



Advanced Domain Decomposition Methods

Parallel Schwarz Preconditioning and an Introduction to FROSch

Alexander Heinlein¹

DCSE Summerschool: Numerical Linear Algebra on High Performance Computers, TU Delft, June 5-9, 2023

¹TU Delft

Contents

1. Classical Schwarz algorithms
2. Extending the ideas to linear preconditioning
3. A parallel Schwarz domain decomposition solver package:
FROSCH (Fast and Robust Overlapping Schwarz)
4. Exercises

Part I — Classical Schwarz Domain Decomposition Methods

1. Literature on Domain Decomposition Methods

2. The Alternating Schwarz Algorithm

3. The Parallel Schwarz Algorithm

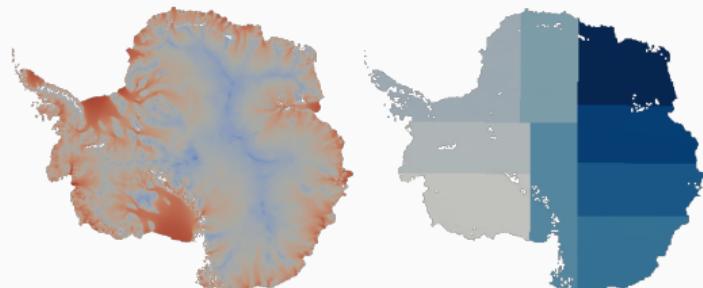
4. Comparison of the two Methods

5. Effect of the Size of the Overlap

1 Literature on Domain Decomposition Methods

-  Alfio Quarteroni and Alberto Valli
Domain decomposition methods for partial differential equations
Oxford University Press, 1999
-  Barry Smith, Petter Bjorstad, and William Gropp
Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations
Cambridge University Press, 2004
-  Andrea Toselli, and Olof Widlund
Domain decomposition methods-algorithms and theory.
Springer Science & Business Media, 2006
-  Victorita Dolean, Pierre Jolivet, Frédéric Nataf
An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation
Society for Industrial and Applied Mathematics, 2016

Domain Decomposition Methods

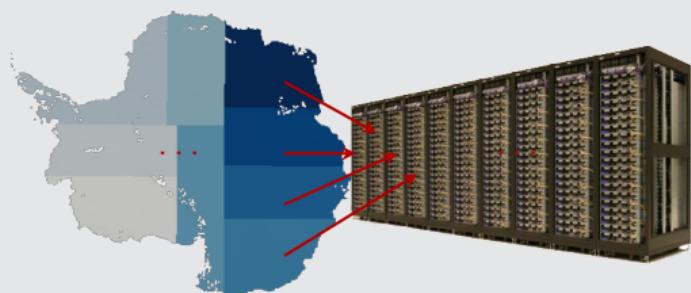


Graphics based on Heinlein, Perego, Rajamanickam (2022)

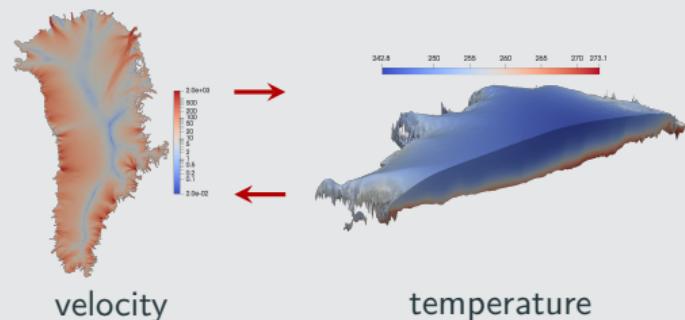
Idea

Decomposition of a large **global problem** into smaller **local problems**.

Parallel solvers



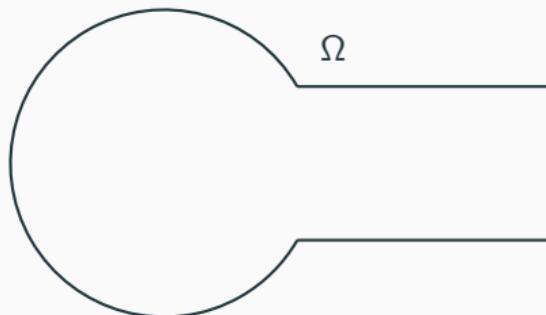
Coupled problems



2 The Alternating Schwarz Algorithm

Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with nonsmooth boundaries**.
- The **regions are constructed recursively** by forming unions of pairs of regions **starting with “simple” regions** for which existence can be established by more elementary means.
- At the core of Schwarz's work is a proof that **this iterative method converges in the maximum norm at a geometric rate**.



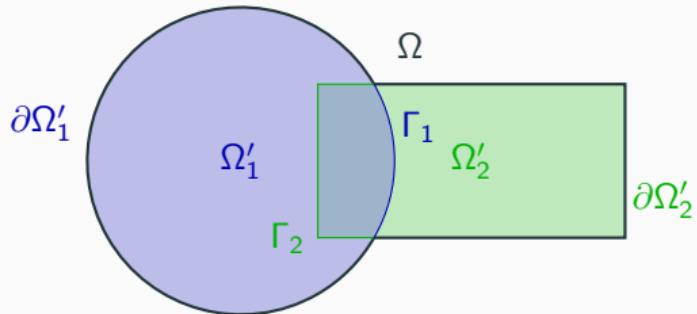
Classical “doorknob” geometry

Overlapping domain decomposition

We solve the Poisson equation

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

on the classical “doorknob” geometry.



The **alternating Schwarz iteration** corresponds to **solving alternately solving the local problems**

$$(D_1) \left\{ \begin{array}{lcl} -\Delta u^{n+1/2} & = & f \quad \text{in } \Omega'_1, \\ u^{n+1/2} & = & u^n \quad \text{auf } \Gamma_1 \\ u^{n+1/2} & = & u^n \quad \text{on } \Omega \setminus \overline{\Omega'_1} \end{array} \right.$$

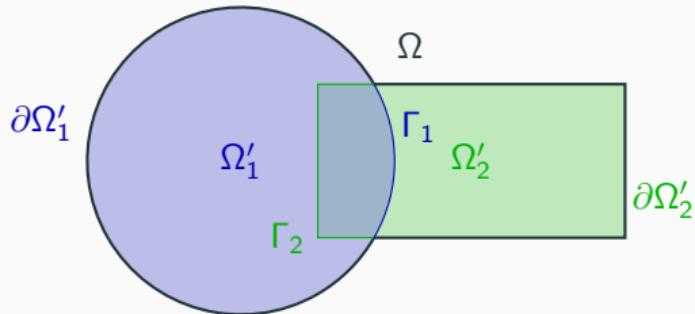
$$(D_2) \left\{ \begin{array}{lcl} -u^{n+1''} & = & f \quad \text{in } \Omega'_2, \\ u^{n+1} & = & u^{n+1/2} \quad \text{auf } \Gamma_2 \\ u^{n+1} & = & u^{n+1/2} \quad \text{on } \Omega \setminus \overline{\Omega'_2} \end{array} \right.$$

Overlapping domain decomposition

We solve the Poisson equation

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

on the classical “doorknob” geometry.



The **alternating Schwarz iteration** corresponds to **solving alternately solving the local problems**

$$(D_1) \left\{ \begin{array}{lcl} -\Delta u^{n+1/2} & = & f & \text{in } \Omega'_1, \\ u^{n+1/2} & = & u^n & \text{auf } \Gamma_1 \\ u^{n+1/2} & = & u^n & \text{on } \Omega \setminus \overline{\Omega'_1} \end{array} \right.$$

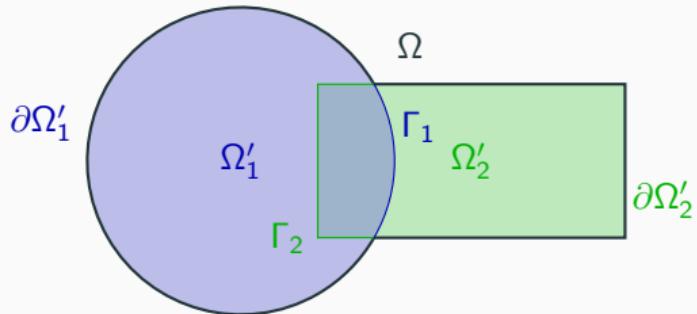
$$(D_2) \left\{ \begin{array}{lcl} -u^{n+1''} & = & f & \text{in } \Omega'_2, \\ u^{n+1} & = & u^{n+1/2} & \text{auf } \Gamma_2 \\ u^{n+1} & = & u^{n+1/2} & \text{on } \Omega \setminus \overline{\Omega'_2} \end{array} \right.$$

Overlapping domain decomposition

We solve the Poisson equation

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

on the classical “doorknob” geometry.



The **alternating Schwarz iteration** corresponds to **solving alternately solving the local problems**

$$(D_1) \left\{ \begin{array}{lcl} -\Delta u^{n+1/2} & = & f \quad \text{in } \Omega'_1, \\ u^{n+1/2} & = & u^n \quad \text{auf } \Gamma_1 \\ u^{n+1/2} & = & u^n \quad \text{on } \Omega \setminus \overline{\Omega'_1} \end{array} \right.$$

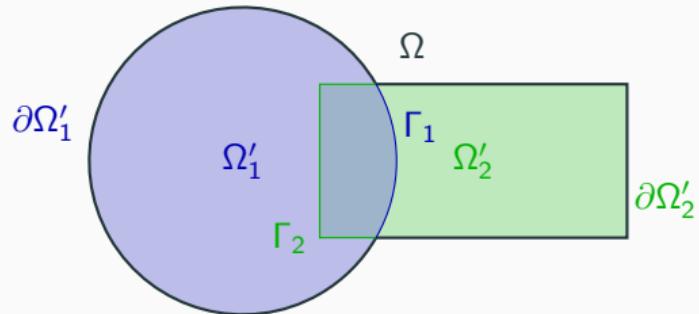
$$(D_2) \left\{ \begin{array}{lcl} -u^{n+1''} & = & f \quad \text{in } \Omega'_2, \\ u^{n+1} & = & u^{n+1/2} \quad \text{auf } \Gamma_2 \\ u^{n+1} & = & u^{n+1/2} \quad \text{on } \Omega \setminus \overline{\Omega'_2} \end{array} \right.$$

Overlapping domain decomposition

We solve the Poisson equation

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

on the classical “doorknob” geometry.

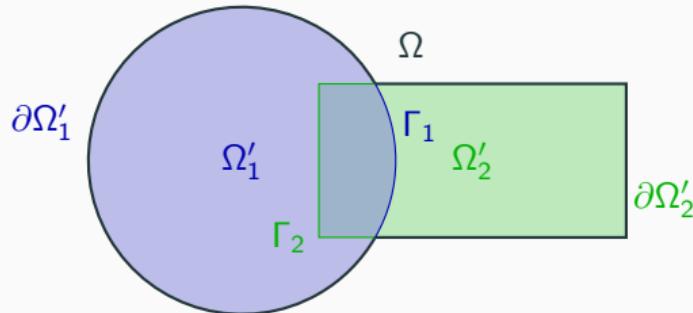


The **alternating Schwarz iteration** corresponds to **solving alternately solving the local problems**

$$(D_1) \left\{ \begin{array}{lcl} -\Delta u^{n+1/2} & = & f \quad \text{in } \Omega'_1, \\ u^{n+1/2} & = & u^n \quad \text{auf } \Gamma_1 \\ u^{n+1/2} & = & u^n \quad \text{on } \Omega \setminus \overline{\Omega'_1} \end{array} \right.$$

$$(D_2) \left\{ \begin{array}{lcl} -u^{n+1''} & = & f \quad \text{in } \Omega'_2, \\ u^{n+1} & = & u^{n+1/2} \quad \text{auf } \Gamma_2 \\ u^{n+1} & = & u^{n+1/2} \quad \text{on } \Omega \setminus \overline{\Omega'_2} \end{array} \right.$$

For the sake of simplicity, instead of the two-dimensional geometry,

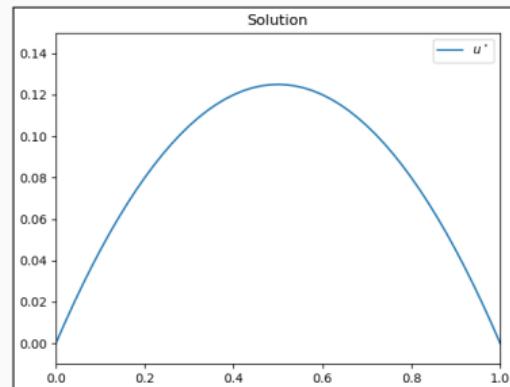


we consider the **one-dimensional Poisson equation**

$$-u'' = 1 \quad \text{in } [0, 1],$$

$$u(0) = u(1) = 0.$$

Solution: $u(x) = -\frac{1}{2}x(x - 1).$



Domain decomposition:



Let us consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform an **alternating Schwarz iteration**:

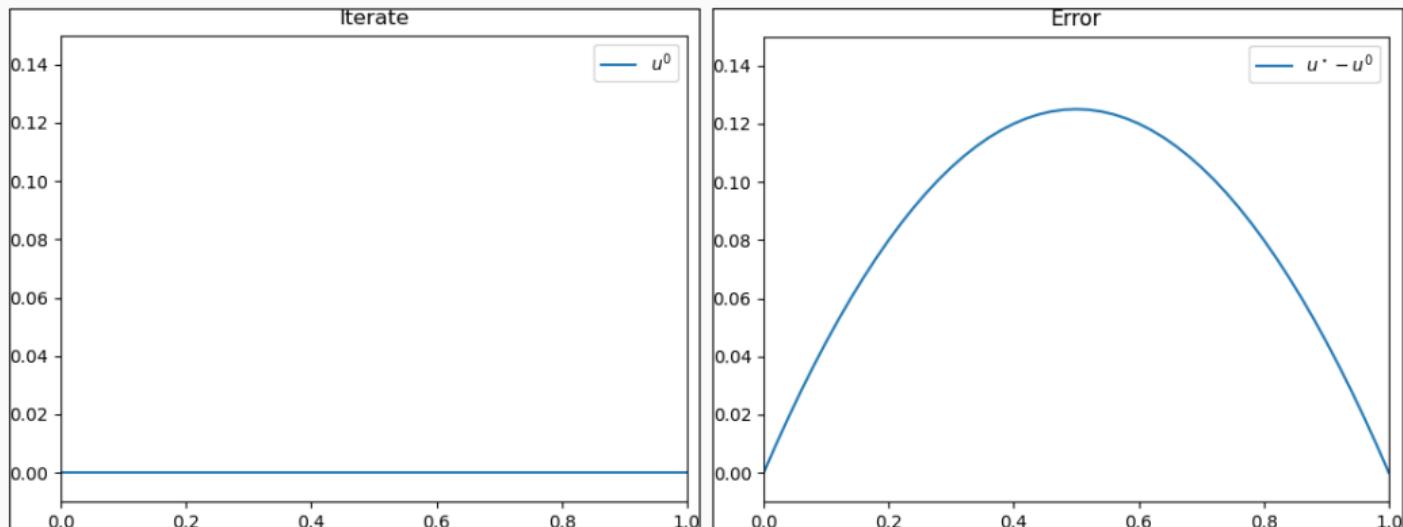


Figure 1: Iterate (left) and error (right) in iteration 0.

Let us consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform an **alternating Schwarz iteration**:

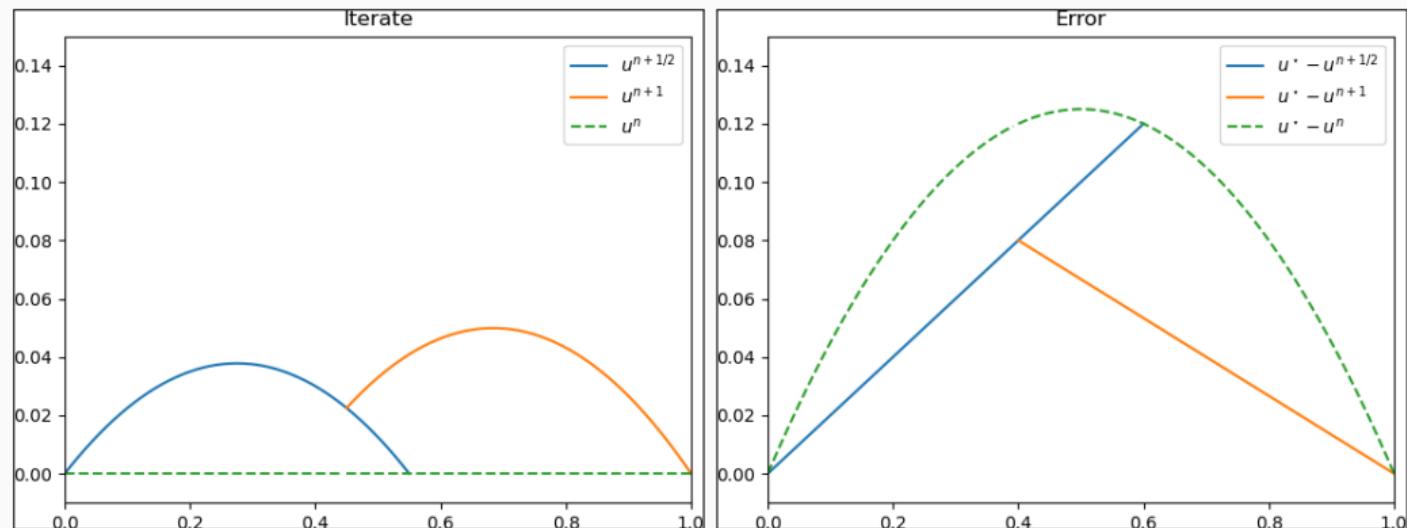


Figure 1: Iterate (left) and error (right) in iteration 1.

Let us consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform an **alternating Schwarz iteration**:

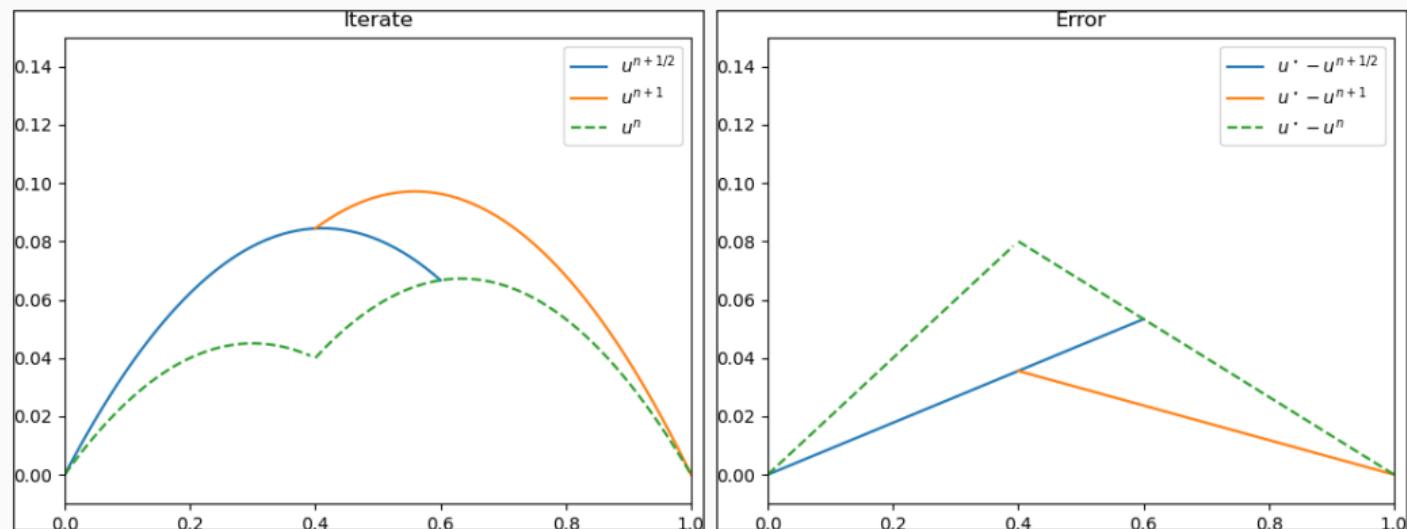


Figure 1: Iterate (left) and error (right) in iteration 2.

