

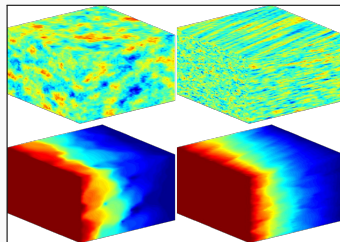
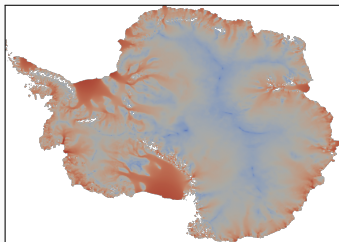
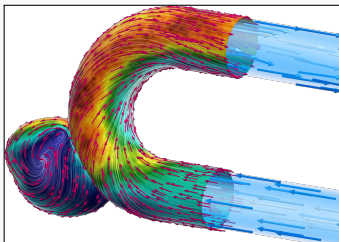


Domain Decomposition-Based Preconditioners and Neural Networks

Alexander Heinlein¹

NHR PerfLab Seminar, Friedrich-Alexander University of Erlangen-Nuremberg, Erlangen, February 3, 2026

¹Delft University of Technology



Numerical methods

Based on physical models

- + Robust and generalizable
- Require availability of mathematical models

Machine learning models

Driven by data

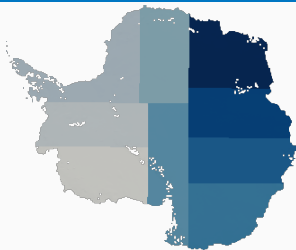
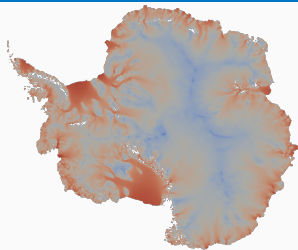
- + Do not require mathematical models
- Sensitive to data, limited extrapolation capabilities

Scientific machine learning

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

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

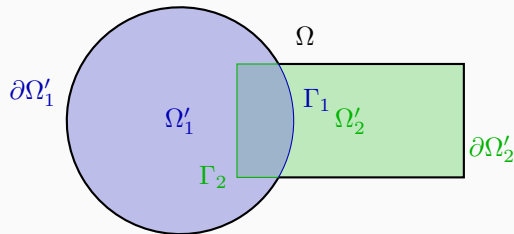
Domain Decomposition Methods



Graphics based on results from [Heinlein, Perego, Rajamanickam \(2022\)](#)

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

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



1 FROSCH – Algebraic and Parallel Schwarz Preconditioners in TRILINOS

Based on joint work with

Axel Klawonn, Jascha Knepper, and Lea Saßmannshausen	(University of Cologne)
Mauro Perego, Siva Rajamanickam and Ichi Yamazaki	(Sandia National Laboratories)
Oliver Rheinbach and Friederike Röver	(TU Bergakademie Freiberg)
Olof Widlund	(New York University)

2 Domain Decomposition for Physics-Informed Neural Networks

Based on joint work with

Victorita Dolean	(Eindhoven University of Technology)
Taniya Kapoor	(Wageningen University & Research)
Siddhartha Mishra	(ETH Zürich)
Ben Moseley	(Imperial College London)
Yong Shang and Fei Wang	(Xi'an Jiaotong University)

FROSch — Algebraic and Parallel Schwarz Preconditioners in Trilinos

Solvers for Partial Differential Equations

Consider a **diffusion model problem**:

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

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

$$Ku = f.$$

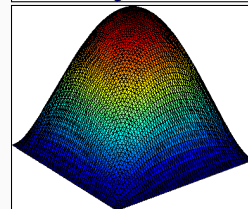
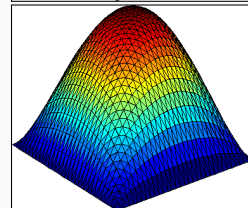
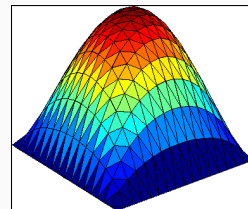
The accuracy of the finite element solution depends on the refinement level of the mesh h : **higher refinement** $\Rightarrow$ **better accuracy**.

