The importance of coarse levels for domain decomposition methods

Alexander Heinlein[1]

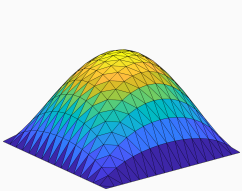
International Conference on Preconditioning Techniques for Scientific and Industrial Applications (Preconditioning 2024), Georgia Institute of Technology, Atlanta, USA, June 10–12, 2024

[1]Delft University of Technology

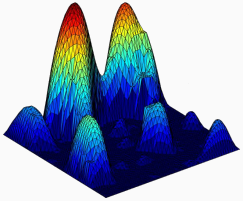
Outline

One- and Two-Level Schwarz Preconditioners

Solving A Model Problem



$\alpha(x) = 1$



heterogeneous $\alpha(x)$

Consider a **diffusion model problem**:

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

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

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

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 linear systems of equations, however, the **convergence rate generally depends on the condition number** $\kappa(\boldsymbol{K})$. It deteriorates, e.g., for

- fine meshes, that is, small element sizes h
- large contrasts $\frac{\max_x \alpha(x)}{\min_x \alpha(x)}$

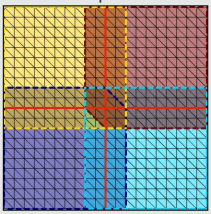
$\Rightarrow$ We introduce a preconditioner $\boldsymbol{M}^{-1} \approx \boldsymbol{K}^{-1}$ to improve the condition number:

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

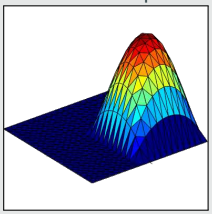
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



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

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

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

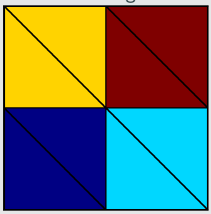
Condition number estimate:

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

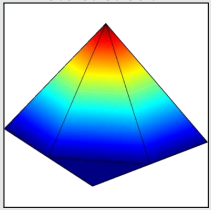
with subdomain size H and overlap width δ.

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.

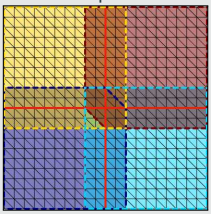
Condition number estimate:

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

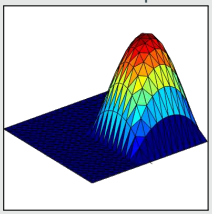
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$

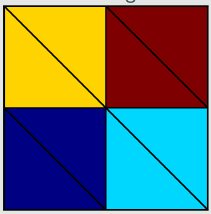


Solution of local problem

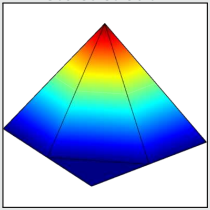


Lagrangian coarse space

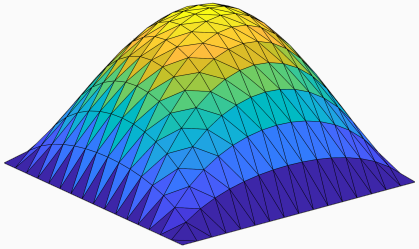
Coarse triangulation

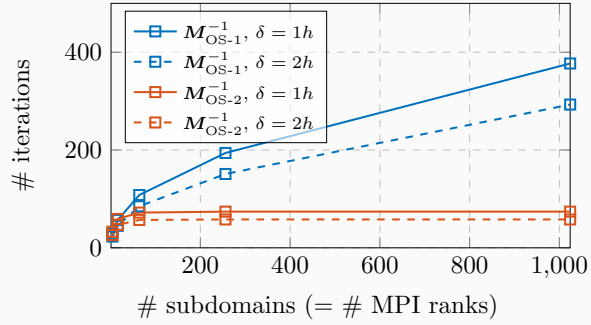


Coarse solution



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

TU Delft

Sandia National Laboratories

Universität zu Köln

TU Bergakademie Freiberg

TUBAF
Die Ressourcenuniversität. Seit 1765.

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)

Partition of Unity

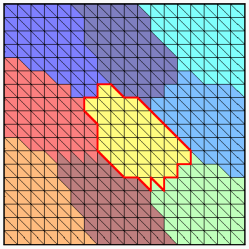
The **energy-minimizing extension** $v_i = H_{\partial\Omega_i \to \Omega_i}(v_{i,\partial\Omega_i})$ solves

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

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

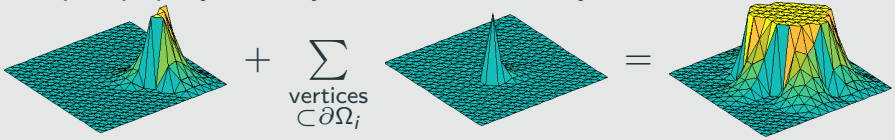
$$\sum_i \varphi_i = \mathbb{1}_{\partial\Omega_i} \Rightarrow \sum_i E_{\partial\Omega_i \to \Omega_i}(\varphi_i) = \mathbb{1}_{\Omega_i}$$



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**:

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



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** $\boldsymbol{K}_i$.

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

Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

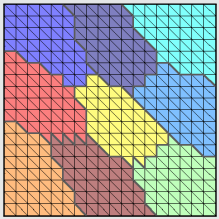
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

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

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

Nonoverlapping DD



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

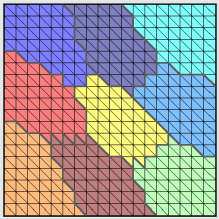
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

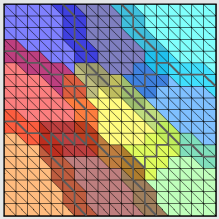
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

Nonoverlapping DD



Overlap $\delta = 1h$



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

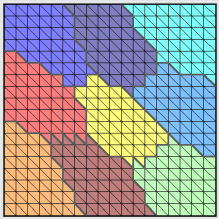
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

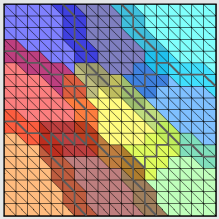
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

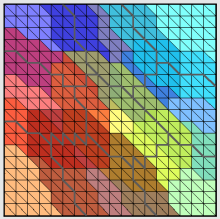
Nonoverlapping DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

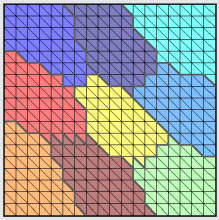
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

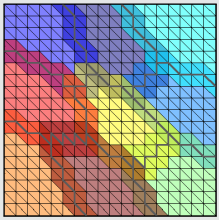
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

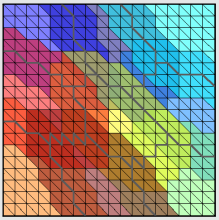
Nonoverlapping DD



Overlap $\delta = 1h$

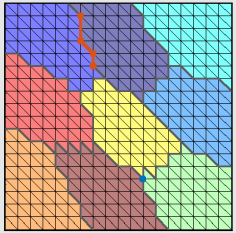


Overlap $\delta = 2h$



Coarse space

1. Interface components



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

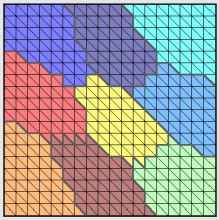
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

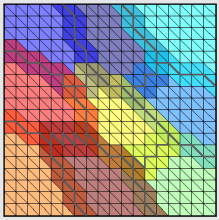
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