Let us consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform an **alternating Schwarz iteration**:

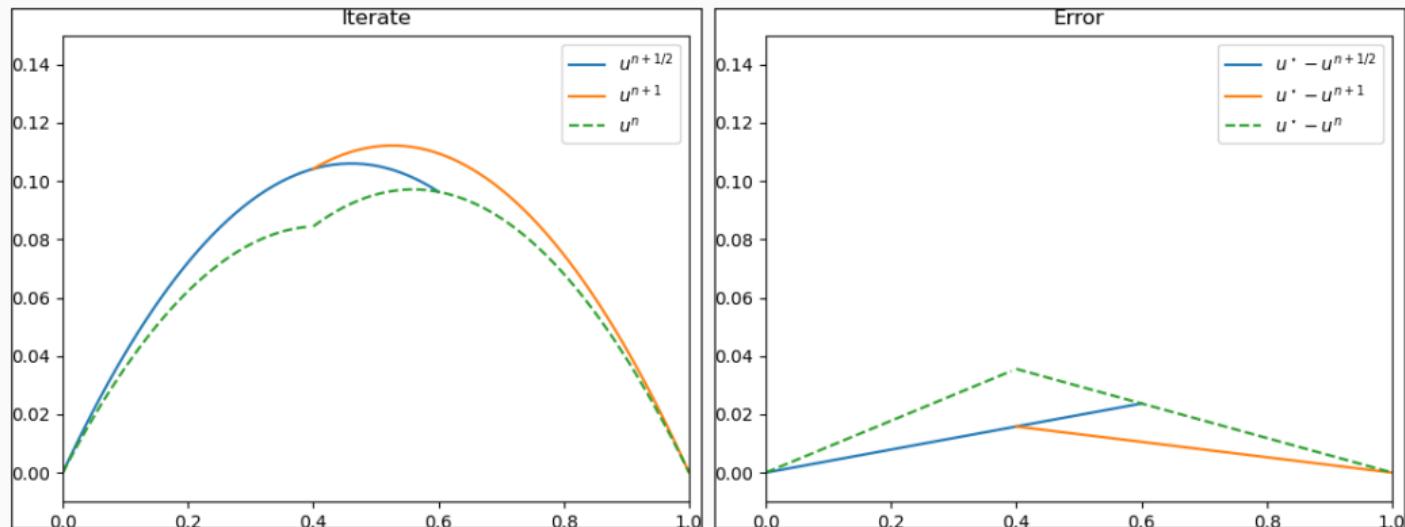


Figure 1: Iterate (left) and error (right) in iteration 3.

Let us consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform an **alternating Schwarz iteration**:

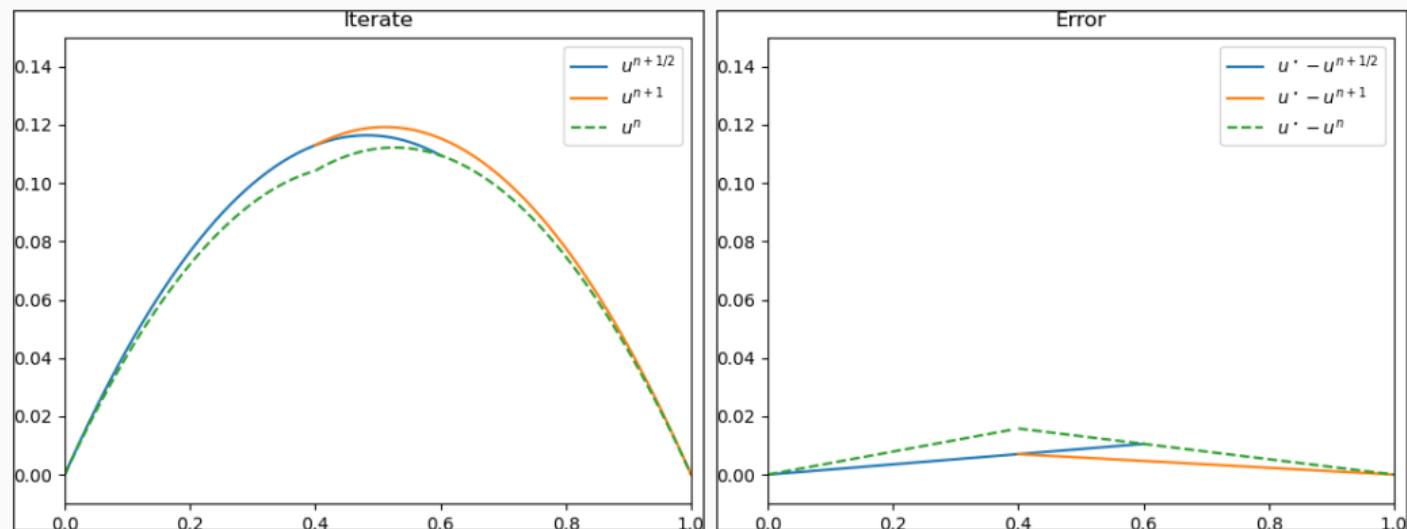


Figure 1: Iterate (left) and error (right) in iteration 4.

Let us consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform an **alternating Schwarz iteration**:

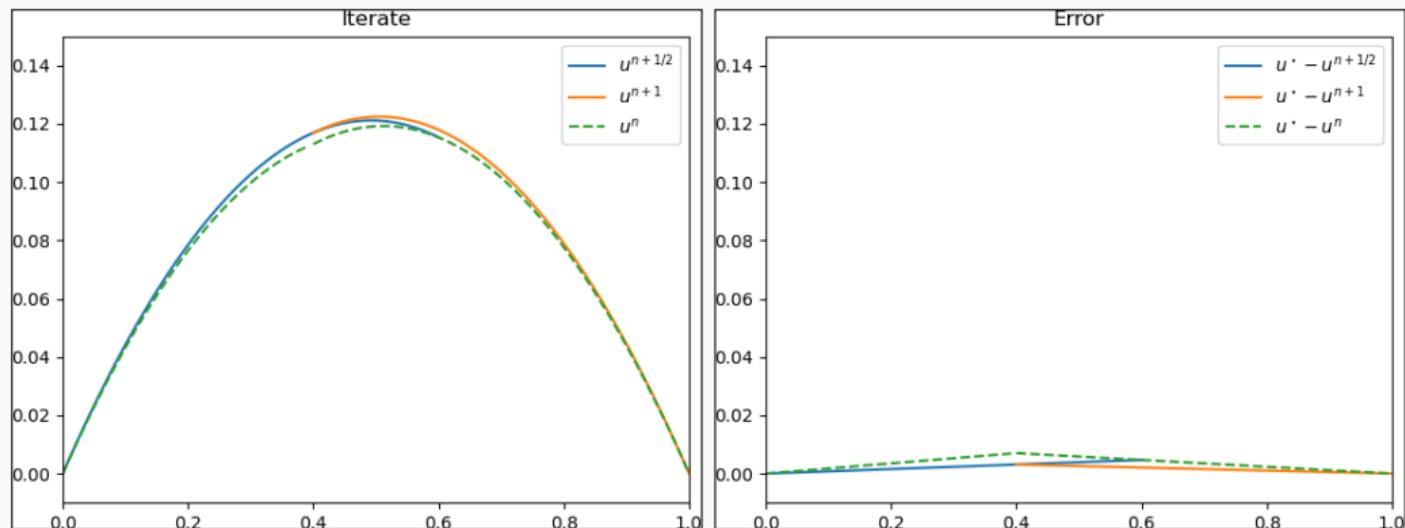


Figure 1: Iterate (left) and error (right) in iteration 5.

The alternating Schwarz algorithm is **sequential** because **each local boundary value problem** depends on the solution of the **previous Dirichlet problem**:

$$(D_1) \begin{cases} -\Delta u^{n+1/2} = f & \text{in } \Omega'_1, \\ u^{n+1/2} = \mathbf{u}^n & \text{on } \partial\Omega'_1 \\ u^{n+1/2} = \mathbf{u}^n & \text{on } \Omega \setminus \overline{\Omega'_1} \end{cases}$$

$$(D_2) \begin{cases} -\Delta u^{n+1} = f & \text{in } \Omega_2, \\ u^{n+1} = \mathbf{u}^{n+1/2} & \text{on } \partial\Omega'_2 \\ u^{n+1} = \mathbf{u}^{n+1/2} & \text{on } \Omega \setminus \overline{\Omega'_2} \end{cases}$$



Idea: For all red terms, we use the values from the previous iteration. Then, the both Dirichlet problem can be solved at the same time.

The alternating Schwarz algorithm is **sequential** because **each local boundary value problem** depends on the solution of the **previous Dirichlet problem**:

$$(D_1) \begin{cases} -\Delta u^{n+1/2} = f & \text{in } \Omega'_1, \\ u^{n+1/2} = \mathbf{u}^n & \text{on } \partial\Omega'_1 \\ u^{n+1/2} = \mathbf{u}^n & \text{on } \Omega \setminus \overline{\Omega'_1} \end{cases}$$

$$(D_2) \begin{cases} -\Delta u^{n+1} = f & \text{in } \Omega_2, \\ u^{n+1} = \mathbf{u}^{n+1/2} & \text{on } \partial\Omega'_2 \\ u^{n+1} = \mathbf{u}^{n+1/2} & \text{on } \Omega \setminus \overline{\Omega'_2} \end{cases}$$

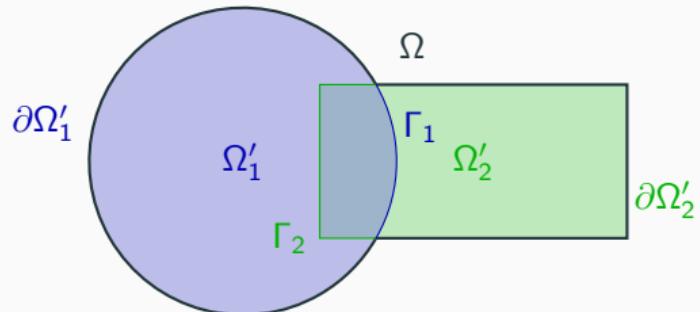


Idea: For all red terms, we **use the values from the previous iteration**. Then, the both Dirichlet problem **can be solved at the same time**.

3 The Parallel Schwarz Algorithm

The **parallel Schwarz algorithm** has been introduced by [Lions \(1988\)](#). Here, we solve the local problems

$$(D_1) \begin{cases} -\Delta u_1^{n+1} = f & \text{in } \Omega'_1, \\ u_1^{n+1} = u_2^n & \text{on } \partial\Omega'_1, \end{cases}$$



$$(D_2) \begin{cases} -\Delta u_2^{n+1} = f & \text{in } \Omega_2, \\ u_2^{n+1} = u_1^n & \text{on } \partial\Omega'_2. \end{cases}$$

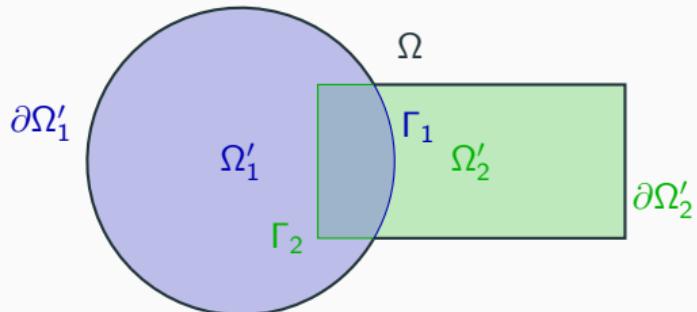
Since u_1^n and u_2^n are both computed in the previous iteration, the problems can be solved independent of each other.

3 The Parallel Schwarz Algorithm

The **parallel Schwarz algorithm** has been introduced by [Lions \(1988\)](#). Here, we solve the local problems

$$(D_1) \begin{cases} -\Delta u_1^{n+1} = f & \text{in } \Omega'_1, \\ u_1^{n+1} = u_2^n & \text{on } \partial\Omega'_1, \end{cases}$$

$$(D_2) \begin{cases} -\Delta u_2^{n+1} = f & \text{in } \Omega_2, \\ u_2^{n+1} = u_1^n & \text{on } \partial\Omega'_2. \end{cases}$$



Since u_1^n and u_2^n are both computed in the previous iteration, the problems can be solved independent of each other.

This method is suitable for **parallel computing!**



Let us again consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform the **parallel Schwarz iteration**:

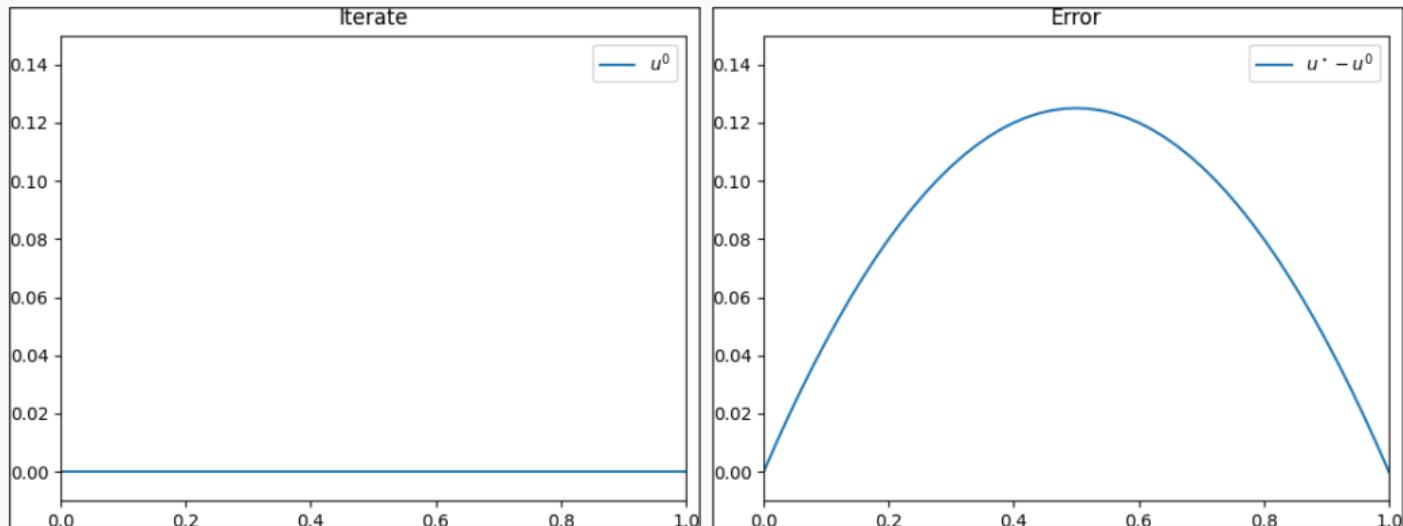


Figure 2: Iterate (left) and error (right) in iteration 0.

Let us again consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform the **parallel Schwarz iteration**:

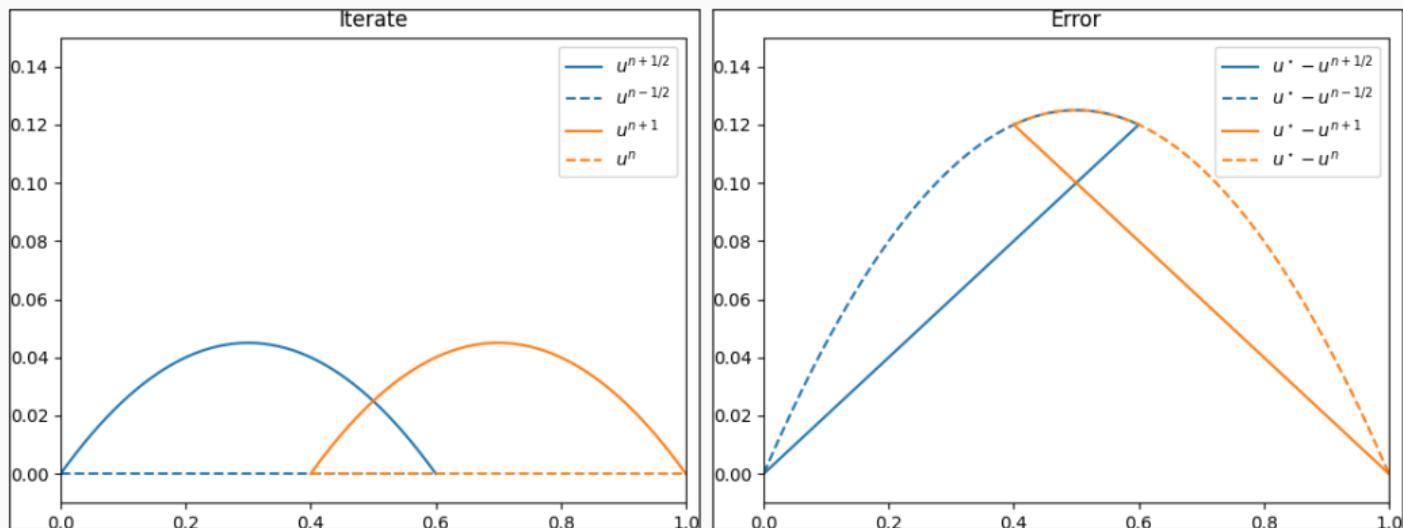


Figure 2: Iterate (left) and error (right) in iteration 1.

