When One Level Is Not Enough

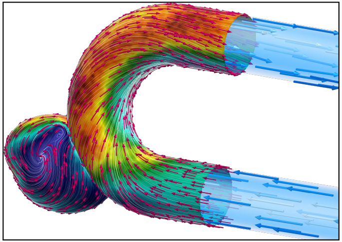
Multilevel Domain Decomposition Methods for Physics and Data-Driven Problems

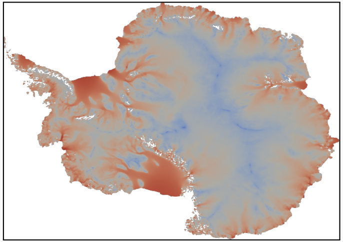
Alexander Heinlein[1]

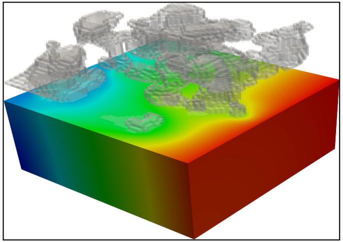
Scientific Computing Seminar, IWR, Heidelberg University, Germany, December 5, 2024

[1]Delft University of Technology

Numerical Analysis and Machine Learning







Numerical methods

Based on physical models

+ Robust and generalizable
− Require availability of mathematical models

Machine learning models

Driven by data

+ Do not require mathematical models
− Sensitive to data, limited extrapolation capabilities

Scientific machine learning (SciML)

Combining the strengths and **compensating the weaknesses** of the individual approaches:

numerical methods **improve** machine learning techniques
machine learning techniques **assist** numerical methods

Outline

1. Classical Schwarz Domain Decomposition Methods

2. Schwarz Domain Decomposition Preconditioners

   Based on joint work with

Axel Klawonn and **Jascha Knepper**	(University of Cologne)
Mauro Perego and **Siva Rajamanickam**	(Sandia National Laboratories)
Oliver Rheinbach and **Friederik Röver**	(TU Bergakademie Freiberg)
Kathrin Smetana	(Stevens Institute of Technology)
Olof Widlund	(New York University)

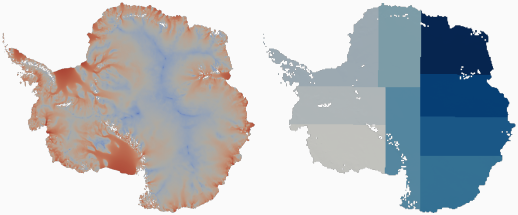
3. Domain Decomposition for Neural Networks

   Based on joint work with

Eric Cyr	(Sandia National Laboratories)
Victorita Dolean	(Eindhoven University of Technology)
Siddhartha Mishra	(ETH Zürich)
Ben Moseley	(Imperial College London)
Corné Verburg	(Delft University of Technology)

Classical Schwarz Domain Decomposition Methods

Domain Decomposition Methods



Images based on **Heinlein, Perego, Rajamanickam (2022)**

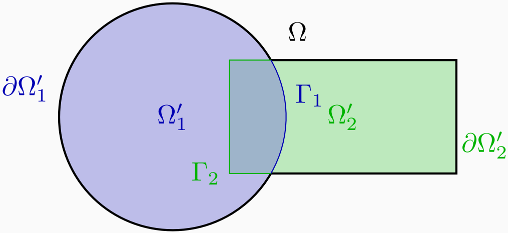
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with non-smooth boundaries**.

Idea

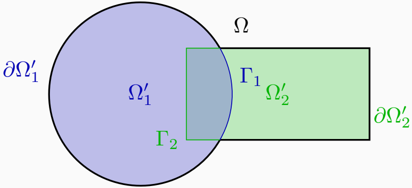
Decomposing a large **global problem** into smaller **local problems**:

- **Better robustness** and **scalability** of numerical solvers
- **Improved computational efficiency**
- Introduce **parallelism**



The Alternating Schwarz Algorithm

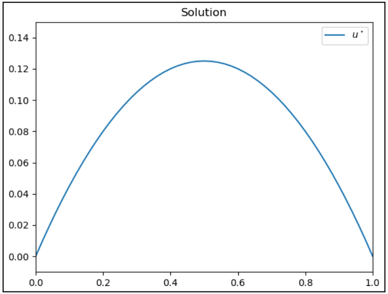
For the sake of simplicity, instead of the two-dimensional geometry,



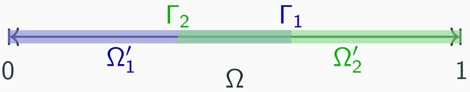
we consider the **one-dimensional Poisson equation**

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

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



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**:

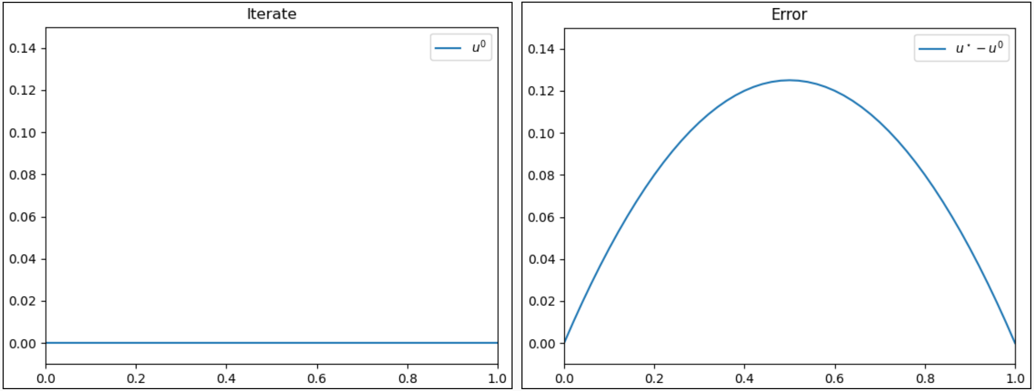


Figure 1: Iterate (left) and error (right) in iteration 0.

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**:

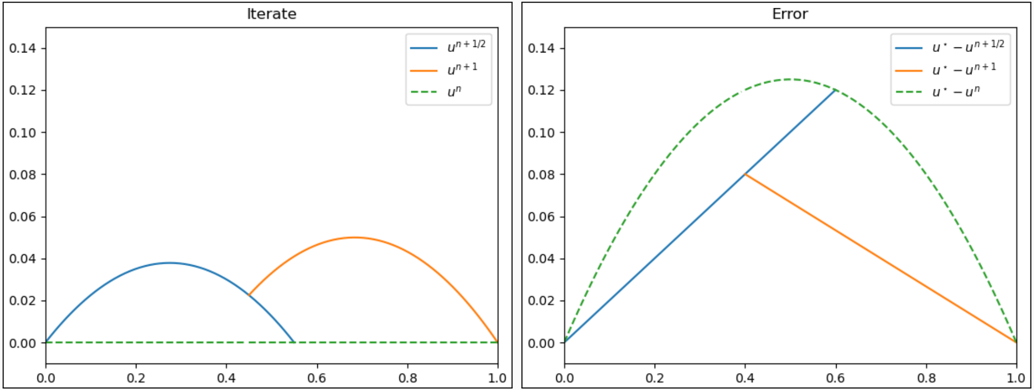


Figure 1: Iterate (left) and error (right) in iteration 1.

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**:

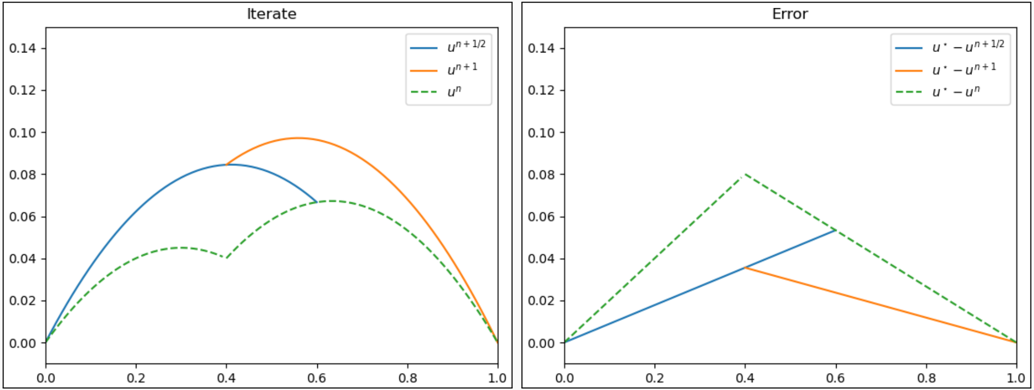


Figure 1: Iterate (left) and error (right) in iteration 2.

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**:

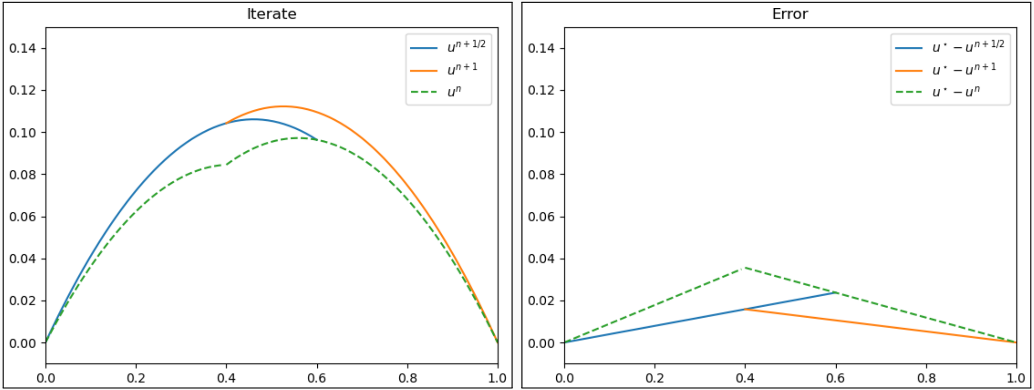


Figure 1: Iterate (left) and error (right) in iteration 3.

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**:

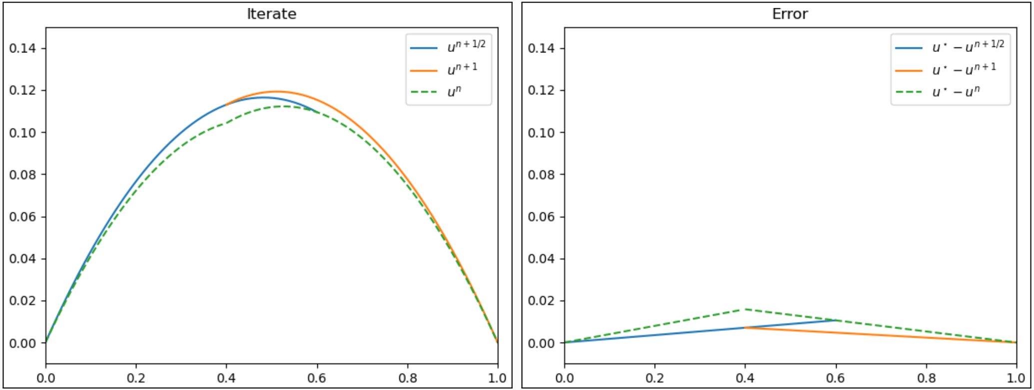


Figure 1: Iterate (left) and error (right) in iteration 4.

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**:

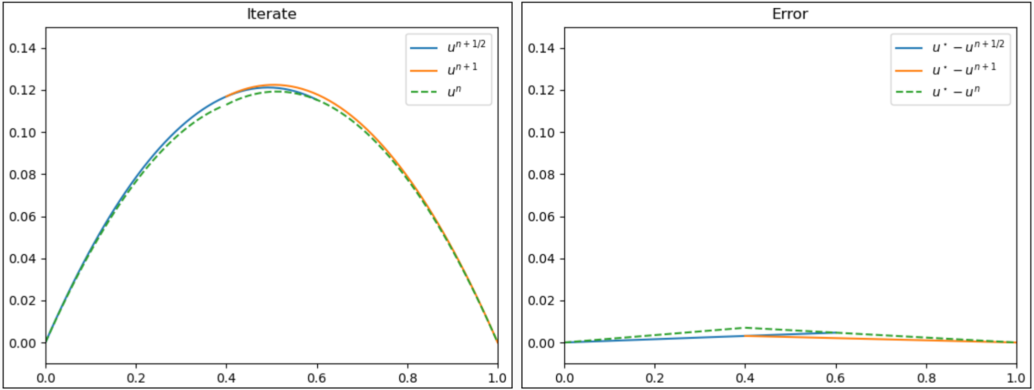


Figure 1: Iterate (left) and error (right) in iteration 5.

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**:

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

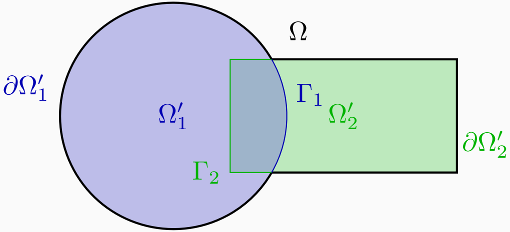
$$
(D_2)\begin{cases} -\Delta u^{n+1} = f & \text{in } \Omega_2, \\ u^{n+1} = \mathbf{u^{n+1/2}} & \text{on } \partial\Omega_2' \\ u^{n+1} = \mathbf{u^{n+1/2}} & \text{on } \Omega \setminus \overline{\Omega_2'} \end{cases}
$$

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**.

