Algebraic Domain Decomposition Solvers for Large-Scale Problems Using Graph Techniques

Alexander Heinlein[1]

Graphs&Data@TUDelft Seminar, Delft University of Technology, April 3, 2025

[1]Delft University of Technology

Outline

1. Introduction to Schwarz Domain Decomposition Methods

2. The FROSch Package – Algebraic and Parallel Schwarz Preconditioners in Trilinos

   Based on joint work with

Axel Klawonn	(University of Cologne)
Siva Rajamanickam	(Sandia National Laboratories)
Oliver Rheinbach and **Friederike Röver**	(TU Bergakademie Freiberg)
Olof Widlund	(New York University)

3. Some Challenging Application Problems

   Based on joint work with

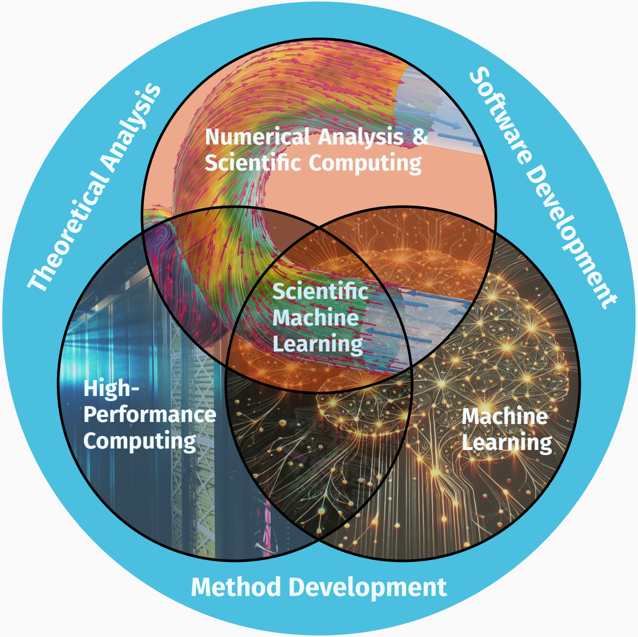
Filipe Cumaru and **Hadi Hajibeygi**	(Delft University of Technology)
Axel Klawonn, **Jascha Knepper**, and **Lea Saßmannshausen**	(University of Cologne)
Mauro Perego and **Siva Rajamanickam**	(Sandia National Laboratories)
Oliver Rheinbach	(TU Bergakademie Freiberg)
Kathrin Smetana	(Stevens Institute of Technology)
Olof Widlund	(New York University)

4. Learning Extension Operators Using Graph Neural Networks

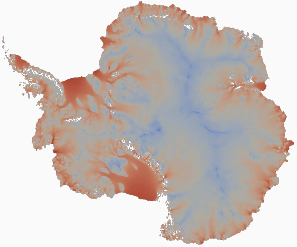
   Based on joint work with

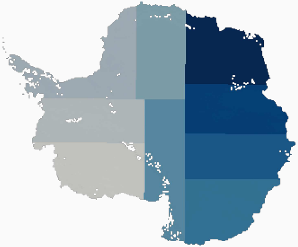
Siva Rajamanickam and **Ichitaro Yamazaki**	(Sandia National Laboratories)

SCaLA – Scalable Scientific Computing and Learning Algorithms



Introduction to Schwarz Domain Decomposition Methods

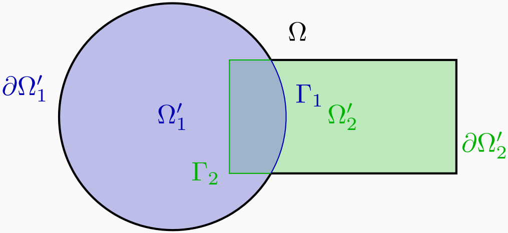




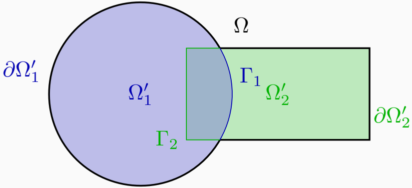
Graphics based on results from **Heinlein, Perego, Rajamanickam (2022)**

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 non-smooth boundaries**.



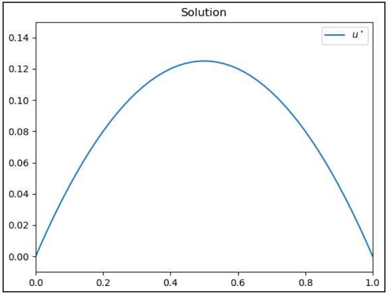
For the sake of simplicity, instead of the two-dimensional geometry,



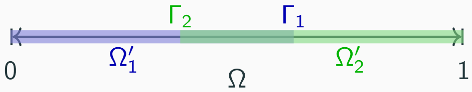
we consider the **one-dimensional Poisson equation**

$$
\begin{aligned}
-u'' &= 1 \quad \text{in } [0,1], \\
u(0) &= u(1) = 0.
\end{aligned}
$$

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



Overlapping 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**:

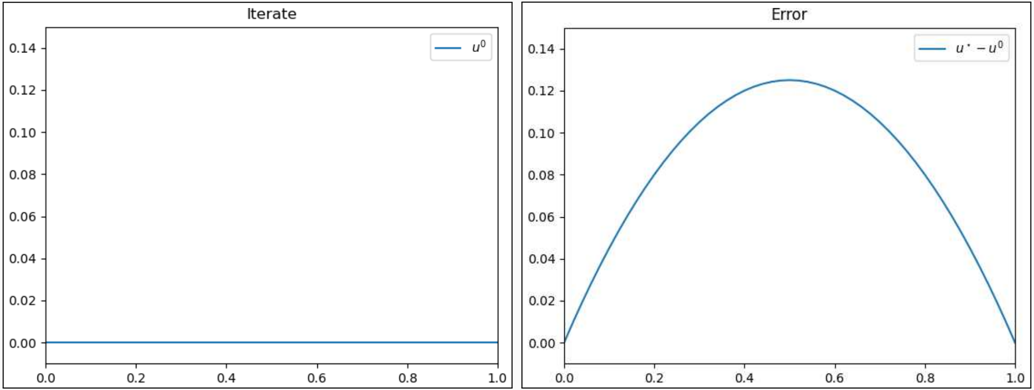


Figure 1: Iterate (left) and error (right) in iteration 0.

The Alternating Schwarz Algorithm – 1D Laplace Results

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**:

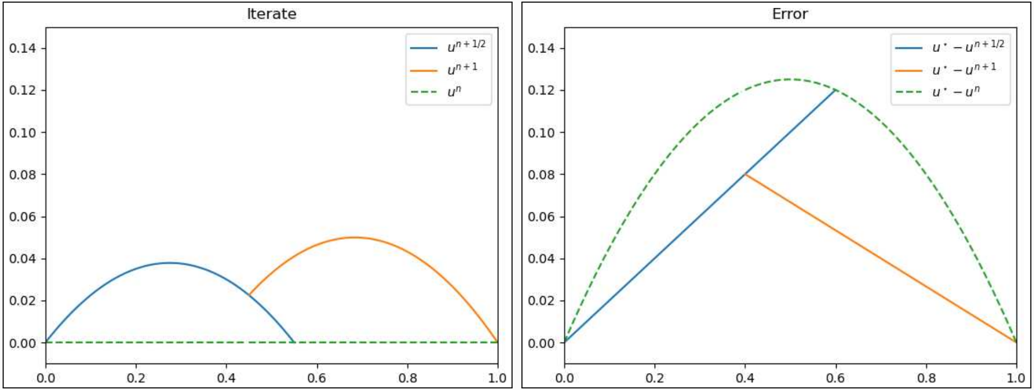


Figure 1: Iterate (left) and error (right) in iteration 1.

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**:

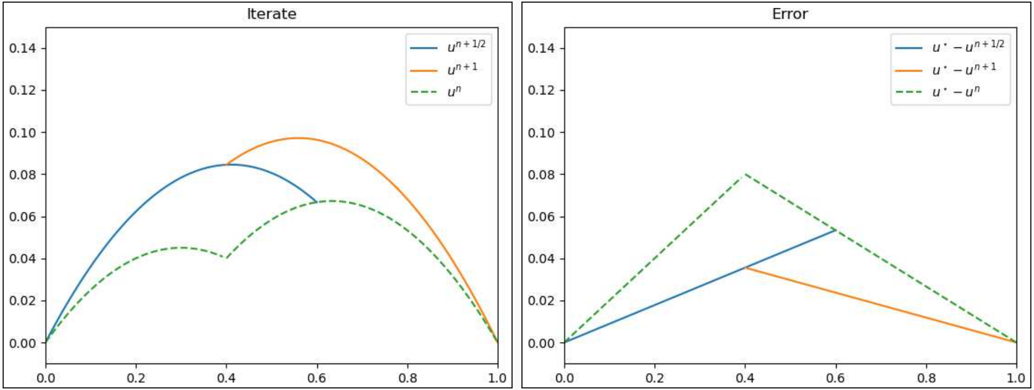


Figure 1: Iterate (left) and error (right) in iteration 2.

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**:

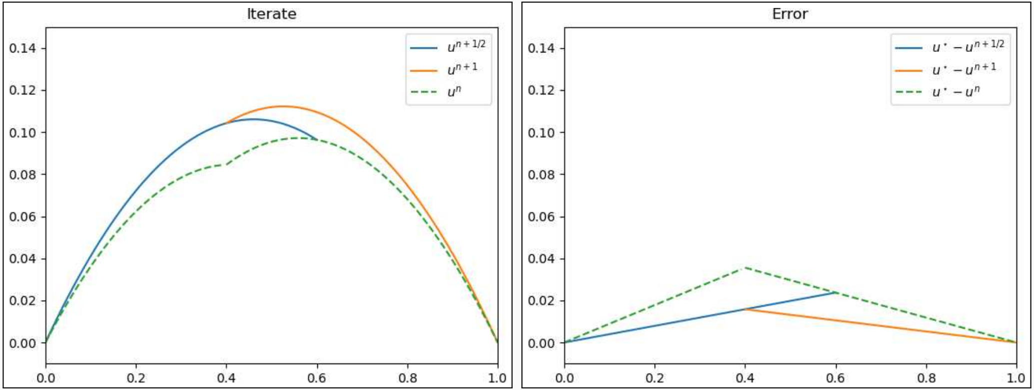


Figure 1: Iterate (left) and error (right) in iteration 3.

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**:

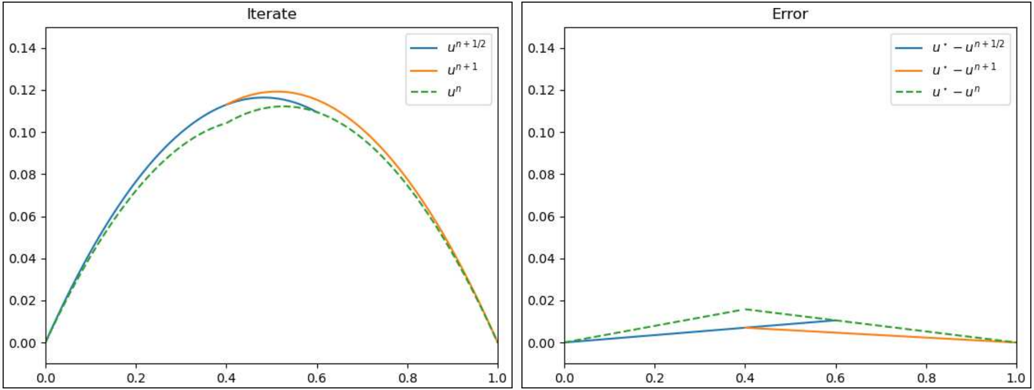


Figure 1: Iterate (left) and error (right) in iteration 4.