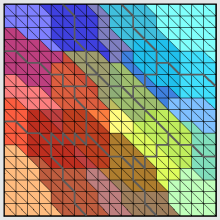
Nonoverlapping DD



Overlap $\delta = 1h$

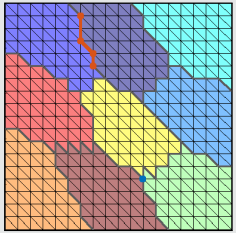


Overlap $\delta = 2h$

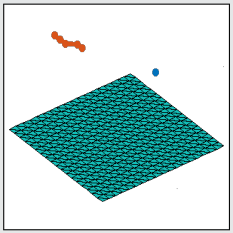


Coarse space

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

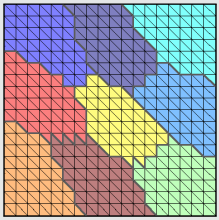
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

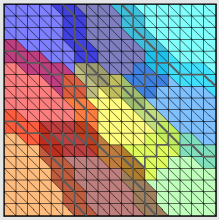
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

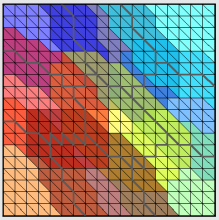
Nonoverlapping DD



Overlap $\delta = 1h$

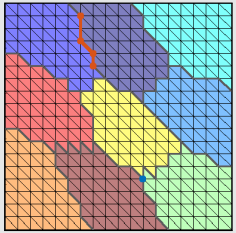


Overlap $\delta = 2h$

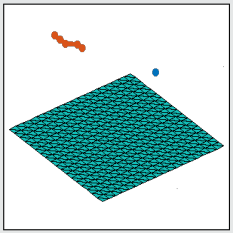


Coarse space

1. Interface components

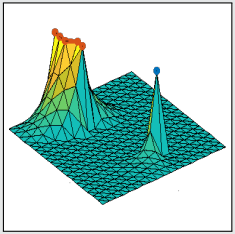


2. Interface basis (partition of unity $\times$ null space)



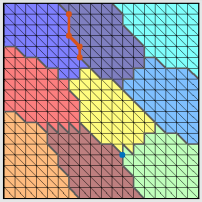
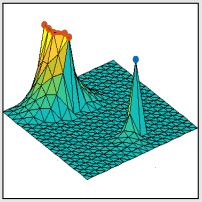
For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

3. Extension



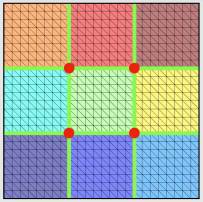
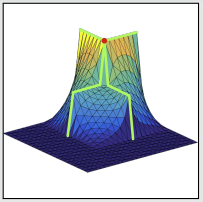
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)

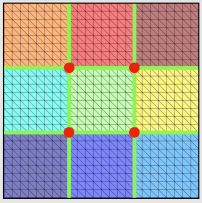
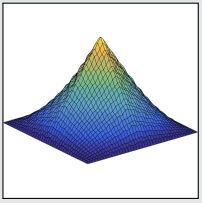
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)

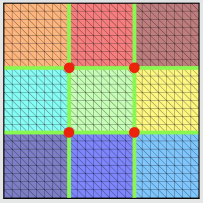
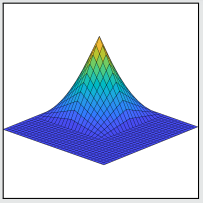
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

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

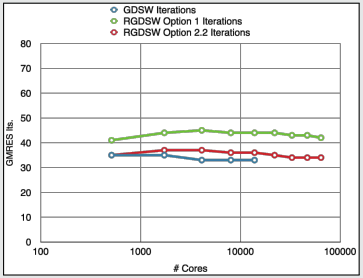



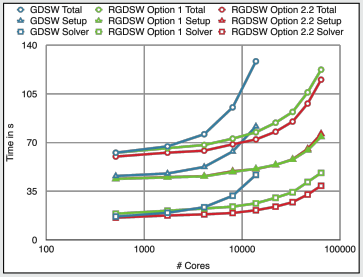
Piecewise linear interface partition of unity functions and a **structured domain decomposition**.

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

GDSW vs RGDSW (reduced dimension)

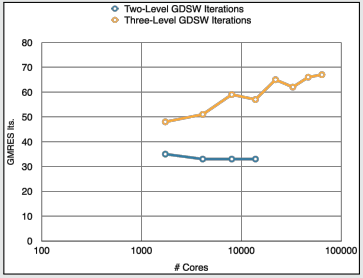
Heinlein, Klawonn, Rheinbach, Widlund (2019).

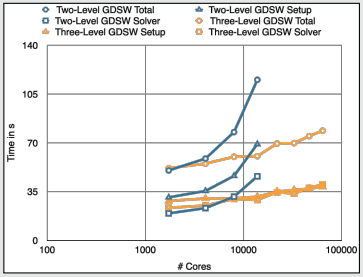




Two-level vs three-level GDSW

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



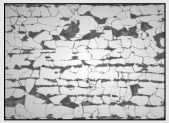


Coarse Spaces for Some Challenging Problems

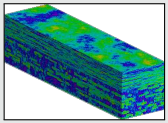
Spectral Extension-Based Coarse Spaces for Schwarz Preconditioners

Highly heterogeneous problems . . .

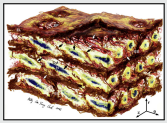
. . . appear in most areas of modern science and engineering:



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)**).

Related works (non-exhaustive)

- **FETI & Neumann–Neumann:** **Bjørstad and Krzyzanowski (2002); Bjørstad, Koster, and Krzyzanowski (2001); Rixen and Spillane (2013); Spillane (2015, 2016)** . . .
- **BDDC & FETI-DP:** **Mandel and Sousedík (2007); Sousedík (2010); Sístek, Mandel, and Sousedík (2012); Dohrmann and Pechstein (2013, 2016); Klawonn, Radtke, and Rheinbach (2014, 2015, 2016); Klawonn, Kühn, and Rheinbach (2015, 2016, 2017); Kim and Chung (2015); Kim, Chung, and Wang (2017); Beirão da Veiga et al. (2017); Calvo and Widlund (2016); Oh et al. (2017)** . . .
- **Overlapping Schwarz:** **Galvis and Efendiev (2010, 2011); Nataf, Xiang, Dolean, and Spillane (2011); Spillane, Dolean, Hauret, Nataf, Pechstein, and Scheichl (2011); Gander, Loneland, and Rahman (preprint 2015); Eikeland, Marcinkowski, and Rahman (TR 2016); Marcinkowski and Rahman (2018)** . . .
- **Spectral AMGe (ρAMGe):** **Chartier, Falgout, Henson, Jones, Manteuffel, McCormick, Ruge, and Vassilevski (2003)** . . .

Spectral Extension-Based Coarse Spaces for Schwarz Preconditioners

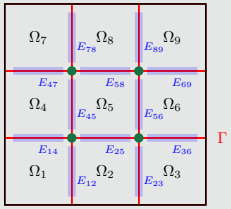
Local eigenvalue problems

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

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

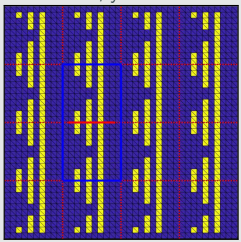
with **Schur complements** S_{EE}, S_{FF} with **Neumann boundary conditions** and **submatrices** K_{EE}, K_{FF} of K. We select eigenfunctions corresponding to **eigenvalues below tolerances** $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$.