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

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

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

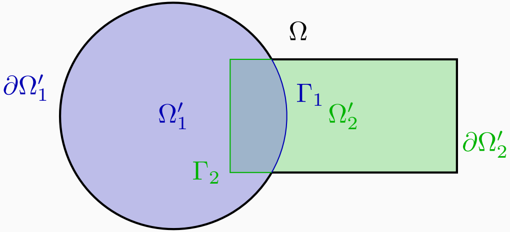


Since u_1^n and u_2^n are both computed in the previous iteration, the problems can be solved independent of each other.

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

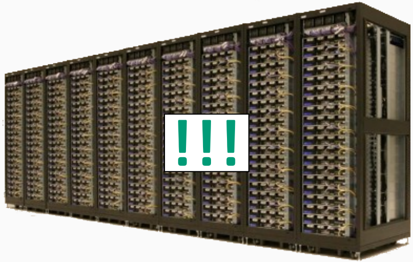
$$(D_1)\begin{cases} -\Delta u_1^{n+1} = f & \text{in } \Omega_1', \\ u_1^{n+1} = u_2^n & \text{on } \partial\Omega_1', \end{cases}$$

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



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**:

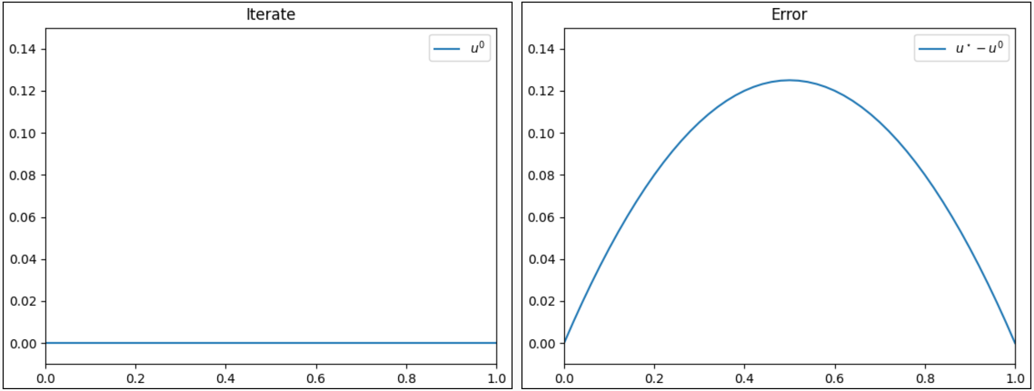


Figure 2: Iterate (left) and error (right) in iteration 0.

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**:

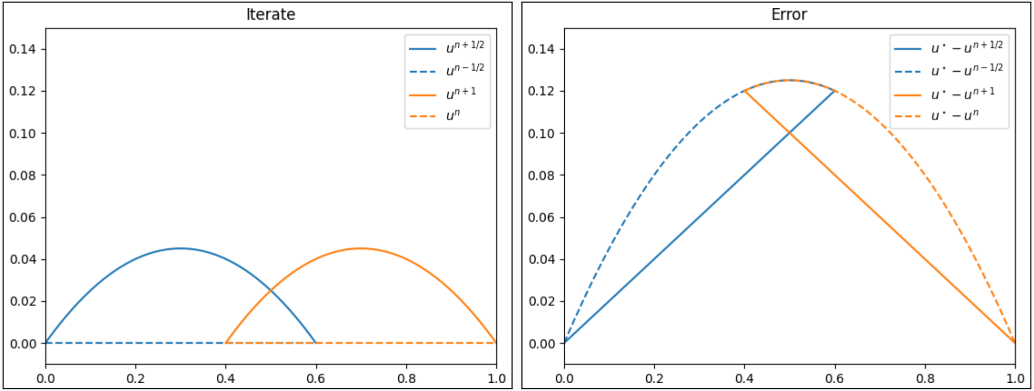


Figure 2: Iterate (left) and error (right) in iteration 1.

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**:

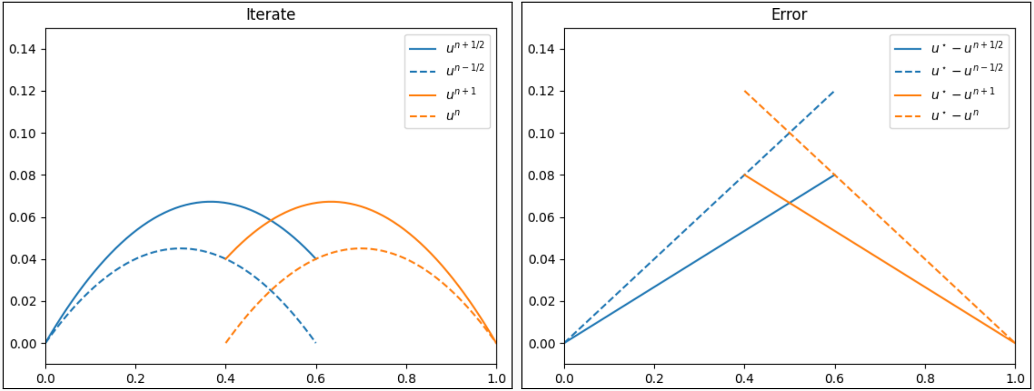


Figure 2: Iterate (left) and error (right) in iteration 2.

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**:

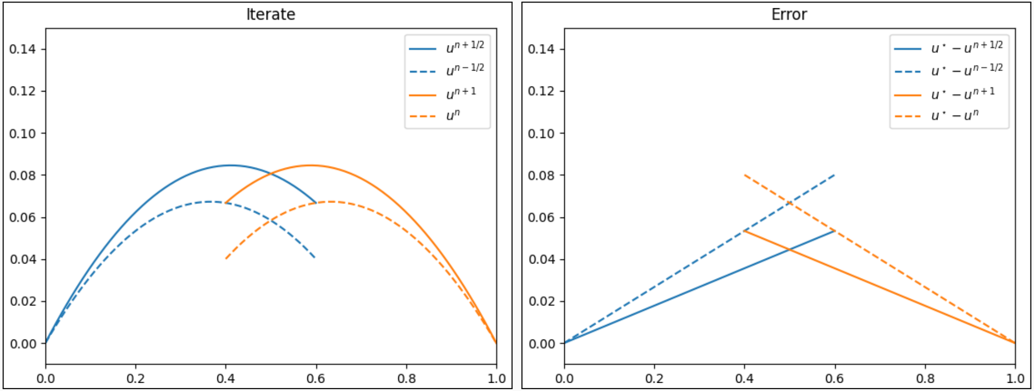


Figure 2: Iterate (left) and error (right) in iteration 3.

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**:

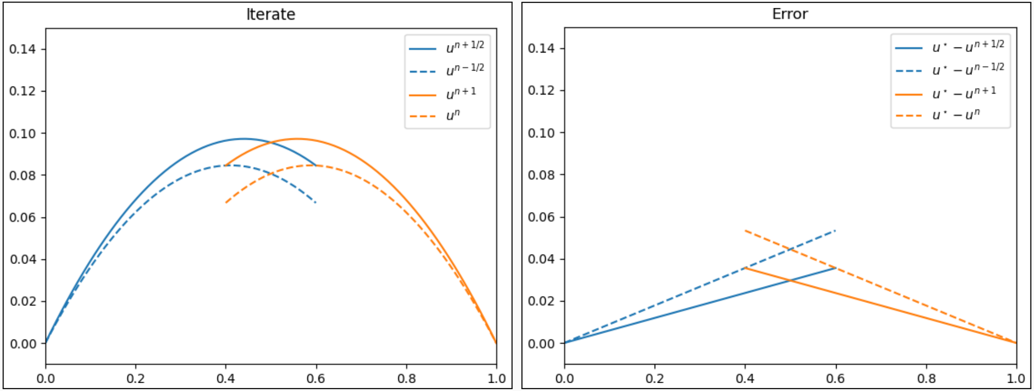


Figure 2: Iterate (left) and error (right) in iteration 4.

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**:

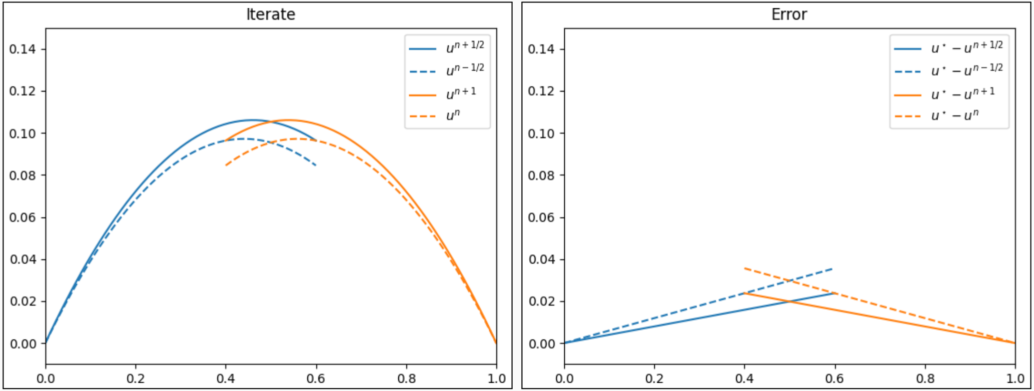
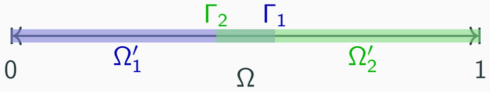


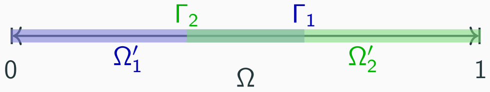
Figure 2: Iterate (left) and error (right) in iteration 5.

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**:

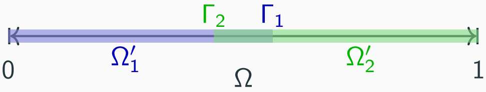


Overlap 0.05

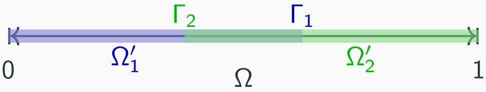


Overlap 0.1

Effect of the Size of the Overlap



Overlap 0.05



Overlap 0.1

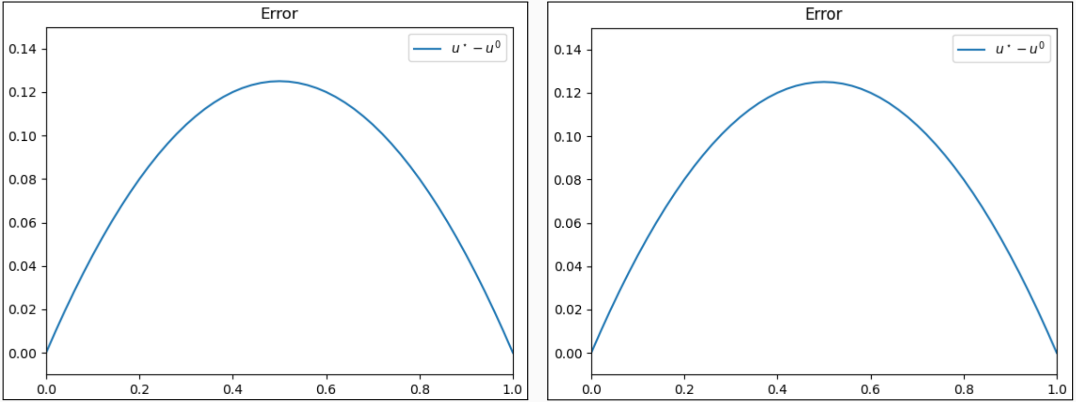
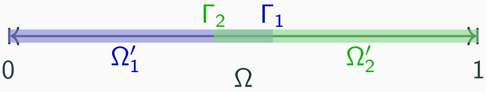
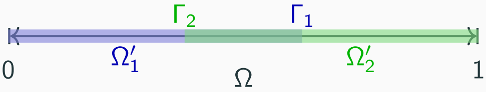


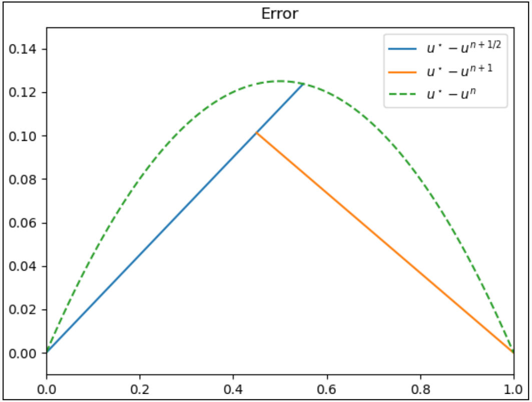
Figure 3: Error in iteration 0.