Direct solvers

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

Iterative solvers

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

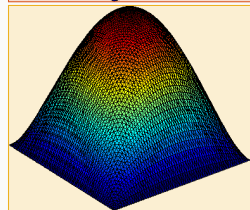
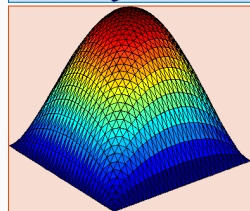
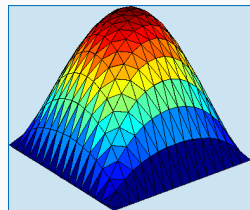
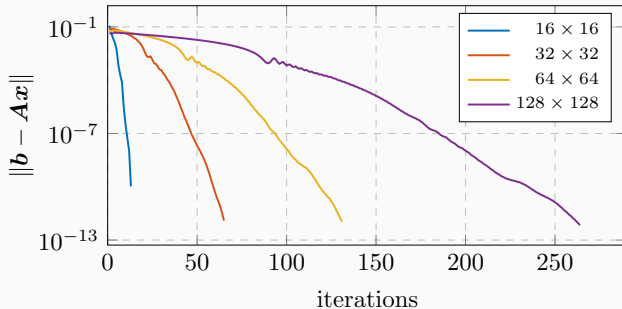


Solvers for Partial Differential Equations

Consider a **diffusion model problem**:

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

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

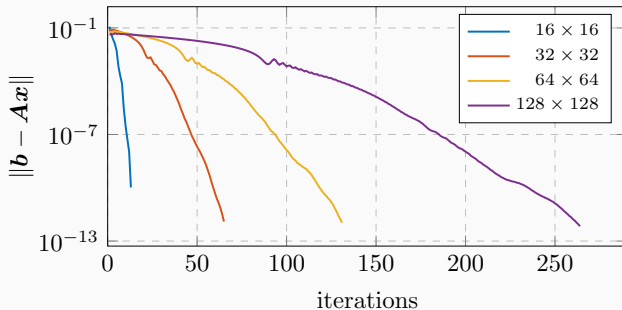


Solvers for Partial Differential Equations

Consider a **diffusion model problem**:

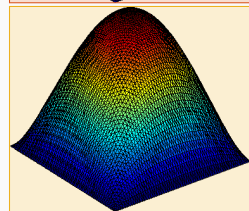
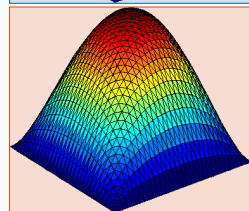
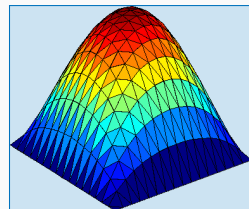
$$\begin{aligned} -\Delta u(x) &= f \quad \text{in } \Omega = [0, 1]^2, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

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



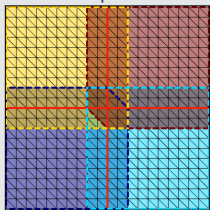
$\Rightarrow$ Introduce a preconditioner $M^{-1} \approx K^{-1}$ to **improve convergence**:

$$M^{-1}Ku = M^{-1}f$$

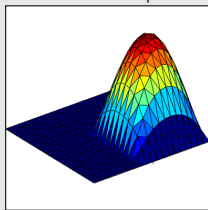


One-level Schwarz preconditioner

Overlap $\delta = 2h$



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

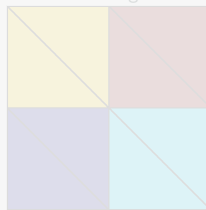
Condition number estimate:

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

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.