Let us again consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform the **parallel Schwarz iteration**:

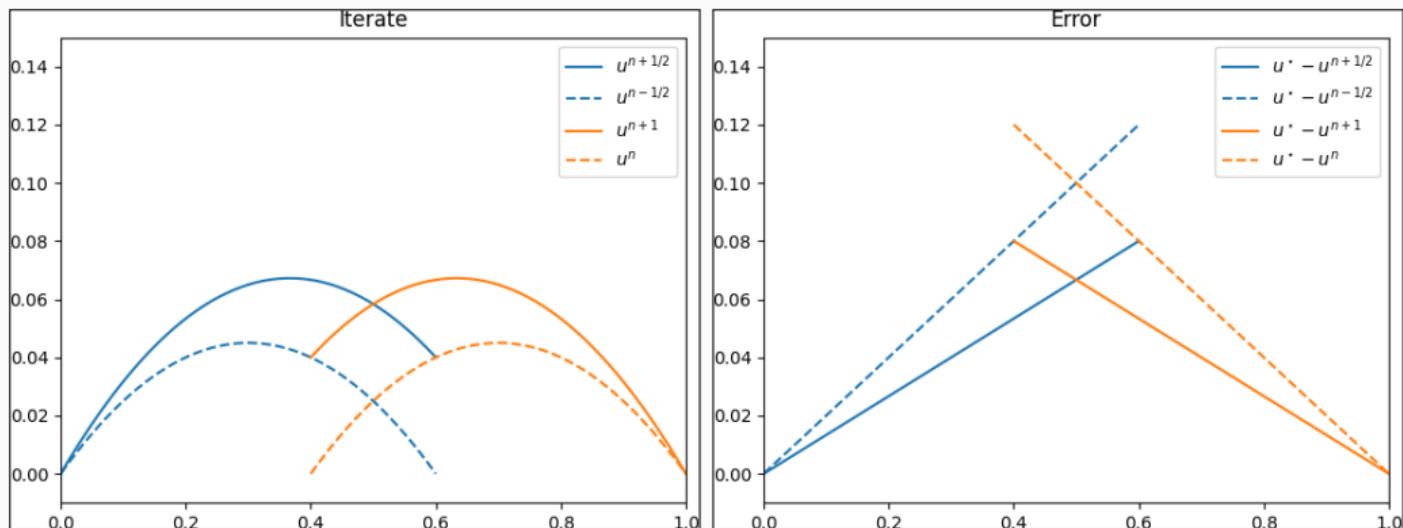


Figure 2: Iterate (left) and error (right) in iteration 2.

Let us again consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform the **parallel Schwarz iteration**:

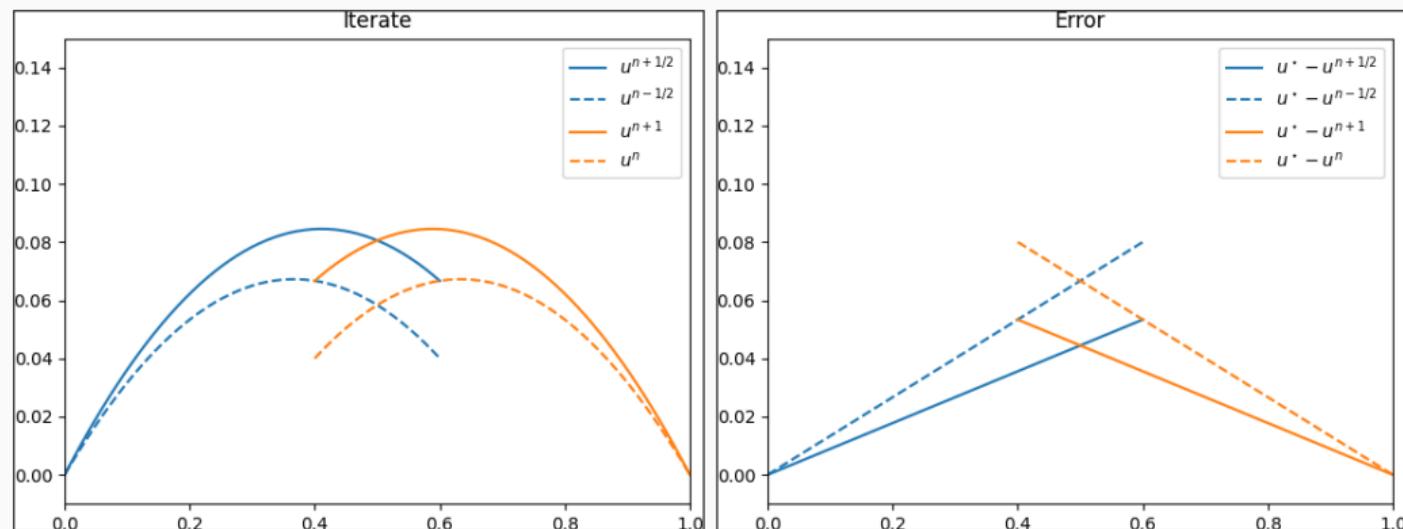


Figure 2: Iterate (left) and error (right) in iteration 3.

Let us again consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform the **parallel Schwarz iteration**:

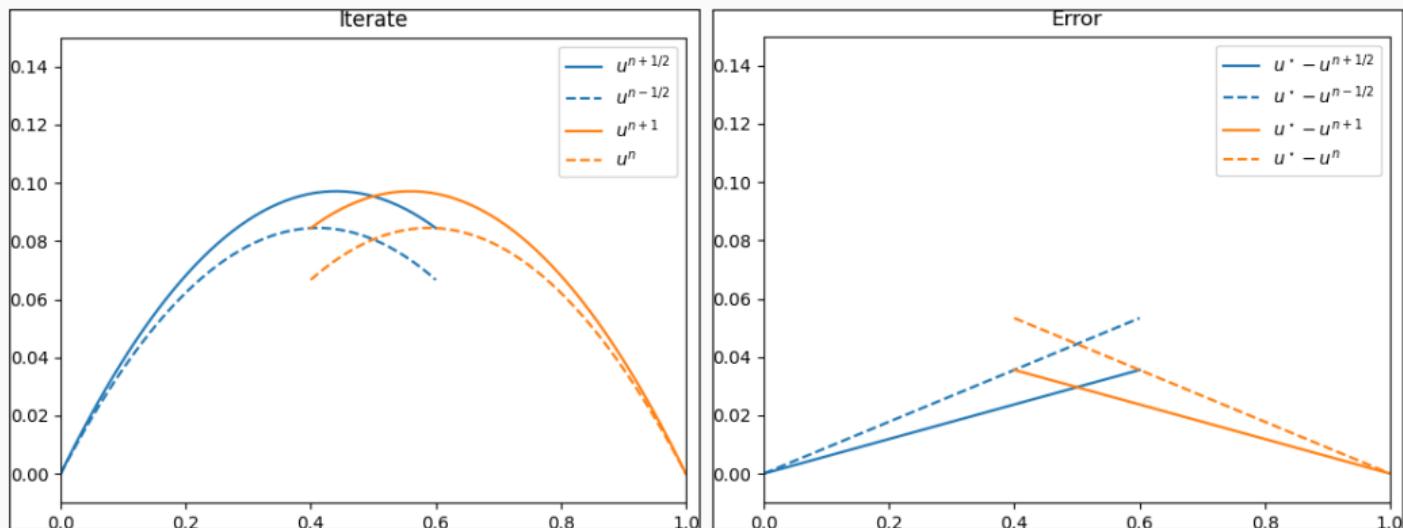


Figure 2: Iterate (left) and error (right) in iteration 4.

Let us again consider the simple boundary value problem: Find u such that

$$-u'' = 1, \text{ in } [0, 1], \quad u(0) = u(1) = 0$$

We perform the **parallel Schwarz iteration**:

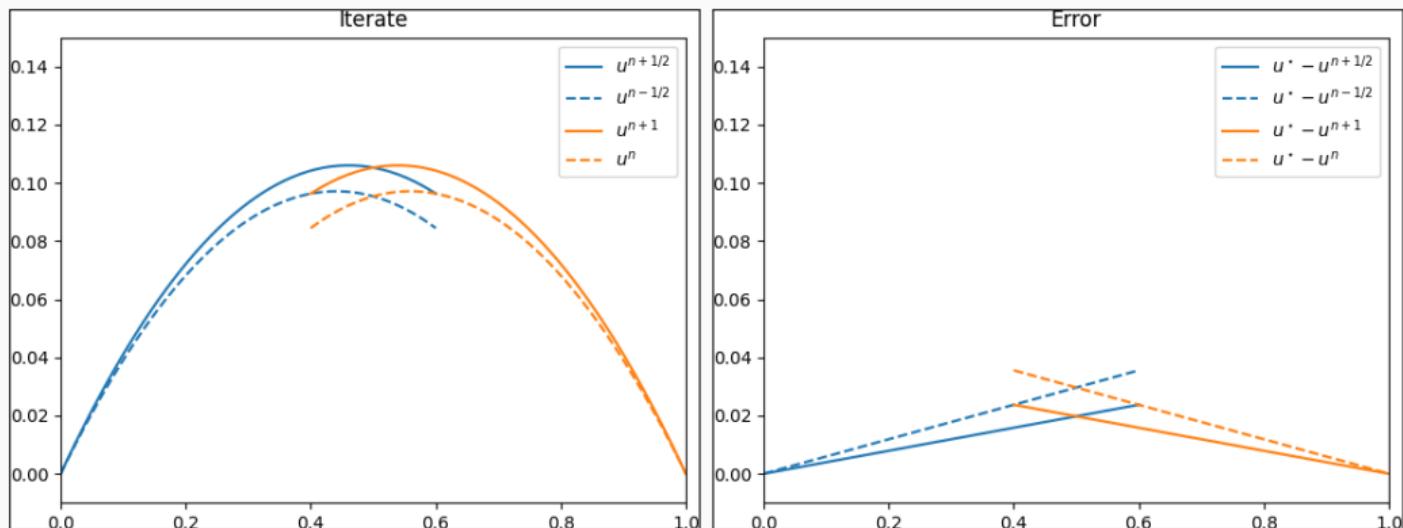


Figure 2: Iterate (left) and error (right) in iteration 5.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

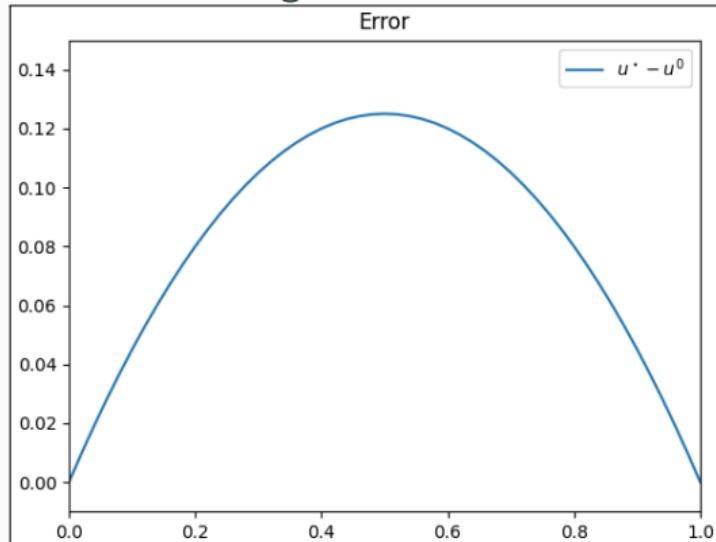


Figure 3: Error in iteration 0.

Parallel Schwarz iteration

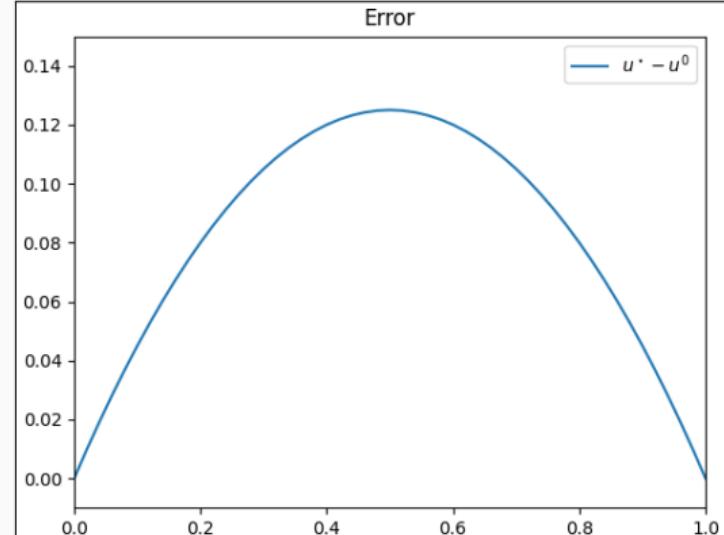


Figure 4: Error in iteration 0.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

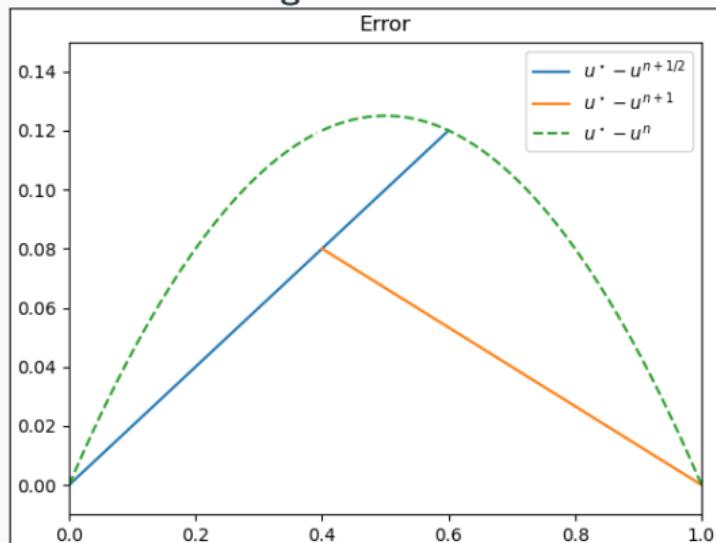


Figure 3: Error in iteration 1.

Parallel Schwarz iteration

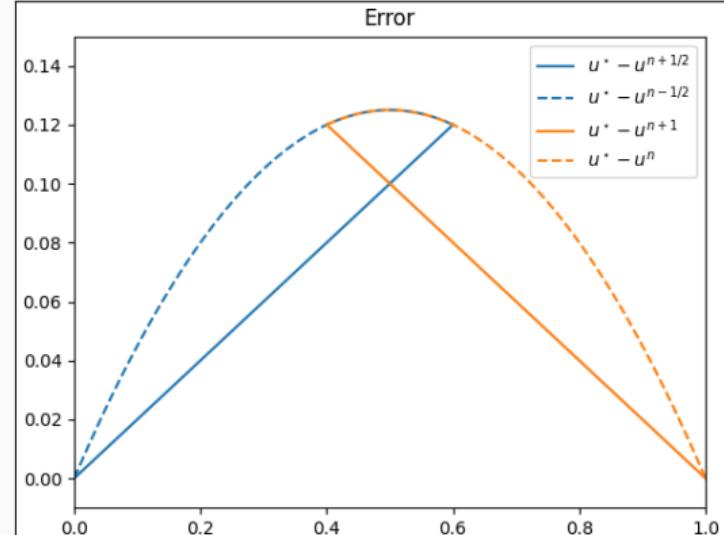


Figure 4: Error in iteration 1.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

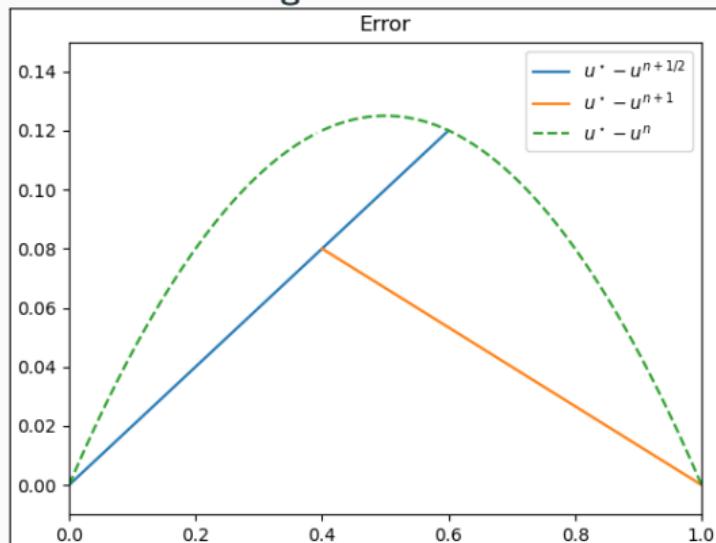


Figure 3: Error in iteration 1.

Parallel Schwarz iteration

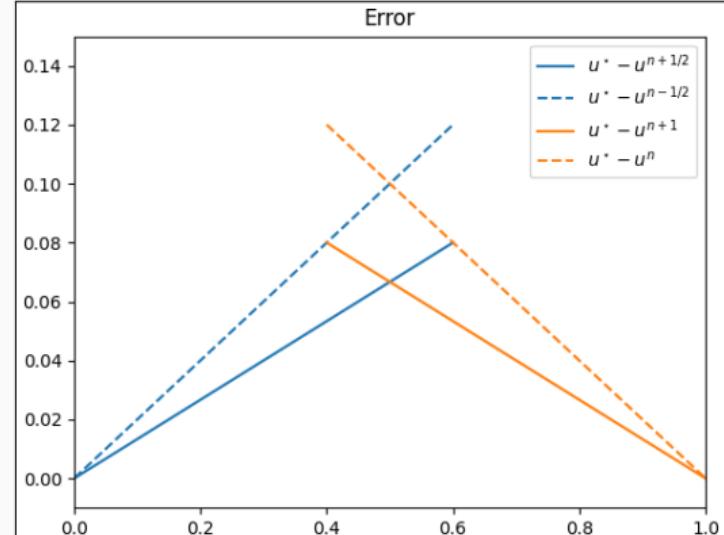


Figure 4: Error in iteration 2.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

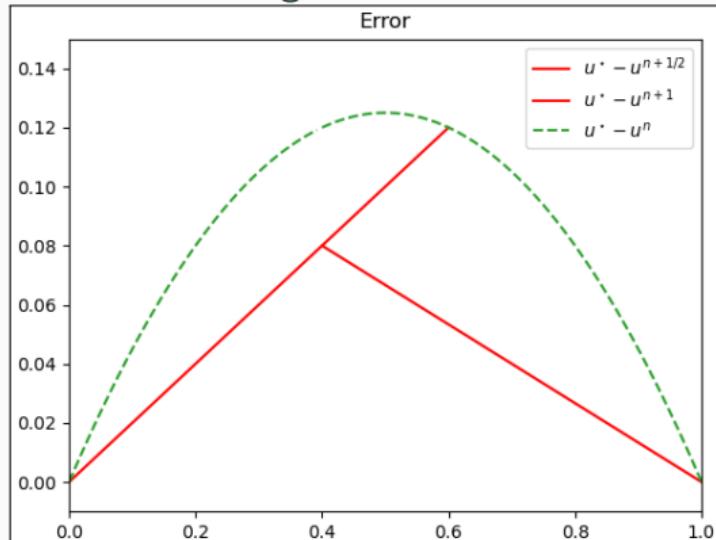


Figure 3: Error in iteration 1.