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**:

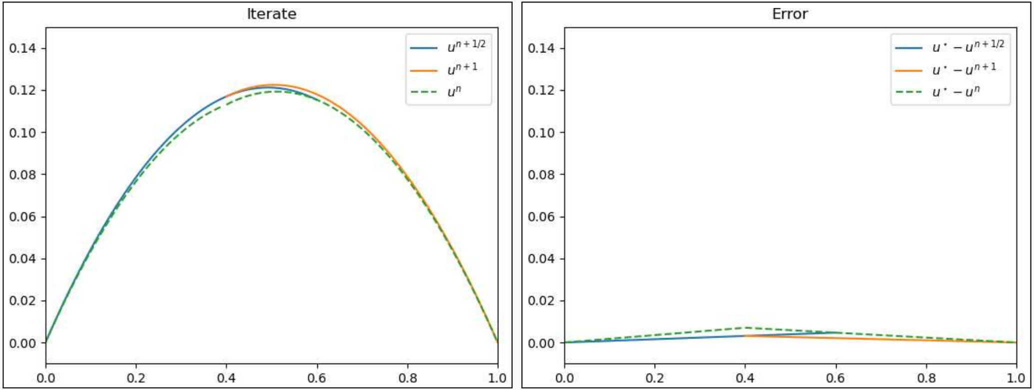


Figure 1: Iterate (left) and error (right) in iteration 5.

Sequential Nature of the Alternating Schwarz Algorithm

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**:

$$(D_1)\begin{cases} -\Delta u^{n+1/2} = f & \text{in } \Omega_1', \\ u^{n+1/2} = \mathbf{u^n} & \text{on } \partial\Omega_1' \\ u^{n+1/2} = \mathbf{u^n} & \text{on } \Omega \setminus \overline{\Omega_1'} \end{cases}$$

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

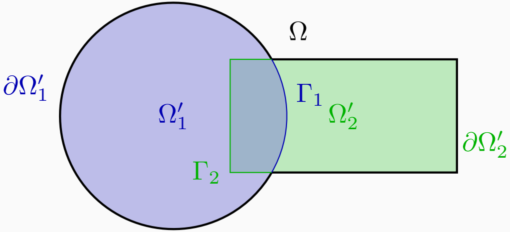
Idea: For all red terms, we **use the values from the previous iteration**. Then, the both Dirichlet problem **can be solved at the same time**.

The Parallel Schwarz Algorithm

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

$$(D_1)\begin{cases} -\Delta u_1^{n+1} &= f \quad \text{in } \Omega_1', \\ u_1^{n+1} &= u_2^n \quad \text{on } \partial\Omega_1', \end{cases}$$

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

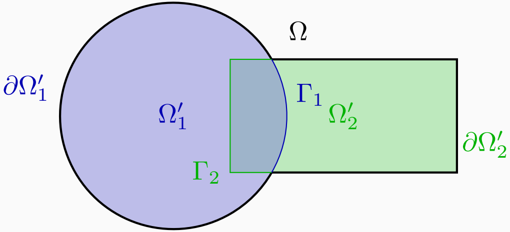


Since u_1^n and u_2^n are both computed in the previous iteration, the problems can be solved independent of each other.

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 a **parallel Schwarz iteration**:

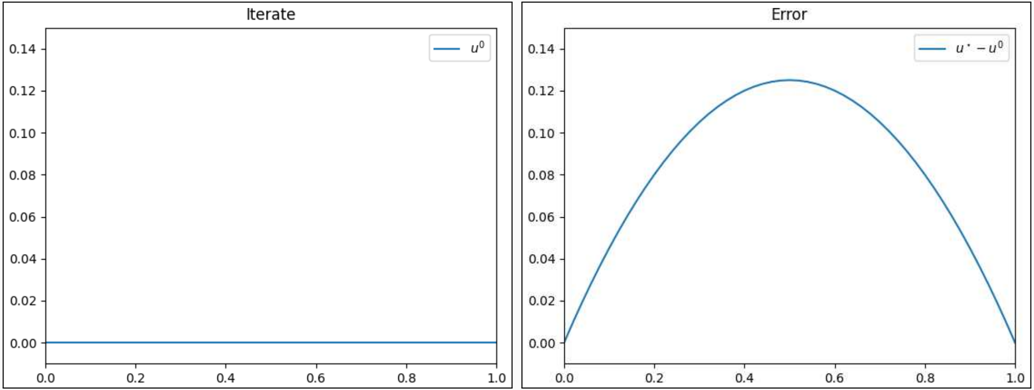


Figure 2: Iterate (left) and error (right) in iteration 0.

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 a **parallel Schwarz iteration**:

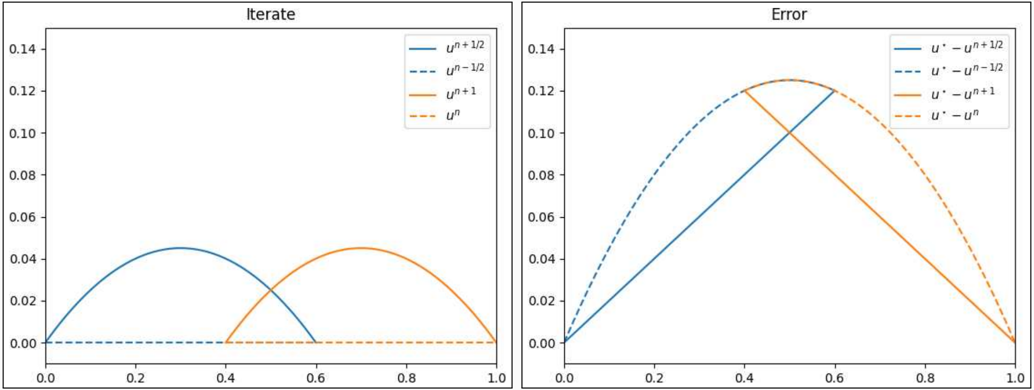


Figure 2: Iterate (left) and error (right) in iteration 1.

The Parallel Schwarz Algorithm – 1D Laplace Results

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 a **parallel Schwarz iteration**:

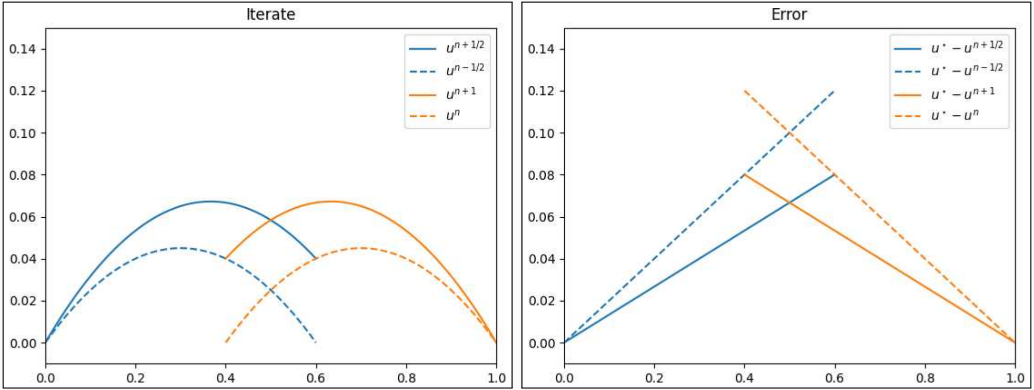


Figure 2: Iterate (left) and error (right) in iteration 2.

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 a **parallel Schwarz iteration**:

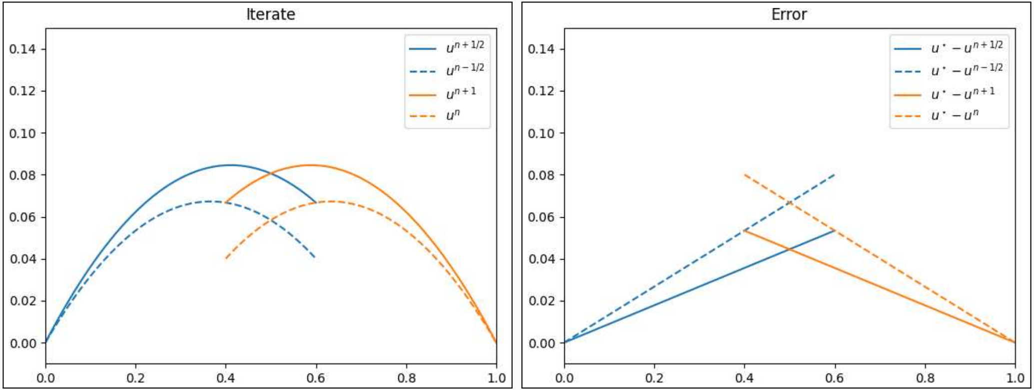


Figure 2: Iterate (left) and error (right) in iteration 3.

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 a **parallel Schwarz iteration**:

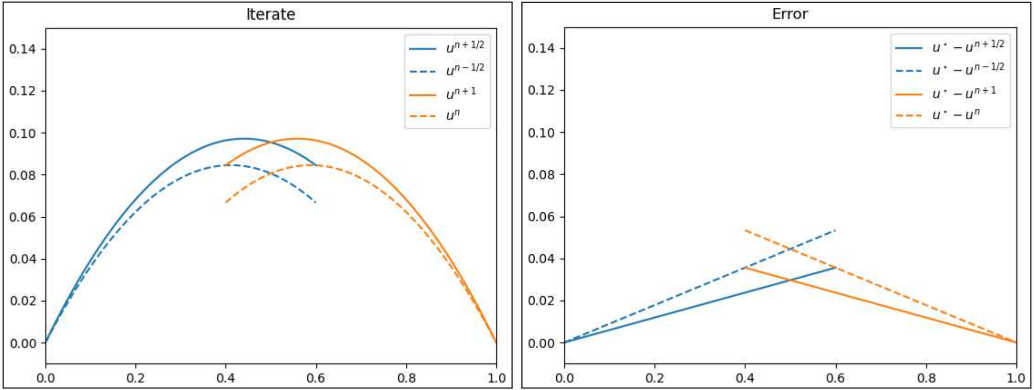


Figure 2: Iterate (left) and error (right) in iteration 4.

The Parallel Schwarz Algorithm – 1D Laplace Results

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 a **parallel Schwarz iteration**:

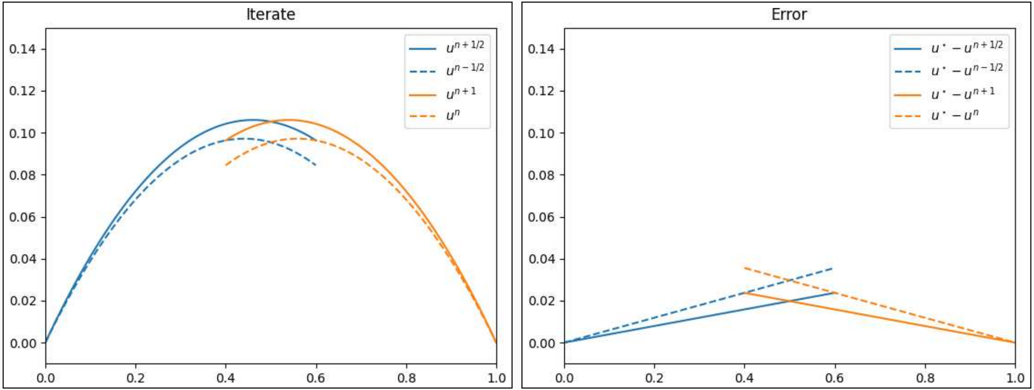
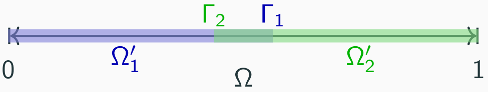


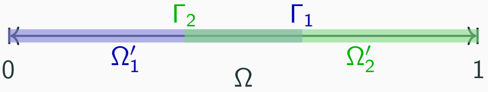
Figure 2: Iterate (left) and error (right) in iteration 5.