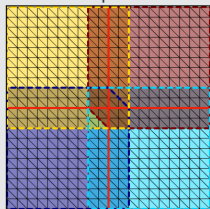
Condition number estimate:

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

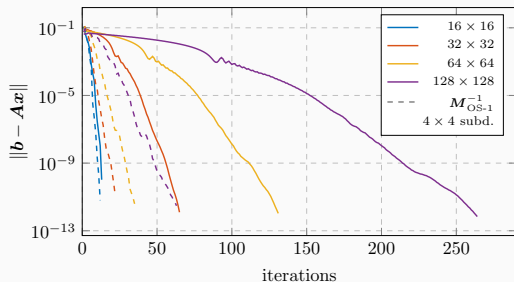
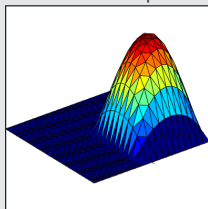
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 2h$

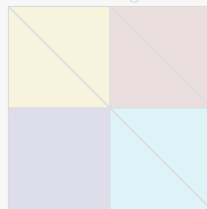


Solution of local problem



Lagrangian coarse space

Coarse triangulation



Coarse solution



The two-level overlapping Schwarz operator reads

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

where Φ contains the coarse basis functions and $K_0 := \Phi^T K \Phi$; cf., e.g., [Toselli, Widlund \(2005\)](#).

The construction of a Lagrangian coarse basis requires a coarse triangulation.

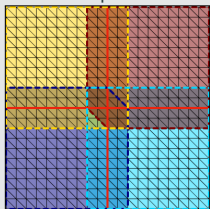
Condition number estimate:

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

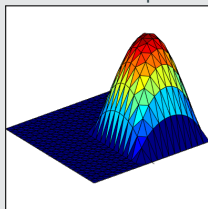
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 2h$



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

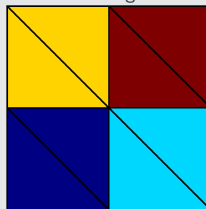
Condition number estimate:

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

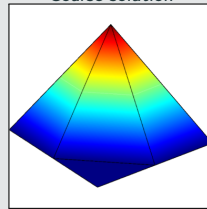
with subdomain size H and overlap width δ .

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.

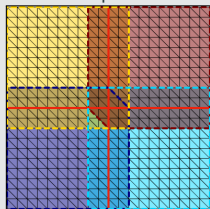
Condition number estimate:

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

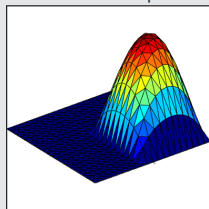
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 2h$

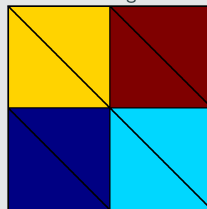


Solution of local problem

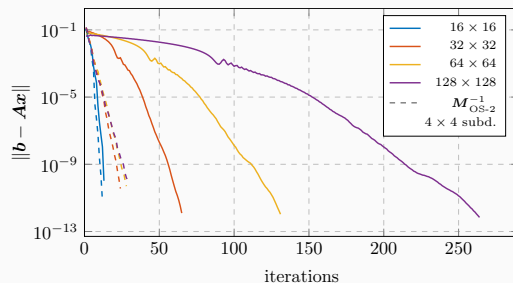
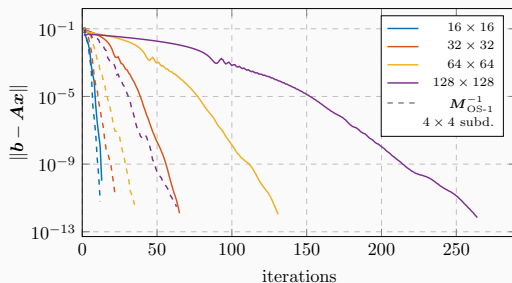
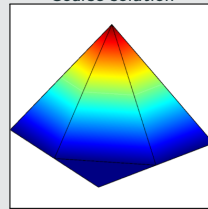