Parallel Schwarz iteration

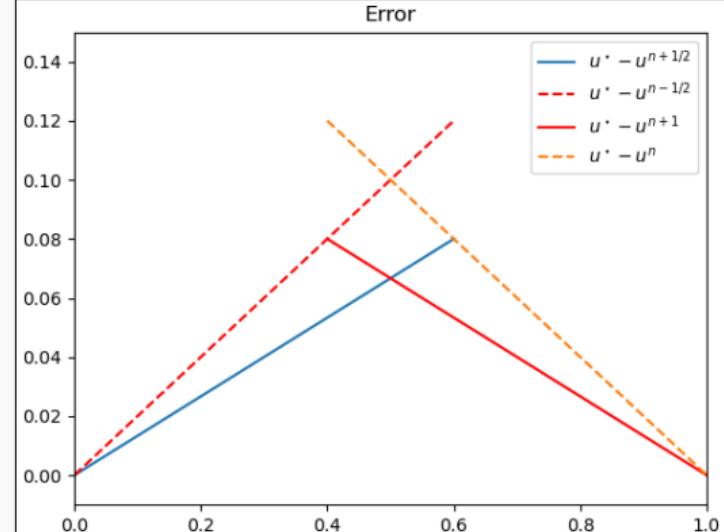


Figure 4: Error in iteration 2.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

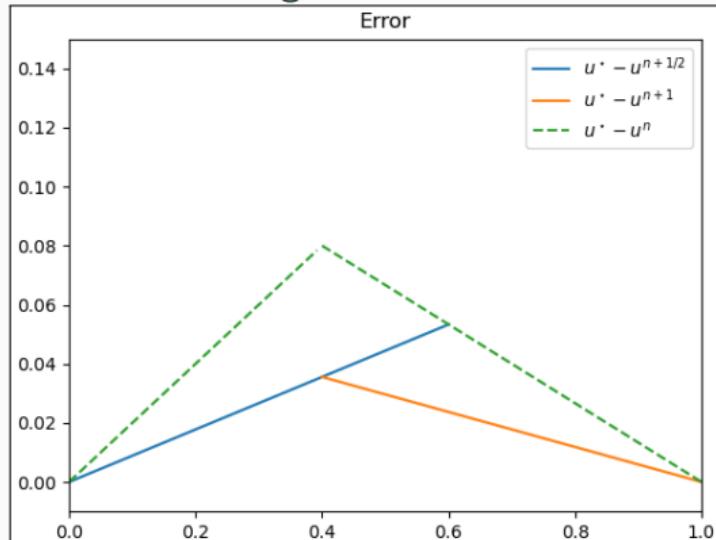


Figure 3: Error in iteration 2.

Parallel Schwarz iteration

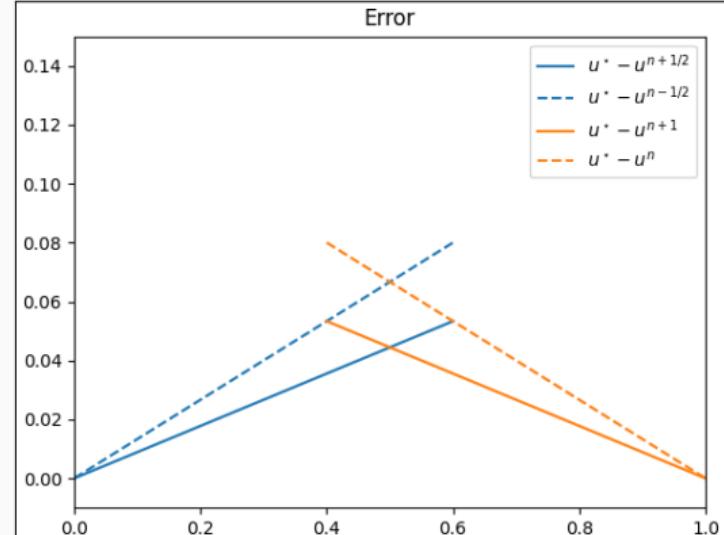


Figure 4: Error in iteration 3.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

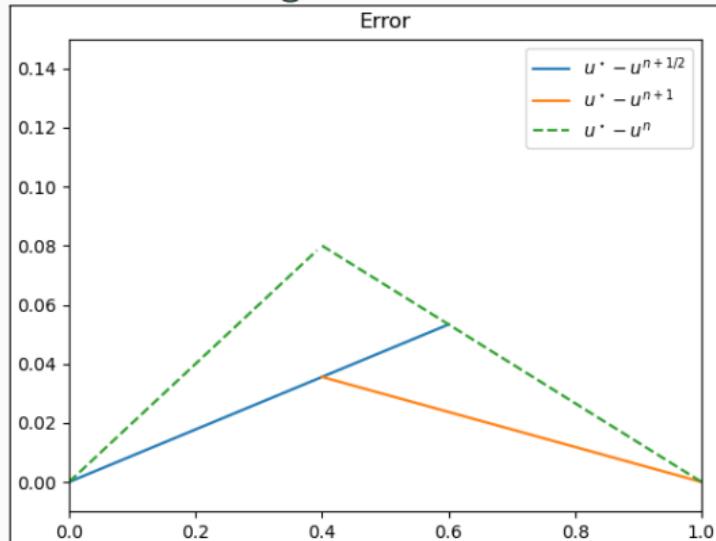


Figure 3: Error in iteration 2.

Parallel Schwarz iteration

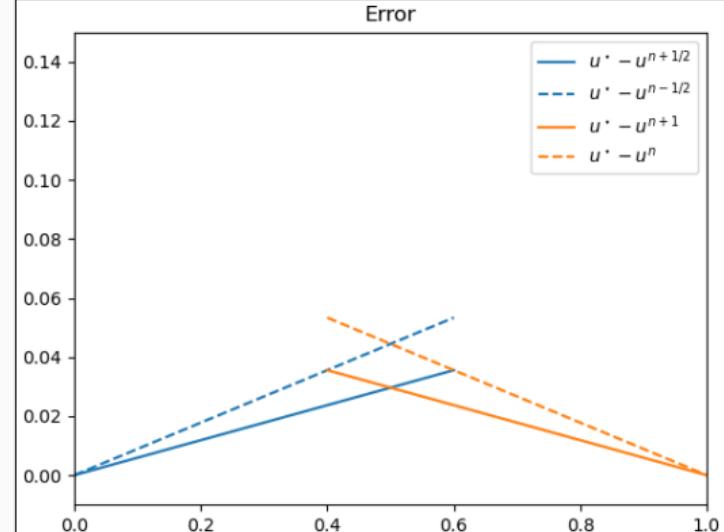


Figure 4: Error in iteration 4.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

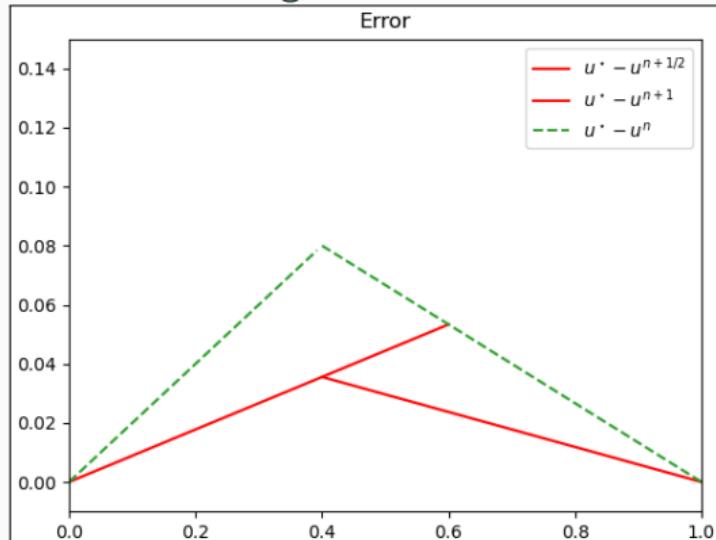


Figure 3: Error in iteration 2.

Parallel Schwarz iteration

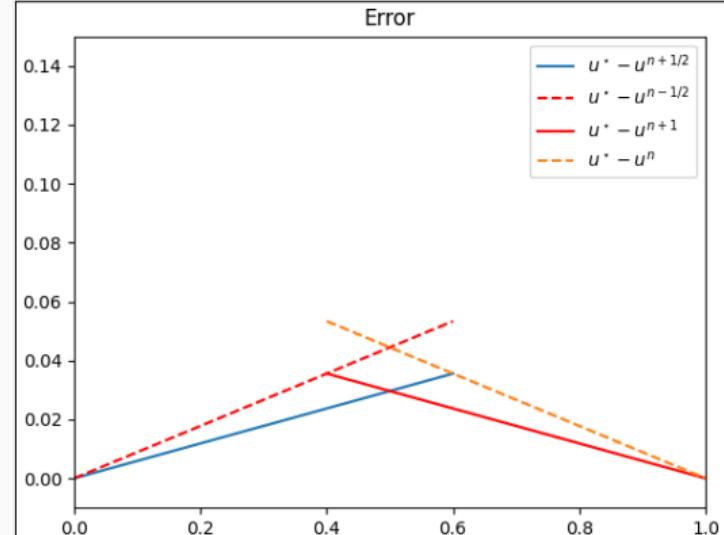


Figure 4: Error in iteration 4.

4 Comparison of the two Methods

Next, we compare the convergence of the two methods using the error plots:

Alternating Schwarz iteration

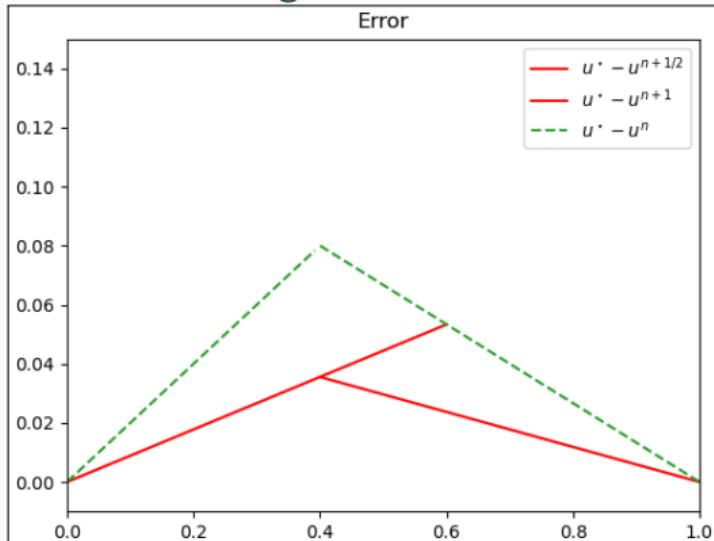


Figure 3: Error in iteration 2.

Parallel Schwarz iteration

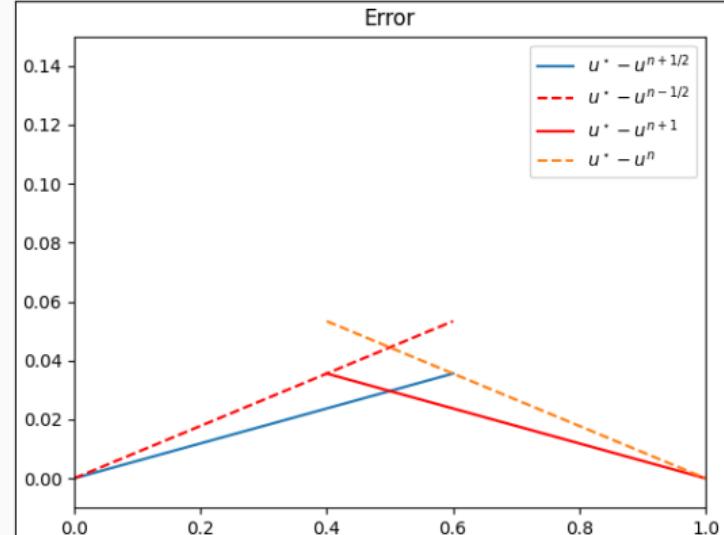


Figure 4: Error in iteration 4.

The alternating Schwarz method **converges twice as fast** as the parallel Schwarz method. However, the **local solutions** have to be computed **sequentially**.

5 Effect of the Size of the Overlap

We investigate the convergence of the methods (using the alternating method as an example) depending on the **size of the overlap**:



Overlap 0.05



Overlap 0.1

5 Effect of the Size of the Overlap

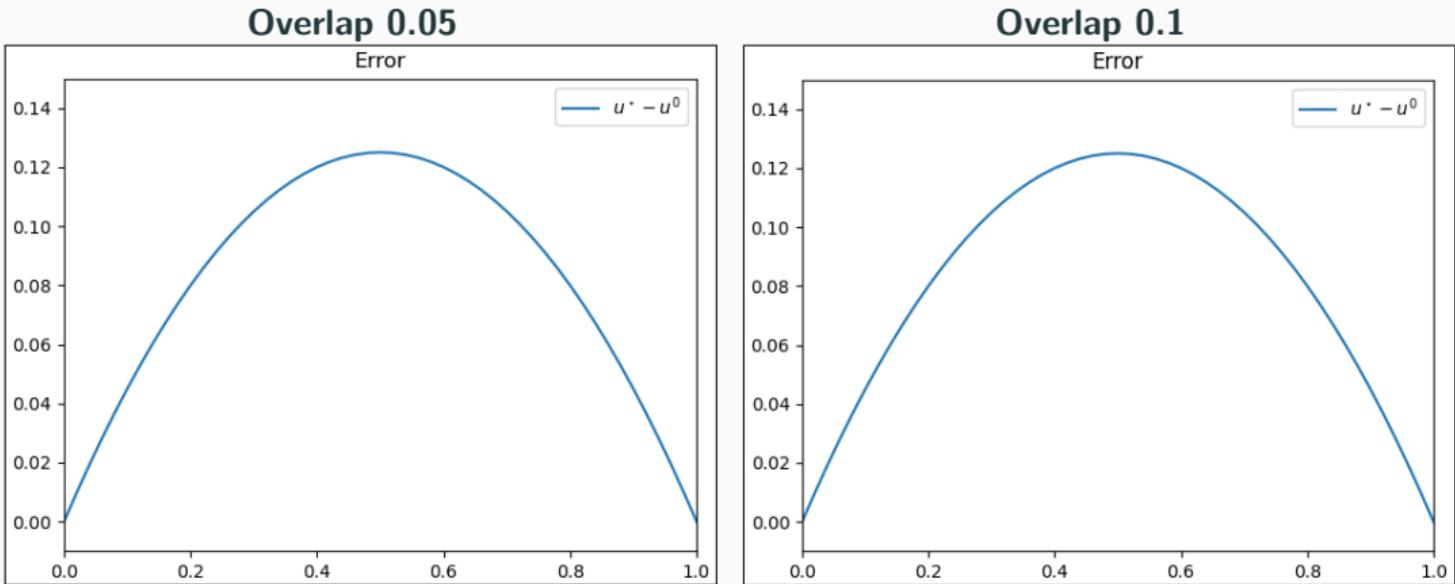


Figure 5: Error in iteration 0.

5 Effect of the Size of the Overlap

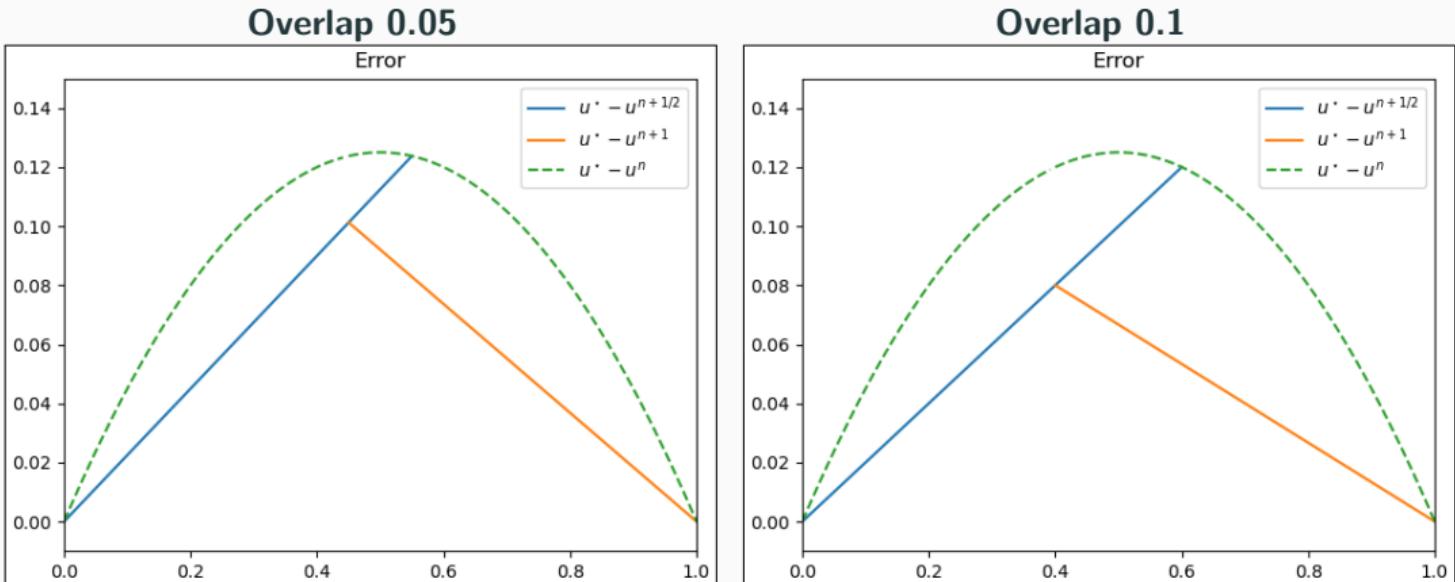


Figure 5: Error in iteration 1.

5 Effect of the Size of the Overlap

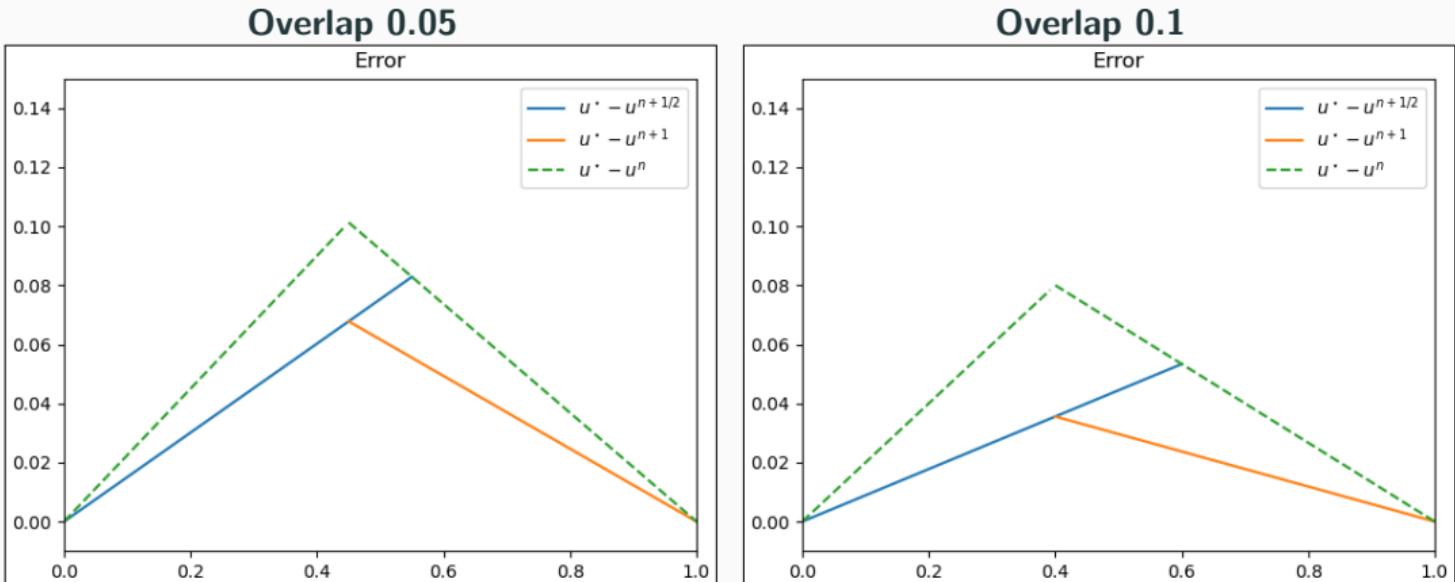


Figure 5: Error in iteration 2.