Effect of the Size of the Overlap





Overlap 0.05



Overlap 0.1

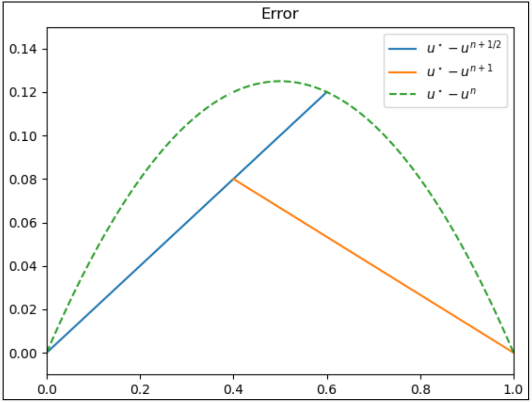
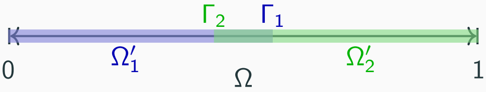
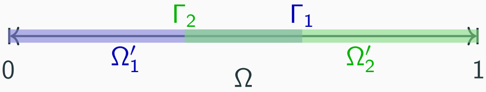


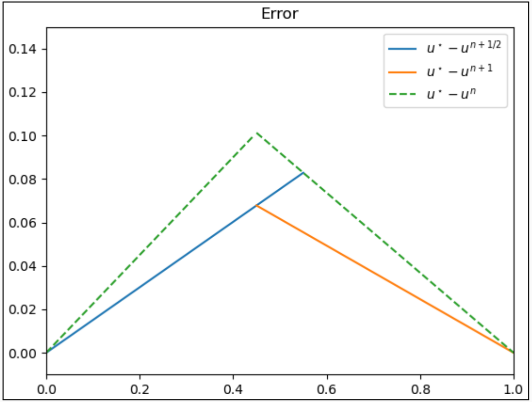
Figure 3: Error in iteration 1.





Overlap 0.05

Overlap 0.1



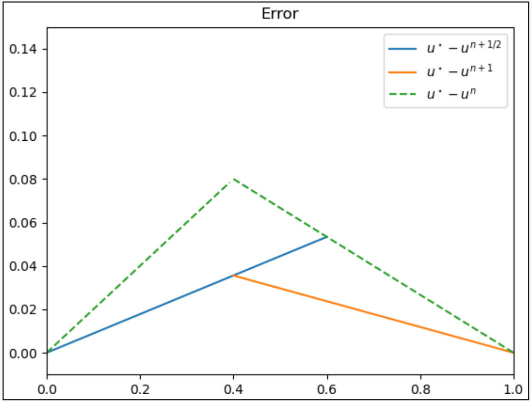
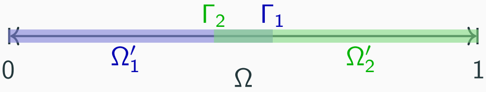
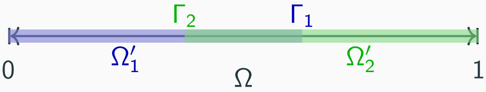


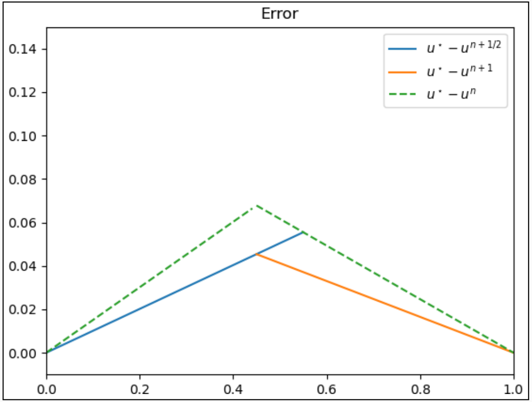
Figure 3: Error in iteration 2.

Effect of the Size of the Overlap





Overlap 0.05



Overlap 0.1

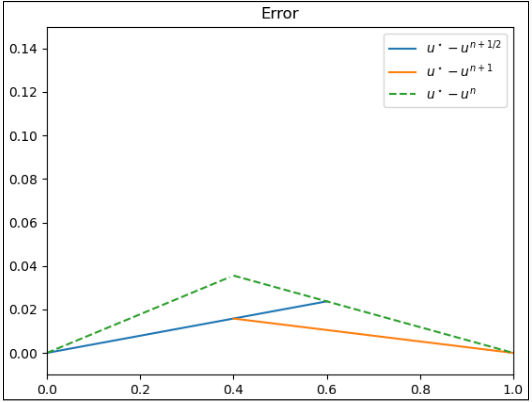
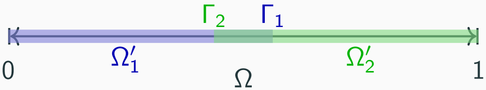
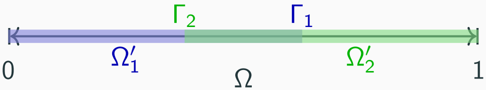


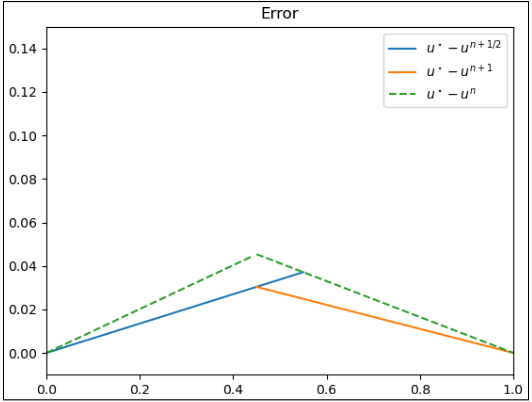
Figure 3: Error in iteration 3.

Effect of the Size of the Overlap





Overlap 0.05



Overlap 0.1

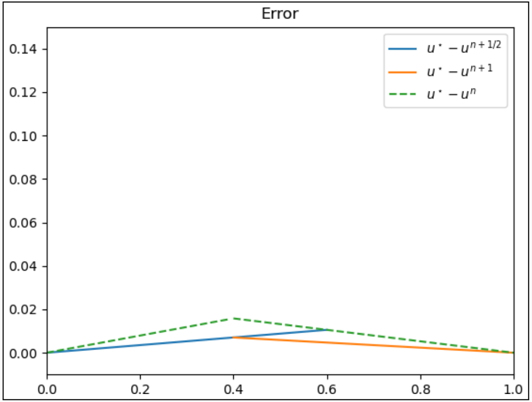
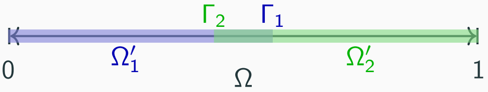
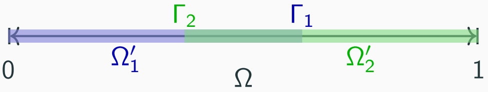


Figure 3: Error in iteration 4.

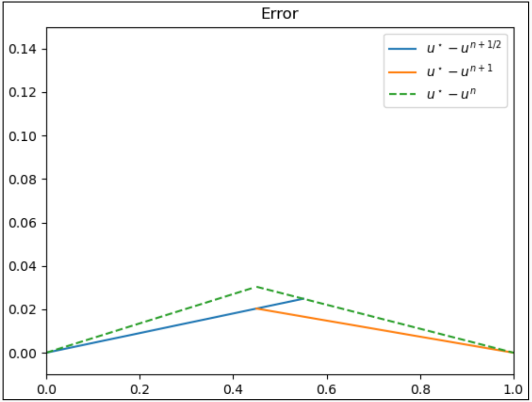
Effect of the Size of the Overlap



Overlap 0.05



Overlap 0.1



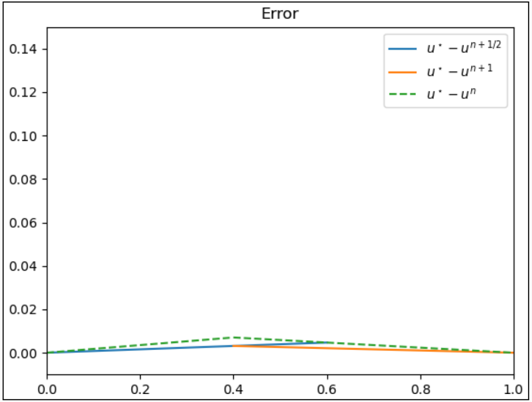


Figure 3: Error in iteration 5.

Effect of the Size of the Overlap

Overlap 0.05 | **Overlap 0.1**

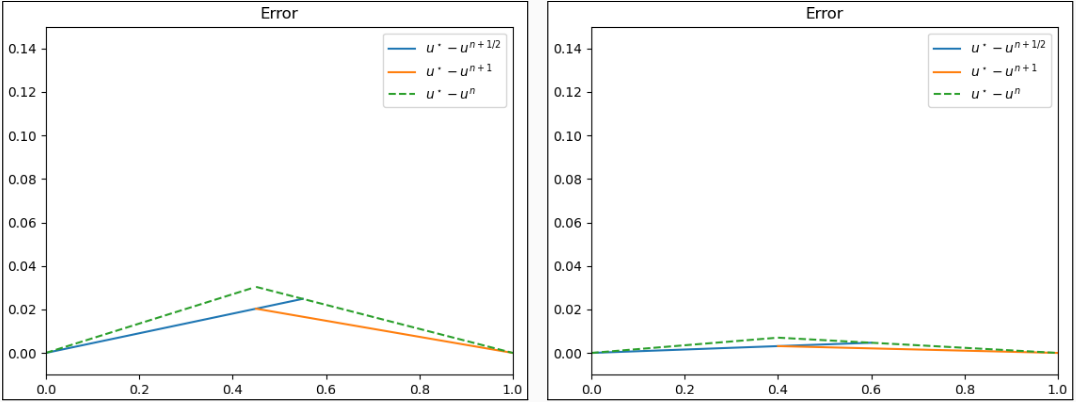


Figure 3: Error in iteration 5.

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

Schwarz Domain Decomposition Preconditioners

Solvers for Partial Different Equations

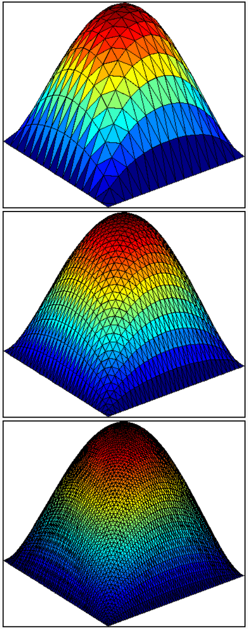
Consider a **diffusion model problem**:

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

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

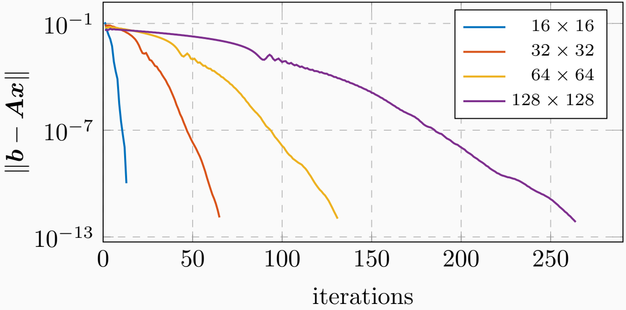
Iterative solvers are efficient for solving **sparse systems**, however, the **convergence rate depends on the condition number**:

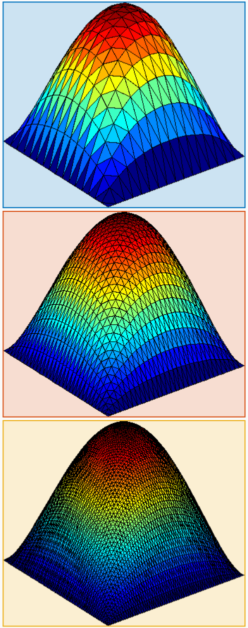
$$\kappa(\boldsymbol{K}) = \frac{\lambda_{\max}(\boldsymbol{K})}{\lambda_{\min}(\boldsymbol{K})} \leq \frac{C}{h^2}$$

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**:

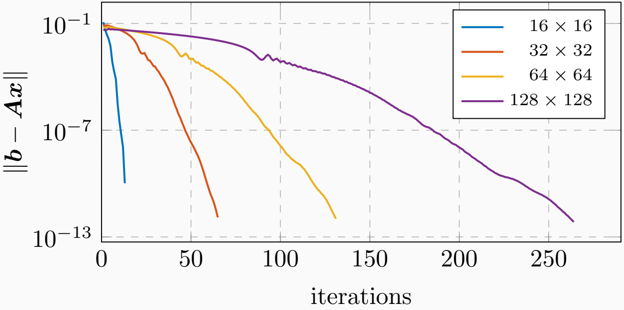


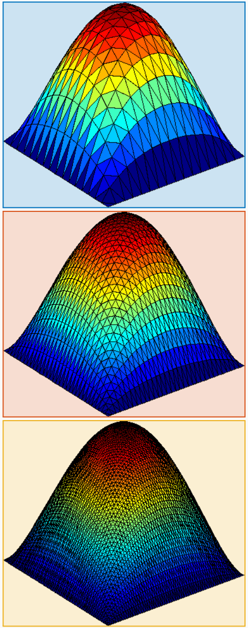


Consider a **diffusion model problem**:

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

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





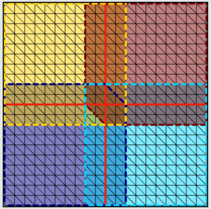
$\Rightarrow$ Introduce a preconditioner $\boldsymbol{M}^{-1} \approx \boldsymbol{K}^{-1}$ to **improve convergence**:

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

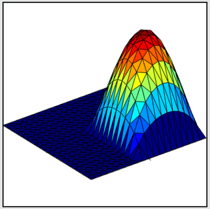
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



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

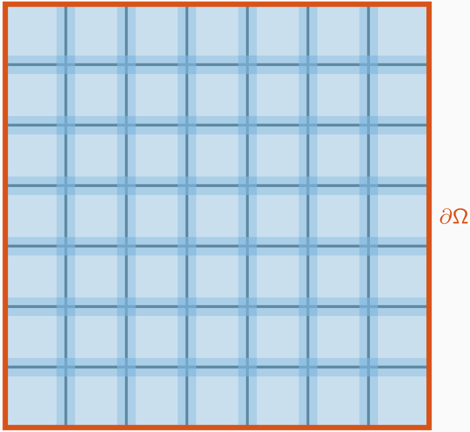
$$\boldsymbol{M}_{\text{OS-1}}^{-1}\boldsymbol{K} = \sum\nolimits_{i=1}^{N} \boldsymbol{R}_i^\top \boldsymbol{K}_i^{-1} \boldsymbol{R}_i \boldsymbol{K},$$

where $\boldsymbol{R}_i$ and $\boldsymbol{R}_i^\top$ are restriction and prolongation operators corresponding to Ω_i', and $\boldsymbol{K}_i := \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^\top$.