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

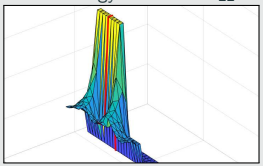


Extensions in the generalized eigenvalue problem

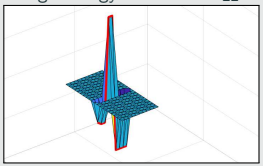
Blue $\alpha = 1$; yellow $\alpha = 10^6$



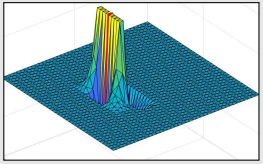
Low energy extension $\boldsymbol{S}_{EE}$



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



Coarse basis function

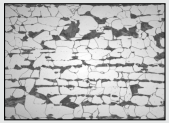


The extensions on the two sides of the generalized eigenvalue problem correspond to **low and high energy extensions of the trace** $\rightarrow$ detects coefficient jumps.

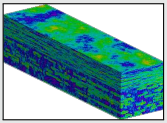
Spectral Extension-Based Coarse Spaces for Schwarz Preconditioners

Highly heterogeneous problems . . .

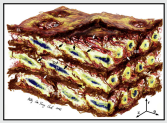
. . . appear in most areas of modern science and engineering:



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)**.

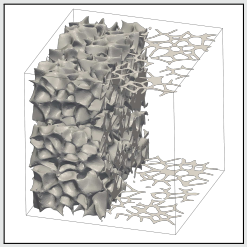
Spectral coarse spaces

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)**.

Algebraic Spectral Extension-Based Coarse Spaces

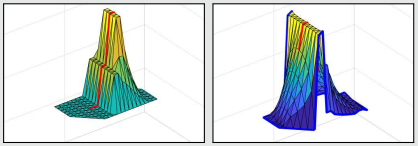
Two algebraic eigenvalue problems

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

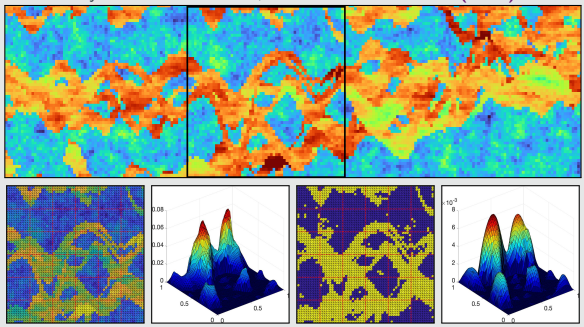
$$\kappa\left(\boldsymbol{M}^{-1}_{\text{DIR\&TR}}\boldsymbol{K}\right) \leq C \max\left\{1/TOL_{\text{DIR}}, TOL_{\text{TR}}/\alpha_{\min}\right\},$$

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

Heinlein & Smetana (subm. 2023; 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	$2.0{\cdot}10^5$	57
V_{AGDSW}	$1.0{\cdot}10^{-2}$		93	19.3	38
$V_{\text{DIR\&TR}-a}$	$1.0{\cdot}10^{-3}$	$1.0{\cdot}10^5$	90	19.4	39
$V_{\text{DIR\&TR}-l^2}$	$1.0{\cdot}10^{-3}$	$1.0{\cdot}10^5$	147	9.6	31
Original coefficient (without thresholding)					
V_{GDSW}	-	-	85	20.6	42

Linear & Nonlinear Preconditioning

Let us consider the nonlinear problem arising from the discretization of a partial differential equation

$$\boldsymbol{F}(\boldsymbol{u}) = 0.$$

We solve the problem using a **Newton-Krylov approach**, i.e., we solve a sequence of linearized problems using a Krylov subspace method:

$$D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right)\Delta\boldsymbol{u}^{(k+1)} = \boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right).$$

Linear preconditioning

In linear preconditioning, we **improve the convergence speed of the linear solver** by constructing a **linear operator** $\boldsymbol{M}^{-1}$ and solve linear systems

$$\boldsymbol{M}^{-1}D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right)\Delta\boldsymbol{u}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{F}(\boldsymbol{u}^{(k)}).$$

Goal:

- $\kappa\left(\boldsymbol{M}^{-1}D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right)\right) \approx 1.$

$\Rightarrow\ \boldsymbol{M}^{-1}D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right) \approx \boldsymbol{I}.$

Nonlinear preconditioning

In nonlinear preconditioning, we **improve the convergence speed of the nonlinear solver** by constructing a **nonlinear operator** G and solve the nonlinear system

$$(\boldsymbol{G}\circ\boldsymbol{F})(\boldsymbol{u}) = 0.$$

Goals:

- $\boldsymbol{G}\circ\boldsymbol{F}$ almost linear.
- Additionally: $\kappa\left(D\left(\boldsymbol{G}\circ\boldsymbol{F}\right)(\boldsymbol{u})\right) \approx 1.$

Linear & Nonlinear Preconditioning

Let us consider the nonlinear problem arising from the discretization of a partial differential equation

$$\boldsymbol{F}(\boldsymbol{u}) = 0.$$

We solve the problem using a **Newton-Krylov approach**, i.e., we solve a sequence of linearized problems using a Krylov subspace method:

$$D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right)\Delta\boldsymbol{u}^{(k+1)} = \boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right).$$

Linear preconditioning

In linear preconditioning, we **improve the convergence speed of the linear solver** by constructing a **linear operator** $\boldsymbol{M}^{-1}$ and solve linear systems

$$\boldsymbol{M}^{-1}D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right)\Delta\boldsymbol{u}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{F}(\boldsymbol{u}^{(k)}).$$

Goal:

- $\kappa\left(\boldsymbol{M}^{-1}D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right)\right) \approx 1.$

$\Rightarrow\ \boldsymbol{M}^{-1}D\boldsymbol{F}\left(\boldsymbol{u}^{(k)}\right) \approx \boldsymbol{I}.$

Nonlinear preconditioning

In nonlinear preconditioning, we **improve the convergence speed of the nonlinear solver** by constructing a **nonlinear operator** G and solve the nonlinear system

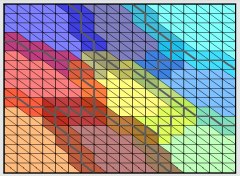
$$(\boldsymbol{G}\circ\boldsymbol{F})(\boldsymbol{u}) = 0.$$

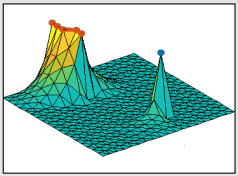
Goals:

- $\boldsymbol{G}\circ\boldsymbol{F}$ almost linear.
- Additionally: $\kappa\left(D\left(\boldsymbol{G}\circ\boldsymbol{F}\right)(\boldsymbol{u})\right) \approx 1.$

Nonlinear Schwarz Methods

Two-level ASPEN & ASPIN methods





In **additive Schwarz preconditioned (in)exact Newton (ASPEN/ASPIN)** **(Cai and Keyes (2002))**, the nonlinear problem is modified

$$F(u) = 0 \Leftrightarrow \sum_{i=0}^{N} R_i^T T_i(u) = 0$$

with corrections $T_i(u)$ given by **nonlinear problems** on the **overlapping subdomains** / **coarse space**

$$R_i F(u - R_i^T T_i(u)) = 0.$$

Coarse space via **Galerkin projection**: **Heinlein, Lanser (2020)**

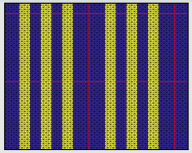
More: **MS6 & MS11** *Nonlinear Preconditioning Techniques and Applications I & II*

