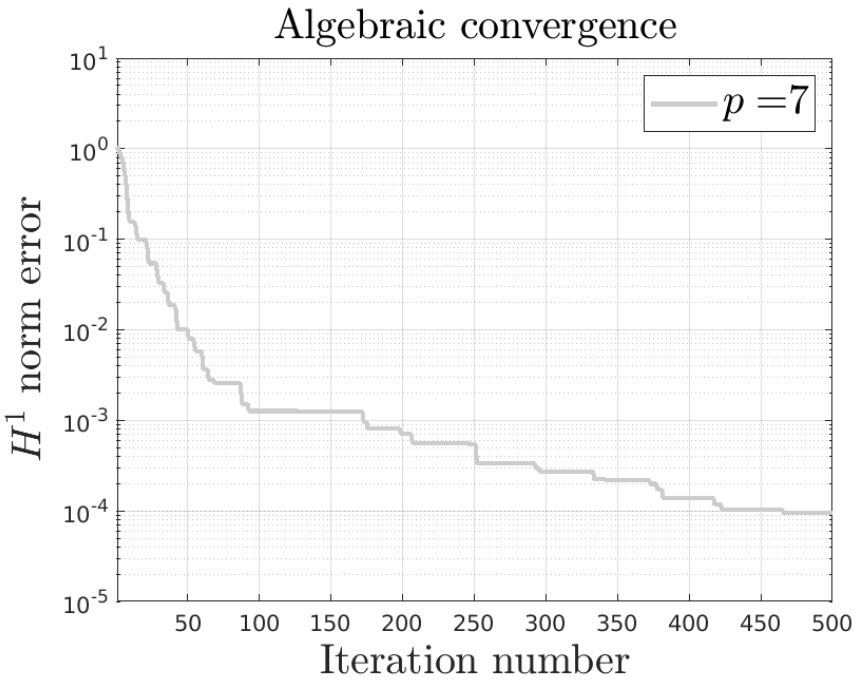


Multigrild : 11 iterations and **CG : + 300 iterations**

Tolerance: $\epsilon = 10^{-5}$

p-robustness





Multrigrid : 5 iterations and **CG : + 500 iterations**

Tolerance: $\epsilon = 10^{-5}$

Future work

While keeping the p-robustness property

- Localize even more (smaller problem than patch)
- Build a solver for other type of Finite Elements Methods
- Build a solver for other problems

Conclusion

Thank you for your attention!