Condition number estimate:

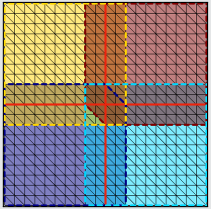
$$\kappa\left(\boldsymbol{M}_{\text{OS-1}}^{-1}\boldsymbol{K}\right) \leq C\left(1 + \frac{1}{H\delta}\right)$$

with subdomain size H and overlap width δ.

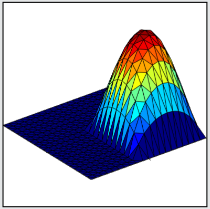


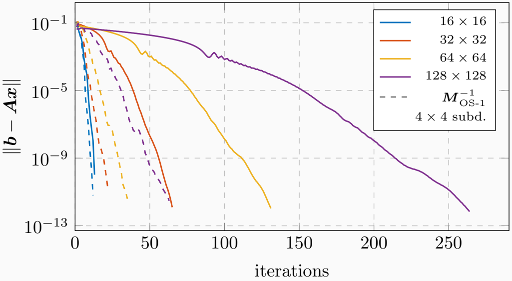
One-level Schwarz preconditioner

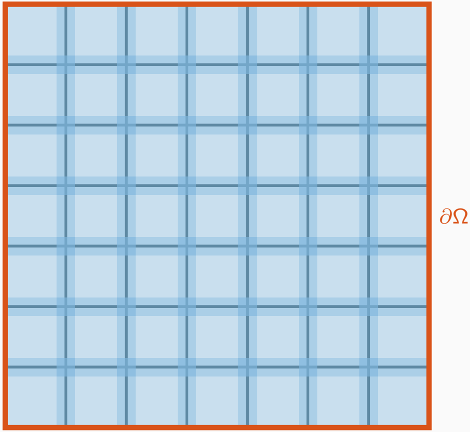
Overlap $\delta = 1h$



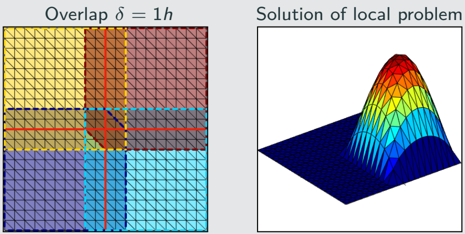
Solution of local problem

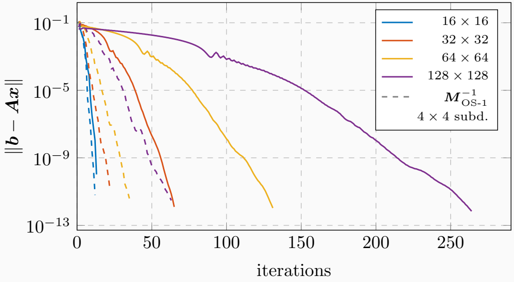


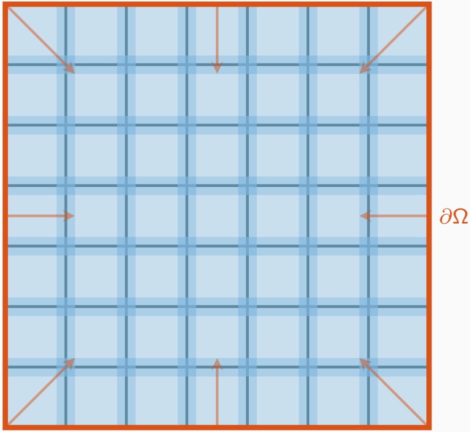




One-level Schwarz preconditioner

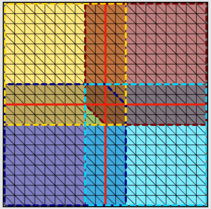


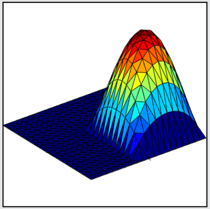


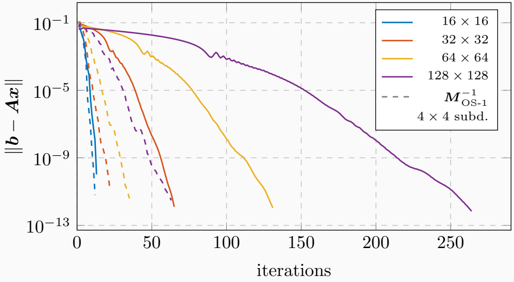


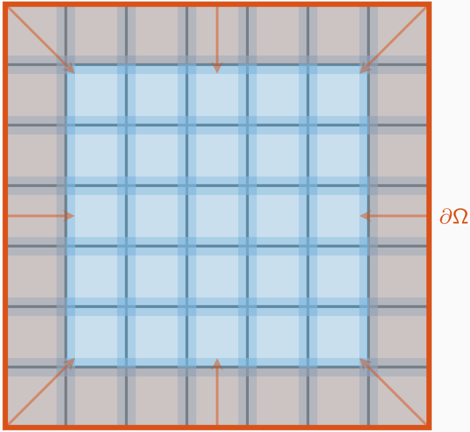
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner





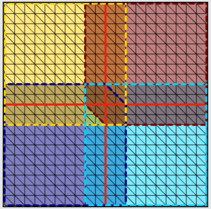




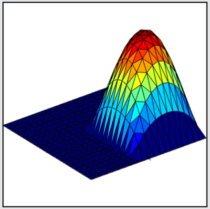
Two-Level Schwarz Preconditioners

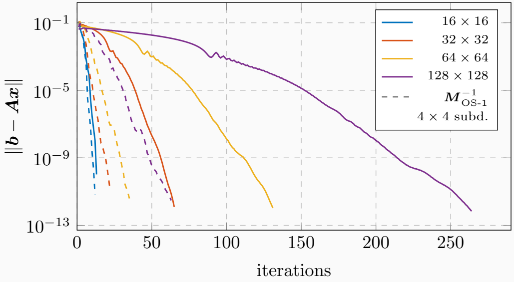
One-level Schwarz preconditioner

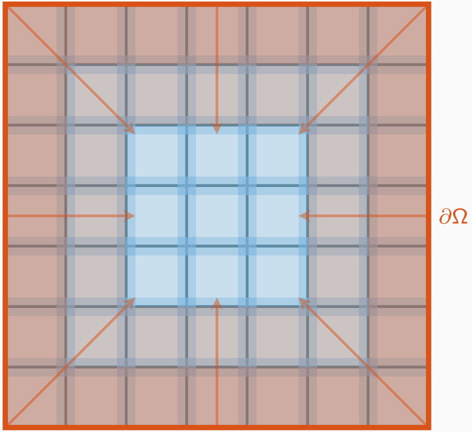
Overlap $\delta = 1h$



Solution of local problem

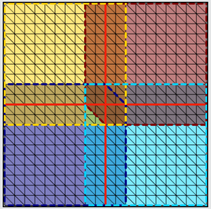




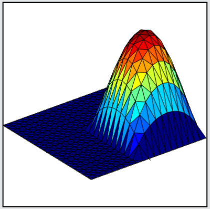


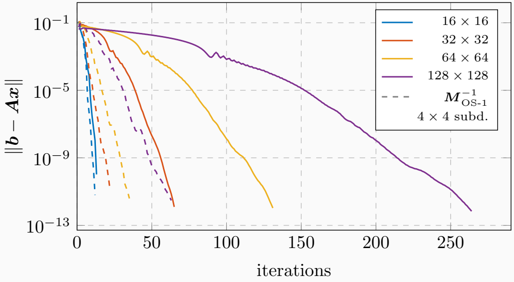
One-level Schwarz preconditioner

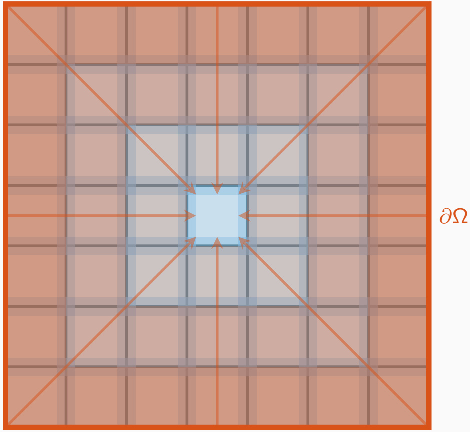
Overlap $\delta = 1h$



Solution of local problem



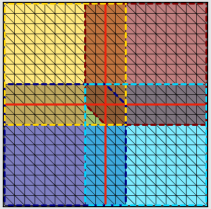




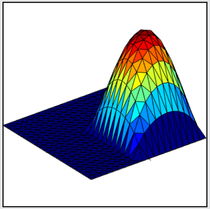
Two-Level Schwarz Preconditioners

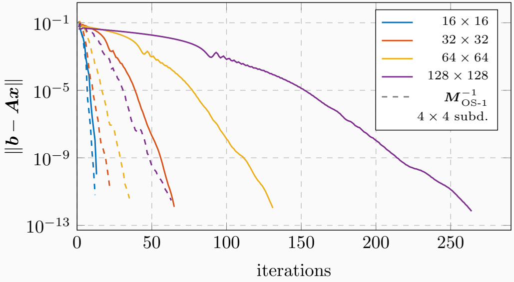
One-level Schwarz preconditioner

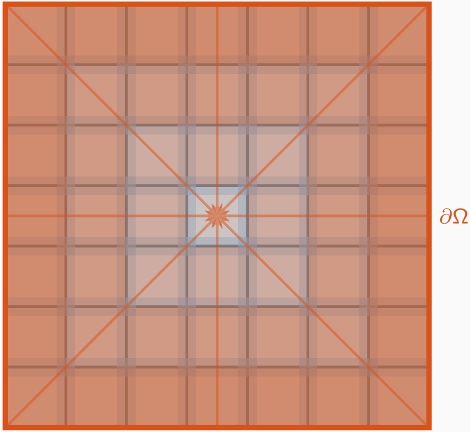
Overlap $\delta = 1h$



Solution of local problem



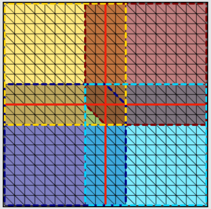




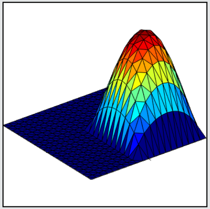
Information (in particular, boundary data) is **only exchanged via the overlapping regions**, leading to **slow convergence** → establish a **faster / global transport of information**.

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 $K_i := R_i K R_i^\top$.

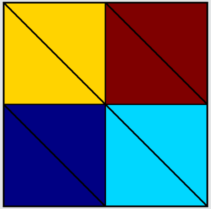
Condition number estimate:

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

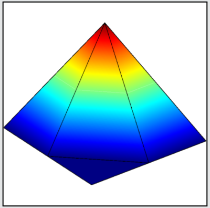
with subdomain size H and overlap width δ.

Lagrangian coarse space

Coarse triangulation



Coarse solution



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

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

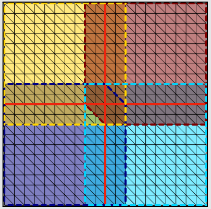
where Φ contains the coarse basis functions and $K_0 := \Phi^\top K \Phi$; cf., e.g., **Toselli, Widlund (2005)**. The construction of a Lagrangian coarse basis requires a coarse triangulation.