Problem configuration

p-**Laplacian model problem** ($p = 4$)

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

with $\alpha \Delta_p u := \operatorname{div}(\alpha |\nabla u|^{p-2} \nabla u)$ on a **domain decomposition into** 6×6 **subdomains** with $H/h = 32$ and overlap $1h$.



yellow: $\alpha = 10^3$
blue: $\alpha = 1$

no globalization						
size cp	method	coarse space	outer it.	local it. (avg.)	coarse it.	GMRES it. (sum)
145	ASPEN	AGDSW	5	27.0	35	77
25	ASPEN	MsFEM	>20	-	-	-
145	NK-AS	AGDSW	>20	-	-	-
Inexact Newton backtracking (INB) **Eisenstat and Walker (1994)**						
145	ASPEN	AGDSW	5	24.8	21	77
25	ASPEN	MsFEM	18	83.9	75	852
145	NK-AS	AGDSW	13	-	-	207

Cf. **Heinlein, Klawonn, Lanser (2022)**

Monolithic (R)GDSW Preconditioners for CFD Simulations

Consider the discrete saddle point problem

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

Monolithic GDSW preconditioner

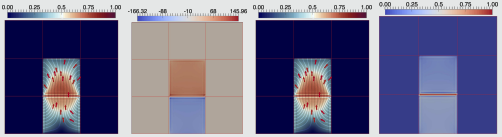
We construct a **monolithic GDSW preconditioner**

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

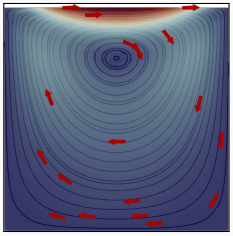
with block matrices $\mathcal{A}_0 = \phi^\top \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^\top$, local pressure projections $\overline{\mathcal{P}}_i$, and

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

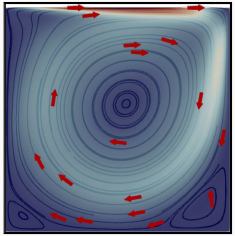
Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



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.

Monolithic (R)GDSW Preconditioners for CFD Simulations

Consider the discrete saddle point problem

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

Monolithic GDSW preconditioner

We construct a **monolithic GDSW preconditioner**

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

with block matrices $\mathcal{A}_0 = \phi^\top \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^\top$.

SIMPLE block preconditioner

We employ the **SIMPLE (Semi-Implicit Method for Pressure Linked Equations)** block preconditioner

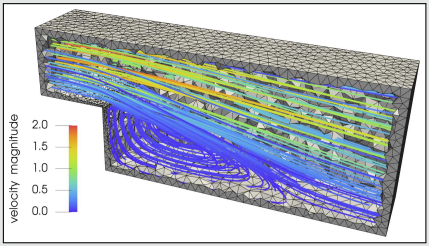
$$M^{-1}_{\text{SIMPLE}} = \begin{bmatrix} I & -D^{-1}B \\ 0 & \alpha I \end{bmatrix} \begin{bmatrix} K^{-1} & 0 \\ -\hat{S}^{-1} B K^{-1} & \hat{S}^{-1} \end{bmatrix};$$

see **Patankar and Spalding (1972)**. Here,

- $\hat{S} = -BD^{-1}B^\top$, with $D = \operatorname{diag} K$
- α is an under-relaxation parameter

We **approximate the inverses** using (R)GDSW preconditioners.

Monolithic vs. SIMPLE preconditioner

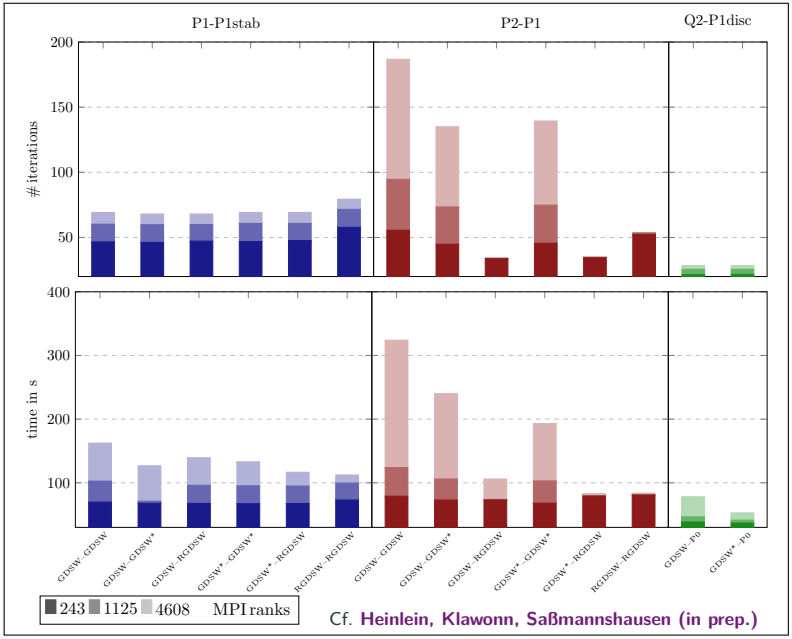


Steady-state Navier–Stokes equations

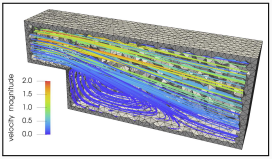
prec.	# MPI ranks	243	1 125	15 562
Monolithic RGDSW (FROSch)	setup	39.6 s	57.9 s	95.5 s
	solve	57.6 s	69.2 s	74.9 s
	total	**97.2 s**	**127.7 s**	**170.4 s**
SIMPLE RGDSW (Teko & FROSch)	setup	39.2 s	38.2 s	68.6 s
	solve	86.2 s	106.6 s	127.4 s
	total	125.4 s	144.8 s	196.0 s

Computations on Piz Daint (CSCS). Implementation in the finite element software FEDDLib.

Balancing the Velocity and Pressure Coarse Spaces

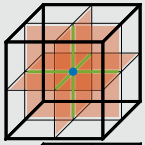


Cf. **Heinlein, Klawonn, Saßmannshausen (in prep.)**

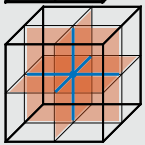


Varying the POU

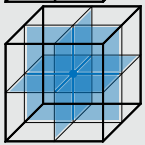
GDSW:



GDSW*:

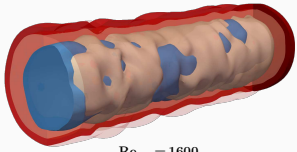


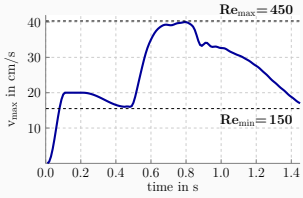
RGDSW:

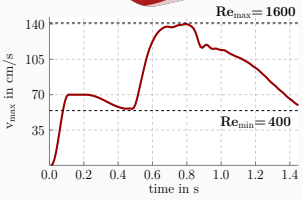


Spectral Extension-Based Coarse Spaces for Schwarz Preconditioners

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





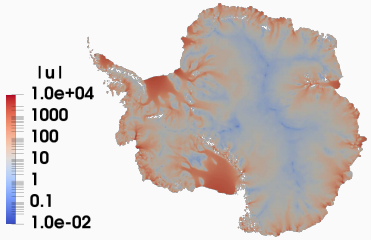


prec.	# MPI ranks	16	64	256
Monolithic RGDSW (FROSch)	avg. #its.	33	31	30
	setup	4 825 s	1 422 s	701 s
	solve	3 198 s	1 004 s	463 s
	total	8 023 s	2 426 s	**1 164 s**
SIMPLE RGDSW (Teko & FROSch)	avg. #its.	82	82	87
	setup	3 046 s	824 s	428 s
	solve	4 679 s	1 533 s	801 s
	total	**7 725 s**	**2 357 s**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