Lagrangian coarse space

Coarse triangulation



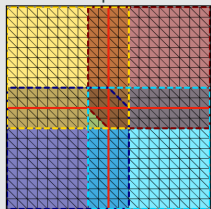
Coarse solution



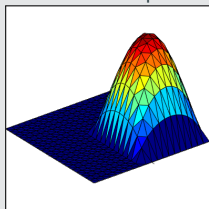
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 2h$

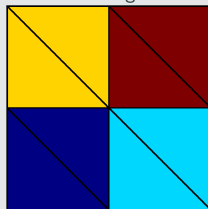


Solution of local problem

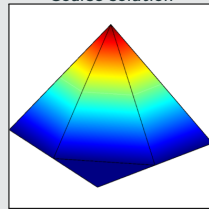


Lagrangian coarse space

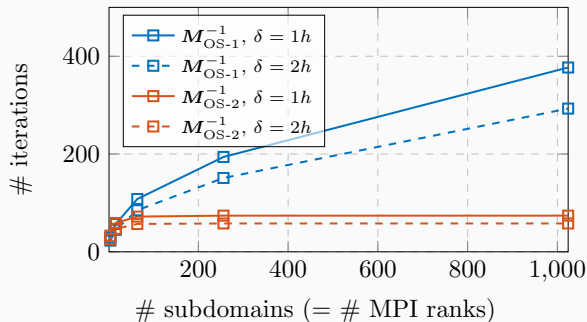
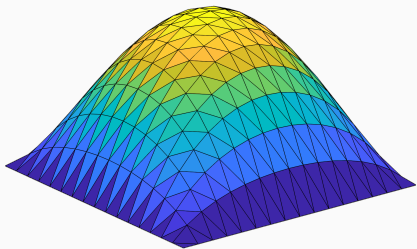
Coarse triangulation



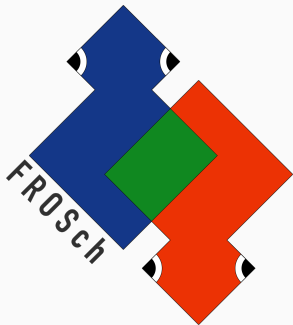
Coarse solution



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



FROSch (Fast and Robust Overlapping Schwarz) Framework in Trilinos



Sandia
National
Laboratories



TUBAF
Die Ressourcenuniversität
Seit 1765

Software

- Object-oriented C++ domain decomposition solver framework with MPI-based distributed memory parallelization
- Part of TRILINOS with the parallel linear algebra based on 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) | ▪ Dane Lacey (FAU) |
| ▪ Alexander Heinlein (TU Delft) | ▪ Friederike Röver (TUBAF) |
| ▪ Kyrill Ho (UCologne) | ▪ Siva Rajamanickam (SNL) |
| ▪ Axel Klawonn (UCologne) | ▪ Oliver Rheinbach (TUBAF) |
| ▪ Jascha Knepper (UCologne) | ▪ Lea Saßmannshausen (UCologne) |
| ▪ Stephan Köhler (TUBAF) | ▪ Ichitaro Yamazaki (SNL) |

Partition of Unity

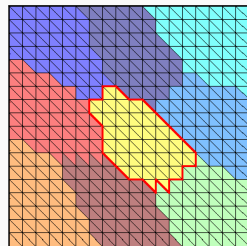
The **energy-minimizing extension** $v_i = H_{\partial\Omega_i \rightarrow \Omega_i}(v_{i,\partial\Omega_i})$ solves

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

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

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

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



Null space property

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

$$\sum_{\text{edges } \subset \partial\Omega_i} \text{[3D plot of a peak on an edge]} + \sum_{\text{vertices } \subset \partial\Omega_i} \text{[3D plot of a peak on a vertex]} = \text{[3D plot of a flat-topped peak covering the entire interface]}$$

Algebraicity of the energy-minimizing extension

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

$$\mathbf{v} = \begin{bmatrix} -K_{II}^{-1} K_{I\Gamma} \\ I_{\Gamma} \end{bmatrix} \mathbf{v}_{\Gamma},$$

Overlapping domain decomposition

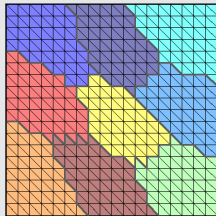
The **overlapping** subdomains are constructed by **recursively adding layers of elements** via the sparsity pattern of K .