Condition number estimate:

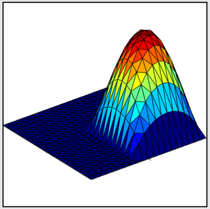
$$\kappa \left(M_{\text{OS-2}}^{-1} K \right) \leq C \left(1 + \frac{H}{\delta} \right)$$

One-level Schwarz preconditioner

Overlap $\delta = 1h$

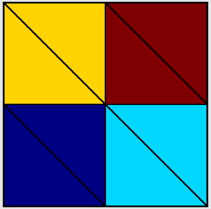


Solution of local problem

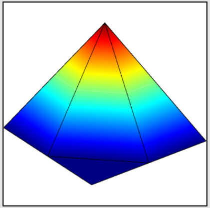


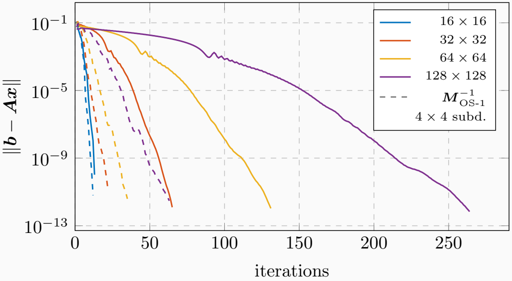
Lagrangian coarse space

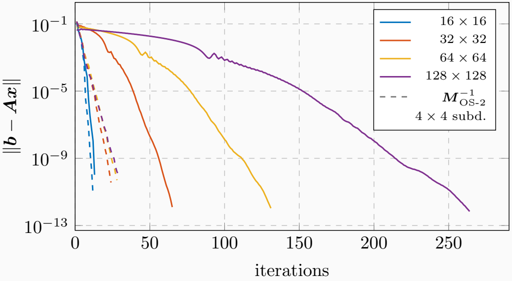
Coarse triangulation



Coarse solution

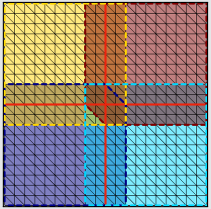




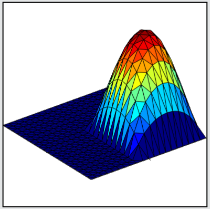


One-level Schwarz preconditioner

Overlap $\delta = 1h$

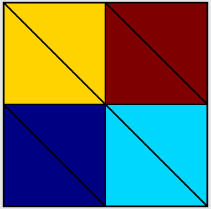


Solution of local problem

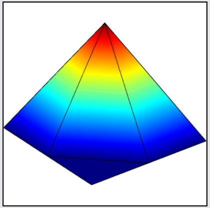


Lagrangian coarse space

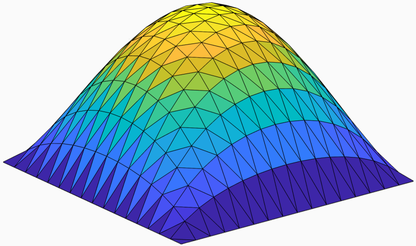
Coarse triangulation

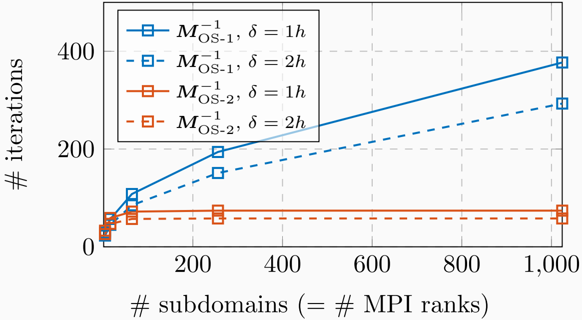


Coarse solution



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





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)

Overlapping domain decomposition

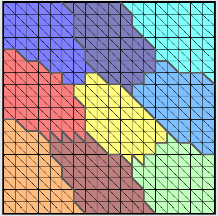
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

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

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

Nonoverlapping DD



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

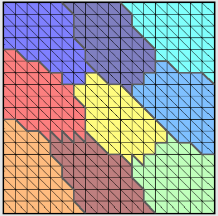
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

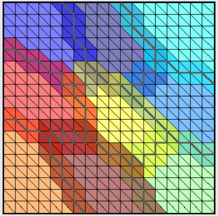
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

Nonoverlapping DD



Overlap $\delta = 1h$



Overlapping domain decomposition

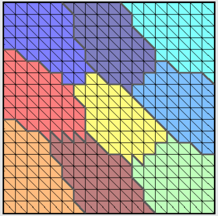
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

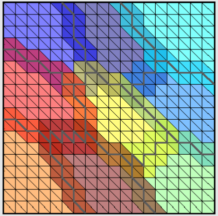
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

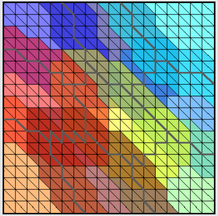
Nonoverlapping DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$



Overlapping domain decomposition

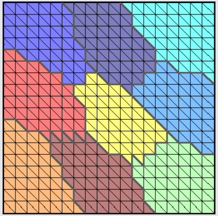
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

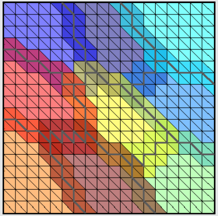
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

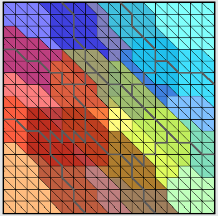
Nonoverlapping DD



Overlap $\delta = 1h$

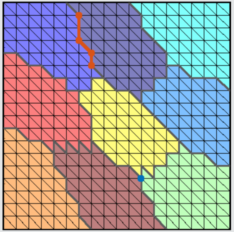


Overlap $\delta = 2h$



Coarse space – Example of Generalized Dryja–Smith–Widlund (GDSW)

1. Interface components



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

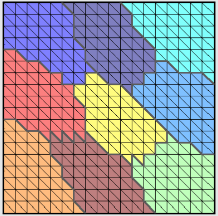
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

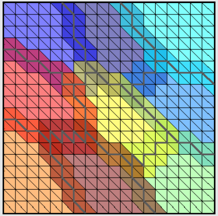
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

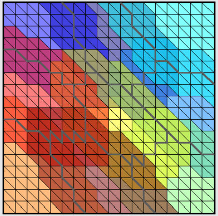
Nonoverlapping DD



Overlap $\delta = 1h$

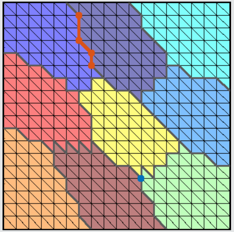


Overlap $\delta = 2h$

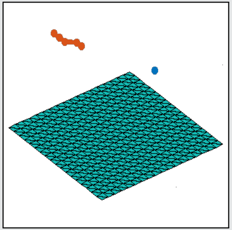


Coarse space – Example of Generalized Dryja–Smith–Widlund (GDSW)

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

Overlapping domain decomposition

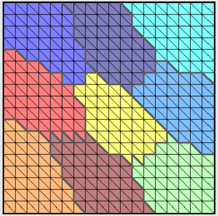
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

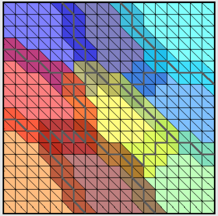
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

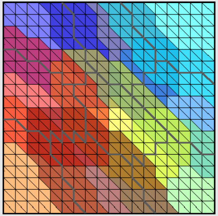
Nonoverlapping DD



Overlap $\delta = 1h$

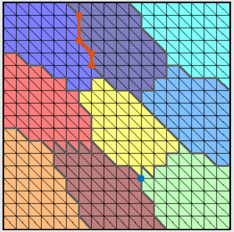


Overlap $\delta = 2h$

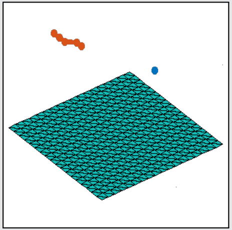


Coarse space – Example of Generalized Dryja–Smith–Widlund (GDSW)

1. Interface components

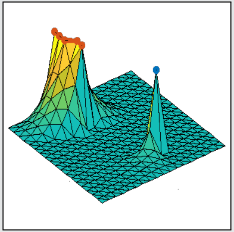


2. Interface basis (partition of unity $\times$ null space)



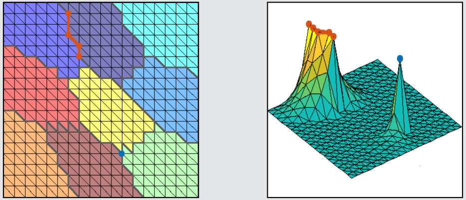
For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

3. Extension



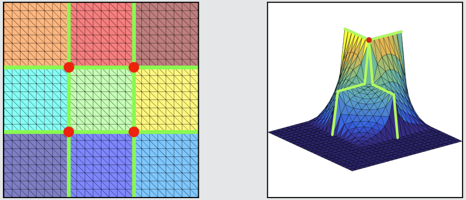
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)



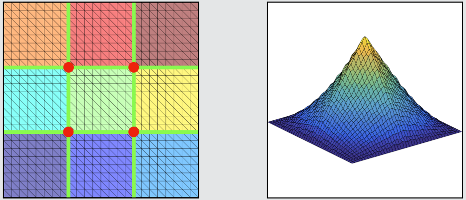
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)



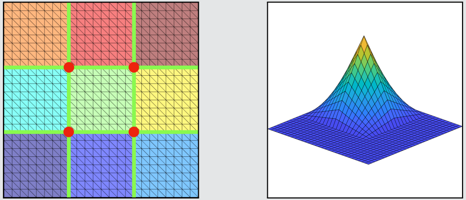
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

MsFEM (Multiscale Finite Element Method)



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

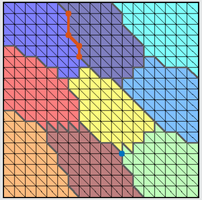
Q1 Lagrangian / piecewise bilinear

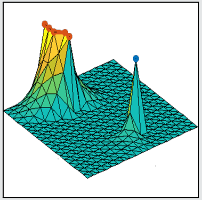


Piecewise linear interface partition of unity functions and a **structured domain decomposition**.

Examples of FROSch Coarse Spaces

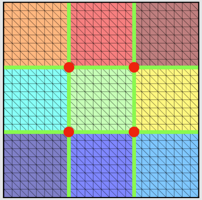
GDSW (Generalized Dryja–Smith–Widlund)

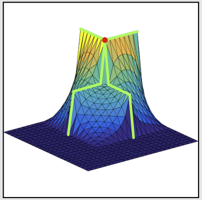




- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

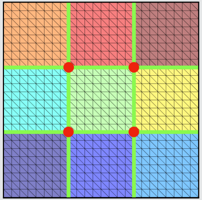
RGDSW (Reduced dimension GDSW)

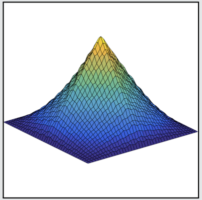




- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

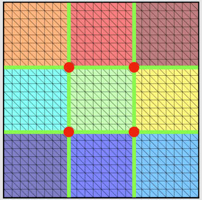
MsFEM (Multiscale Finite Element Method)

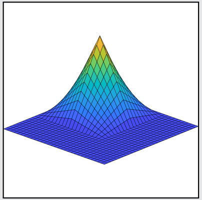




- Hou (1997), Efendiev and Hou (2009)
- Buck, Iliev, and Andrä (2013)
- H., Klawonn, Knepper, Rheinbach (2018)

Q1 Lagrangian / piecewise bilinear



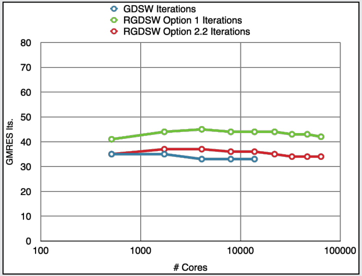


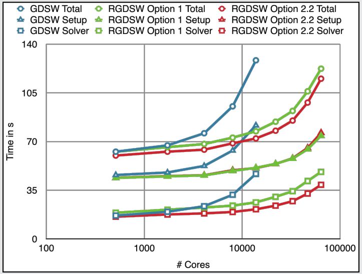
Piecewise linear interface partition of unity functions and a **structured domain decomposition**.

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

GDSW vs RGDSW (reduced dimension)

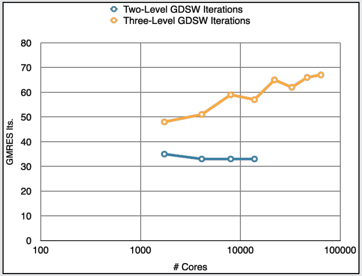
Heinlein, Klawonn, Rheinbach, Widlund (2019).

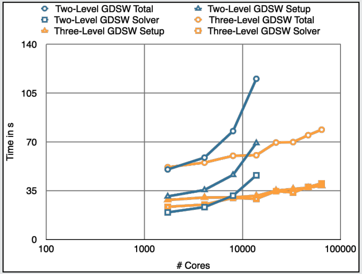




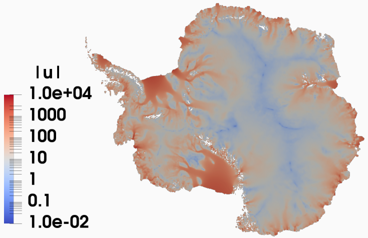
Two-level vs three-level GDSW

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





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**,

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

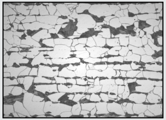
with a **nonlinear viscosity model** (Glen's law); cf., e.g., **Blatter (1995)** and **Pattyn (2003)**.

	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	-	-	-

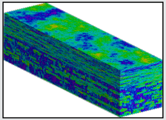
Computations performed on Cori (NERSC). **Heinlein, Perego, Rajamanickam (2022)**

Highly heterogeneous problems ...

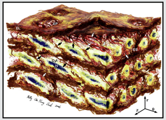
...appear in most areas of modern science and engineering:



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)**).