5 Effect of the Size of the Overlap

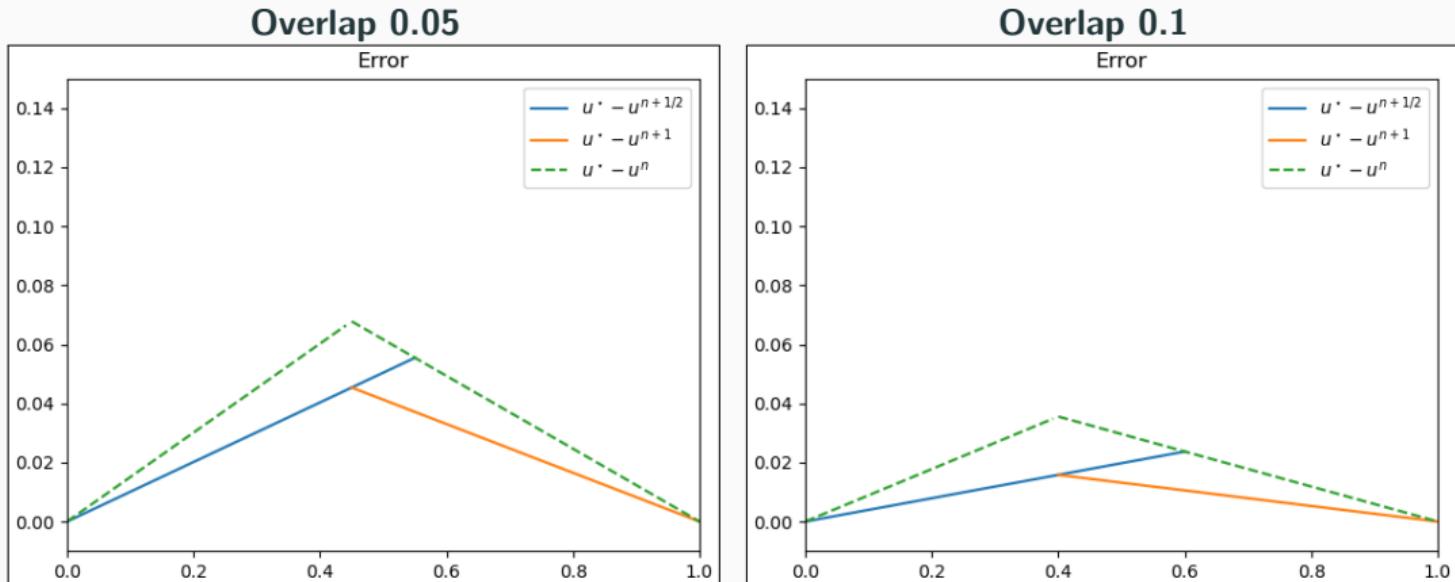


Figure 5: Error in iteration 3.

5 Effect of the Size of the Overlap

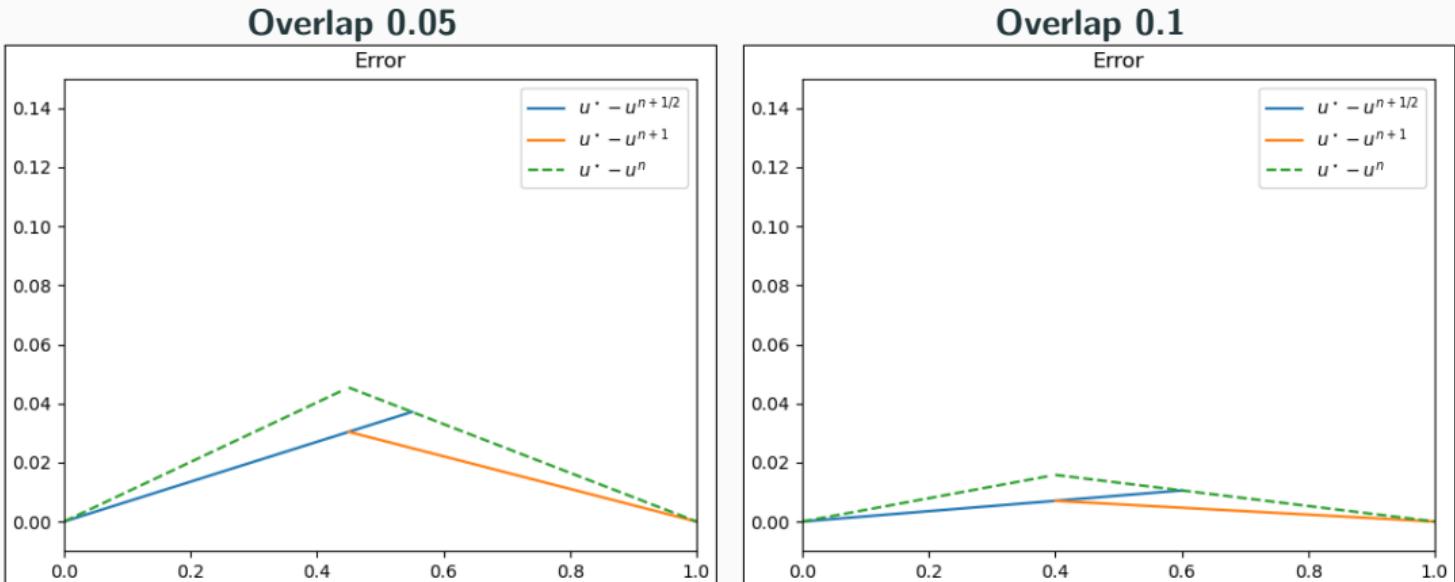


Figure 5: Error in iteration 4.

5 Effect of the Size of the Overlap

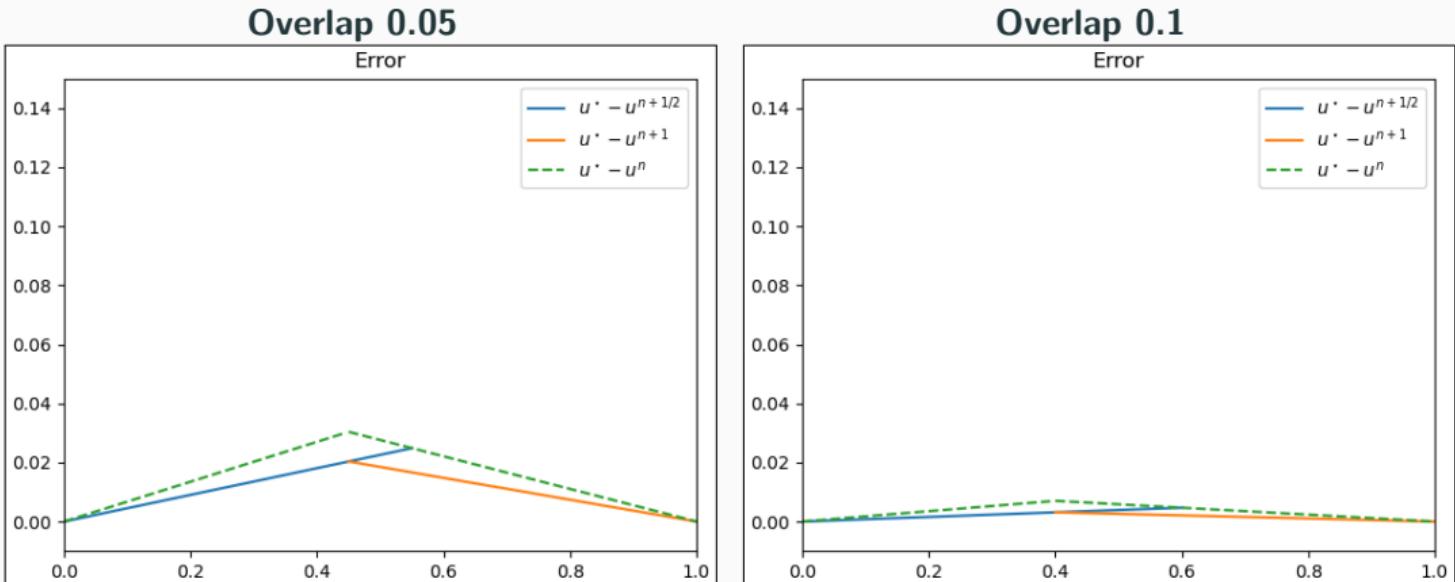


Figure 5: Error in iteration 5.

5 Effect of the Size of the Overlap

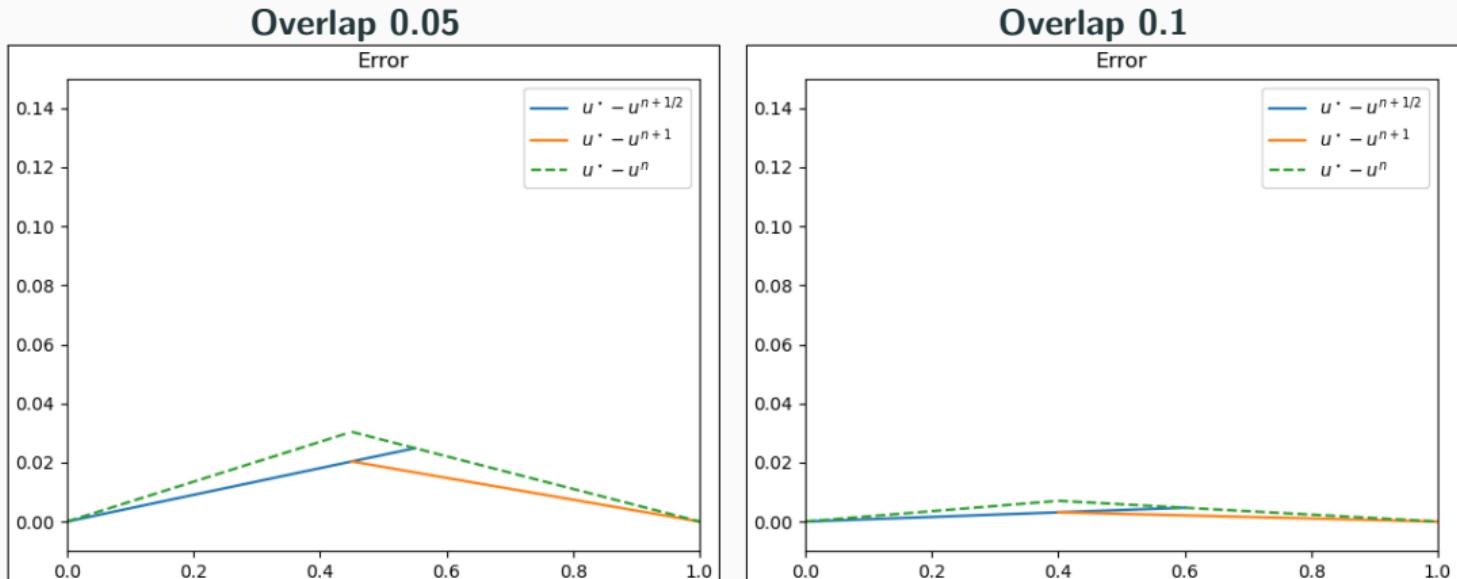


Figure 5: Error in iteration 5.

⇒ A **larger overlap** leads to **faster convergence**.

Part II — Schwarz Domain Decomposition Preconditioners

6. Model Problem

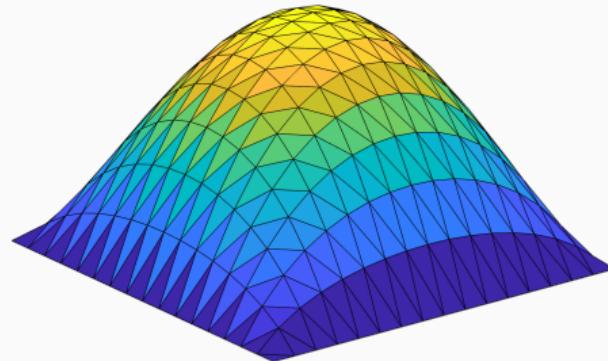
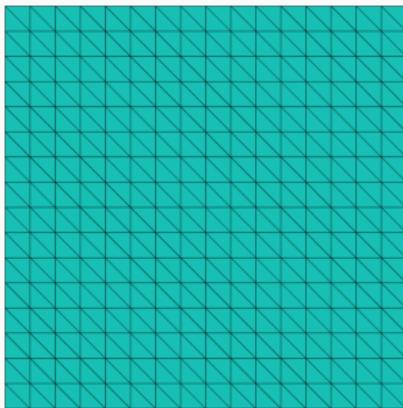
7. One-Level Overlapping Schwarz Preconditioners

8. Two-Level Overlapping Schwarz Preconditioners

9. A Brief Overview Over the Theoretical Framework

10. Some Comments on Constructing Schwarz Preconditioners

6 Model Problem

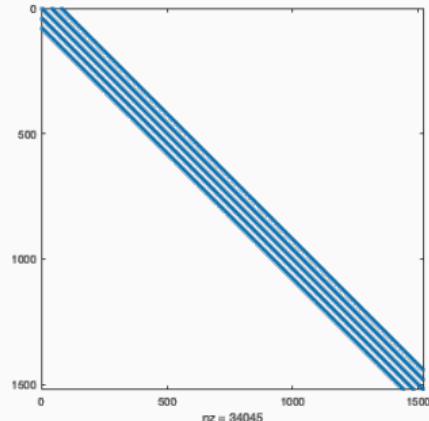
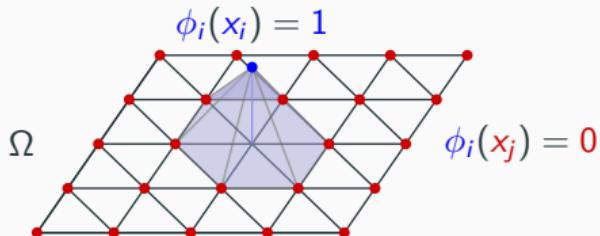


Let us consider the simple **diffusion model problem**:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [0, 1]^2, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

Discretization using finite elements yields the linear equation system

$$\mathbf{A}\mathbf{u} = \mathbf{f}.$$



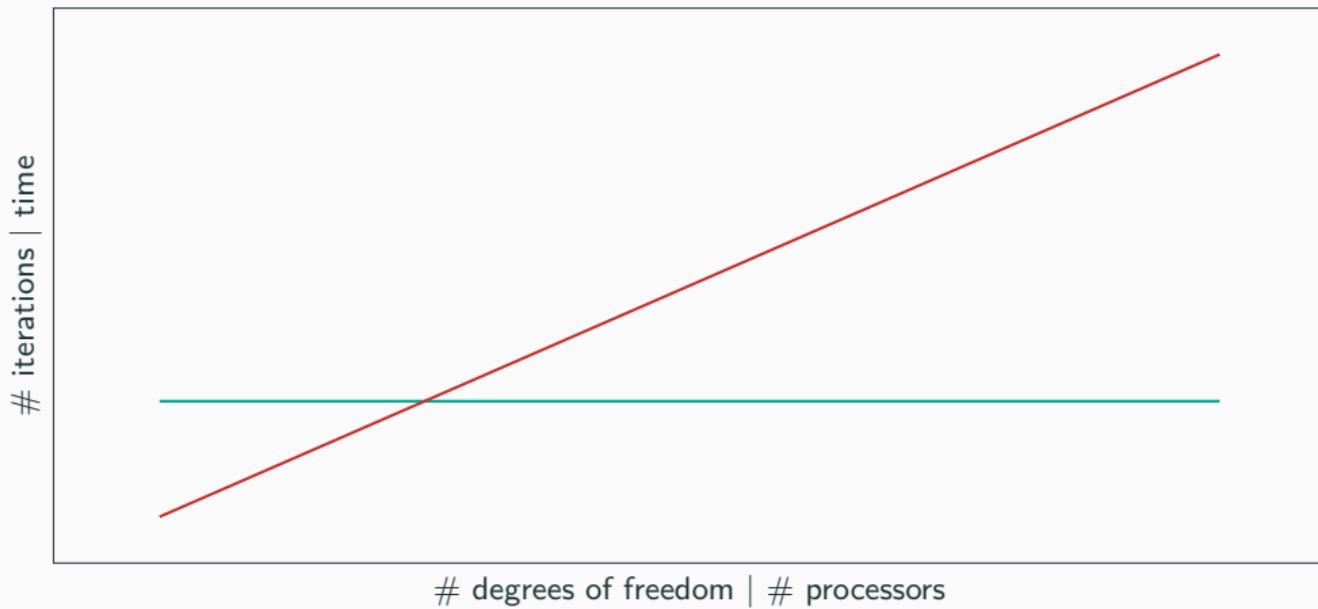
- Due to the local support of the finite element basis functions, the resulting system is **sparse**.
 - However, due to the **superlinear complexity and memory cost**, the use of direct solvers becomes infeasible for fine meshes, that is, for the **resulting large sparse equation systems**.
- We will employ iterative solvers:
For our elliptic model problem, the system matrix is symmetric positive definite, such that we can use the **preconditioner gradient descent (PCG) method**.

Goal – Numerical & Parallel (Weak) Scalability

Increase the problem size while keeping

$$\frac{\# \text{ degrees of freedom}}{\# \text{ processors}}$$

fixed.



Preconditioned Conjugate Gradient (PCG) Method

Algorithm 1: Preconditioned conjugate gradient (PCG) method

Result: Approximate solution of the linear equation system $\mathbf{A}\mathbf{x} = \mathbf{b}$

Given: Initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$ and tolerance $\varepsilon > 0$

$$\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$$

$$\mathbf{p}^{(0)} := \mathbf{y}^{(0)} := \mathbf{M}^{-1}\mathbf{r}^{(0)}$$

while $\|\mathbf{r}^{(k)}\| \geq \varepsilon \|\mathbf{r}^{(0)}\|$ **do**

$$\alpha_k := \frac{(\mathbf{p}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{A}\mathbf{p}^{(k)}, \mathbf{p}^{(k)})}$$

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \alpha_k \mathbf{y}^{(k)}$$

$$\mathbf{r}^{(k+1)} := \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{p}^{(k)}$$

$$\mathbf{y}^{(k+1)} := \mathbf{M}^{-1}\mathbf{r}^{(k+1)}$$

$$\beta_k := \frac{(\mathbf{y}^{(k+1)}, \mathbf{A}\mathbf{p}^{(k)})}{(\mathbf{p}^{(k)}, \mathbf{A}\mathbf{p}^{(k)})}$$

$$\mathbf{p}^{(k+1)} := \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}$$

end

Theorem 1

Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. Then the **PCG method** converges and the following error estimate holds:

$$\|\mathbf{e}^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\kappa(M^{-1}A)} - 1}{\sqrt{\kappa(M^{-1}A)} + 1} \right)^k \|\mathbf{e}^{(0)}\|_A,$$

where $\kappa(M^{-1}A) = \frac{\lambda_{\max}(M^{-1}A)}{\lambda_{\min}(M^{-1}A)}$ is condition number of the preconditioned matrix $M^{-1}A$.

Do we need a preconditioner?