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**:

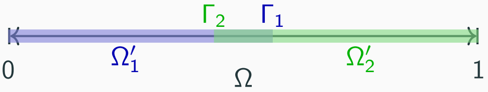


Overlap 0.05

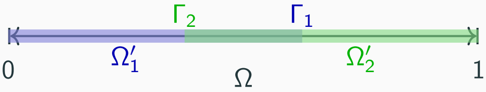


Overlap 0.1

Effect of the Size of the Overlap



Overlap 0.05



Overlap 0.1

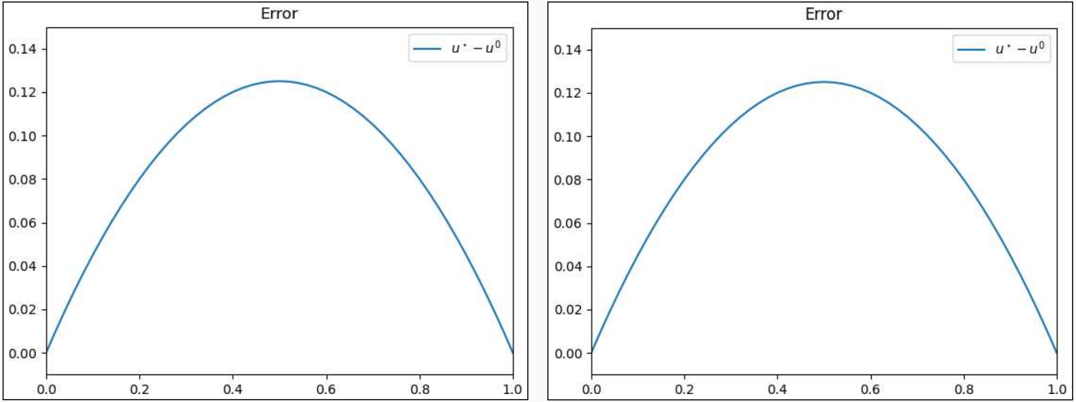
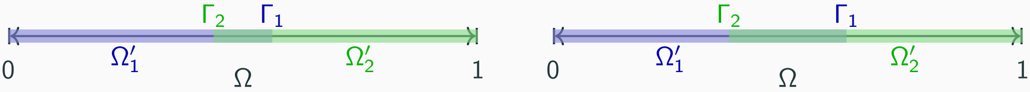


Figure 3: Error in iteration 0.



Overlap 0.05 | **Overlap 0.1**

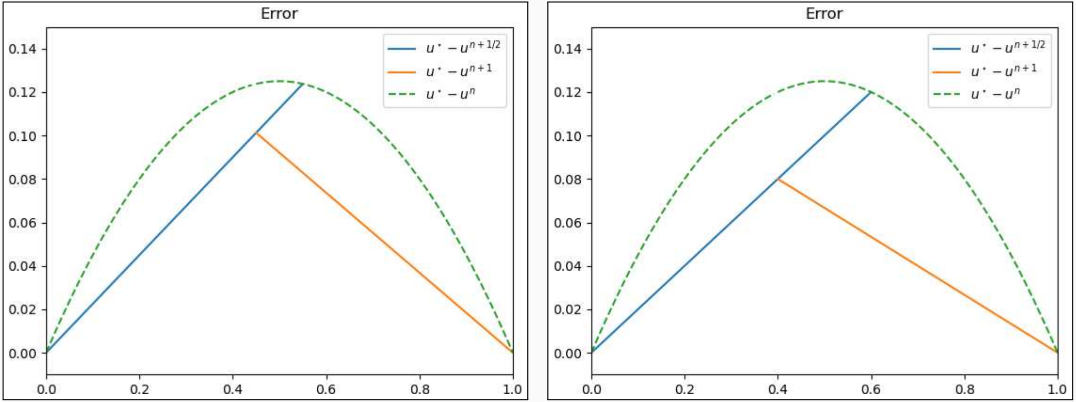
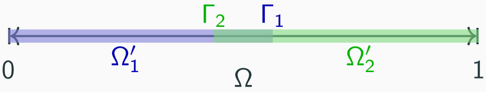
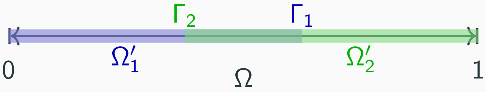


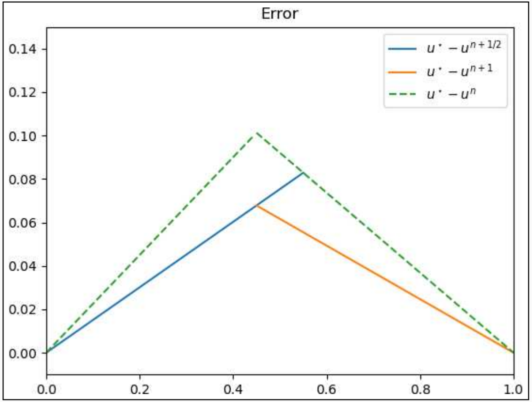
Figure 3: Error in iteration 1.



Overlap 0.05



Overlap 0.1



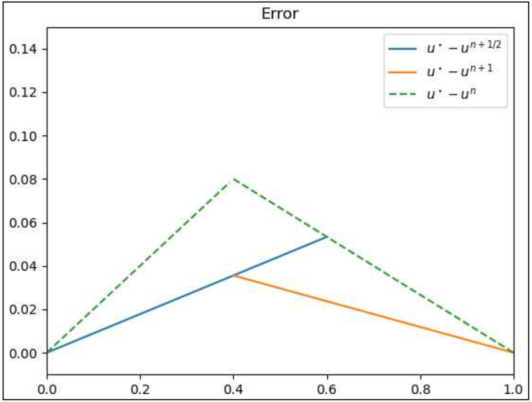
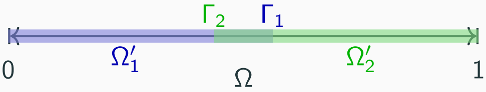
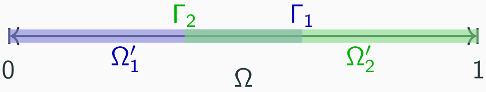


Figure 3: Error in iteration 2.

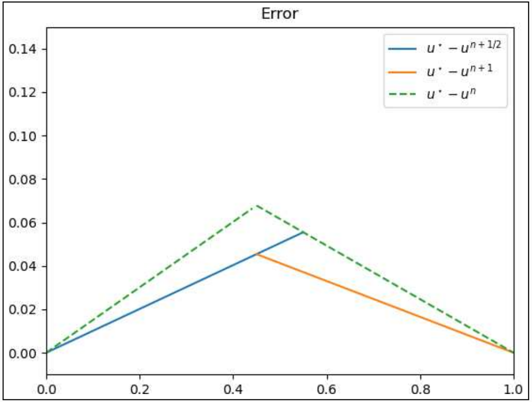
Effect of the Size of the Overlap





Overlap 0.05

Overlap 0.1



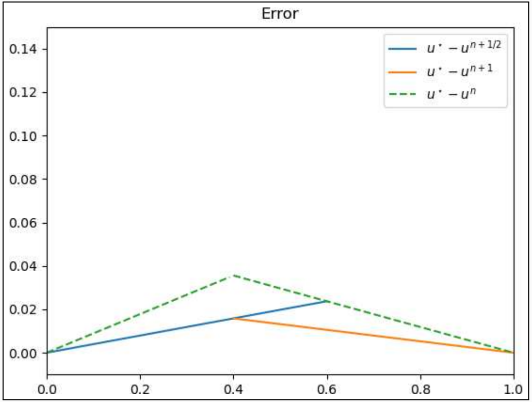
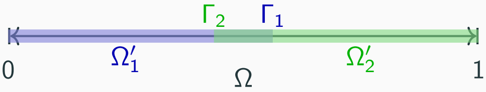
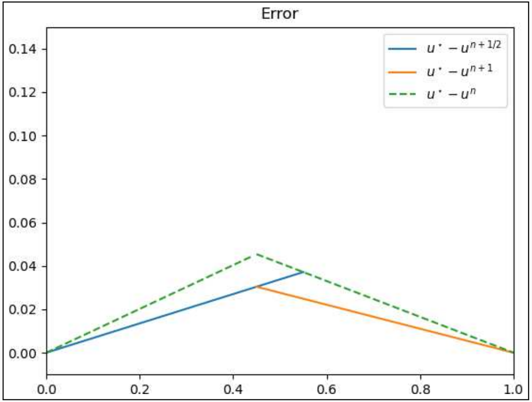
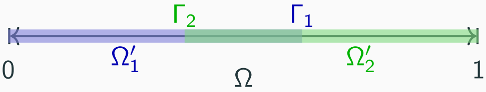


Figure 3: Error in iteration 3.



Overlap 0.05





Overlap 0.1

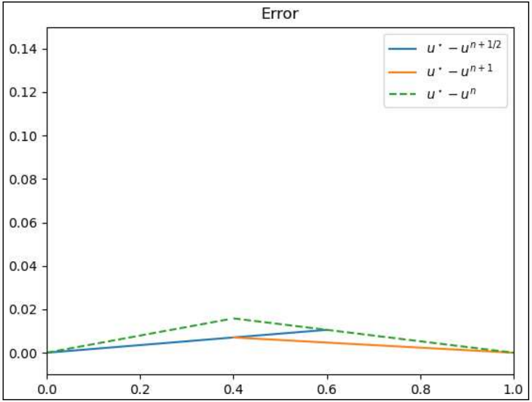
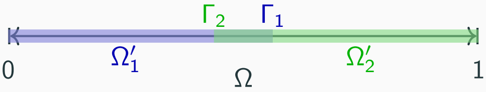
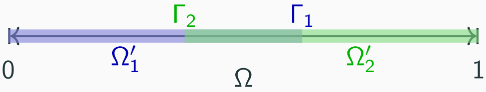


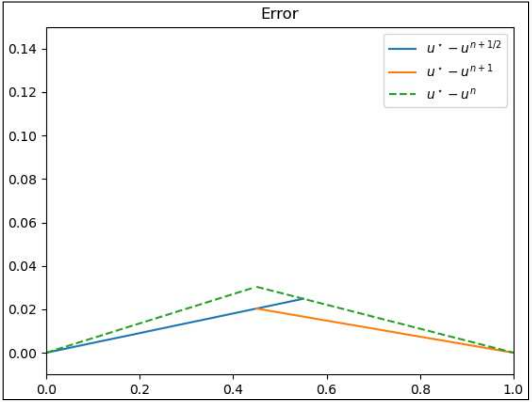
Figure 3: Error in iteration 4.





Overlap 0.05

Overlap 0.1



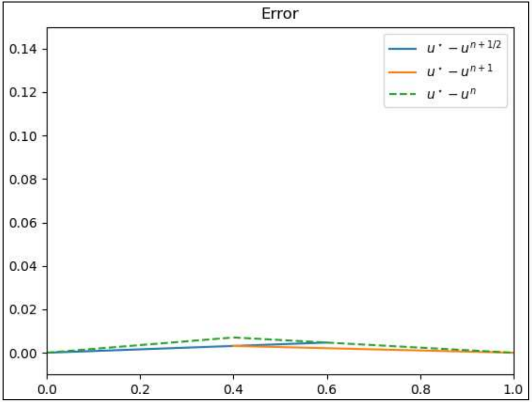
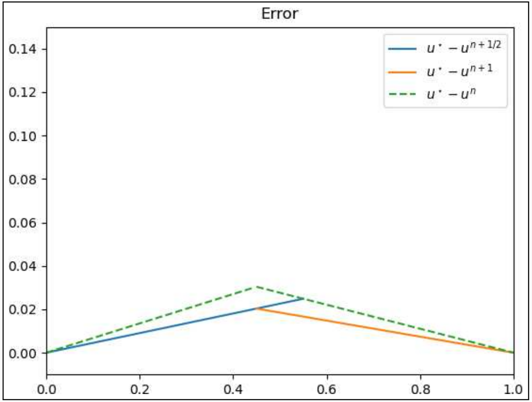


Figure 3: Error in iteration 5.

Overlap 0.05



Overlap 0.1

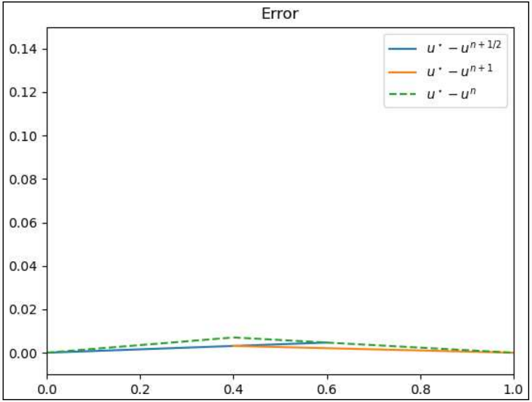


Figure 3: Error in iteration 5.

$\Rightarrow$ A **larger overlap** leads to **faster convergence**.

Solvers for Partial Different Equations

Consider a **diffusion model problem**:

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

Discretization using finite elements yields a **sparse** system of linear equations

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}.$$

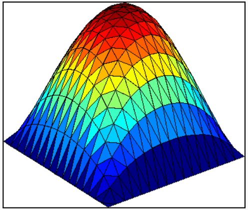
The accuracy of the finite element solution depends on the refinement level of the mesh h: **higher refinement** $\Rightarrow$ **better accuracy**.