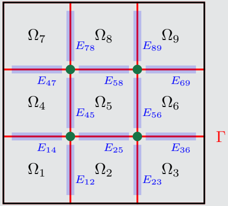
Related works (non-exhaustive)

- FETI & Neumann–Neumann: **Bjørstad, Krzyzanowski (2002); Bjørstad, Koster, Krzyzanowski (2001); Rixen, Spillane (2013); Spillane (2015, 2016)** ...
- BDDC & FETI-DP: **Mandel, Sousedík (2007); Sousedík (2010); Šístek, Mandel, Sousedík (2012); Dohrmann, Pechstein (2013, 2016); Klawonn, Radtke, Rheinbach (2014, 2015, 2016); Klawonn, Kühn, Rheinbach (2015, 2016, 2017); Kim, Chung (2015); Kim, Chung, Wang (2017); Beirão da Veiga et al. (2017); Calvo, Widlund (2016); Oh et al. (2017)** ...
- Overlapping Schwarz: **Galvis, Efendiev (2010, 2011); Nataf, Xiang, Dolean, Spillane (2011); Spillane et al. (2011); Gander, Loneland, Rahman (preprint 2015); Eikeland, Marcinkowski, Rahman (TR 2016); Marcinkowski, Rahman (2018), Al Daas, Grigori, Jolivet, Tournier (2021); Bastian, Scheichl, Seelinger, Strehlow (2022); Spillane (preprint 2021, acc. 2024); Al Daas, Jolivet (2022); Bootland, Dolean, Graham, Ma, Scheichl (2023)** ...
- Spectral AMGe (ρAMGe): **Chartier, Falgout, Henson, Jones, Manteuffel, McCormick, Ruge, Vassilevski (2003)** ...

Local eigenvalue problems

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

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

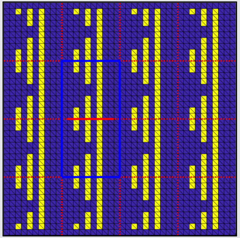


with **Schur complements** S_{EE}, S_{FF} with **Neumann boundary conditions** and **submatrices** K_{EE}, K_{FF} of K. We select eigenfunctions corresponding to **eigenvalues below tolerances** $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$.

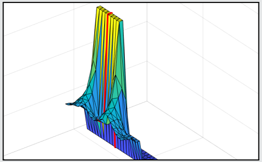
$\rightarrow$ The corresponding coarse basis functions are **energy-minimizing extensions** into the interior of the subdomains.

Extensions in the generalized eigenvalue problem

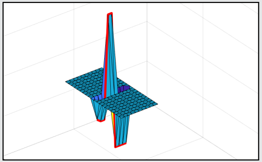
Blue $\alpha = 1$; yellow $\alpha = 10^6$



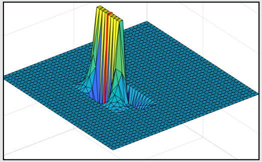
Low energy extension $\boldsymbol{S}_{EE}$



High energy extension $\boldsymbol{K}_{EE}$



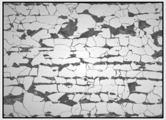
Coarse basis function



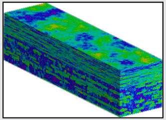
The extensions on the two sides of the generalized eigenvalue problem correspond to **low and high energy extensions of the trace** $\rightarrow$ detects coefficient jumps.

Highly heterogeneous problems . . .

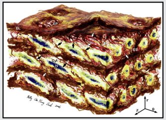
. . . appear in most areas of modern science and engineering:



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

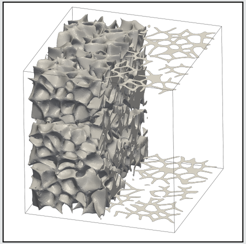
The coarse space is **enhanced** by eigenfunctions of **local edge and face eigenvalue problems** with eigenvalues below tolerances $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$:

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

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**	$1.3\cdot10^6$	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)**.

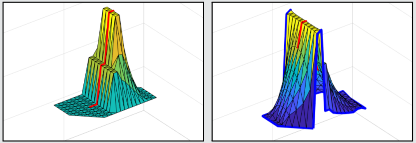
Two algebraic eigenvalue problems

Use the a-orthogonal decomposition

$$V_{\Omega_e} = V^0_{\Omega_e} \oplus \{E_{\partial\Omega_e \to \Omega_e}(v) : v \in V_{\partial\Omega_e}\}$$

to ***"split the AGDSW (Neumann) eigenvalue problem"*** into two:

- **Dirichlet eigenvalue problem** on $V^0_{\Omega_e}$
- **Transfer eigenvalue problem** on $V_{\Omega_e,\text{harm}}$; cf. **Smetana, Patera (2016)**



Condition number estimate

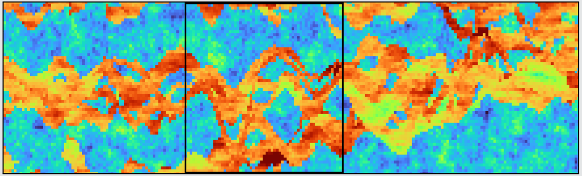
$$\kappa\left(\boldsymbol{M}^{-1}_{\text{DIR\&TR}}\boldsymbol{K}\right) \leq C \max\left\{{}^{1}/_{TOL_{\text{DIR}}}, {}^{TOL_{\text{TR}}}/_{\alpha_{\min}}\right\},$$

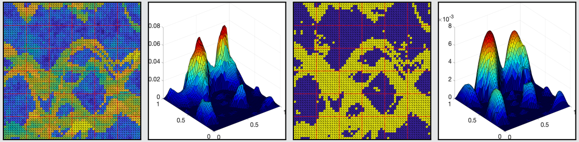
where C is independent of H, h, and the contrast of the coefficient function α.

Heinlein & Smetana (acc. 2024; preprint arXiv).

Numerical results – SPE10 benchmark

Layer 70 from model 2; cf. **Christie and Blunt (2001)**





V_0	TOL_{DIR}	TOL_{TR}	dim V_0	κ	its.
V_{GDSW}	-	-	85	$2.0\cdot10^5$	57
V_{AGDSW}	$1.0\cdot10^{-2}$		93	19.3	38
$V_{\text{DIR\&TR}-a}$	$1.0\cdot10^{-3}$	$1.0\cdot10^{5}$	90	19.4	39
$V_{\text{DIR\&TR}-l^2}$	$1.0\cdot10^{-3}$	$1.0\cdot10^{5}$	147	9.6	31
Original coefficient (without thresholding)					
V_{GDSW}	-	-	85	20.6	42

Domain Decomposition for Neural Networks

Domain Decomposition Methods and Machine Learning – Literature

A **non-exhaustive literature overview**:

- **Machine Learning for adaptive BDDC, FETI–DP, and AGDSW**: **Heinlein, Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022); Klawonn, Lanser, Weber (2024)**
- **cPINNs, XPINNs**: **Jagtap, Kharazmi, Karniadakis (2020); Jagtap, Karniadakis (2020)**
- **Classical Schwarz iteration for PINNs or DeepRitz (D3M, DeepDDM, etc)**:: **Li, Tang, Wu, and Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Dolean, Heinlein, Mercier, Gratton (subm. 2024 / arXiv:2408.12198); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2022, arXiv 2023); Kim, Yang (2022, arXiv 2023)**
- **FBPINNs**, **FBKANs**: **Moseley, Markham, and Nissen-Meyer (2023); Dolean, Heinlein, Mishra, Moseley (2024, 2024); Heinlein, Howard, Beecroft, Stinis (acc. 2024 / arXiv:2401.07888); Howard, Jacob, Murphy, Heinlein, Stinis (arXiv:2406.19662)**
- **DDMs for CNNs**: **Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (2024); Verburg, Heinlein, Cyr (subm. 2024)**

An overview of the state-of-the-art in early 2021:

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.

An overview of the state-of-the-art in mid 2024:

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

Physics-Informed Neural Networks (PINNs)

In the **physics-informed neural network (PINN)** approach introduced by **Raissi et al. (2019)**, a **neural network** is employed to **discretize a partial differential equation**

$$\mathcal{N}[u] = f, \quad \text{in } \Omega.$$

PINNs use a **hybrid loss function**:

$$\mathcal{L}(\boldsymbol{\theta}) = \omega_{\text{data}} \mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}),$$

where ω_{data} and ω_{PDE} are **weights** and

$$\mathcal{L}_{\text{data}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(u(\hat{\mathbf{x}}_i, \boldsymbol{\theta}) - u_i \right)^2,$$

$$\mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \left(\mathcal{N}[u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i) \right)^2.$$

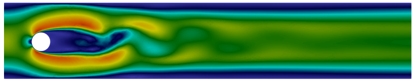
See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.

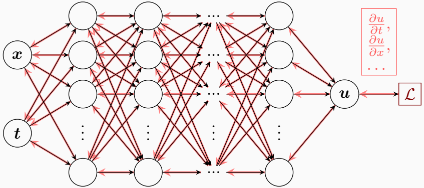
Advantages

- **"Meshfree"**
- **Small data**
- **Generalization properties**
- **High-dimensional problems**
- **Inverse** and **parameterized problems**

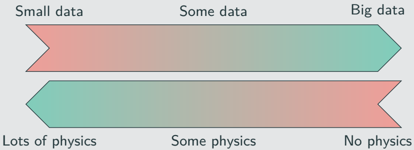
Drawbacks

- **Training cost** and **robustness**
- **Convergence not well-understood**
- **Difficulties with scalability** and **multi-scale problems**





Hybrid loss



- **Known solution values** can be included in $\mathcal{L}_{\text{data}}$
- **Initial and boundary conditions** are also included in $\mathcal{L}_{\text{data}}$

Theoretical Result for PINNs

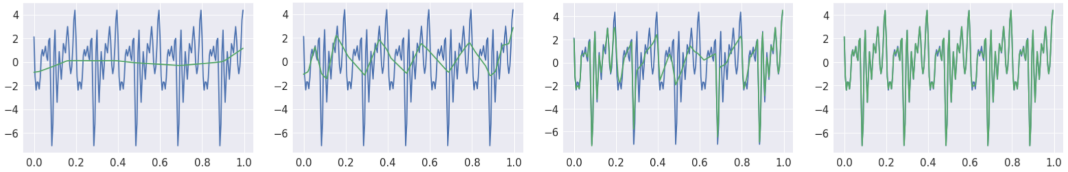
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

$$\mathcal{E}_G \leq C_{\text{PDE}} \mathcal{E}_T + C_{\text{PDE}} C_{\text{quad}}^{1/p} N^{-\alpha/p}$$

- $\mathcal{E}_G = \mathcal{E}_G(\boldsymbol{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **general. error** (V Sobolev space, $\boldsymbol{X}$ training data set)
- $\mathcal{E}_T$ **training error** (l^p **loss** of the residual of the PDE)
- N **number of the training points** and α **convergence rate of the quadrature**
- C_{PDE} and C_{quad} **constants** depending on the **PDE**, **quadrature**, and **neural network**

Rule of thumb: **"As long as the PINN is trained well, it also generalizes well"**



100 iterations | 1 000 iterations | 10 000 iterations | 80 000 iterations

Rahaman et al., *On the spectral bias of neural networks*, **ICML (2019)**

Finite Basis Physics-Informed Neural Networks (FBPINNs)

In the **finite basis physics informed neural network (FBPINNs) method** introduced in **Moseley, Markham, and Nissen-Meyer (2023)**, we employ the **PINN** approach and **hard enforcement of the boundary conditions**; cf. **Lagaris et al. (1998)**.

FBPINNs use the **network architecture**

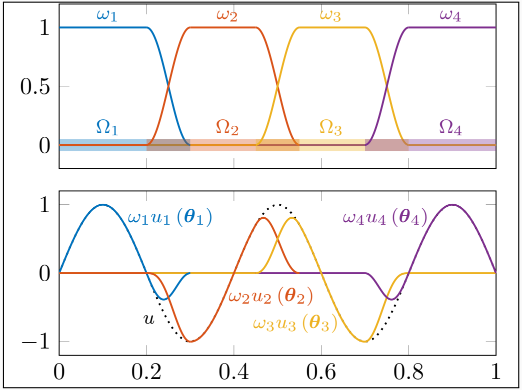
$$u(\theta_1, \ldots, \theta_J) = \mathcal{C} \sum\nolimits_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

$$\mathcal{L}(\theta_1, \ldots, \theta_J) = \frac{1}{N} \sum_{i=1}^{N} \Big(\mathcal{N}[\mathcal{C} \sum_{x_i \in \Omega_j} \omega_j u_j](x_i, \theta_j) - f(x_i) \Big)^2.$$

Here:

- **Overlapping DD**: $\Omega = \bigcup_{l=1}^{J} \Omega_j$
- **Partition of unity** ω_j with $\mathrm{supp}(\omega_j) \subset \Omega_j$ and $\sum_{j=1}^{J} \omega_j \equiv 1$ on Ω



Hard enf. of boundary conditions

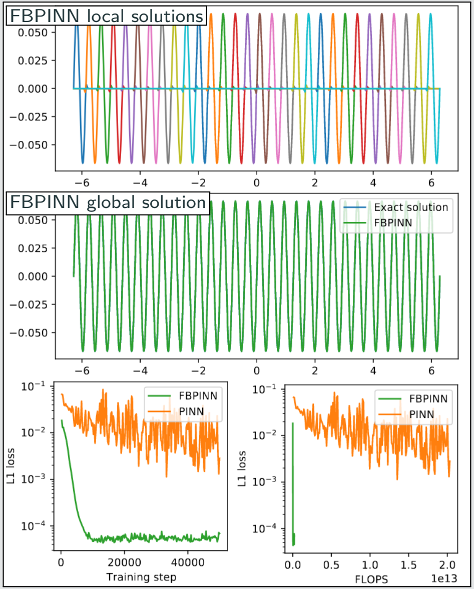
Loss function

$$\mathcal{L}(\theta) = \frac{1}{N} \sum\nolimits_{i=1}^{N} \left(\mathcal{N}[\mathcal{C}u](x_i, \theta) - f(x_i) \right)^2,$$

with constraining operator $\mathcal{C}$, which **explicitly enforces the boundary conditions**.