FROSch Preconditioners for Land Ice Simulations

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)

Multilevel 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 oret DeepRitz (D3M, DeepDDM, etc)**: **Li, Tang, Wu, and Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2022, arXiv 2023); Kim, Yang (2022, arXiv 2023)**
- **FBPINNs**: **Moseley, Markham, and Nissen-Meyer (2023); Dolean, Heinlein, Mishra, Moseley (2024, acc. 2024 / arXiv:2306.05486); Heinlein, Howard, Beecroft, Stinis (subm. 2024 / arXiv:2401.07888)**
- **DDMs for CNNs**: **Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (acc. 2024); Verburg, Heinlein, Cyr (in prep.)**

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 the **end of 2023**:

A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods — A survey

arXiv:2312.14050. 2023

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

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

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

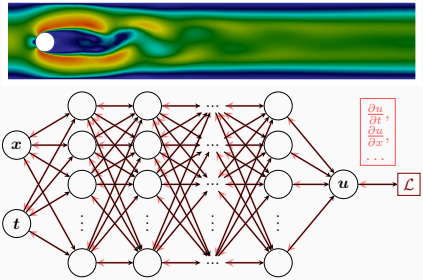
See also **Dissanayake and Phan-Thien (1994); Lagaris et al. (1998)**.

Advantages

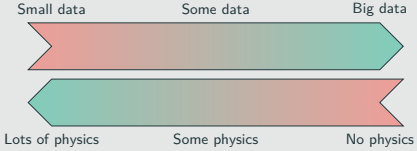
- **"Meshfree"**
- **Small data**
- **Generalization properties**
- **High-dimensional problems**
- **Inverse** and **parameterized problems**

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 $\mathcal{L}_{\text{data}}$

Motivation – Some Observations on the Performance of PINNs

Solve

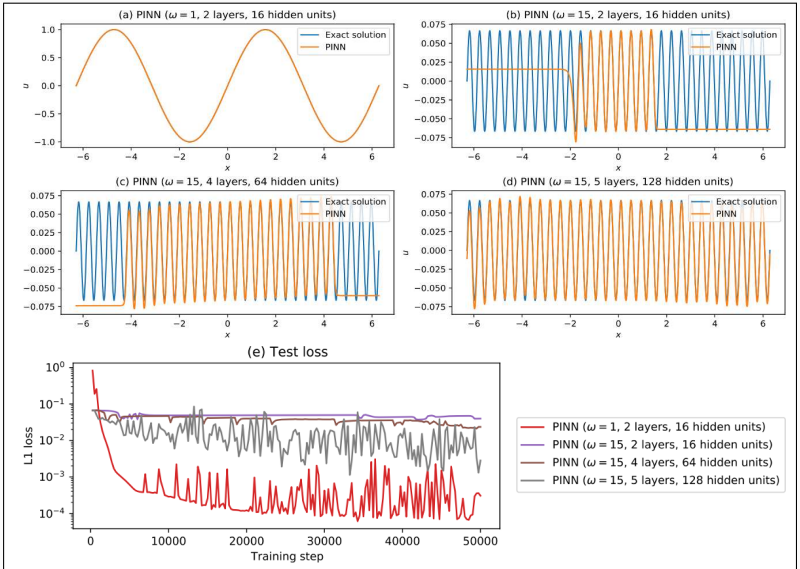
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

for different values of ω using **PINNs with varying network capacities**.

Scaling issues

- Large computational domains
- Small frequencies

Cf. **Moseley, Markham, and Nissen-Meyer (2023)**



(a) 321 free parameters (d) 66 433 free parameters

Motivation – Some Observations on the Performance of PINNs

Solve

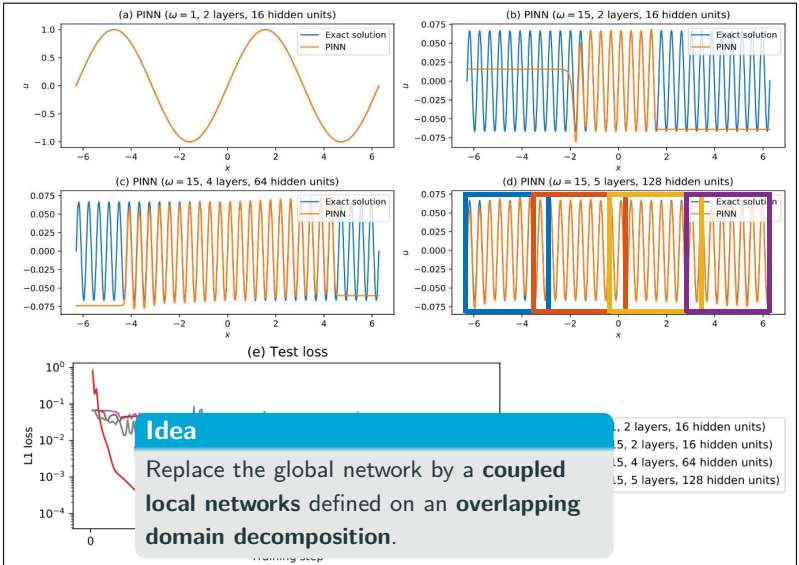
$$\begin{aligned} u' &= \cos(\omega x), \\ u(0) &= 0, \end{aligned}$$

for different values of ω using **PINNs with varying network capacities**.

Scaling issues

- Large computational domains
- Small frequencies

Cf. **Moseley, Markham, and Nissen-Meyer (2023)**



Idea

Replace the global network by a **coupled local networks** defined on an **overlapping domain decomposition**.

(a) 321 free parameters (d) 66 433 free parameters

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**

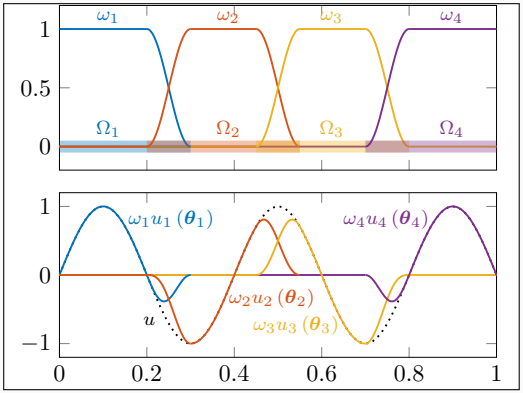
$$u(\theta_1, \dots, \theta_J) = \mathcal{C} \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

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

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

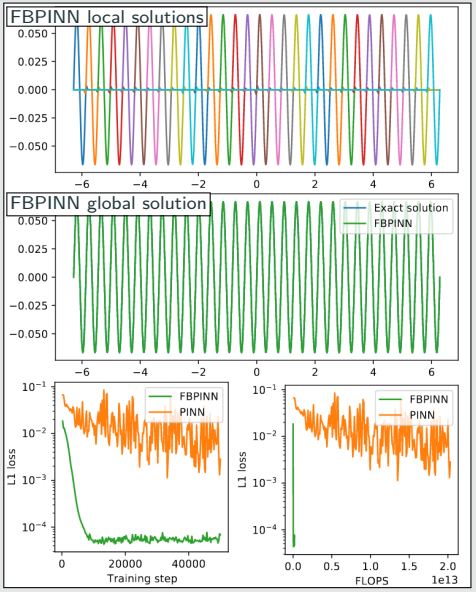
Loss function

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

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

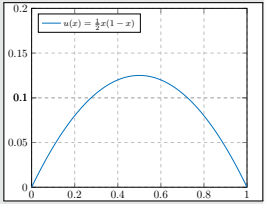
Consider the **simple boundary value problem**