The corresponding matrices

$$K_i = R_i K R_i^T$$

can easily be extracted from K .

Nonoverlapping DD



Overlapping domain decomposition

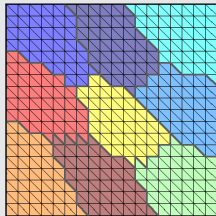
The **overlapping** subdomains are constructed by **recursively adding layers of elements** via the sparsity pattern of K .

The corresponding matrices

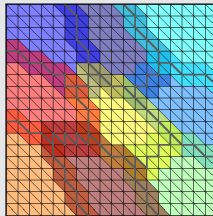
$$K_i = R_i K R_i^T$$

can easily be extracted from K .

Nonoverlapping DD



Overlap $\delta = 1h$



Overlapping domain decomposition

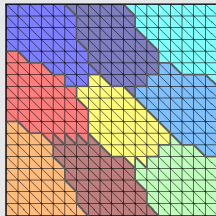
The **overlapping** subdomains are constructed by **recursively adding layers of elements** via the sparsity pattern of K .

The corresponding matrices

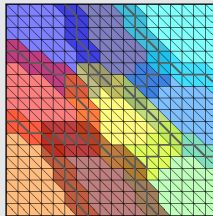
$$K_i = R_i K R_i^T$$

can easily be extracted from K .

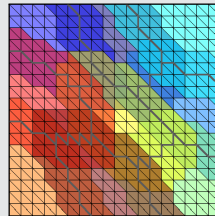
Nonoverlapping DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$



Overlapping domain decomposition

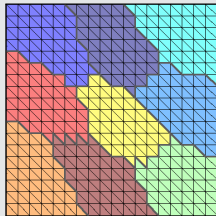
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of K .

The corresponding matrices

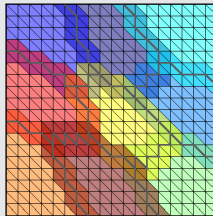
$$K_i = R_i K R_i^T$$

can easily be extracted from K .

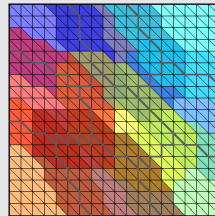
Nonoverlapping DD



Overlap $\delta = 1h$

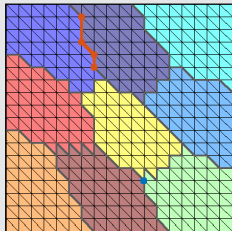


Overlap $\delta = 2h$



Coarse space

1. Interface components



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

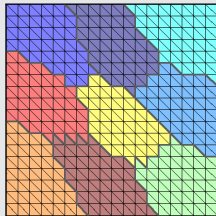
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of K .

The corresponding matrices

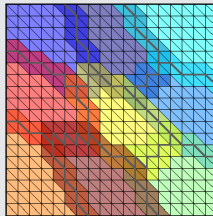
$$K_i = R_i K R_i^T$$

can easily be extracted from K .

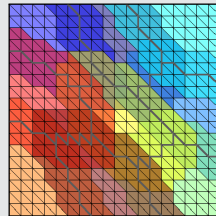
Nonoverlapping DD



Overlap $\delta = 1h$

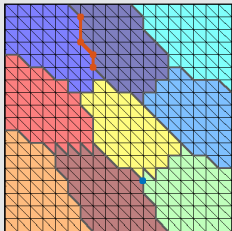


Overlap $\delta = 2h$

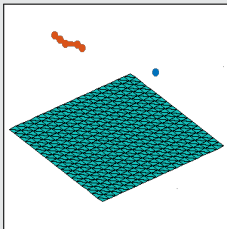


Coarse space

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

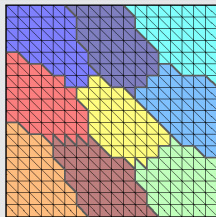
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of K .