Numerical Results for FBPINNs

PINN vs FBPINN (Moseley et al. (2023))



Scalability of FBPINNs

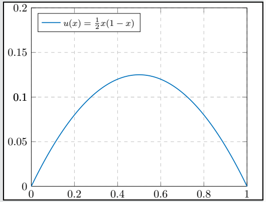
Consider the **simple boundary value problem**

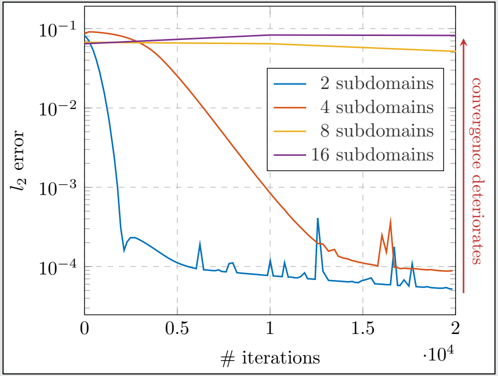
$$-u'' = 1 \text{ in } [0,1],$$

$$u(0) = u(1) = 0,$$

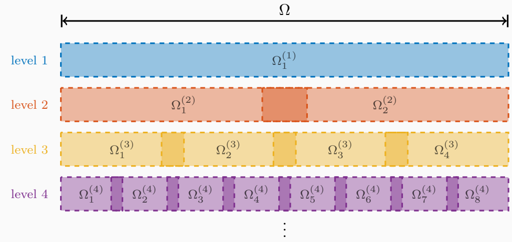
which has the **solution**

$$u(x) = 1/2\,x(1-x).$$



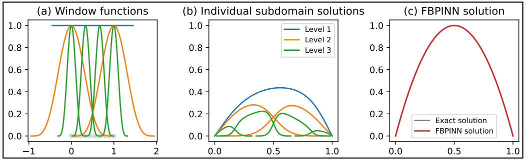


Extension of FBPINNs to L levels; Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.



L-level network architecture

$$u\left(\theta_1^{(1)}, \ldots, \theta_{J^{(L)}}^{(L)}\right) = \mathcal{C}\left(\sum_{l=1}^{L} \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\theta_j^{(l)}\right)\right)$$



Multi-Frequency Problem

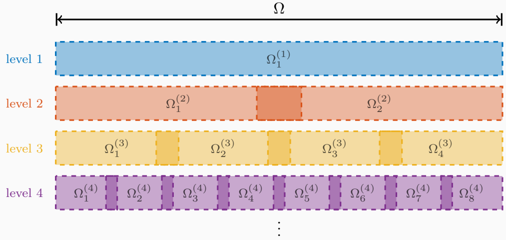
Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

with $\omega_i = 2^i$.

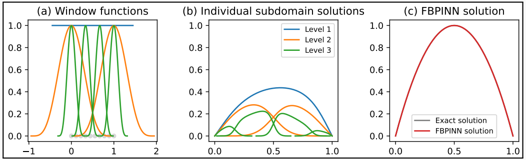
For increasing values of n, we obtain the **analytical solutions**:

Extension of FBPINNs to L levels; Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.



L-level network architecture

$$u\left(\theta_1^{(1)}, \ldots, \theta_{J^{(L)}}^{(L)}\right) = \mathcal{C}\left(\sum_{l=1}^{L} \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\theta_j^{(l)}\right)\right)$$



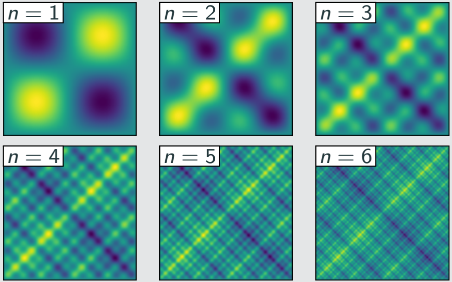
Multi-Frequency Problem

Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

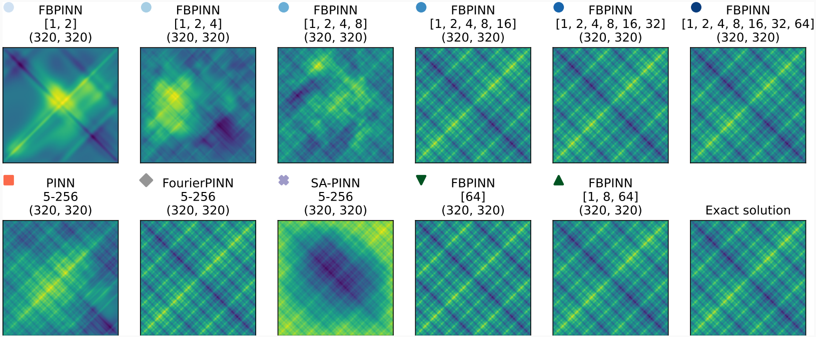
$$\begin{aligned} -\Delta u &= 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

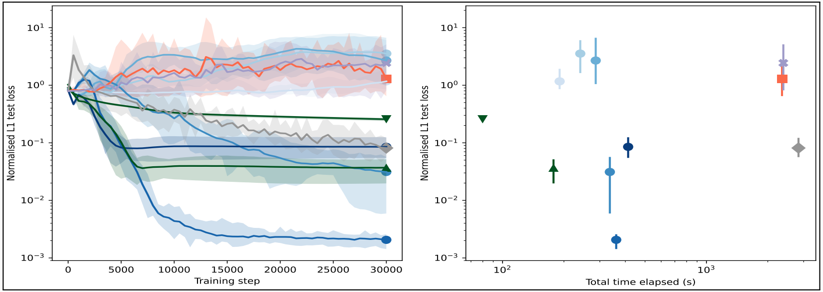
with $\omega_i = 2^i$.

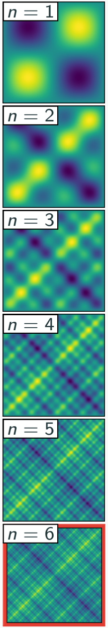
For increasing values of n, we obtain the **analytical solutions**:



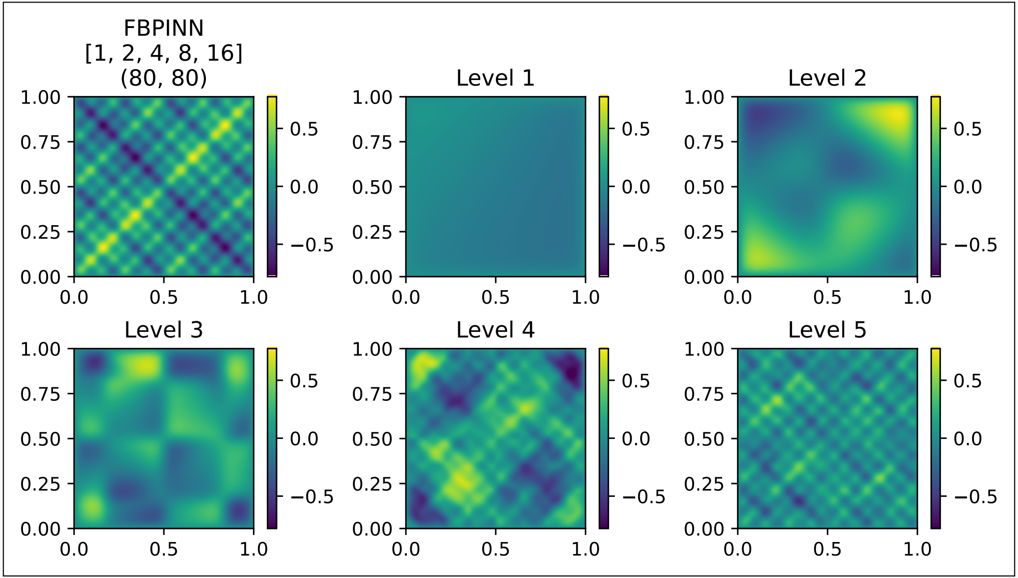
Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling





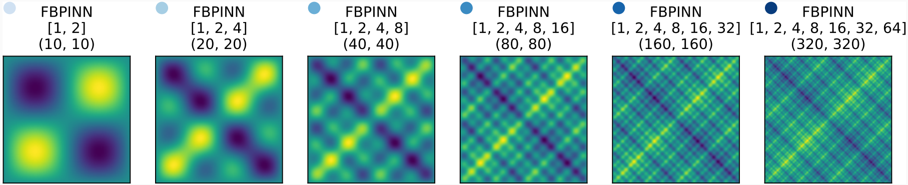


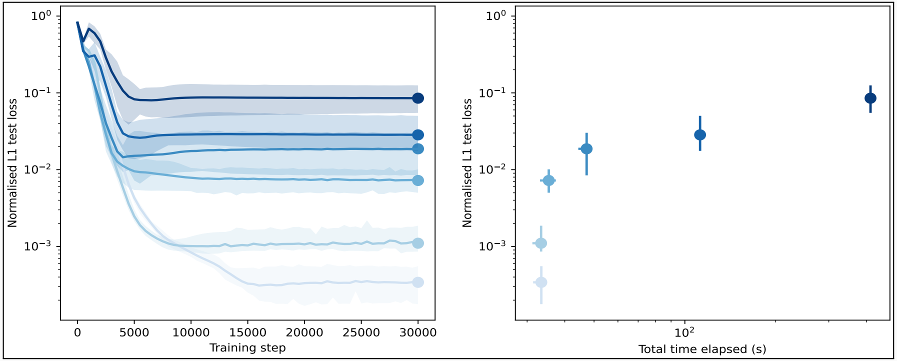
Multi-Frequency Problem – What the FBPINN Learns

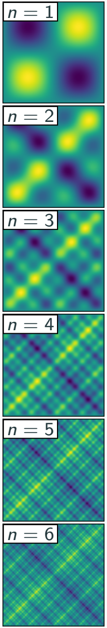


Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

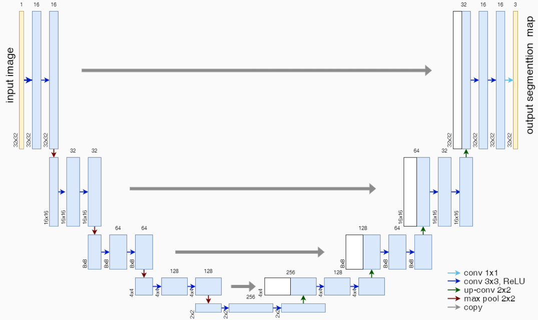
Multi-Level FBPINNs for a Multi-Frequency Problem – Weak Scaling







Cf. **Dolean, Heinlein, Mishra, Moseley (2024)**.

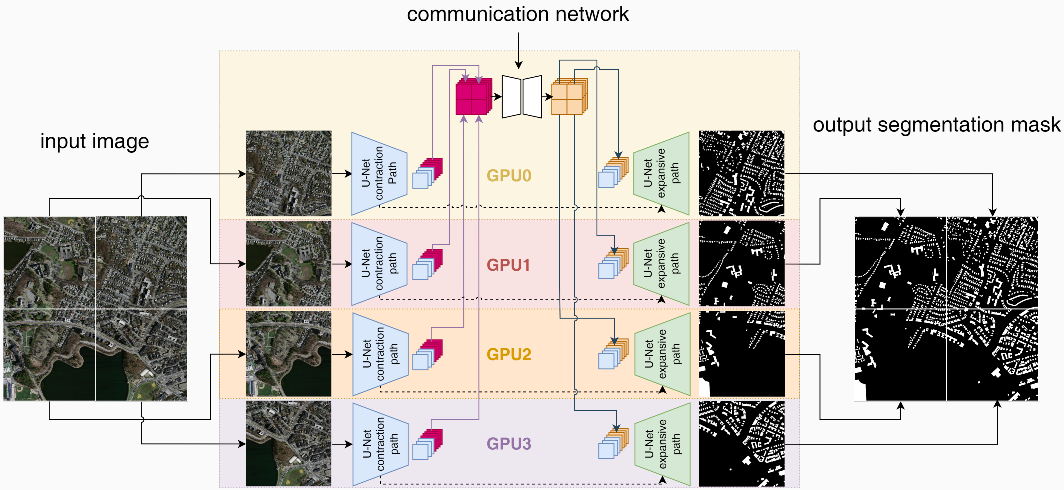


- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy in semantic image segmentation** and other **image-to-image tasks**.