The condition number of the stiffness matrix K for the diffusion problem behaves as follows:

$$\kappa(K) \leq C \frac{(\max_{T \in \tau_h} h_T)^d}{(\min_{T \in \tau_h} h_T)^{d+2}} \stackrel{\text{quasi uniform}}{\equiv} C \frac{1}{h^2},$$

where τ_h is the triangulation and d is the problem dimension (for instance, $d = 2, 3$).

⇒ **Convergence of the PCG method will deteriorate** when refining the mesh.

Theorem 1

Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. Then the **PCG method** converges and the following error estimate holds:

$$\|\mathbf{e}^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\kappa(M^{-1}A)} - 1}{\sqrt{\kappa(M^{-1}A)} + 1} \right)^k \|\mathbf{e}^{(0)}\|_A,$$

where $\kappa(M^{-1}A) = \frac{\lambda_{\max}(M^{-1}A)}{\lambda_{\min}(M^{-1}A)}$ is condition number of the preconditioned matrix $M^{-1}A$.

Do we need a preconditioner?

The condition number of the stiffness matrix K for the diffusion problem behaves as follows:

$$\kappa(K) \leq C \frac{(\max_{T \in \tau_h} h_T)^d}{(\min_{T \in \tau_h} h_T)^{d+2}} \stackrel{\text{quasi uniform}}{\equiv} C \frac{1}{h^2},$$

where τ_h is the triangulation and d is the problem dimension (for instance, $d = 2, 3$).

⇒ Convergence of the PCG method will deteriorate when refining the mesh.

Theorem 1

Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. Then the **PCG method** converges and the following error estimate holds:

$$\|\mathbf{e}^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\kappa(M^{-1}A)} - 1}{\sqrt{\kappa(M^{-1}A)} + 1} \right)^k \|\mathbf{e}^{(0)}\|_A,$$

where $\kappa(M^{-1}A) = \frac{\lambda_{\max}(M^{-1}A)}{\lambda_{\min}(M^{-1}A)}$ is condition number of the preconditioned matrix $M^{-1}A$.

Do we need a preconditioner?

The condition number of the stiffness matrix K for the diffusion problem behaves as follows:

$$\kappa(K) \leq C \frac{(\max_{T \in \tau_h} h_T)^d}{(\min_{T \in \tau_h} h_T)^{d+2}} \stackrel{\text{quasi uniform}}{\equiv} C \frac{1}{h^2},$$

where τ_h is the triangulation and d is the problem dimension (for instance, $d = 2, 3$).

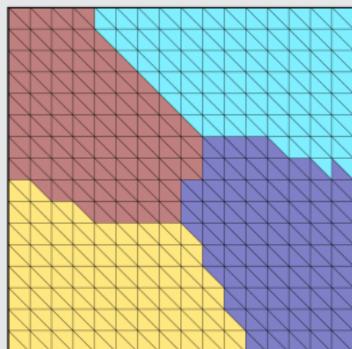
⇒ **Convergence of the PCG method will deteriorate** when refining the mesh.

7 One-Level Overlapping Schwarz Preconditioners

Overlapping domain decomposition

As the classical alternating and parallel Schwarz method (**overlapping**) Schwarz preconditioners are based on **overlapping decompositions** of the computational domain

$$\Omega = \bigcup_{i=1}^N \Omega'_i.$$



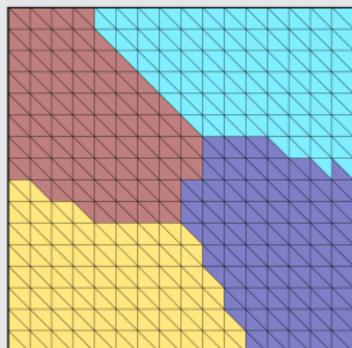
Nonoverlap. DD

7 One-Level Overlapping Schwarz Preconditioners

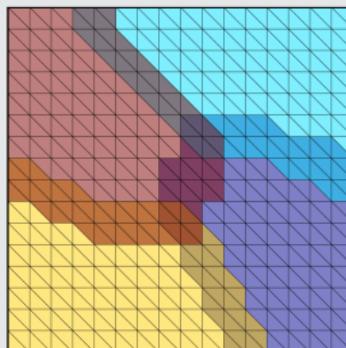
Overlapping domain decomposition

As the classical alternating and parallel Schwarz method (**overlapping**) Schwarz preconditioners are based on **overlapping decompositions** of the computational domain

$$\Omega = \bigcup_{i=1}^N \Omega'_i.$$



Nonoverlap. DD



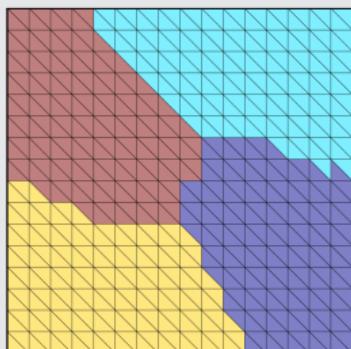
Overlap $\delta = 1h$

7 One-Level Overlapping Schwarz Preconditioners

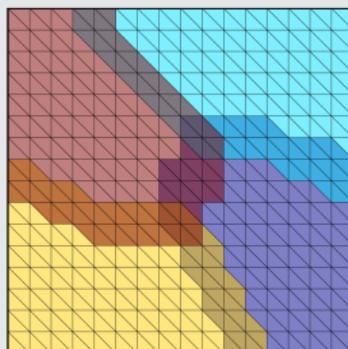
Overlapping domain decomposition

As the classical alternating and parallel Schwarz method (**overlapping**) Schwarz preconditioners are based on **overlapping decompositions** of the computational domain

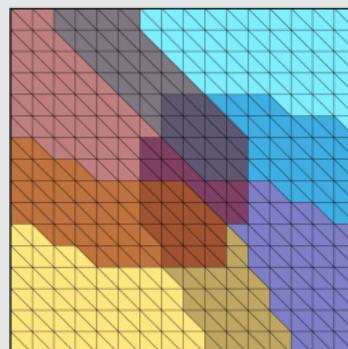
$$\Omega = \bigcup_{i=1}^N \Omega'_i.$$



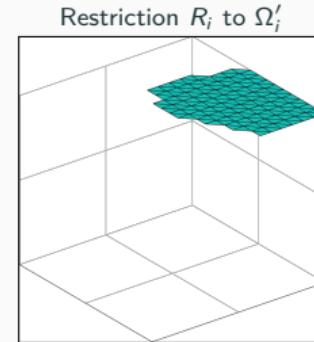
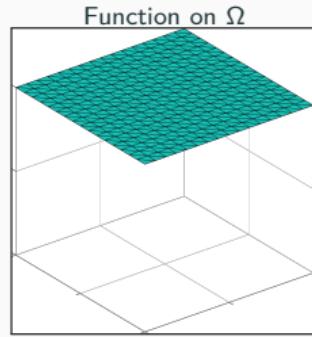
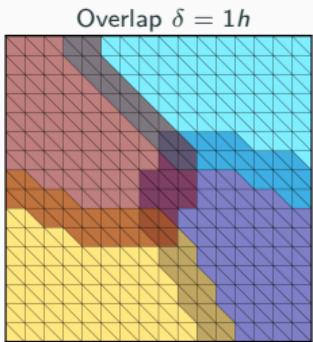
Nonoverlap. DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$



Based on an **overlapping domain decomposition**, we define an additive **one-level Schwarz preconditioner**

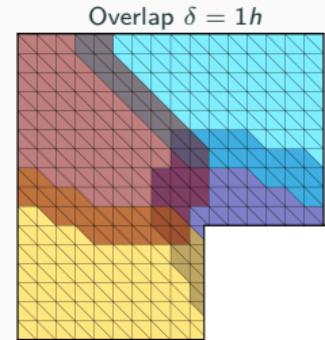
$$\mathbf{M}_{\text{OS-1}}^{-1} = \sum_{i=1}^N \mathbf{R}_i^T \mathbf{K}_i^{-1} \mathbf{R}_i,$$

where \mathbf{R}_i and \mathbf{R}_i^T are restriction and prolongation operators corresponding to Ω'_i , and $\mathbf{K}_i := \mathbf{R}_i \mathbf{K} \mathbf{R}_i^T$. The \mathbf{K}_i correspond to **local Dirichlet problems** on the overlapping subdomains.

Condition number bound:

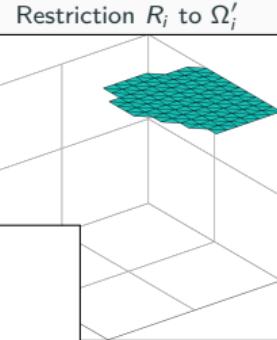
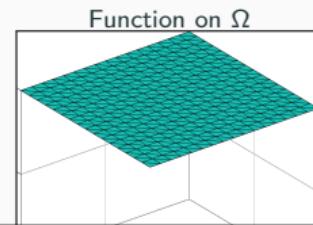
$$\kappa(\mathbf{M}_{\text{OS-1}}^{-1} \mathbf{K}) \leq C \left(1 + \frac{1}{H\delta} \right)$$

where the constant C is **independent of the subdomain size H and the width of the overlap δ** .

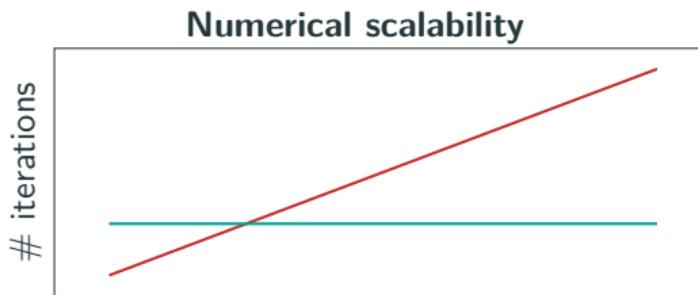


Based on an **overlap preconditioner**

where \mathbf{R}_i and \mathbf{R}_i^T
 $\mathbf{K}_i := \mathbf{R}_i \mathbf{K} \mathbf{R}_i^T$. The \mathbf{K}_i correspond to **local Dirichlet problems** on the overlapping subdomains.



e-level Schwarz



$$\# \text{ subdomains} = 1/H^d$$

Ω'_i , and

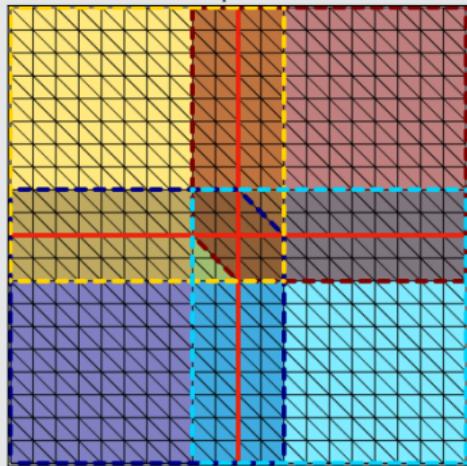
Condition number bound:

$$\kappa(\mathbf{M}_{\text{OS-1}}^{-1} \mathbf{K}) \leq C \left(1 + \frac{1}{H\delta} \right)$$

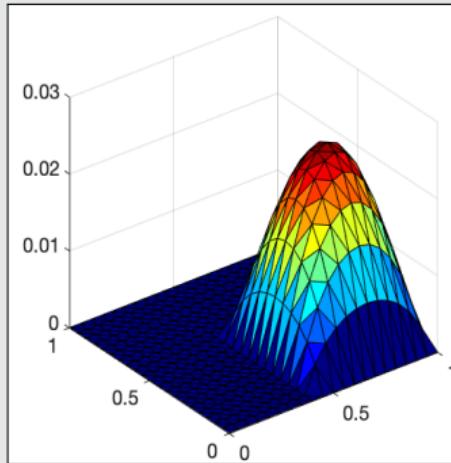
where the constant C is **independent of the subdomain size H and the width of the overlap δ** .

Solving a local subdomain problem

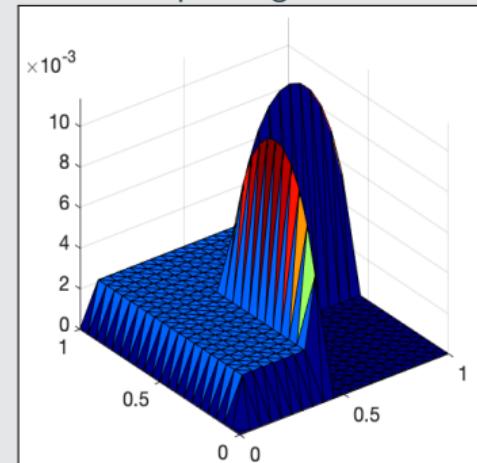
Overlap $\delta = 2h$



Solution on Ω_2

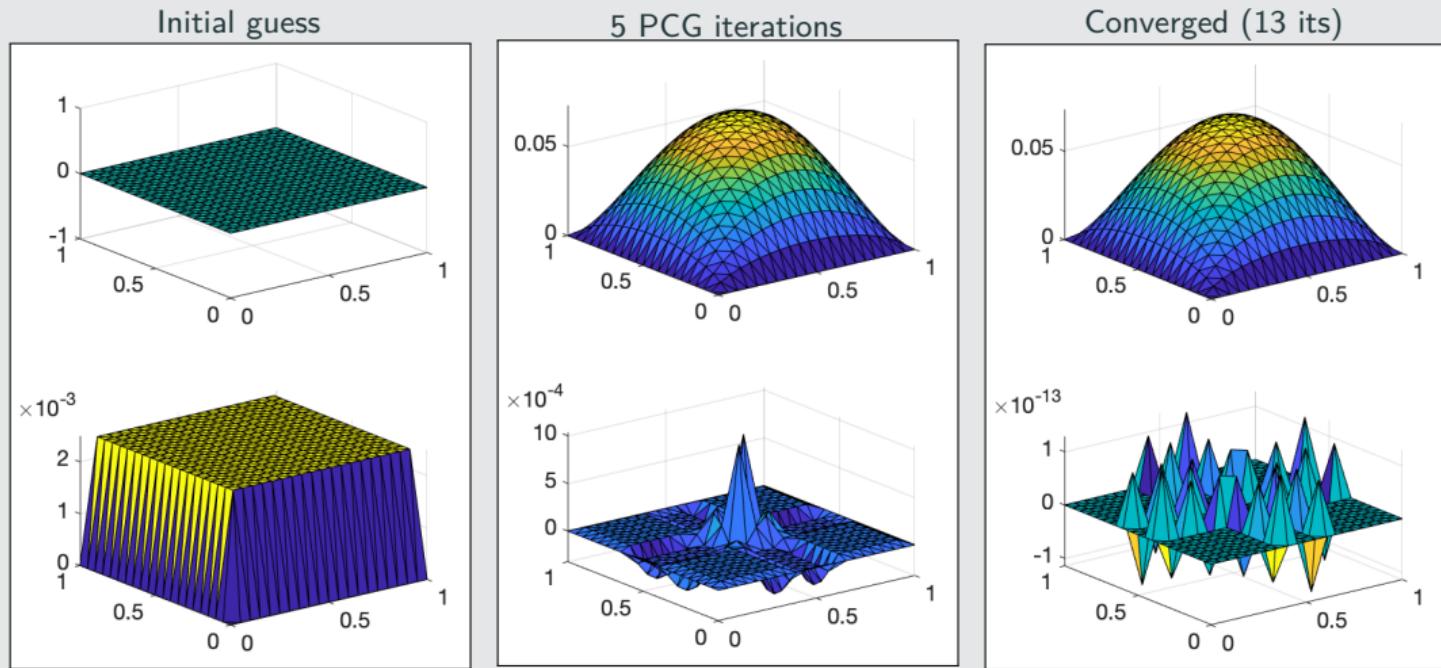


Corresponding residual



→ Zero residual only inside this subdomain but particularly large residual inside the overlap.

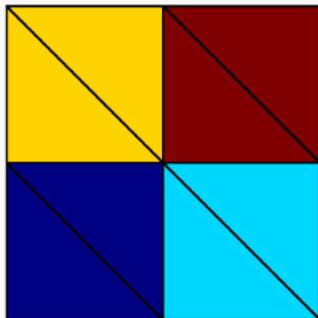
Convergence of the PCG method with a one-level Schwarz preconditioner



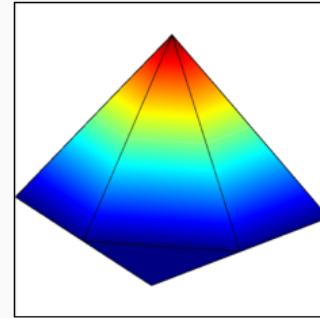
→ **Fast convergence** of the preconditioned gradient decent (PCG) method (**low number of subdomains**).

8 Two-Level Overlapping Schwarz Preconditioners

Coarse triangulation



Nodal bilinear basis function



The additive **two-level Schwarz preconditioner** reads

$$\mathbf{M}_{\text{OS-2}}^{-1} = \underbrace{\Phi \mathbf{K}_0^{-1} \Phi^T}_{\text{coarse level - global}} + \underbrace{\sum_{i=1}^N \mathbf{R}_i^T \mathbf{K}_i^{-1} \mathbf{R}_i}_{\text{first level - local}},$$

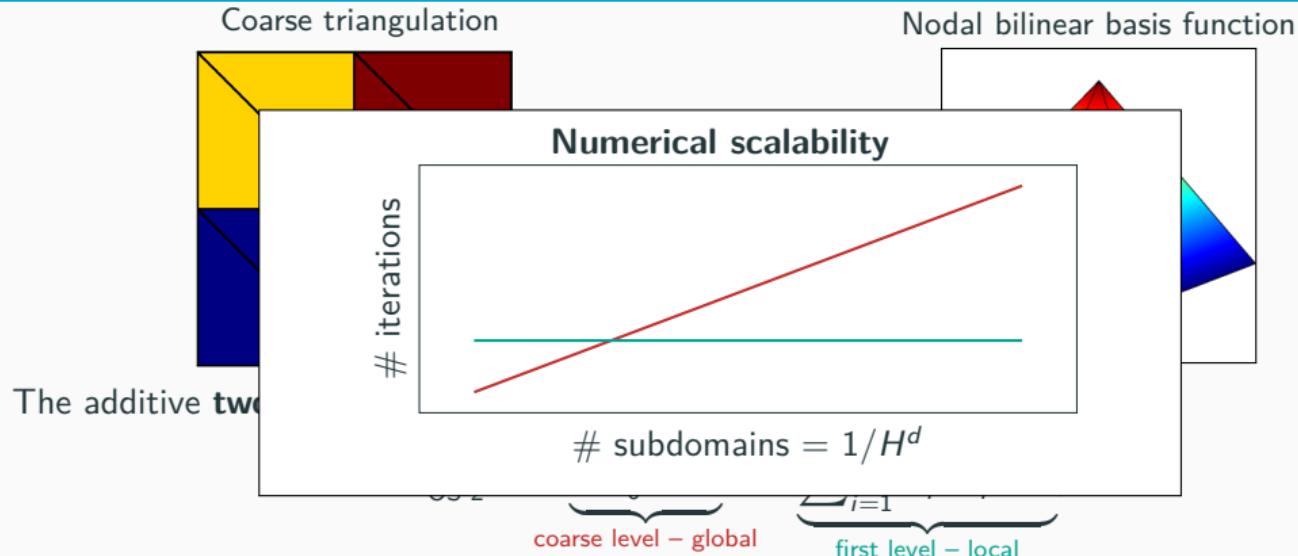
where Φ contains the coarse basis functions and $\mathbf{K}_0 := \Phi^T \mathbf{K} \Phi$.