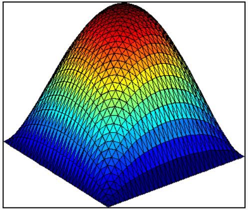
Direct solvers

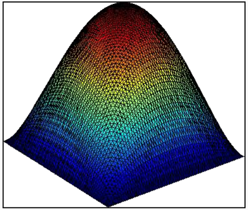
For fine meshes, solving the system using a direct solver is not feasible due to **superlinear complexity and memory cost**.

Iterative solvers

Iterative solvers are efficient for solving **sparse systems**, however, the **convergence rate depends on the spectral properties of $\boldsymbol{K}$**.



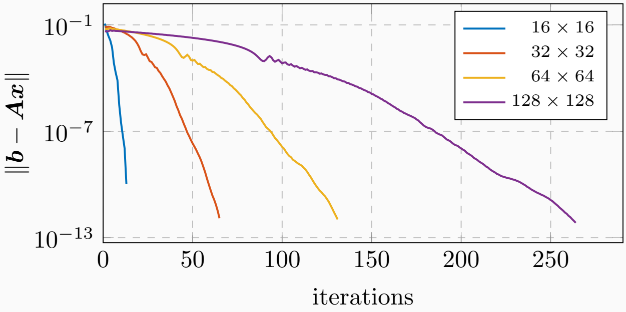


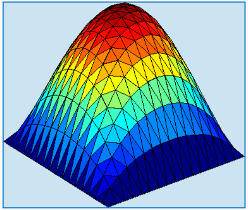


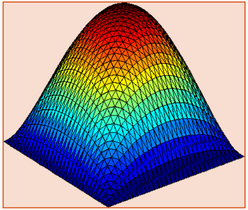
Consider a **diffusion model problem**:

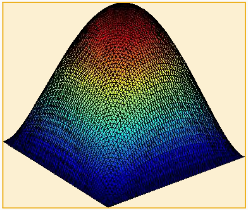
$$-\Delta u(x) = f \quad \text{in } \Omega = [0,1]^2,$$
$$u = 0 \quad \text{on } \partial\Omega.$$

We solve $\boldsymbol{Ku} = \boldsymbol{f}$ using the **conjugate gradient (CG) method**:



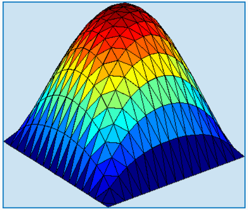


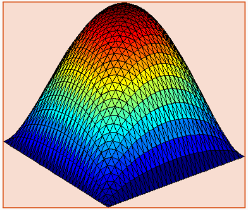


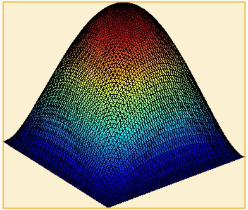


Consider a **diffusion model problem**:

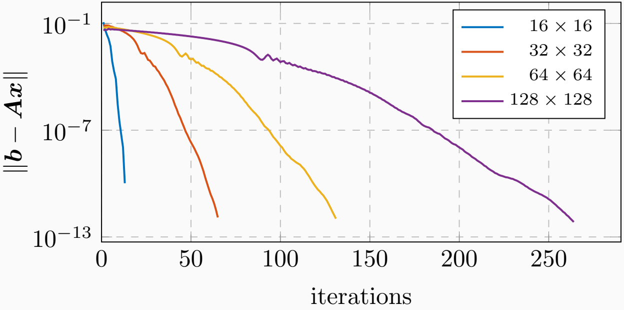
$$-\Delta u(x) = f \quad \text{in } \Omega = [0,1]^2,$$
$$u = 0 \quad \text{on } \partial\Omega.$$







We solve $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$ using the **conjugate gradient (CG) method**:

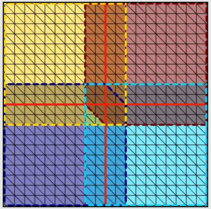


$\Rightarrow$ Introduce a preconditioner $\boldsymbol{M}^{-1} \approx \boldsymbol{K}^{-1}$ to **improve convergence**:

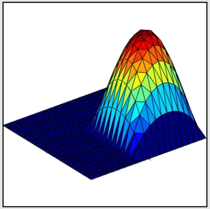
$$\boldsymbol{M}^{-1}\boldsymbol{K}\boldsymbol{u} = \boldsymbol{M}^{-1}\boldsymbol{f}$$

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



Based on an **overlapping domain decomposition**, we define a **one-level Schwarz operator**

$$M_{\text{OS-1}}^{-1} K = \sum_{i=1}^{N} R_i^\top K_i^{-1} R_i K,$$

where R_i and $R_i^\top$ are restriction and prolongation operators corresponding to Ω_i', and $K_i := R_i K R_i^\top$.

Condition number estimate:

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

with subdomain size H and overlap width δ.

Lagrangian coarse space

Coarse triangulation

Coarse solution

The **two-level overlapping Schwarz operator** reads

$$M_{\text{OS-2}}^{-1} K = \underbrace{\Phi K_0^{-1} \Phi^\top K}_{\text{coarse level – global}} + \underbrace{\sum_{i=1}^{N} R_i^\top K_i^{-1} R_i K}_{\text{first level – local}},$$

where Φ contains the coarse basis functions and $K_0 := \Phi^\top K \Phi$; cf., e.g., **Toselli, Widlund (2005)**. The construction of a Lagrangian coarse basis requires a coarse triangulation.

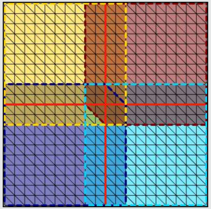
Condition number estimate:

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

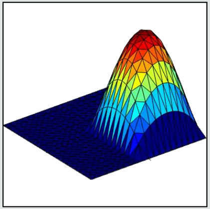
Two-Level Schwarz Preconditioners

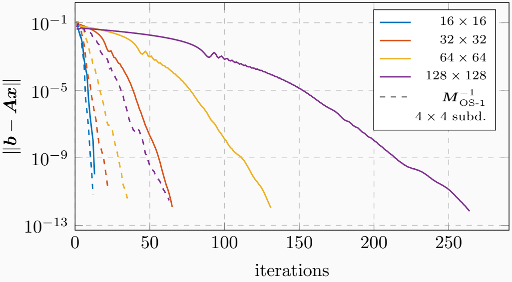
One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem





Lagrangian coarse space

Coarse triangulation

Coarse solution

The **two-level overlapping Schwarz operator** reads

$$M_{\text{OS-2}}^{-1}K = \underbrace{\Phi K_0^{-1}\Phi^\top K}_{\text{coarse level -- global}} + \underbrace{\sum_{i=1}^{N} R_i^\top K_i^{-1} R_i K}_{\text{first level -- local}},$$

where Φ contains the coarse basis functions and $K_0 := \Phi^\top K \Phi$; cf., e.g., **Toselli, Widlund (2005)**.
The construction of a Lagrangian coarse basis requires a coarse triangulation.

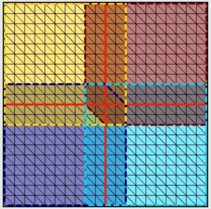
Condition number estimate:

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

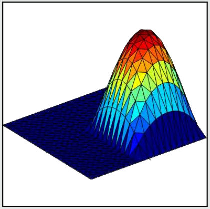
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



Based on an **overlapping domain decomposition**, we define a **one-level Schwarz operator**

$$M_{\text{OS-1}}^{-1} K = \sum_{i=1}^{N} R_i^\top K_i^{-1} R_i K,$$

where R_i and $R_i^\top$ are restriction and prolongation operators corresponding to Ω_i', and $K_i := R_i K R_i^\top$.

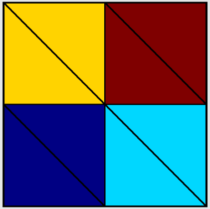
Condition number estimate:

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

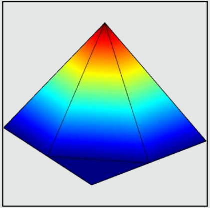
with subdomain size H and overlap width δ.

Lagrangian coarse space

Coarse triangulation



Coarse solution



The **two-level overlapping Schwarz operator** reads

$$M_{\text{OS-2}}^{-1} K = \underbrace{\Phi K_0^{-1} \Phi^\top K}_{\text{coarse level -- global}} + \underbrace{\sum_{i=1}^{N} R_i^\top K_i^{-1} R_i K}_{\text{first level -- local}},$$

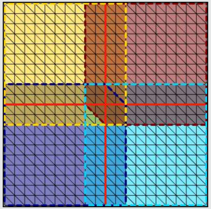
where Φ contains the coarse basis functions and $K_0 := \Phi^\top K \Phi$; cf., e.g., **Toselli, Widlund (2005)**. The construction of a Lagrangian coarse basis requires a coarse triangulation.

Condition number estimate:

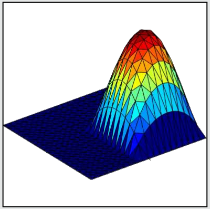
$$\kappa\left(M_{\text{OS-2}}^{-1} K\right) \leq C \left(1 + \frac{H}{\delta}\right)$$

One-level Schwarz preconditioner

Overlap $\delta = 1h$

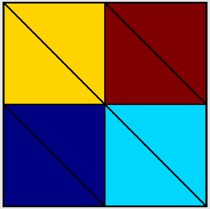


Solution of local problem

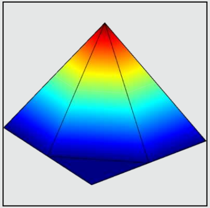


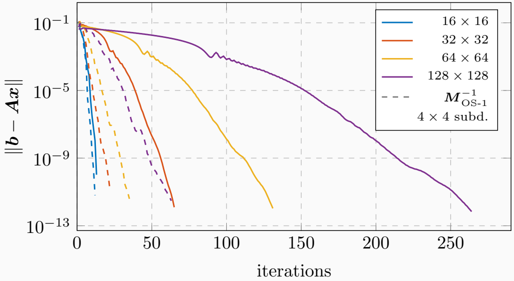
Lagrangian coarse space

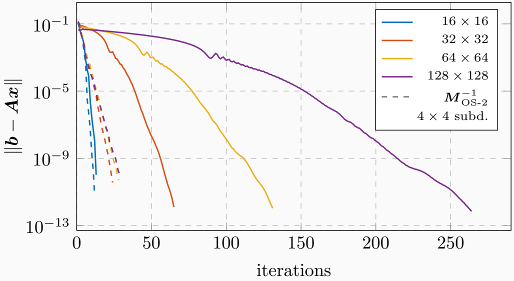
Coarse triangulation



Coarse solution

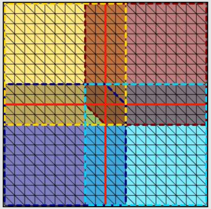




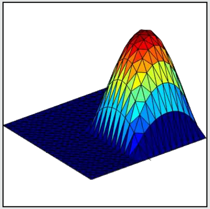


One-level Schwarz preconditioner

Overlap $\delta = 1h$

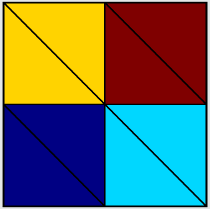


Solution of local problem

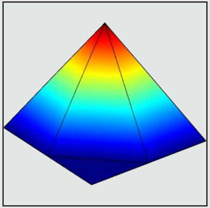


Lagrangian coarse space

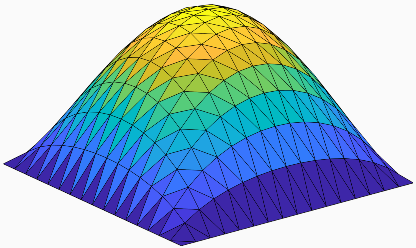
Coarse triangulation

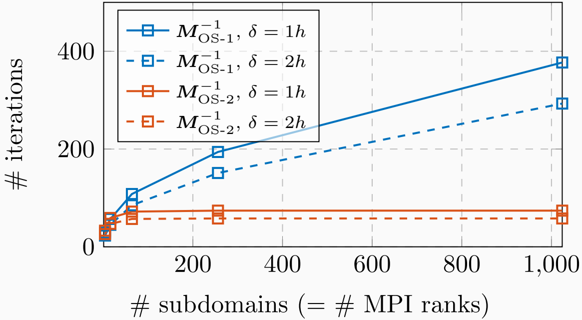


Coarse solution



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





The FROSch Package – Algebraic and Parallel Schwarz Preconditioners in Trilinos

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)

- Filipe Cumaru (TU Delft)
- Kyrill Ho (UCologne)
- Jascha Knepper (UCologne)
- Friederike Röver (TUBAF)
- Lea Saßmannshausen (UCologne)
- Alexander Heinlein (TU Delft)
- Axel Klawonn (UCologne)
- Siva Rajamanickam (SNL)
- Oliver Rheinbach (TUBAF)
- Ichitaro Yamazaki (SNL)

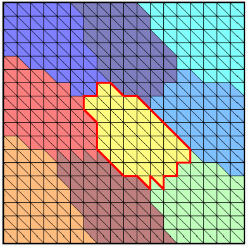
The **energy-minimizing extension** $v_i = H_{\partial\Omega_i \to \Omega_i}(v_{i,\partial\Omega_i})$ solves

$$\begin{aligned} -\Delta v_i &= 0 && \text{in } \Omega_i, \\ v_i &= v_{i,\partial\Omega_i} && \text{on } \partial\Omega_i. \end{aligned}$$

Hence, $v_i = E_{\partial\Omega_i \to \Omega_i}(\mathbb{1}_{\partial\Omega_i}) = \mathbb{1}$.