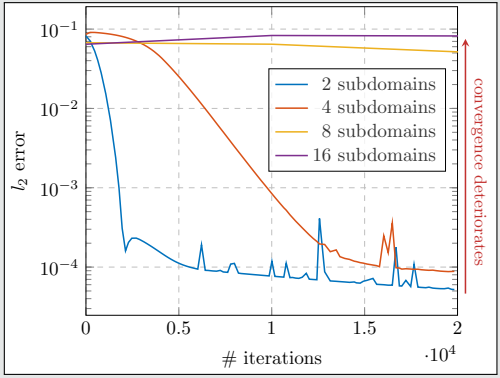
$$-u'' = 1 \text{ in } [0,1],$$

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

which has the **solution**

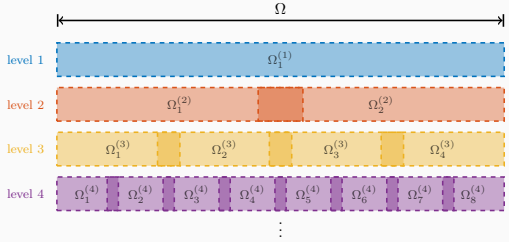
$$u(x) = 1/2 x(1-x).$$





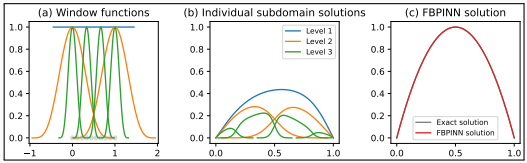
Multi-Level FBPINN Algorithm

Extension of FBPINNs to L levels; Cf. **Dolean, Heinlein, Mishra, Moseley (accepted 2024 / arXiv:2306.05486)**.



L-level network architecture

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



Multi-Frequency Problem

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

$$-\Delta u = 2\sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x)\sin(\omega_i \pi y) \quad \text{in } \Omega,$$

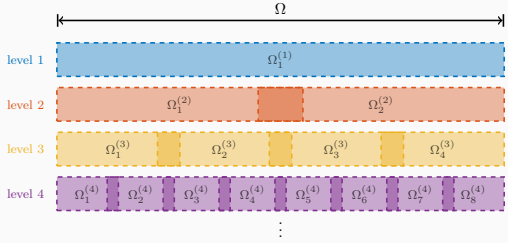
$$u = 0 \quad \text{on } \partial\Omega,$$

with $\omega_i = 2^i$.

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

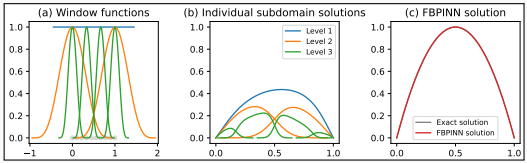
Multi-Level FBPINN Algorithm

Extension of FBPINNs to L levels; Cf. **Dolean, Heinlein, Mishra, Moseley (accepted 2024 / arXiv:2306.05486)**.



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

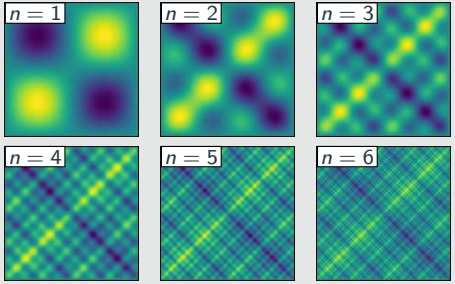
Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

$$-\Delta u = 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) \quad \text{in } \Omega,$$

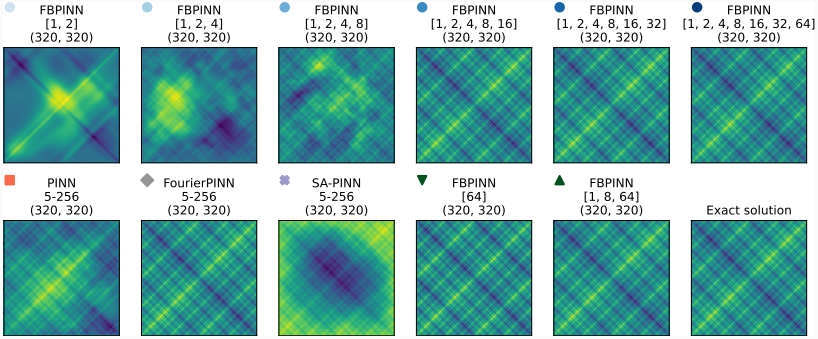
$$u = 0 \quad \text{on } \partial\Omega,$$

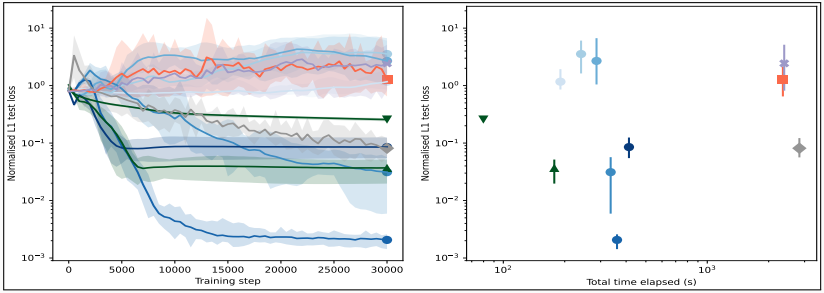
with $\omega_i = 2^i$.

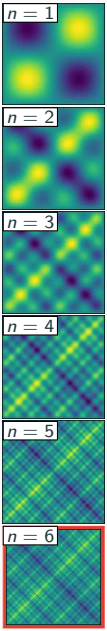
For increasing values of n, we obtain the **analytical solutions**:



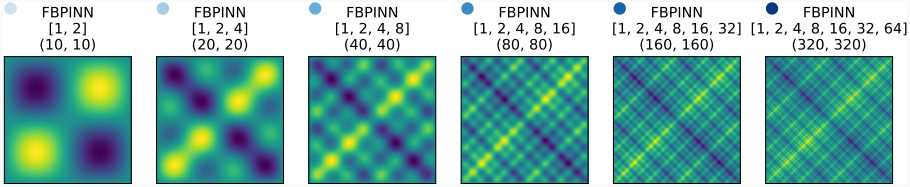
Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling

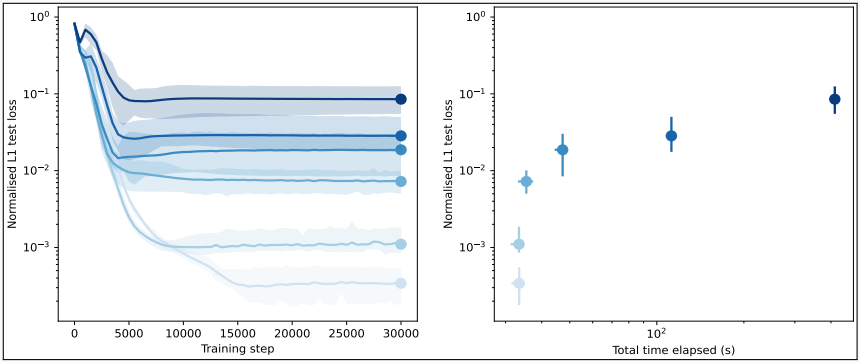


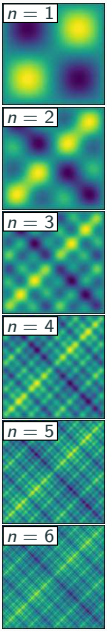




Multi-Level FBPINNs for a Multi-Frequency Problem – Weak Scaling



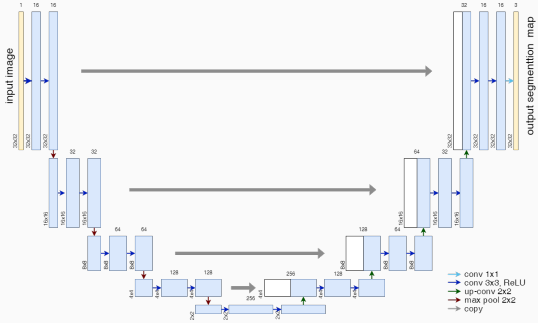




- Ongoing: **analysis and improvement of the convergence**

Cf. **Dolean, Heinlein, Mishra, Moseley (accepted 2024 / arXiv:2306.05486)**. Implementation using JAX

Memory Requirements for CNN Training

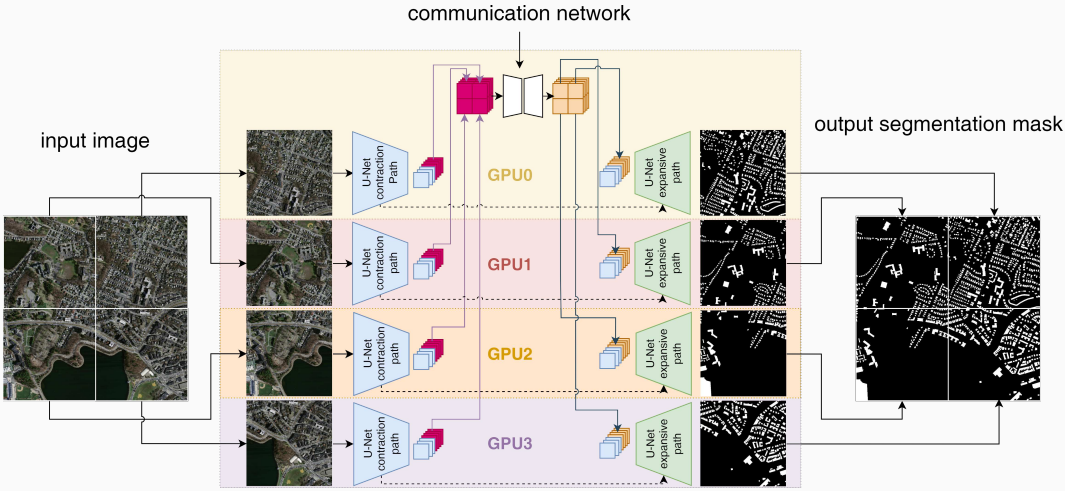


- As an example for a **convolutional neural network (CNN)**, we employ the **U-Net architecture** introduced in **Ronneberger, Fischer, and Brox (2015)**.
- The U-Net yields **state-of-the-art accuracy in semantic image segmentation** and other **image-to-image tasks**.

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

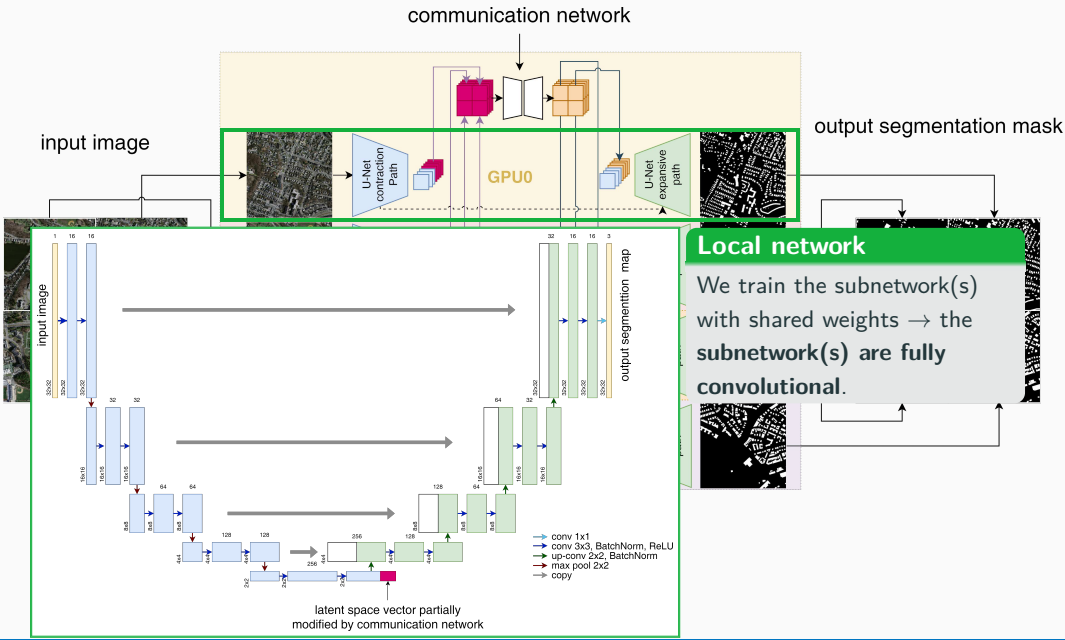
name	size	# channels		mem. feature maps		mem. weights	
		input	output	# of values	MB	# of values	MB
input block	1 024	3	64	268 M	**1 024.0**	38 848	**0.148**
encoder block 1	512	64	128	167 M	**704.0**	221 696	**0.846**
encoder block 2	256	128	256	84 M	**352.0**	885 760	**3.379**
encoder block 3	128	256	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 (in preparation).**