Condition number bound:

$$\kappa(\mathbf{M}_{\text{OS-2}}^{-1} \mathbf{K}) \leq C \left(1 + \frac{H}{\delta} \right)$$

where the constant C is **independent of h , δ , and H** ; cf., e.g., [Toselli, Widlund \(2005\)](#).

8 Two-Level Overlapping Schwarz Preconditioners



where Φ contains the coarse basis functions and $\mathbf{K}_0 := \Phi^T \mathbf{K} \Phi$.

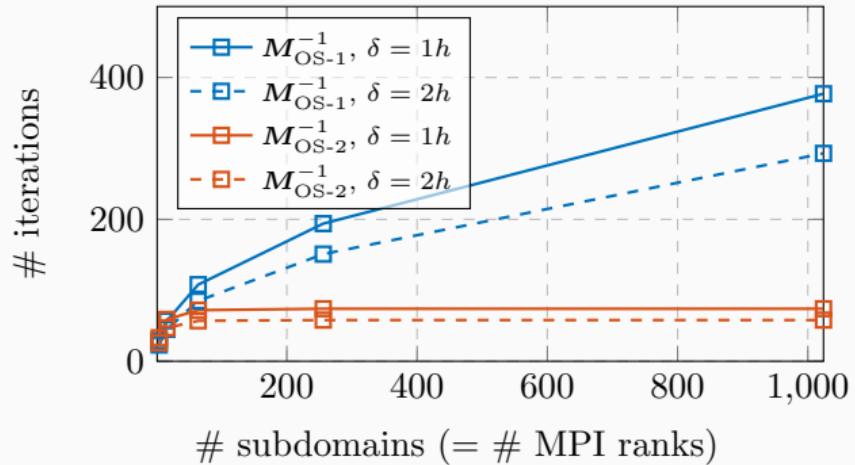
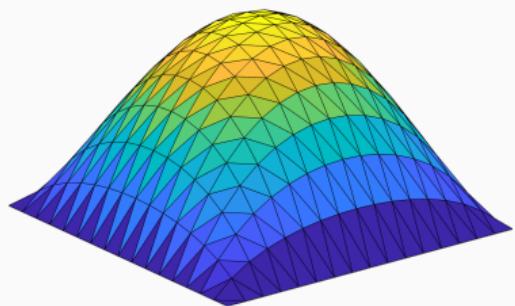
Condition number bound:

$$\kappa(\mathbf{M}_{\text{OS-2}}^{-1} \mathbf{K}) \leq C \left(1 + \frac{H}{\delta} \right)$$

where the constant C is **independent of h , δ , and H** ; cf., e.g., [Toselli, Widlund \(2005\)](#).

One- Vs Two-Level Schwarz Preconditioners

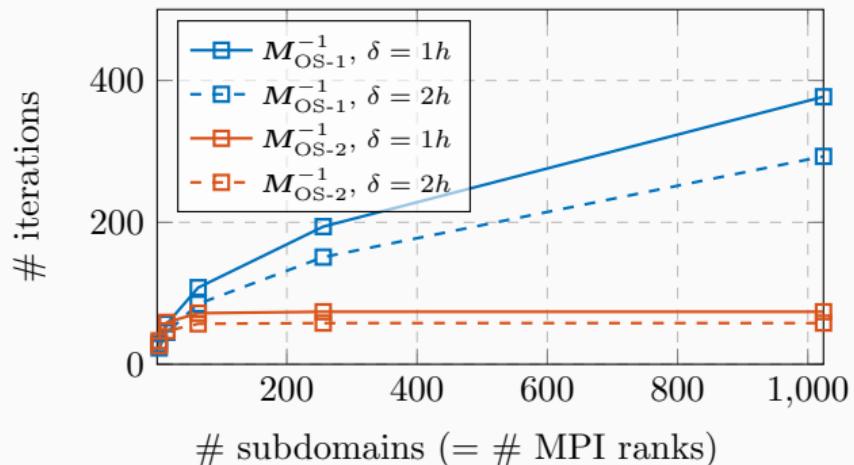
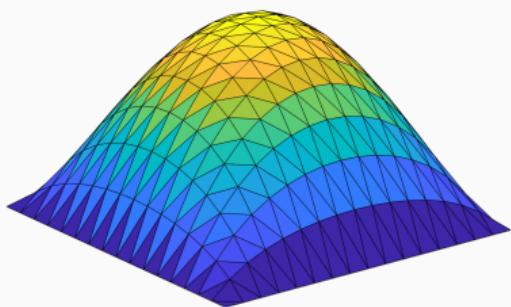
Diffusion model problem in two dimensions, # subdomains = # cores, $H/h = 100$



- We only obtain **numerical scalability** if a **coarse level** is used.
- Convergence is **faster** for larger overlaps.

One- Vs Two-Level Schwarz Preconditioners

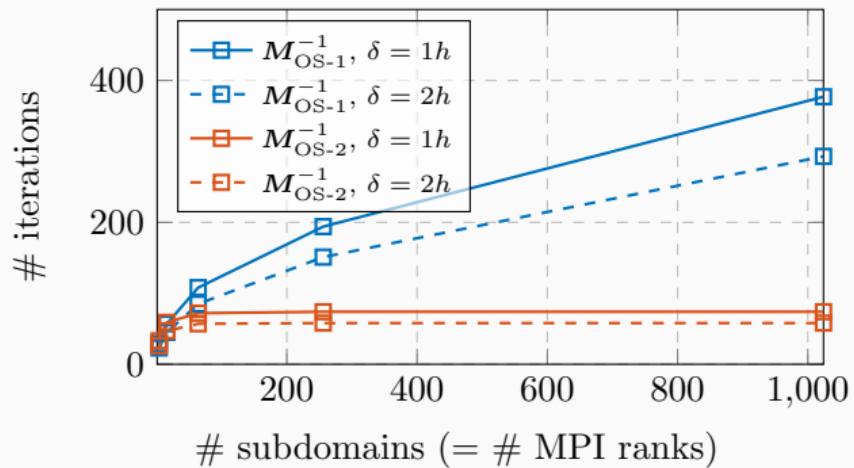
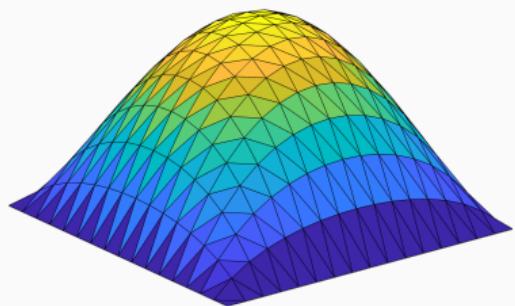
Diffusion model problem in two dimensions, # subdomains = # cores, $H/h = 100$



- We only obtain **numerical scalability** if a **coarse level** is used.
- Convergence is **faster** for larger overlaps.

One- Vs Two-Level Schwarz Preconditioners

Diffusion model problem in two dimensions, # subdomains = # cores, $H/h = 100$



- We only obtain **numerical scalability** if a **coarse level** is used.
- Convergence is **faster** for **larger overlaps**.

9 A Brief Overview Over the Theoretical Framework

In order to establish a condition number bound for $\kappa(M_{\text{ad}}^{-1}K)$ based on the **abstract Schwarz framework**, we have to verify the following **three assumptions**:

Assumption 1: Stable Decomposition

There exists a constant C_0 such that, for every $u \in V$, there exists a decomposition $u = \sum_{i=0}^N R_i^T u_i$, $u_i \in V_i$, with

$$\sum_{i=0}^N a_i(u_i, u_i) \leq C_0^2 a(u, u).$$

Assumption 2: Strengthened Cauchy-Schwarz Inequality

There exist constants $0 \leq \epsilon_{ij} \leq 1$, $1 \leq i, j \leq N$, such that

$$|a(R_i^T u_i, R_j^T u_j)| \leq \epsilon_{ij} \left(a(R_i^T u_i, R_i^T u_i) \right)^{1/2} \left(a(R_j^T u_j, R_j^T u_j) \right)^{1/2}$$

for $u_i \in V_i$ and $u_j \in V_j$. (Consider $\mathcal{E} = (\epsilon_{ij})$ and $\rho(\mathcal{E})$ its spectral radius)

Assumption 3: Local Stability

There exists $\omega < 0$, such that

$$a(R_i^T u_i, R_i^T u_i) \leq \omega a_i(u_i, u_i), \quad u_i \in \text{range}(\tilde{P}_i), \quad 0 \leq i \leq N.$$

General Condition Number Bound

With Assumption 1–3, we have

$$\kappa(M_{\text{ad}}^{-1} K) \leq C_0^2 \omega (\rho(\mathcal{E}) + 1)$$

for

$$M_{\text{ad}}^{-1} = \sum_{i=0/1}^N R_i^T K_i^{-1} R_i;$$

see, e.g., [Toselli, Wildund \(2005\)](#).

To obtain a condition number bound for a specific additive Schwarz preconditioner, we have to bound ω , $\rho(\mathcal{E})$, and C_0^2 .

The constants ω and $\rho(\mathcal{E})$ can often be handled easily.

Exact Solvers

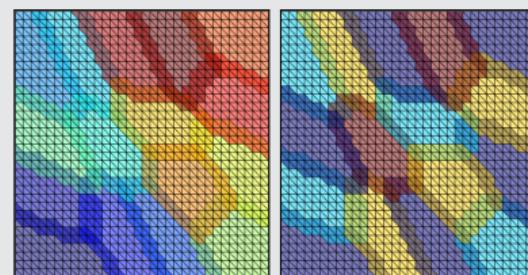
If we choose the local bilinear forms as

$$a_i(u_i, u_i) := a(R_i^T u_i, R_i^T u_i),$$

we obtain $K_i = R_i K R_i^T$ and $\omega = 1$.

→ For exact **exact local and coarse solvers**, ω does not depend on the coefficient.

Coloring Constant



The spectral radius $\rho(\mathcal{E})$ is bounded by the number of colors N^c of the domain decomposition.

→ N^c depends only on the **domain decomposition** but not on the coefficient function.

Assumption 3 is typically proved by constructing functions $u_i \in V_i$, $i = 0, \dots, N$, such that

$$u = \sum_{i=0}^N R_i^T u_i \text{ and } \sum_{i=0}^N a_i(u_i, u_i) \leq C_0^2 a(u, u)$$

for any given function $u \in V$. Let us sketch the difference between the one- and two-level preconditioners.

One-level Schwarz preconditioner

During the proof of the condition number, we have to use an L^2 -norm using Friedrich's inequality globally on Ω :

$$\sum_{i=1}^N \|u\|_{L_2(\Omega_i)}^2 = \|u\|_{L_2(\Omega)}^2 \leq C |u|_{H^1(\Omega)}^2,$$

This results in

$$\sum_{i=1}^N a_i(u_i, u_i) \leq C \left(1 + \frac{H}{\delta}\right) a(u, u) + C \frac{1}{H\delta} a(u, u)$$

Since $\frac{H}{\delta} \leq \frac{1}{H\delta}$, we obtain

$$\sum_{i=1}^N a_i(u_i, u_i) \leq C \left(1 + \frac{1}{H\delta}\right) a(u, u).$$

Two-level Schwarz preconditioner

In contrast to the one-level method, we can estimate the L^2 -norm locally since we instead have the term $u - u_0$

$$\sum_{i=1}^N \|u - u_0\|_{L_2(\Omega'_i)}^2 \leq \sum_{i=1}^N CH^2 |u|_{H^1(\omega_{\Omega'_i})}^2.$$

Different from the one-level preconditioner, we obtain an H^2 term in the final estimate:

$$\begin{aligned} \sum_{i=1}^N a_i(u_i, u_i) &\leq C \left(1 + \frac{H}{\delta}\right) a(u, u) + C \frac{1}{H\delta} \textcolor{red}{H^2} a(u, u) \\ &\leq C \left(1 + \frac{H}{\delta}\right) a(u, u) \end{aligned}$$

10 Some Comments on Constructing Schwarz Preconditioners

Restricted Schwarz Preconditioner (Cai and Sarkis (1999))

Replace the prolongation \mathbf{R}_i^T by $\tilde{\mathbf{R}}_i^T$,

$$\mathbf{M}_{\text{OS-1}}^{-1} = \sum_{i=1}^N \tilde{\mathbf{R}}_i^T \mathbf{K}_i^{-1} \mathbf{R}_i,$$

where

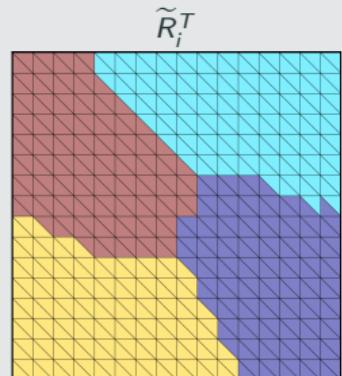
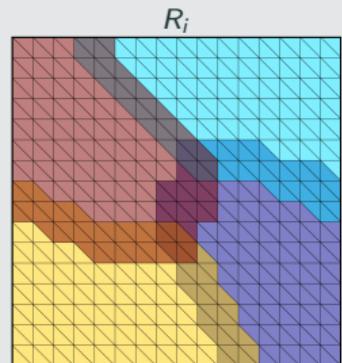
$$\sum_{i=1}^N \tilde{\mathbf{R}}_i^T = \mathbf{I}.$$

Therefore, we can just introduce a diagonal scaling matrix \mathbf{D} , such that

$$\tilde{\mathbf{R}}_i^T = \mathbf{D} \mathbf{R}_i^T,$$

for example based on a nonoverlapping domain decomposition or an inverse multiplicity scaling.

This often **improves the convergence**, however, the preconditioner becomes **unsymmetric**.



Changing the local and coarse solvers

For solving

$$\mathbf{K}_i^{-1}, \quad i = 0, \dots, N,$$

we can employ **inexact solvers** instead of direct solvers, such as

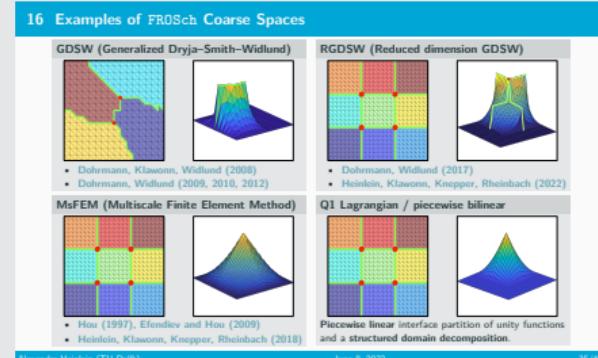
- iterative solvers
- preconditioners

to **speedup the computing times**. Of course, **convergence might slow down** a bit at the same time.

Choose another coarse basis

As it turns out, the choice of a **suitable coarse basis** is one of the more important ingredients for a **scalable and robust domain decomposition solver**.

We will discuss this again in a few slides.



11. Wishlist for a Parallel Schwarz Preconditioning Package

12. FROSCH (Fast and Robust Overlapping Schwarz) Framework in TRILINOS

13. Algorithmic Framework for FROSCH Coarse Spaces

14. Examples of FROSCH Coarse Spaces

15. Some Numerical Results

16. Exercises

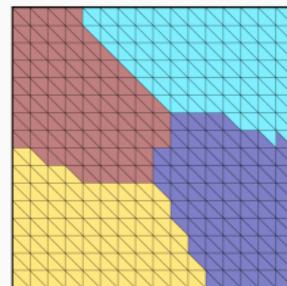
11 Wishlist for a Parallel Schwarz Preconditioning Package

Parallel distributed system

$$Ax = b$$

with

$$A = \begin{bmatrix} & & \\ \vdots & \ddots & \\ & & \end{bmatrix} \quad b = \begin{bmatrix} & \\ \vdots & \\ & \end{bmatrix}$$



Wishlist:

- **Parallel scalability** (includes numerical scalability)
- Usability → **algebraicity**
- **Generality**
- **Robustness**

12 FROSch (Fast and Robust Overlapping Schwarz) Framework in Trilinos



Software

- Object-oriented C++ domain decomposition solver framework with MPI-based distributed memory parallelization
- Part of TRILINOS with support for both parallel linear algebra packages EPETRA and TPETRA
- Node-level parallelization and performance portability on CPU and GPU architectures through KOKKOS and KOKKOSKERNELS
- Accessible through unified TRILINOS solver interface STRATIMIKOS

Methodology

- Parallel scalable multi-level Schwarz domain decomposition preconditioners
- Algebraic construction based on the parallel distributed system matrix
- Extension-based coarse spaces

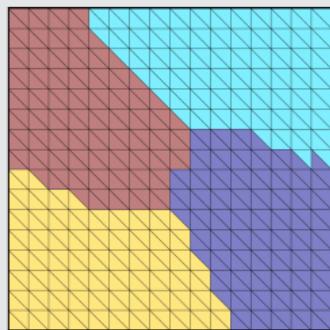
Team (active)

- | | |
|---|--|
| <ul style="list-style-type: none">▪ Alexander Heinlein (TU Delft)▪ Siva Rajamanickam (Sandia)▪ Friederike Röver (TUBAF) | <ul style="list-style-type: none">▪ Axel Klawonn (Uni Cologne)▪ Oliver Rheinbach (TUBAF)▪ Ichitaro Yamazaki (Sandia) |
|---|--|

Algorithmic Framework for FROSCH Overlapping Domain Decompositions

Overlapping domain decomposition

In FROSCH, the overlapping subdomains $\Omega'_1, \dots, \Omega'_N$ are constructed by **recursively adding layers of elements** to the nonoverlapping subdomains; this can be performed based on the sparsity pattern of K .

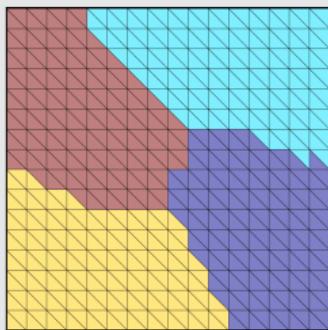


Nonoverlapping DD

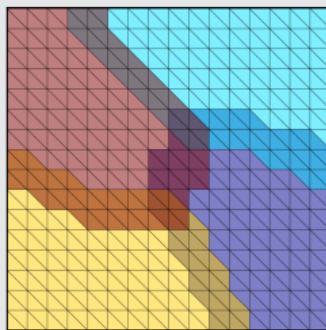
Algorithmic Framework for FROSCH Overlapping Domain Decompositions

Overlapping domain decomposition

In FROSCH, the overlapping subdomains $\Omega'_1, \dots, \Omega'_N$ are constructed by **recursively adding layers of elements** to the nonoverlapping subdomains; this can be performed based on the sparsity pattern of K .