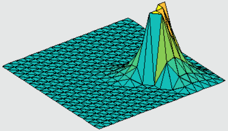
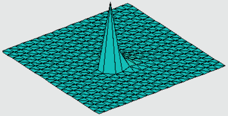
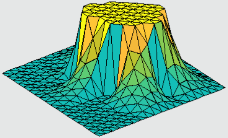
Due to **linearity of the extension operator**, we have

$$\sum_i \varphi_i = \mathbb{1}_{\partial\Omega_i} \Rightarrow \sum_i E_{\partial\Omega_i \to \Omega_i}(\varphi_i) = \mathbb{1}_{\Omega_i}$$



Null space property

Any extension-based coarse space built from a partition of unity on the domain decomposition interface satisfies the **null space property necessary for numerical scalability**:

$\sum_{\text{edges} \subset \partial\Omega_i}$  $+ \sum_{\text{vertices} \subset \partial\Omega_i}$  $=$ 

Algebraicity of the energy-minimizing extension

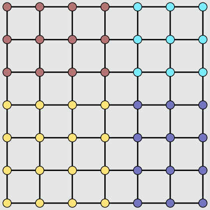
The computation of energy-minimizing extensions only requires $\boldsymbol{K}_{II}$ and $\boldsymbol{K}_{I\Gamma}$, **submatrices of the fully assembled matrix** $\boldsymbol{K}_i$.

$$\boldsymbol{v} = \begin{bmatrix} -\boldsymbol{K}_{II}^{-1}\boldsymbol{K}_{I\Gamma} \\ \boldsymbol{I}_\Gamma \end{bmatrix} \boldsymbol{v}_\Gamma,$$

FROSch Construction – Graph View

In the algebraic construction of FROSCH preconditioners, we use **two different graphs**:

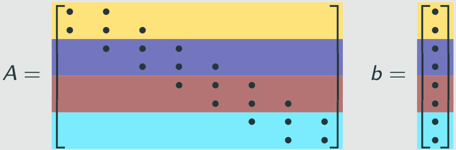
Node graph



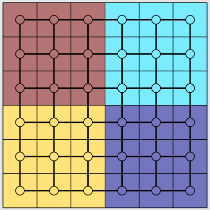
The **node graph coincides with the simulation mesh** of the computational domain:

- Graph nodes $\equiv$ mesh nodes
- Graph edges $\equiv$ mesh element edges

In **parallel simulations**:



Dual graph



The **dual graph represents the connectivity of mesh elements**:

- Graph nodes $\equiv$ mesh elements
- Graph edges $\equiv$ shared edges between mesh elements

In **parallel finite element simulations**, where the matrix assembly is element-based, the **distribution of the data is done based on a partition of the dual graph**.

Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

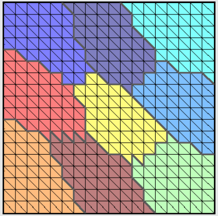
The **overlapping subdomains** are constructed by **recursively adding layers of elements**.

The corresponding matrices

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

can easily be extracted from $\boldsymbol{K}$.

Nonoverlapping DD



Overlapping domain decomposition

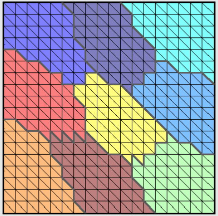
The **overlapping subdomains** are constructed by **recursively adding layers of elements**.

The corresponding matrices

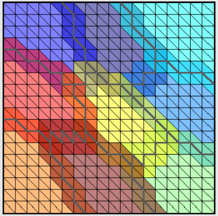
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

can easily be extracted from $\boldsymbol{K}$.

Nonoverlapping DD



Overlap $\delta = 1h$



Overlapping domain decomposition

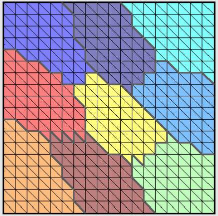
The **overlapping subdomains** are constructed by **recursively adding layers of elements**.

The corresponding matrices

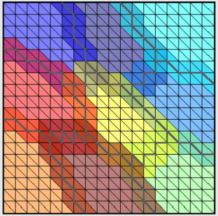
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

can easily be extracted from $\boldsymbol{K}$.

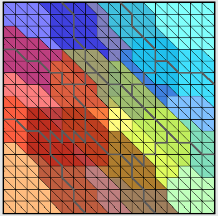
Nonoverlapping DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$



Overlapping domain decomposition

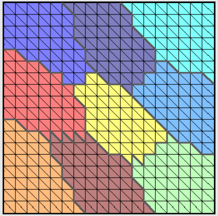
The **overlapping subdomains** are constructed by **recursively adding layers of elements**.

The corresponding matrices

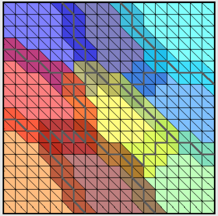
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

can easily be extracted from $\boldsymbol{K}$.

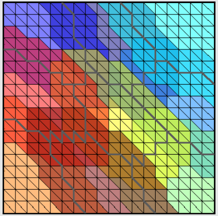
Nonoverlapping DD



Overlap $\delta = 1h$

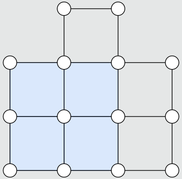


Overlap $\delta = 2h$



Implementation as a graph search problem

The overlapping subdomains can be seen as the result of a limited **breadth first search** on the graph representing the sparsity pattern of **K**.



Algorithmic Framework for FROSch Preconditioners

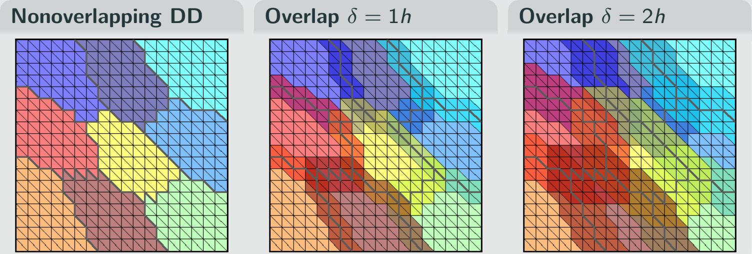
Overlapping domain decomposition

The **overlapping subdomains** are constructed by **recursively adding layers of elements**.

The corresponding matrices

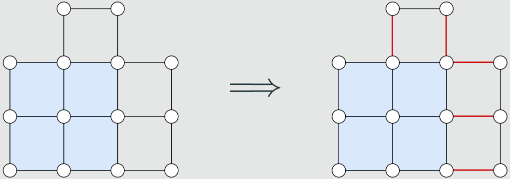
$$K_i = R_i K R_i^T$$

can easily be extracted from K.



Implementation as a graph search problem

The overlapping subdomains can be seen as the result of a limited **breadth first search** on the graph representing the sparsity pattern of **K**.



Algorithmic Framework for FROSch Preconditioners

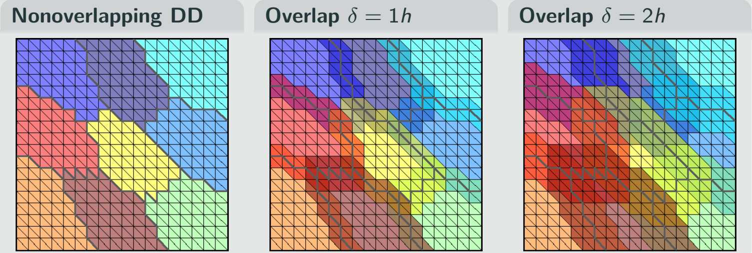
Overlapping domain decomposition

The **overlapping subdomains** are constructed by **recursively adding layers of elements**.

The corresponding matrices

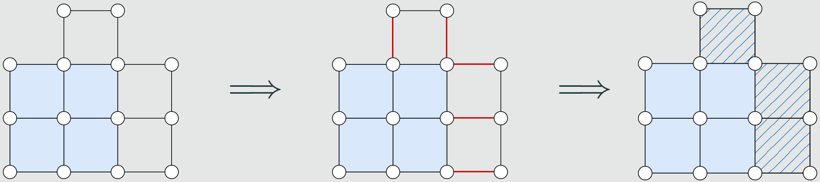
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

can easily be extracted from $\boldsymbol{K}$.



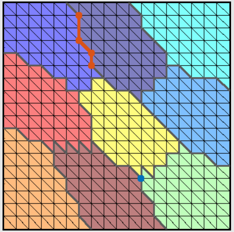
Implementation as a graph search problem

The overlapping subdomains can be seen as the result of a limited **breadth first search** on the graph representing the sparsity pattern of **K**.



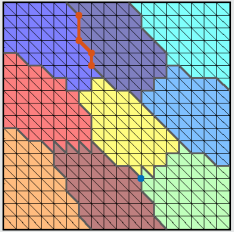
Coarse space

1. Interface components

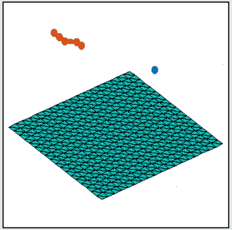


Coarse space

1. Interface components



2. Interface basis (partition of unity $\times$ null space)

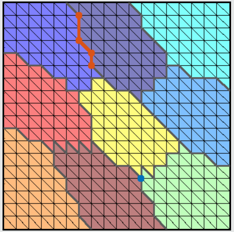


For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

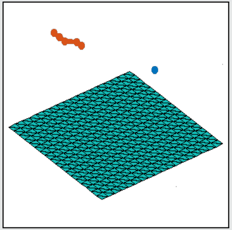
Algorithmic Framework for FROSch Preconditioners

Coarse space

1. Interface components

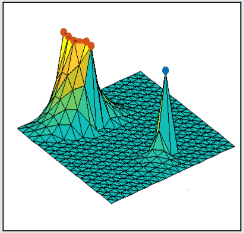


2. Interface basis (partition of unity $\times$ null space)



For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

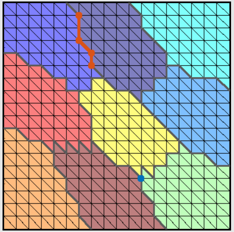
3. Extension



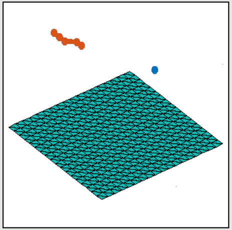
Algorithmic Framework for FROSch Preconditioners

Coarse space

1. Interface components

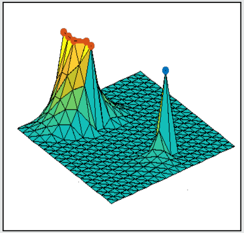


2. Interface basis (partition of unity $\times$ null space)



For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

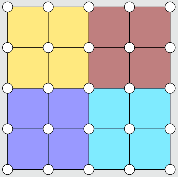
3. Extension



Construction of the interface entities

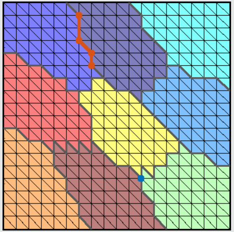
A partition of interface entities can be found by inspecting the **dual graph of the mesh**.

If it is **not given as an input**, it **can generally not be retained** from the **sparsity pattern of K**, but we can approximate it.