Below: *memory consumption for training on a single* 1024×1024 *image.*

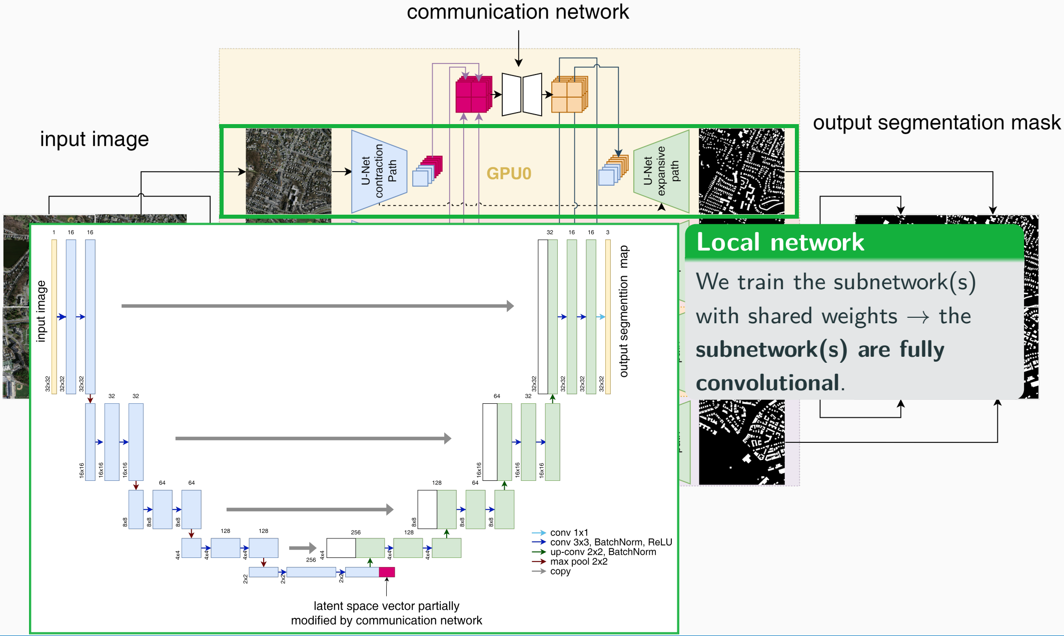
name	size	# channels input	# channels output	mem. feature maps # of values	mem. feature maps MB	mem. weights # of values	mem. weights MB
input block	1 024	3	64	268 M	**1 024.0**	38 848	**0.148**
encoder block 1	512	64	128	167 M	**704.0**	221 696	**0.846**
encoder block 2	256	128	256	84 M	**352.0**	885 760	**3.379**
encoder block 3	128	256	512	42 M	**176.0**	3 540 992	**13.508**
encoder block 4	64	512	1 024	21 M	**88.0**	14 159 872	**54.016**
decoder block 1	64	1,024	512	50 M	**192.0**	9 177 088	**35.008**
decoder block 2	128	512	256	101 M	**384.0**	2 294 784	**8.754**
decoder block 3	256	256	128	201 M	**768.0**	573 952	**2.189**
decoder block 4	512	128	64	402 M	**1 536.0**	143 616	**0.548**
output block	1 024	64	3	3.1 M	**12.0**	195	**0.001**

Decomposing the U-Net



Cf. **Verburg, Heinlein, Cyr (subm. 2024).**

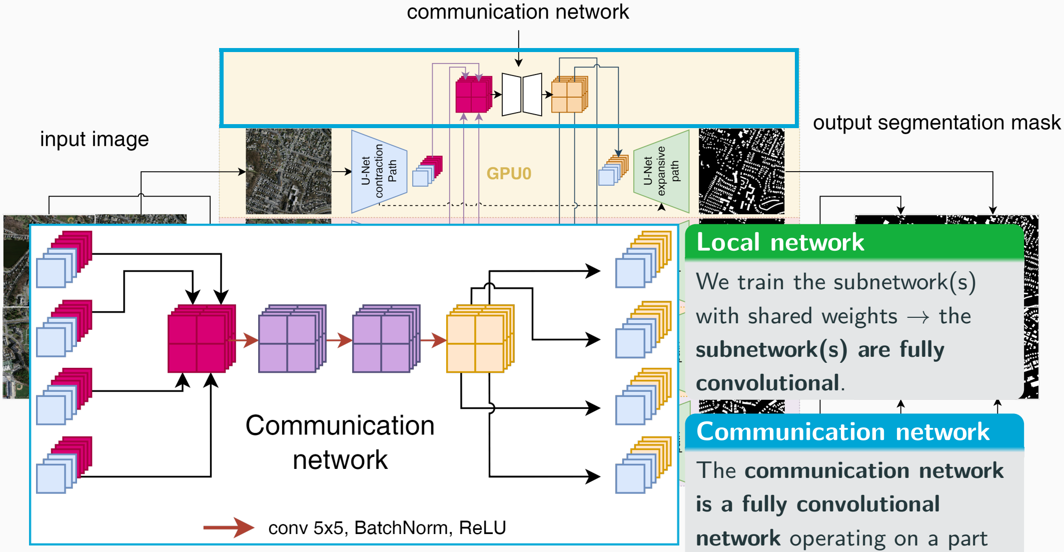
Decomposing the U-Net



Local network

We train the subnetwork(s) with shared weights → the **subnetwork(s) are fully convolutional**.

Decomposing the U-Net

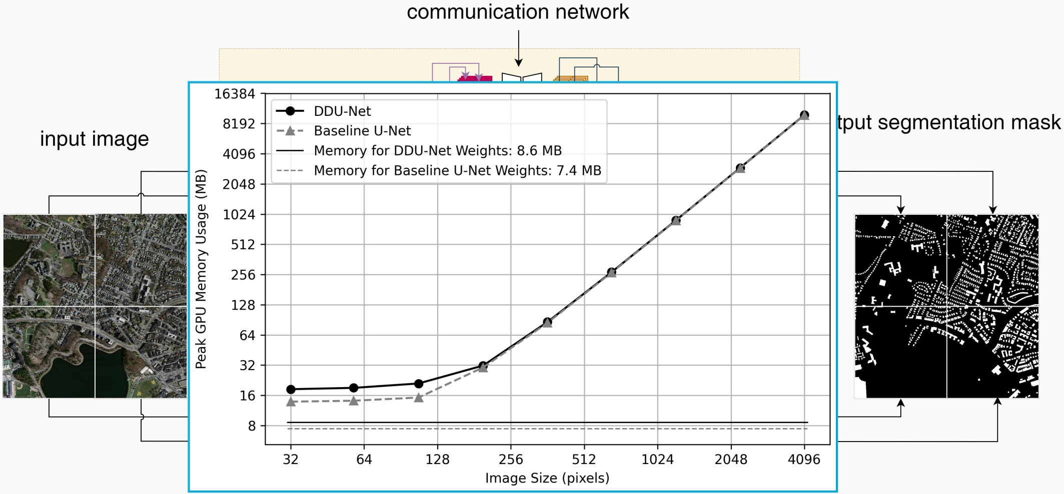


Local network

We train the subnetwork(s) with shared weights $\rightarrow$ the **subnetwork(s) are fully convolutional**.

Communication network

The **communication network is a fully convolutional network** operating on a part of the coarse feature maps.

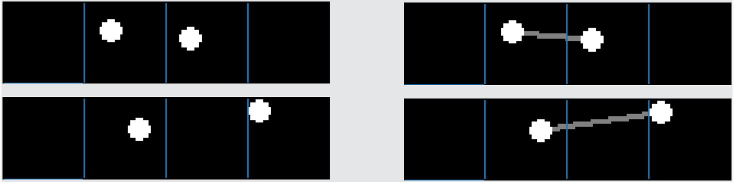


- Distribution of feature maps results in **significant reduction of memory usage on a single GPU**
- **Moderate additional memory usage** due to the **communication network**

Results – Synthetic Data Set

Task: Connect two dots via a line segment

Input | Target (segmentation mask)

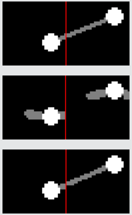


Result: Communication

True mask

Pred. (no comm.)

Pred. (comm.)



Training & testing on # of subimages (32 × 32 pixels) vs. # feature maps communicated

	0	1	2	4	8	16	32	Baseline
2	0.14	0.55	0.57	0.87	0.91	0.92	0.93	0.95
3	0.14	0.44	0.66	0.77	0.84	0.90	0.92	0.81
4	0.14	0.58	0.61	0.77	0.84	0.88	0.90	0.74
6	0.14	0.23	0.54	0.67	0.72	0.82	0.87	0.68
8	0.14	0.18	0.40	0.50	0.63	0.70	0.69	0.39
16	0.08	0.18	0.18	0.18	0.18	0.18	0.20	0.19

Testing on 6 subimages

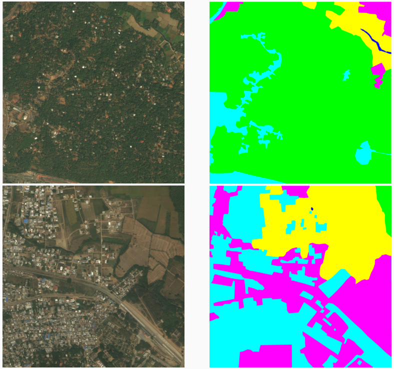
Training on # subimages (32 × 32 pixels) vs. # feature maps communicated

	0	1	2	4	8	16	32	Baseline
2	0.13	0.20	0.20	0.29	0.28	0.30	0.40	0.53
3	0.14	0.39	0.65	0.75	0.79	0.88	0.92	0.76
4	0.14	0.59	0.55	0.77	0.78	0.83	0.73	0.75
6	0.14	0.23	0.54	0.67	0.72	0.82	0.87	0.68
8	0.13	0.18	0.39	0.49	0.64	0.71	0.70	0.42
16	0.13	0.14	0.14	0.16	0.18	0.16	0.26	0.26

DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %

Input Target

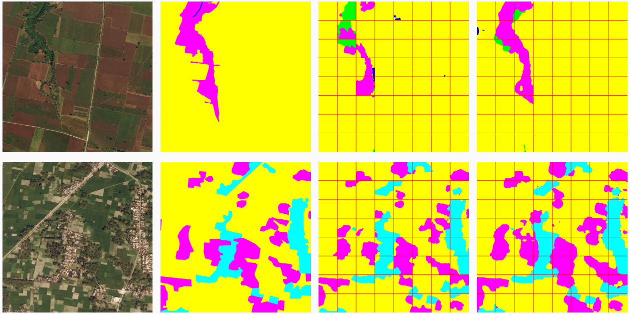


Avoiding overfitting

The data set includes **only** 803 **images**. To **avoid overfitting**, we

- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (**He, Zhang, Ren, and Sun (2016)**)

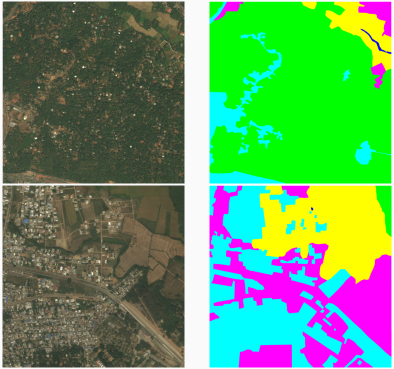
image true mask without comm. with comm.



DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

class	pixel count	proportion
urban	642.4M	9.35 %
agriculture	3898.0M	56.76 %
rangeland	701.1M	10.21 %
forest	944.4M	13.75 %
water	256.9M	3.74 %
barren	421.8M	6.14 %
unknown	3.0M	0.04 %

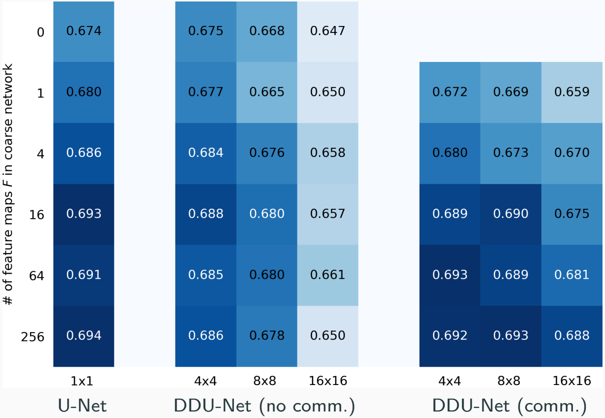
Input **Target**



Avoiding overfitting

The data set includes **only** 803 **images**. To **avoid overfitting**, we

- apply **batch normalization**, use **random dropout** layers and **data augmentation**, and
- **initialize the encoder** using the **ResNet-18** (**He, Zhang, Ren, and Sun (2016)**)



Schwarz Domain Decomposition Preconditioners

- **Numerical scalability** and **robust convergence** for
  - **heterogeneous** problems
  - **multiphysics** problems
  - **highly nonlinear** problems

→ **Algebraic** and **parallel** implementation in FROSCH

Domain Decomposition for Neural Networks

- **Schwarz domain decomposition architectures** **improve the scalability of PINNs** to **large domains** / **high frequencies**, **keeping the complexity of the local networks low**.
- **Novel DDU-Net** approach **decouples the training on the sub-images**, allowing us to **distribute the memory load** among **multiple GPUs**. It **limits communication** to **deepest level** of the U-Net architecture using a **communication network**.

Thank you for your attention!