Nonoverlapping DD

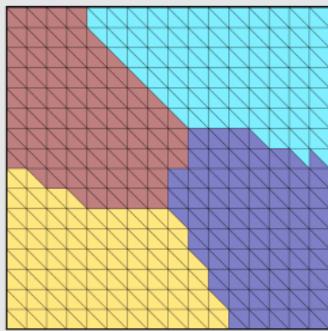


Overlap $\delta = 1h$

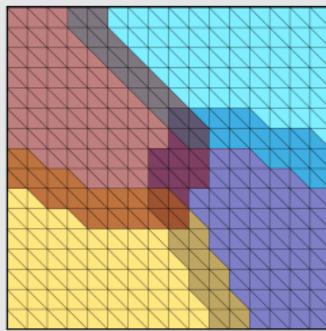
Algorithmic Framework for FROSCH Overlapping Domain Decompositions

Overlapping domain decomposition

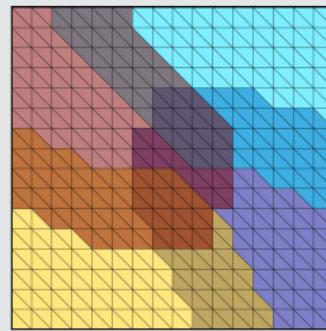
In FROSCH, the overlapping subdomains $\Omega'_1, \dots, \Omega'_N$ are constructed by **recursively adding layers of elements** to the nonoverlapping subdomains; this can be performed based on the sparsity pattern of K .



Nonoverlapping DD



Overlap $\delta = 1h$

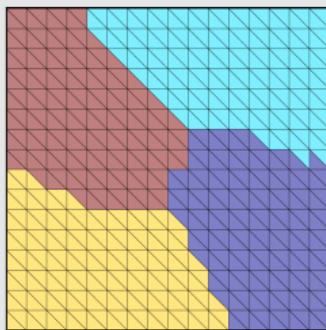


Overlap $\delta = 2h$

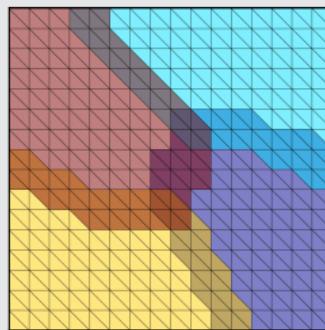
Algorithmic Framework for FROSCH Overlapping Domain Decompositions

Overlapping domain decomposition

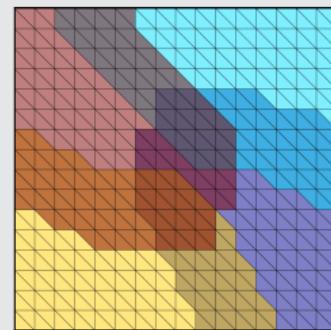
In FROSCH, the overlapping subdomains $\Omega'_1, \dots, \Omega'_N$ are constructed by **recursively adding layers of elements** to the nonoverlapping subdomains; this can be performed based on the sparsity pattern of K .



Nonoverlapping DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$

Computation of the overlapping matrices

The overlapping matrices

$$\mathbf{K}_i = \mathbf{R}_i \mathbf{K} \mathbf{R}_i^T$$

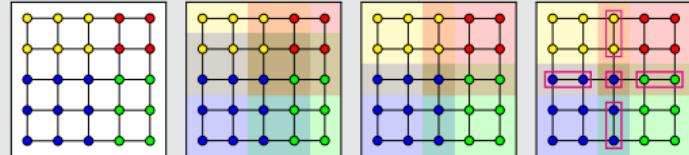
can easily be extracted from K since R_i is just a **global-to-local index mapping**.

13 Algorithmic Framework for FROSch Coarse Spaces

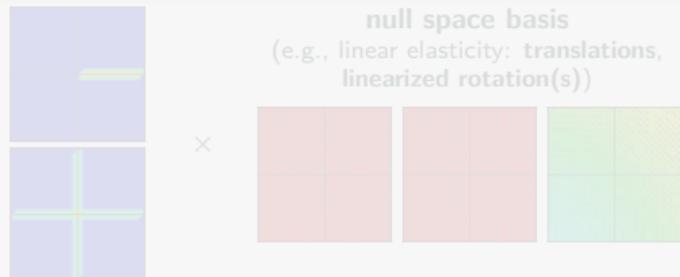
1. Identification interface components

$$K = \begin{bmatrix} & \\ & \\ & \\ & \\ & \end{bmatrix} \quad f = \begin{bmatrix} & \\ & \\ & \\ & \\ & \end{bmatrix}$$

Identification from parallel distribution of matrix:
distributed map overlapping map repeated map interface comp.



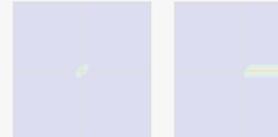
3. Interface basis



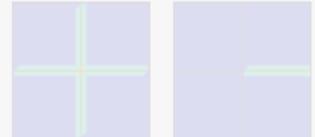
The interface values of the basis of the coarse space is obtained by **multiplication with the null space**.

2. Interface partition of unity (IPOU)

vertex & edge functions



vertex functions



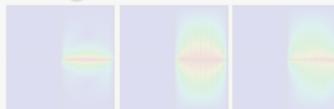
Based on the interface components,
construct an **interface partition of unity**:

$$\sum_i \pi_i = 1 \text{ on } \Gamma$$



4. Extension into the interior

edge basis function



vertex basis function



The values in the interior of the subdomains are computed via the **extension operator**:

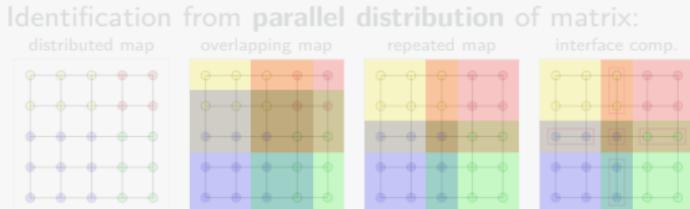
$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -K_{II}^{-1} K_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}.$$

(For elliptic problems: energy-minimizing extension)

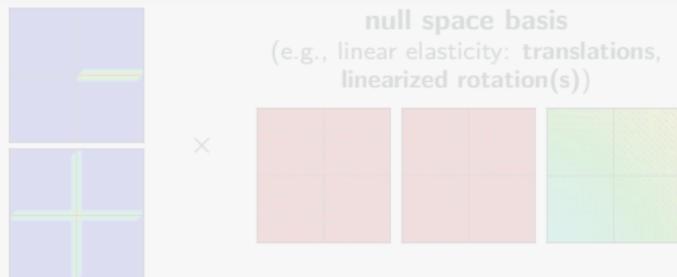
13 Algorithmic Framework for FROSch Coarse Spaces

1. Identification interface components

$$K = \begin{bmatrix} & & \\ & \cdot & \end{bmatrix} \quad f = \begin{bmatrix} & & \\ & \cdot & \end{bmatrix}$$



3. Interface basis



The interface values of the basis of the coarse space is obtained by **multiplication with the null space**.

2. Interface partition of unity (IPOU)

vertex & edge functions

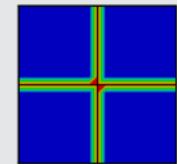


vertex functions



Based on the interface components, construct an **interface partition of unity**:

$$\sum_i \pi_i = 1 \text{ on } \Gamma$$



4. Extension into the interior

edge basis function



vertex basis function

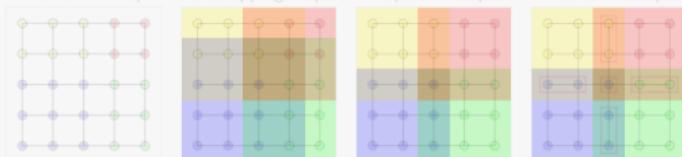


The values in the interior of the subdomains are computed via the extension operator:

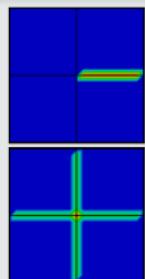
$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -K_{II}^{-1} K_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}.$$

(For elliptic problems: energy-minimizing extension)

13 Algorithmic Framework for FROSch Coarse Spaces



3. Interface basis

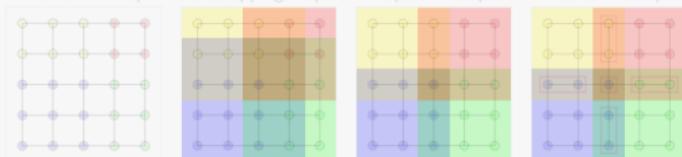


null space basis
(e.g., linear elasticity: translations,
linearized rotation(s))



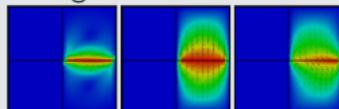
The interface values of the basis of the coarse space is obtained by **multiplication with the null space**.

13 Algorithmic Framework for FROSch Coarse Spaces

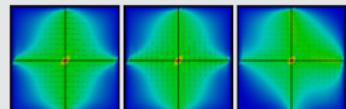


4. Extension into the interior

edge basis function



vertex basis function



The values in the interior of the subdomains are computed via the **extension operator**:

$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -K_{II}^{-1} K_{I\Gamma}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}.$$

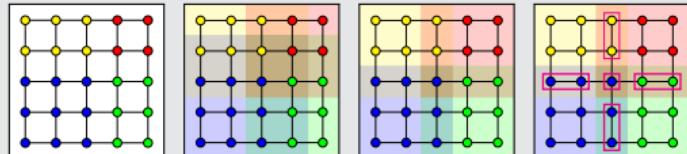
(For elliptic problems: energy-minimizing extension)

13 Algorithmic Framework for FROSch Coarse Spaces

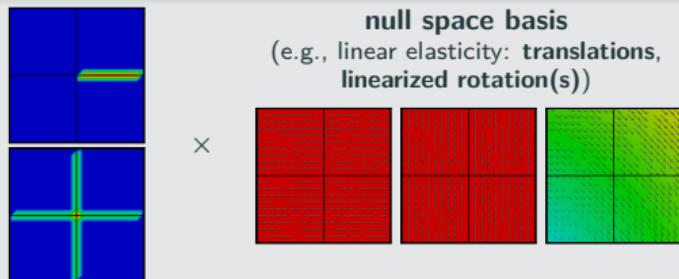
1. Identification interface components

$$K = \begin{bmatrix} & & \\ & \vdots & \\ & & \end{bmatrix} \quad f = \begin{bmatrix} & \\ & \vdots \\ & \end{bmatrix}$$

Identification from parallel distribution of matrix:
distributed map overlapping map repeated map interface comp.



3. Interface basis



The interface values of the basis of the coarse space is obtained by **multiplication with the null space**.

2. Interface partition of unity (IPOU)

vertex & edge functions

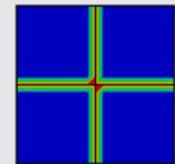


vertex functions



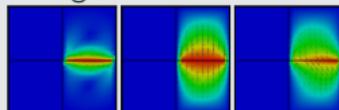
Based on the interface components, construct an **interface partition of unity**:

$$\sum_i \pi_i = 1 \text{ on } \Gamma$$

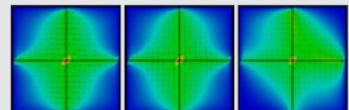


4. Extension into the interior

edge basis function



vertex basis function



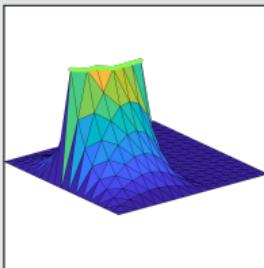
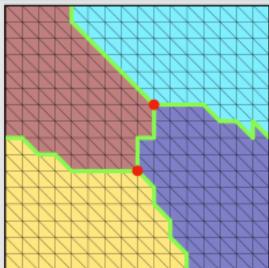
The values in the interior of the subdomains are computed via the **extension operator**:

$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -K_{II}^{-1} K_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}.$$

(For elliptic problems: energy-minimizing extension)

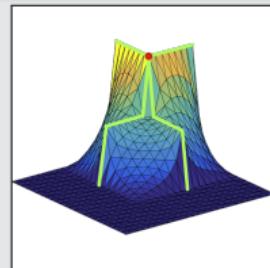
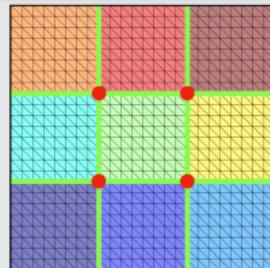
14 Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)



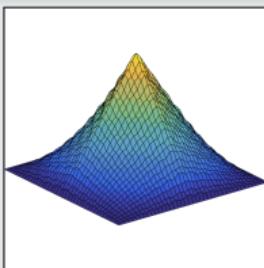
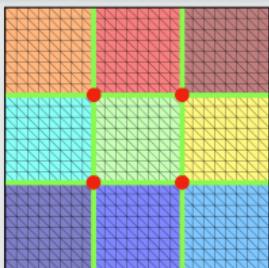
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)



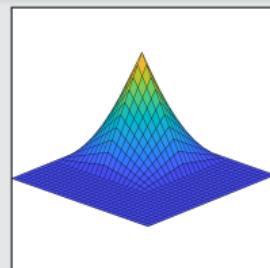
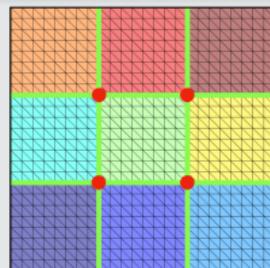
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

MsFEM (Multiscale Finite Element Method)



- Hou (1997), Efendiev and Hou (2009)
- Buck, Iliev, and Andrä (2013)
- H., Klawonn, Knepper, Rheinbach (2018)

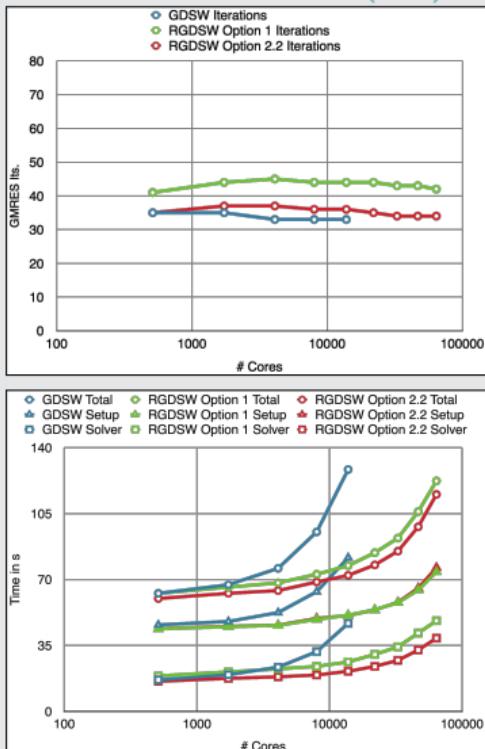
Q1 Lagrangian / piecewise bilinear



Piecewise linear interface partition of unity functions and a structured domain decomposition.

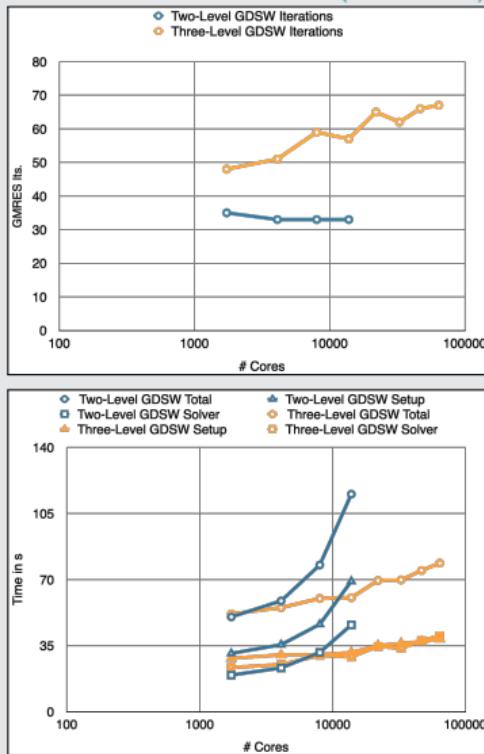
GDSW vs RGDSW (reduced dimension)

Heinlein, Klawonn, Rheinbach, Widlund (2019).



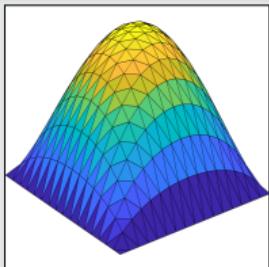
Two-level vs three-level GDSW

Heinlein, Klawonn, Rheinbach, Röver (2019, 2020).

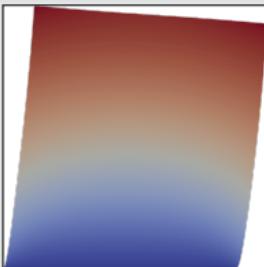


Exemplary Applications

Elliptic problems

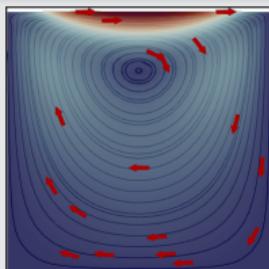


Diffusion

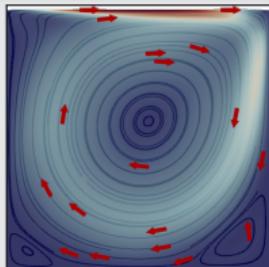


Elasticity

Fluid flow problems

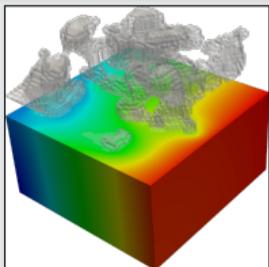


Stokes flow

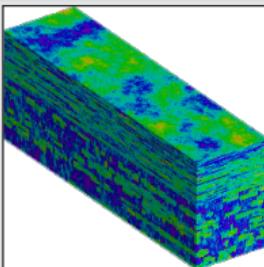


Navier–Stokes flow

Heterogeneous problems

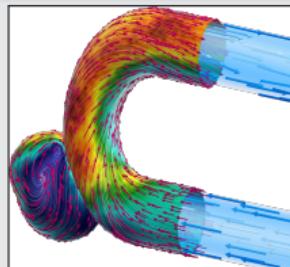


Dual-phase steel

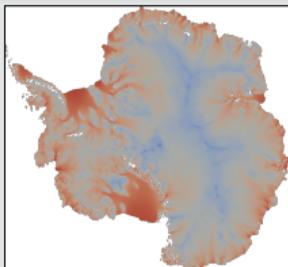


Groundwater flow

Multi-physics problems



Fluid–structure
interaction



Land ice simulations

16 Exercises

All the material for the exercises of this session can be found in the folder `lab2` of the **GitHub repository** of the summer school: <https://github.com/jthies/dcse-summerschool>

Important are the last two **exercises**

- exercise 3 – Implementing a One-Level Schwarz Preconditioner Using FROSCH
- exercise 4 – Implementing a GDSW Preconditioner Using FROSCH

and the **corresponding solutions** (in the subfolder `solution`),

Each exercise has **two parts**:

1. **Implement the missing code**; step-by-step explanations in the `README.md` files.
2. **Perform numerical experiments** to investigate the behavior of the methods.

Parallelization

The code assumes a **one-to-one correspondence of MPI ranks and subdomains**. In order to run with > 1 subdomains, you have to increase the number of MPI ranks. For instance, for 4 MPI ranks / subdomains: `mpirun -n 4 ./main.x`

Thank you for your attention!

Questions?

Want to **try out FROSch at home?**

→ <https://github.com/searhein/frosch-demo> for a **demo** with **simple installation via Docker**.