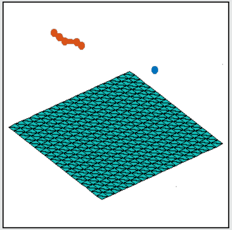


Coarse space

1. Interface components

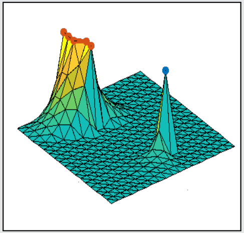


2. Interface basis (partition of unity × null space)



For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

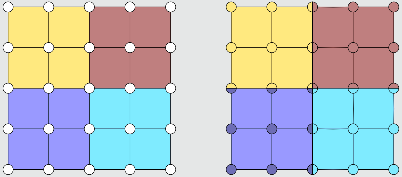
3. Extension



Construction of the interface entities

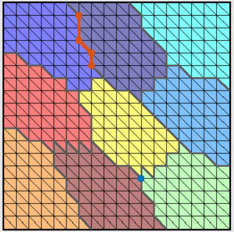
A partition of interface entities can be found by inspecting the **dual graph of the mesh**.

If it is **not given as an input**, it **can generally not be retained** from the **sparsity pattern of K**, but we can approximate it.

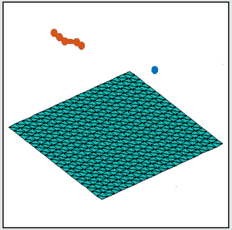


Coarse space

1. Interface components

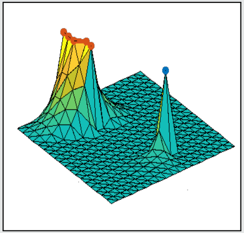


2. Interface basis (partition of unity × null space)



For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

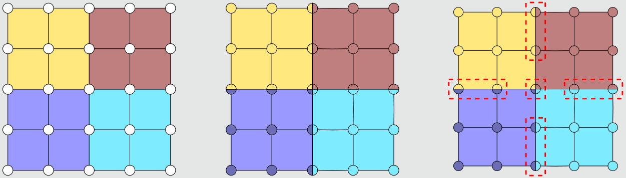
3. Extension



Construction of the interface entities

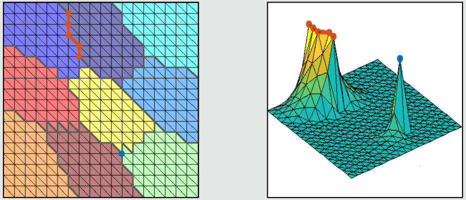
A partition of interface entities can be found by inspecting the **dual graph of the mesh**.

If it is **not given as an input**, it **can generally not be retained** from the **sparsity pattern of K**, but we can approximate it.



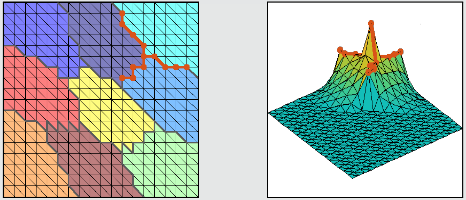
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)



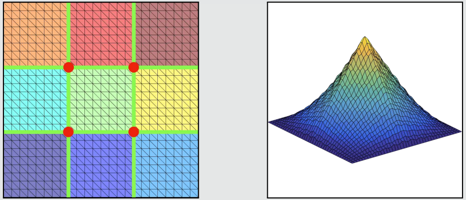
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)



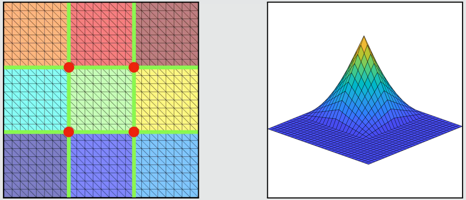
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

MsFEM (Multiscale Finite Element Method)



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

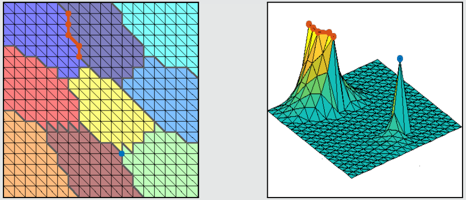
Q1 Lagrangian / piecewise bilinear



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

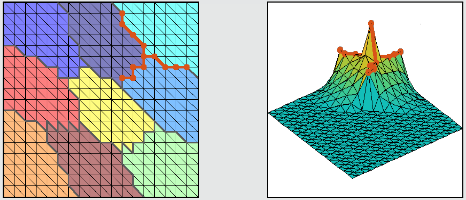
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)



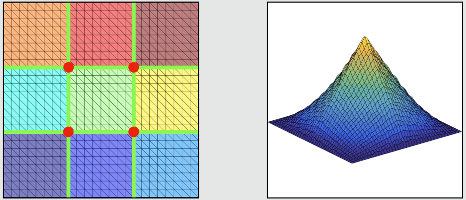
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)



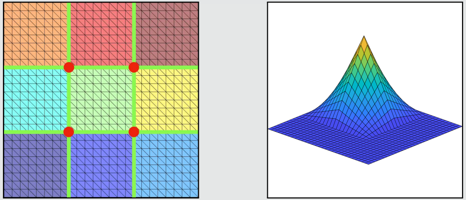
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

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

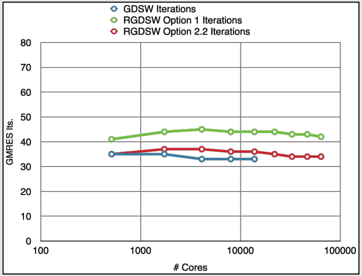


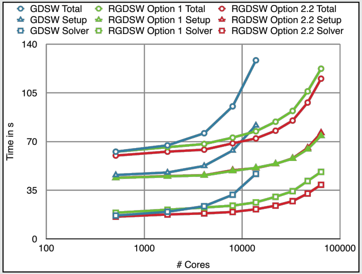
Piecewise linear interface partition of unity functions and a **structured domain decomposition**.

Weak Scalability up to $64\,\mathrm{k}$ MPI Ranks / $1.7\,\mathrm{b}$ Unknowns (3D Poisson; Juqueen)

GDSW vs RGDSW (reduced dimension)

Heinlein, Klawonn, Rheinbach, Widlund (2019).





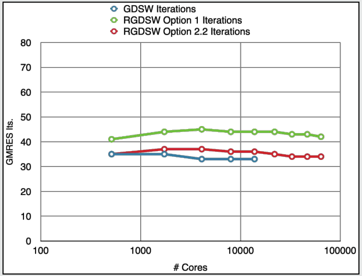
Two-level vs three-level GDSW

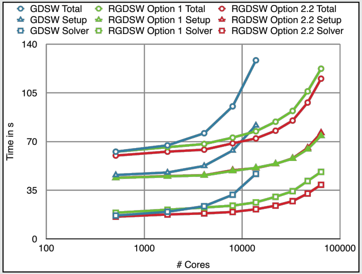
Heinlein, Klawonn, Rheinbach, Röver (2019, 2020).

Weak Scalability up to $64\,\mathrm{k}$ MPI Ranks / $1.7\,\mathrm{b}$ Unknowns (3D Poisson; Juqueen)

GDSW vs RGDSW (reduced dimension)

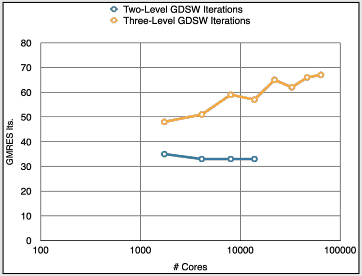
Heinlein, Klawonn, Rheinbach, Widlund (2019).

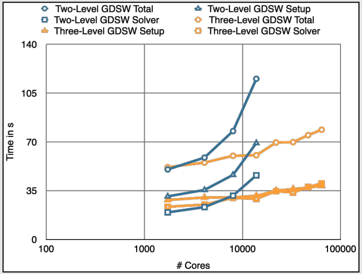




Two-level vs three-level GDSW

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





Some Challenging Application Problems

Monolithic (R)GDSW Preconditioners for CFD Simulations

Consider the discrete saddle point problem

$$\mathcal{A}x = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{B}^\top \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} = \mathcal{b}.$$

Monolithic GDSW preconditioner

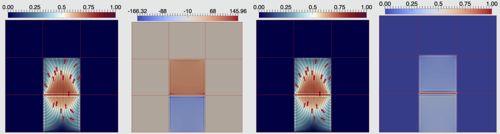
We construct a **monolithic GDSW preconditioner**

$$\mathcal{M}_{\mathrm{GDSW}}^{-1} = \phi \mathcal{A}_0^{-1} \phi^\top + \sum_{i=1}^{N} \mathcal{R}_i^\top \overline{\mathcal{P}}_i \mathcal{A}_i^{-1} \mathcal{R}_i,$$

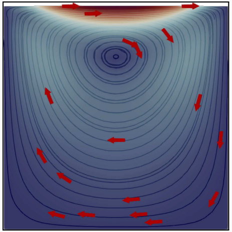
with block matrices $\mathcal{A}_0 = \phi^\top \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^\top$, local pressure projections $\overline{\mathcal{P}}_i$, and

$$\mathcal{R}_i = \begin{bmatrix} \mathcal{R}_{u,i} & \boldsymbol{0} \\ \boldsymbol{0} & \mathcal{R}_{p,i} \end{bmatrix} \quad \text{and} \quad \phi = \begin{bmatrix} \Phi_{u,u_0} & \Phi_{u,p_0} \\ \Phi_{p,u_0} & \Phi_{p,p_0} \end{bmatrix}.$$

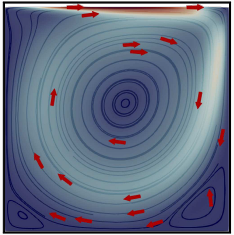
Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



Φ_{u,u_0} Φ_{p,u_0} Φ_{u,p_0} Φ_{p,p_0}.



Stokes flow



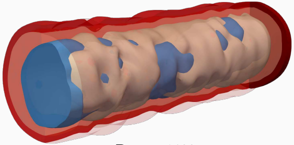
Navier–Stokes flow

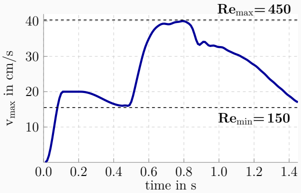
Related work:

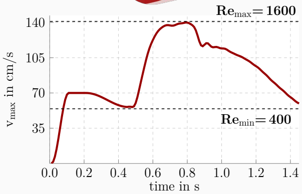
- Original work on monolithic Schwarz preconditioners: **Klawonn and Pavarino (1998, 2000)**
- Other publications on monolithic Schwarz preconditioners: e.g., **Hwang and Cai (2006)**, **Barker and Cai (2010)**, **Wu and Cai (2014)**, and the presentation **Dohrmann (2010)** at the *Workshop on Adaptive Finite Elements and Domain Decomposition Methods* in Milan.

Results for Blood Flow Simulations

- **3D unsteady flow simulation** within the **geometry of a realistic artery** (from **Balzani et al. (2012)**) and kinematic viscosity $\nu = 0.03\,\text{cm}^2/\text{s}$
- **Parabolic inflow profile** is prescribed at inlet of geometry
- **Time discretization**: BDF-2; **space discretization**: P2-P1 elements





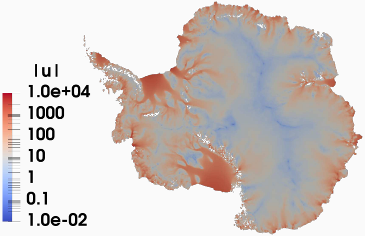


prec.	# MPI ranks	16	64	256
Monolithic RGDSW (FROSch)	avg. #its.	33	31	30
	setup	4 825 s	1 422 s	701 s
	solve	3 198 s	1 004 s	463 s
	total	8 023 s	2 426 s	**1 164 s**
SIMPLE RGDSW (Teko & FROSch)	avg. #its.	82	82	87
	setup	3 046 s	824 s	428 s
	solve	4 679 s	1 533 s	801 s
	total	**7 725 s**	**2 357 s**	1 229 s

prec.	# MPI ranks	16	64	256
Monolithic RGDSW (FROSch)	avg. #its.	36	36	36
	setup	4 808 s	1 448 s	688 s
	solve	3 490 s	1 186 s	538 s
	total	**8 298 s**	**2 634 s**	**1 226 s**
SIMPLE RGDSW (Teko & FROSch)	avg. #its.	157	164	169
	setup	3 071 s	842 s	432 s
	solve	9 541 s	3 210 s	1 585 s
	total	12 612 s	4 052 s	2 017 s

Albany

https://github.com/SNLComputation/Albany



The velocity of the ice sheet in Antarctica and Greenland is modeled by a **first-order-accurate Stokes approximation model**,

$$-\nabla \cdot (2\mu\dot{\epsilon}_1) + \rho g \frac{\partial s}{\partial x} = 0, \quad -\nabla \cdot (2\mu\dot{\epsilon}_2) + \rho g \frac{\partial s}{\partial y} = 0,$$

with a **nonlinear viscosity model** (Glen's law); cf., e.g., **Blatter (1995)** and **Pattyn (2003)**.