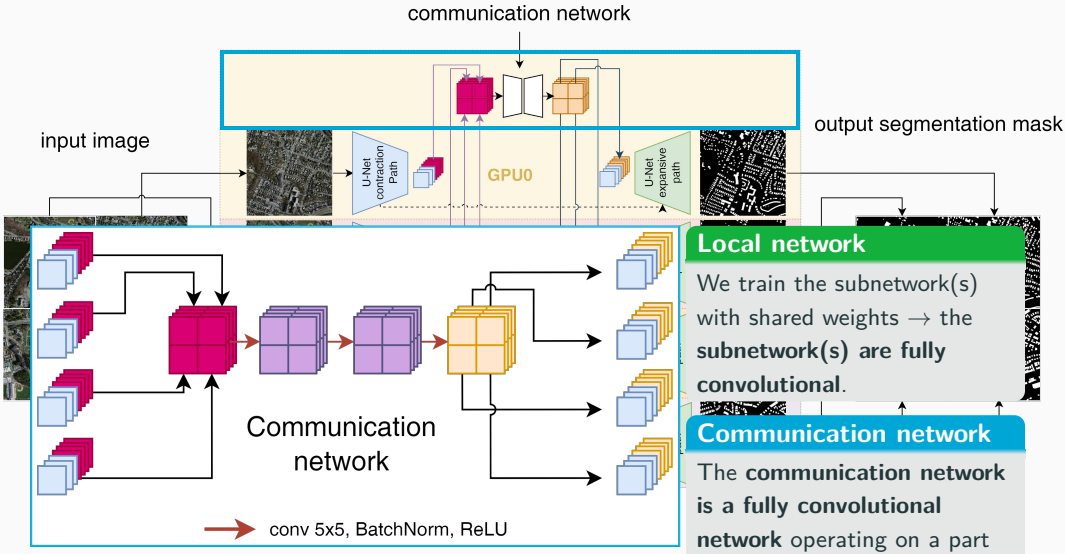
Decomposing the U-Net



Local network

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

Decomposing the U-Net



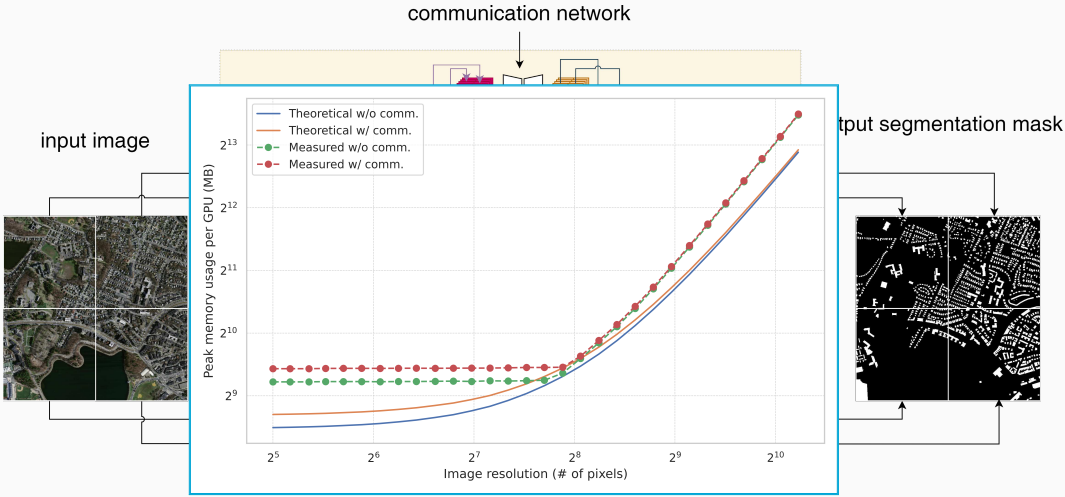
Local network

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.

Decomposing the U-Net

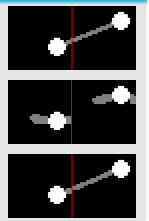


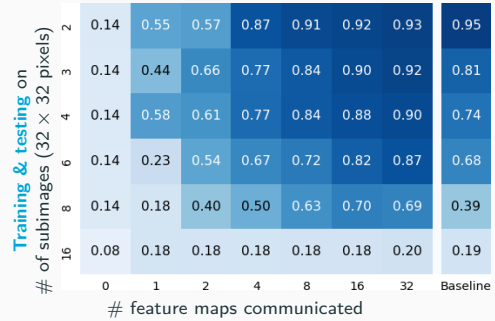
- Distribution of feature maps results in **significant reduction of memory usage on a single GPU**
- **Moderate additional memory usage** due to the **communication network**

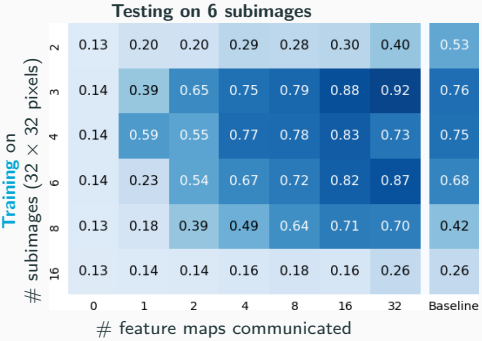
Results – Synthetic Data Set

Task: Connect two dots via a line segment

Result: Communication



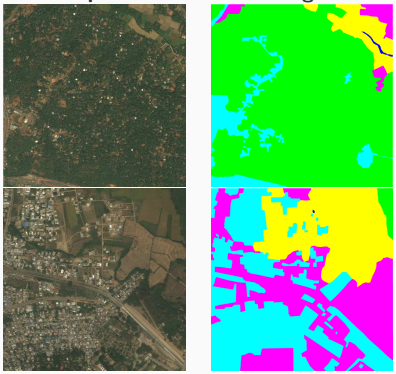




DeepGlobe 2018 Satellite Image Data Set (Demir et al. (2018))

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

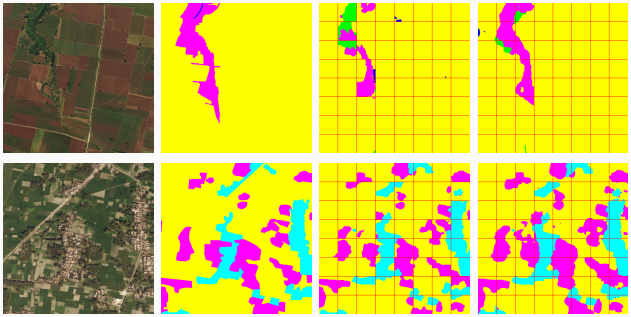


Avoiding overfitting

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

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

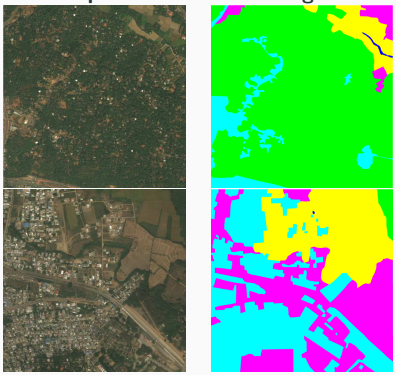
image true mask without comm. with comm.



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 %

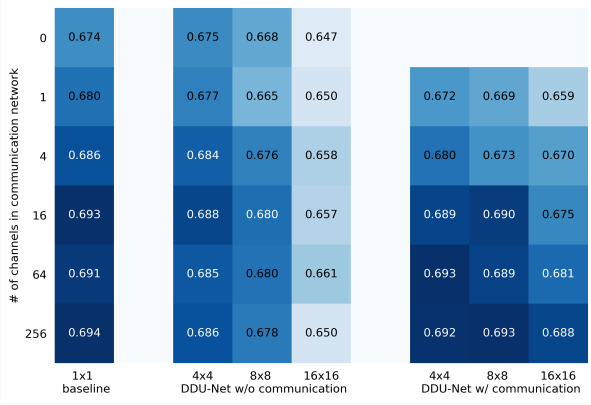
Input Target



Avoiding overfitting

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

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



Coarse levels for domain decomposition methods

- **Numerical scalability** and **robust convergence** for
  - **heterogeneous** problems
  - **multiphysics** problems
  - **highly nonlinear** problems
- **Representation** of **coarse scale features**
- **Fast transport** of **global information**
- **Challenges**:
  - Implementation is often **intrusive**
  - **Bottleneck** for **parallel scalability**

→ **Algebraic** and **parallel** implementation in FROSch

Acknowledgements

- **Financial support:** DFG (KL2094/3-1, RH122/4-1), DFG SPP 2311 project number 465228106, DOE SciDAC-5 FASTMath Institute (Contract no. DE-AC02-05CH11231)
- **Computing resources:** Summit (OLCF), Cori (NERSC), magnitUDE (UDE), Piz Daint (CSCS), Fritz (FAU), DelftBlue (TU Delft)

Thank you for your attention!