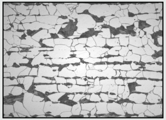
	Antarctica (**velocity**) 4 km resolution, 20 layers, 35 m dofs			Greenland (**multiphysics vel. & temperature**) 1-10 km resolution, 20 layers, 69 m dofs		
MPI ranks	avg. its	avg. setup	avg. solve	avg. its	avg. setup	avg. solve
512	**41.9** (11)	25.10 s	12.29 s	**41.3** (36)	18.78 s	4.99 s
1 024	**43.3** (11)	9.18 s	5.85 s	**53.0** (29)	8.68 s	4.22 s
2 048	**41.4** (11)	4.15 s	2.63 s	**62.2** (86)	4.47 s	4.23 s
4 096	**41.2** (11)	1.66 s	1.49 s	**68.9** (40)	2.52 s	2.86 s
8 192	**40.2** (11)	1.26 s	1.06 s	-	-	-

Computations performed on Cori (NERSC).

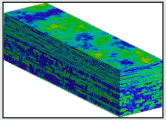
Heinlein, Perego, Rajamanickam (2022)

Highly heterogeneous problems . . .

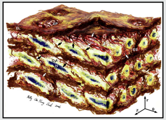
. . . appear in most areas of modern science and engineering:



Micro section of a dual-phase steel. Courtesy of **J. Schröder**.



Groundwater flow (SPE10); cf. **Christie and Blunt (2001)**.



Composition of arterial walls; taken from **O'Connell et al. (2008)**.

Spectral coarse spaces

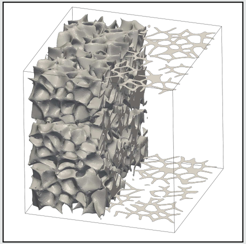
The coarse space is **enhanced** by eigenfunctions of **local edge and face eigenvalue problems** with eigenvalues below tolerances $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$:

$$\kappa\left(M_*^{-1}K\right) \leq C\left(1 + \frac{1}{tol_{\mathcal{E}}} + \frac{1}{tol_{\mathcal{F}}} + \frac{1}{tol_{\mathcal{E}} \cdot tol_{\mathcal{F}}}\right);$$

C does not depend on h, H, or the coefficients.

OS-ACMS & adaptive GDSW (AGDSW) (**Heinlein, Klawonn, Knepper, Rheinbach (2018, 2018, 2019)**).

Foam coefficient function example



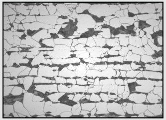
Solid phase: $\alpha = 10^6$; **transparent phase:** $\alpha = 1$; 100 subdomains

V_0	$tol_{\mathcal{E}}$	$tol_{\mathcal{F}}$	it.	κ	dim V_0	dim V_0/ dof
V_{GDSW}	—	—	**565**	$1.3 \cdot 10^6$	1 601	0.27 %
V_{AGDSW}	0.05	0.05	**60**	30.2	1 968	0.33 %
$V_{\mathrm{OS-ACMS}}$	0.001	0.001	**57**	30.3	690	0.12 %

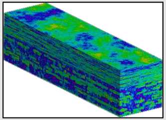
Cf. **Heinlein, Klawonn, Knepper, Rheinbach (2018, 2019)**.

Highly heterogeneous problems ...

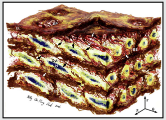
...appear in most areas of modern science and engineering:



Micro section of a dual-phase steel. Courtesy of **J. Schröder**.



Groundwater flow (SPE10); cf. **Christie and Blunt (2001)**.



Composition of arterial walls; taken from **O'Connell et al. (2008)**.

Spectral coarse spaces

The coarse space is **enhanced** by eigenfunctions of **local edge and face eigenvalue problems** with eigenvalues below tolerances $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$:

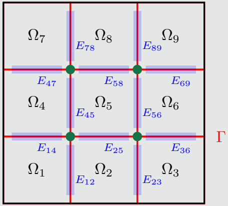
$$\kappa\left(\boldsymbol{M}_*^{-1}\boldsymbol{K}\right) \leq C\left(1 + \frac{1}{tol_{\mathcal{E}}} + \frac{1}{tol_{\mathcal{F}}} + \frac{1}{tol_{\mathcal{E}} \cdot tol_{\mathcal{F}}}\right);$$

C does not depend on h, H, or the coefficients.
OS-ACMS & **adaptive GDSW (AGDSW)** (**Heinlein, Klawonn, Knepper, Rheinbach (2018, 2018, 2019)**).

Local eigenvalue problems

Local generalized eigenvalue problems corresponding to the edges $\mathcal{E}$ and faces $\mathcal{F}$ of the domain decomposition:

$$\begin{aligned}
\forall E \in \mathcal{E}: \qquad & \boldsymbol{S}_{EE}\tau_{*,E} = \lambda_{*,E}\boldsymbol{K}_{EE}\tau_{*,E}, \quad \forall \tau_{*,E} \in V_E, \\
\forall F \in \mathcal{F}: \qquad & \boldsymbol{S}_{FF}\tau_{*,F} = \lambda_{*,F}\boldsymbol{K}_{FF}\tau_{*,F}, \quad \forall \tau_{*,F} \in V_F,
\end{aligned}$$

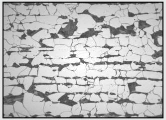


with **Schur complements** S_{EE}, S_{FF} with **Neumann boundary conditions** and **submatrices** K_{EE}, K_{FF} of K. We select eigenfunctions corresponding to **eigenvalues below tolerances** $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$.

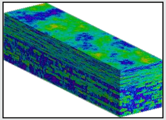
$\rightarrow$ The corresponding coarse basis functions are **energy-minimizing extensions** into the interior of the subdomains.

Highly heterogeneous problems ...

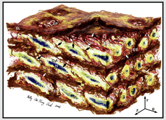
...appear in most areas of modern science and engineering:



Micro section of a dual-phase steel. Courtesy of **J. Schröder**.



Groundwater flow (SPE10); cf. **Christie and Blunt (2001)**.



Composition of arterial walls; taken from **O'Connell et al. (2008)**.

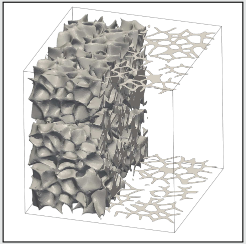
Spectral coarse spaces

The coarse space is **enhanced** by eigenfunctions of **local edge and face eigenvalue problems** with eigenvalues below tolerances $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$:

$$\kappa\left(M_*^{-1}K\right) \leq C\left(1 + \frac{1}{tol_{\mathcal{E}}} + \frac{1}{tol_{\mathcal{F}}} + \frac{1}{tol_{\mathcal{E}} \cdot tol_{\mathcal{F}}}\right);$$

C does not depend on h, H, or the coefficients. OS-ACMS & adaptive GDSW (AGDSW) (**Heinlein, Klawonn, Knepper, Rheinbach (2018, 2018, 2019)**).

Foam coefficient function example



Solid phase: $\alpha = 10^6$; **transparent phase:** $\alpha = 1$; 100 subdomains

V_0	$tol_{\mathcal{E}}$	$tol_{\mathcal{F}}$	it.	κ	dim V_0	dim V_0/ dof
V_{GDSW}	—	—	**565**	$1.3 \cdot 10^6$	1 601	0.27 %
V_{AGDSW}	0.05	0.05	**60**	30.2	1 968	0.33 %
$V_{\mathrm{OS-ACMS}}$	0.001	0.001	**57**	30.3	690	0.12 %

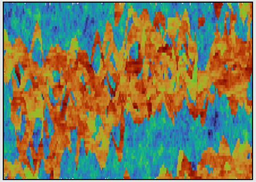
Cf. **Heinlein, Klawonn, Knepper, Rheinbach (2018, 2019)**.

Domain Decomposition for Reservoir Simulations

Algebraic multiscale coarse space

- We investigate **scalable and robust** simulation methods for underground hydrogen storage.
- We consider **two-level domain decomposition solver** with **algebraic multiscale solver (AMS)** coarse space; cf. **Wang, Hajibeygi and Tchelepi (2014)**.

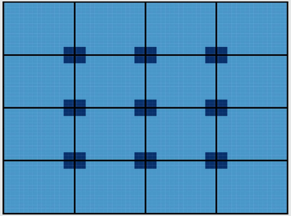
Numerical results - SPE10 benchmark



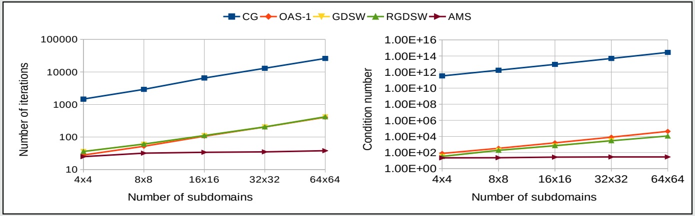
Layer 72 from model 2; cf. **Christie and Blunt (2001)**.

preconditioner	its.	κ
-	$> \mathbf{10^4}$	$8.61 \cdot 10^8$
one-level	**174**	$1.01 \cdot 10^5$
two-level w\ AMS	**78**	144.33

Numerical results - Weak scalability for high coefficient inclusions



Dark blue: $\alpha = 10^8$; **light blue:** $\alpha = 1$



Cf. **Alves, Heinlein and Hajibeygi (2024; preprint arXiv)**

Learning Extension Operators Using Graph Neural Networks

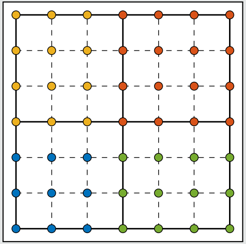
Learning Extension Operators

Most coarse spaces for Schwarz preconditioners are constructed based on a **characteristic functions**

$$\varphi_i(\omega_j) = \delta_{ij},$$

on specifically chosen sets of nodes $\{\omega_j\}_j$. The **values in the remaining nodes** are then obtained by **extending the values into the adjacent subdomains**. Examples:

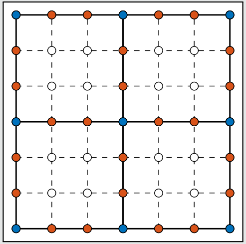
Subdomain-based



- The ω_j are based on nonoverl. subdomains Ω_j
- **No extensions** needed

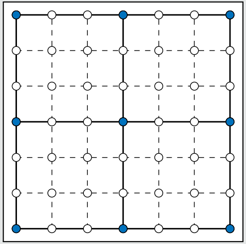
Cf. **Nicolaides (1987)**.

GDSW



- The ω_j are based on **partition of the interface**
- **Energy-minimizing** exts.

Vertex-based



- **Lagrangian**: **geometric** ext.
- **MsFEM**: **geometric** and **energy-minimizing** exts.
- **RGDSW**: **algebraic** and **energy-minimizing** exts.

Learning Extension Operators

Most coarse spaces for Schwarz preconditioners are constructed based on a **characteristic functions**

$$\varphi_i(\omega_j) = \delta_{ij},$$

on specifically chosen sets of nodes $\{\omega_j\}_j$. The **values in the remaining nodes** are then obtained by **extending the values into the adjacent subdomains**. Examples:

Observation 1

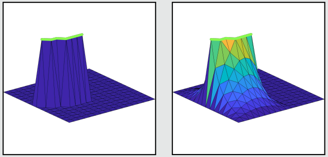
Energy-minimizing extensions

- are algebraic:

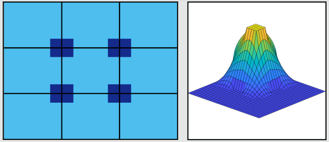
$$\boldsymbol{v}_I = -\boldsymbol{K}_{II}^{-1}\boldsymbol{K}_{I\Gamma}\boldsymbol{v}_\Gamma$$

(with Dirichlet b. c.)

- can be costly: solving a **problem in the interior**



Observation 2

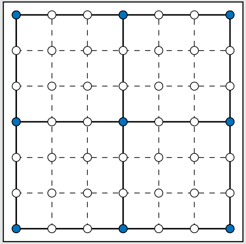


Heterogeneous: $\alpha_{\text{light}} = 1$; $\alpha_{\text{dark}} = 10^8$

The performance may **strongly depend on extension operator**:

coarse space	its.	κ
—	**163**	$4.06 \cdot 10^7$
Q1	**138**	$1.07 \cdot 10^6$
MsFEM	**24**	8.05

Vertex-based



- Lagrangian: **geometric** ext.
- MsFEM: **geometric** and **energy-minimizing** exts.
- RGDSW: **algebraic** and **energy-minimizing** exts.

→ **Improving efficiency & robustness** via machine learning.

Related Works

This overview is **not exhaustive**:

Coarse spaces for domain decomposition methods

- **Prediction of the geometric location of adaptive constraints (adaptive BDDC & FETI–DP as well as AGDSW)**: Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022)
- **Prediction of coarse basis functions**: Chung, Kim, Lam, Zhao (2021); Klawonn, Lanser, Weber (2024, 2024); Kopaničáková, Karniadakis (2025)
- **Learning interface conditions and coarse interpolation operators**: Taghibakhshi et al. (2022, 2023)

Algebraic multigrid (AMG)

- **Prediction of coarse grid operators**: Luz et al. (2020); Tomasi, Krause (2023); Zhang et al. (2024)
- **Coarsening**: Taghibakhshi, MacLachlan, Olson, West (2021); Antonietti, Caldana, Dede (2023)

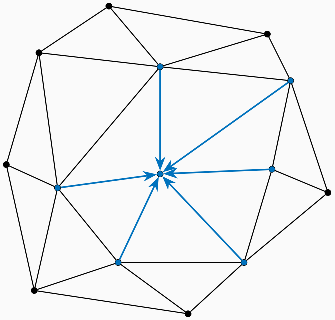
An overviews of the **state-of-the-art on domain decomposition and machine learning** in early 2021 and 2023:

A. Heinlein, A. Klawonn, M. Lanser, J. Weber
Combining machine learning and domain decomposition methods for the solution of partial differential equations — A review
GAMM-Mitteilungen. 2021.

A. Klawonn, M. Lanser, J. Weber
Machine learning and domain decomposition methods – a survey
Computational Science and Engineering. 2024

Graph neural networks (GNNs) **Gori, Monfardini, and Scarselli (2005)** are a natural choice for learning on data defined over simulation meshes:

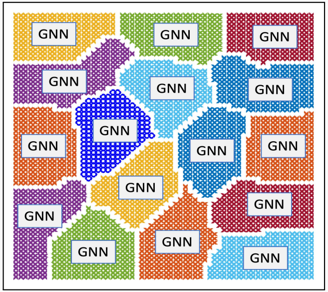
- Generalize **CNNs** **LeCun (1998)** to irregular, **graph-structured data**.
- Learn via **iterative aggregation** from **neighboring nodes**.
- Naturally **permutation invariant** and **geometrically robust**.

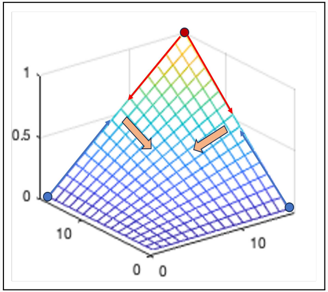


Further references: Scarselli et al. (2005), Bruna et al. (2014), Henaff et al. (2015), Defferrard et al. (2016), Kipf, Welling (2017), . . .

Local approach

- **Input:** subdomain matrix K_i
- **Output:** basis functions $\{\varphi_j^{\Omega_i}\}_j$ on the same subdomain
- Training on **subdomains with varying geometry**
- Inference on **unseen subdomains**

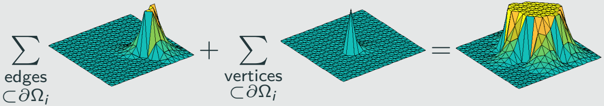




Theory-Inspired Design of the GNN-Based Coarse Space

Null space property

Any extension-based coarse space built from a partition of unity on the domain decomposition interface satisfies the **null space property necessary for numerical scalability**:



Explicit partition of unity

To **explicitly enforce** that the basis functions $(\varphi_j)_j$ form **a partition of unity**

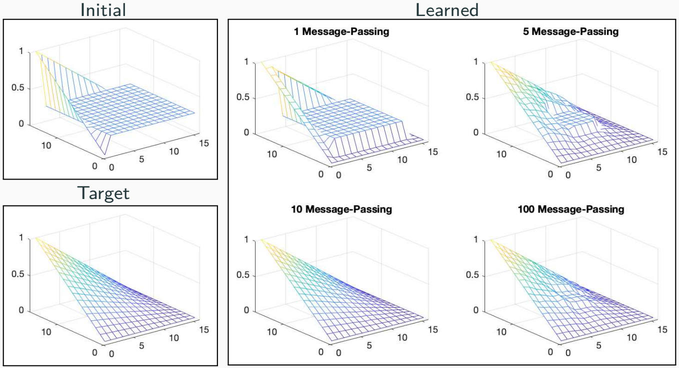
$$\varphi_j = \frac{\hat{\varphi}_j}{\sum_k \hat{\varphi}_k},$$

where the $\hat{\varphi}_k$ are the outputs of the GNN.

Initial and target

- **Initial function:** partition of unity that is constant in the interior
- **Target function:**
  - linear on the edges
  - energy-minimizing in the interior

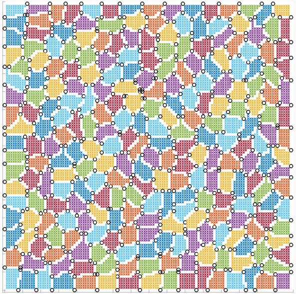
$\rightarrow$ **Information transport** via **message passing**



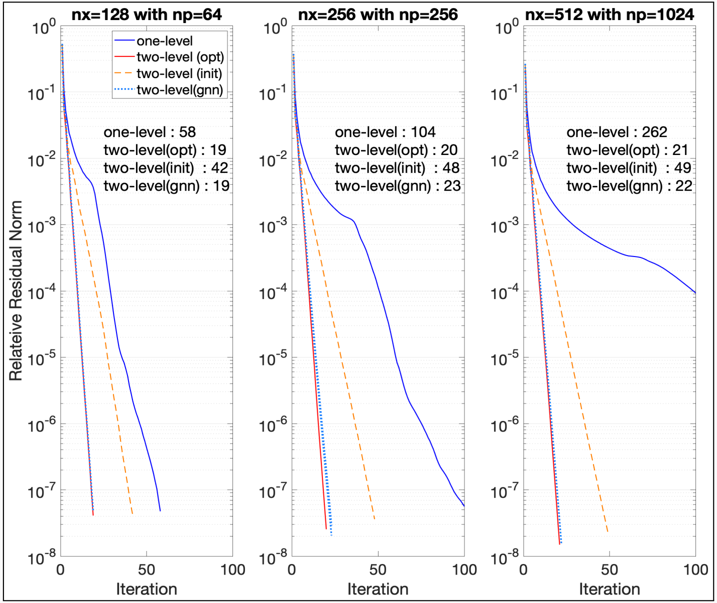
Model problem: 2D Laplacian model problem discretized using finite differences on a structured grid

$$-\Delta u = 1 \quad \text{in } \Omega,$$
$$u = 0 \quad \text{on } \partial\Omega,$$

decomposed using **METIS**:



- The GNN has been **trained on 64 subdomains**.

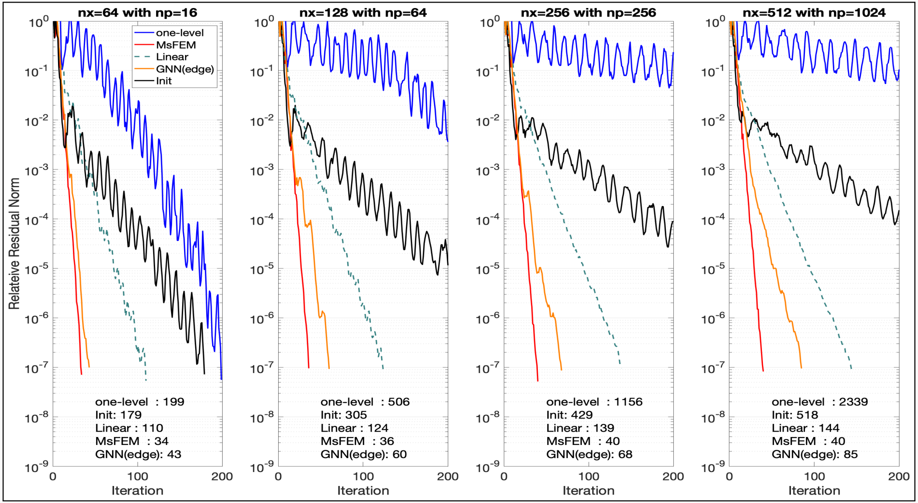


Yamazaki, Heinlein, Rajamanickam (subm. 2024)

Numerical Results for Heterogeneous Laplacian – Weak Scaling Study

Heterogeneous Laplacian with $\alpha_{\max}/\alpha_{\min} = 10^3$:

$$-\nabla \cdot (\alpha(x)\nabla u(x)) = f \text{ in } \Omega = [0,1]^2, \qquad u = 0 \text{ on } \partial\Omega.$$



Yamazaki, Heinlein, Rajamanickam (subm. 2024)

Scientific Machine Learning in Academia and Beyond:
From Theory to Real-World Impact (in Industry)

- **Dates**: June 17, 2025, 12.30–17.30
- **Location**: Crowne Plaza Hotel, Utrecht
- **Lunch & networking**: 12.30–13.30; closing discussion & drinks to follow.
- An afternoon with **talks, case studies, and lively discussions** on advancing scientific machine learning from theory to real-world deployment — tackling core challenges like *uncertainty quantification, data assimilation, graph-based modelling, and operator learning*.

- **Confirmed plenary speakers**:
  - Max Welling (UvA, CUSP AI)
  - Stefan Kurz (ETH Zürich & Bosch)
  - Koen Strien (Ignition Computing)
  - Maxim Pisarenco (ASML)
  - Jan Willem van de Meent (UvA)

Computational Science NL

CWI Research Semester Programme:

Bridging Numerical Analysis and Scientific Machine Learning: Advances and Applications

Co-organizers: Victorita Dolean (TU/e), Alexander Heinlein (TU Delft), Benjamin Sanderse (CWI), Jemima Tabbeart (TU/e), Tristan van Leeuwen (CWI)

- **Autumn School** (October 27–31, 2025):
  - Chris Budd (University of Bath)
  - Ben Moseley (Imperial College London)
  - Gabriele Steidl (Technische Universität Berlin)
  - Andrew Stuart (California Institute of Technology)
  - Andrea Walther (Humboldt-Universität zu Berlin)
  - Ricardo Baptista (University of Toronto)
- **Workshop** (December 1–3, 2025):
  - 3 days with plenary talks (academia & industry) and an industry panel
  - Confirmed plenary speakers:
    - Marta d'Elia (Atomic Machines)
    - Benjamin Peherstorfer (New York University)
    - Andreas Roskopf (Fraunhofer Institute)

Centrum Wiskunde & Informatica

Join us for inspiring talks, hands-on sessions, and industry collaboration!

FROSch (Fast and Robust Overlapping Schwarz)

- FROSch is based on the **Schwarz framework** and **energy-minimizing coarse spaces**, which provide **numerical scalability** using **only algebraic information** for a **variety of applications**
- FROSch is well-integrated into the Trilinos software framework, enabling
  - **large-scale distributed memory parallelization** and
  - **node-level performance on CPU and/or GPU architectures**

Learning extension operators

- **Extensions** are a major component in the **construction of coarse spaces** for domain decomposition methods.
- Using **GNNs** and **known properties from the theory**, we can **learn extension operators** that lead to a scalable coarse spaces.

Thank you for your attention!

Topical Activity Group
Scientific Machine Learning