The corresponding matrices

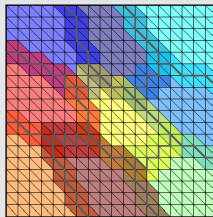
$$K_i = R_i K R_i^T$$

can easily be extracted from K .

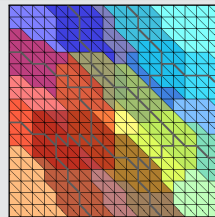
Nonoverlapping DD



Overlap $\delta = 1h$

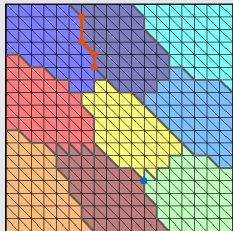


Overlap $\delta = 2h$

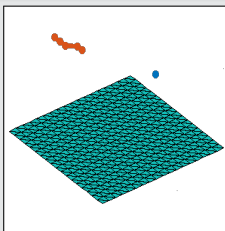


Coarse space

1. Interface components

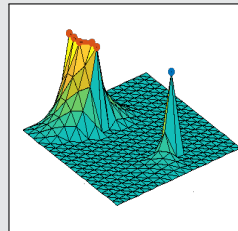


2. Interface basis (partition of unity $\times$ null space)



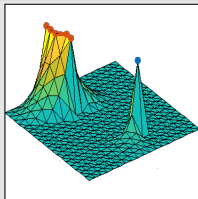
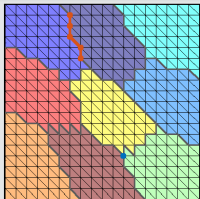
For scalar elliptic problems, the null space consists only of constant functions.

3. Extension



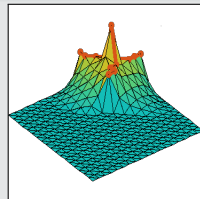
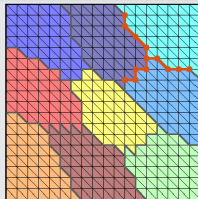
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)



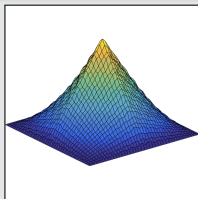
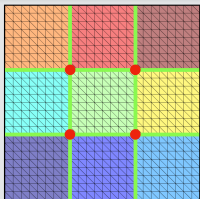
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)



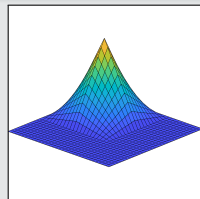
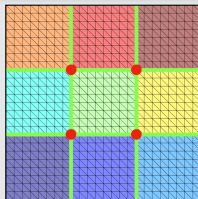
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

MsFEM (Multiscale Finite Element Method)



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

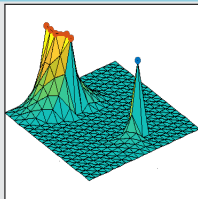
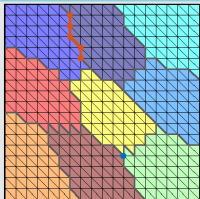
Q1 Lagrangian / piecewise bilinear



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

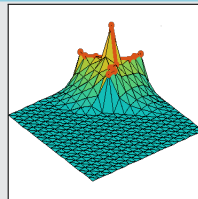
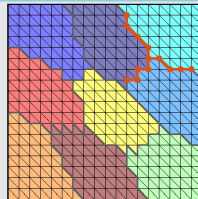
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)



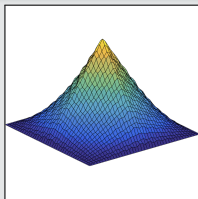
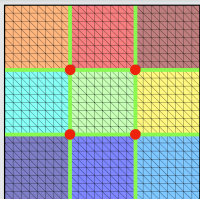
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)



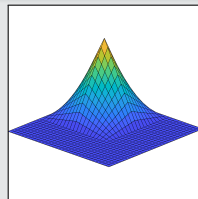
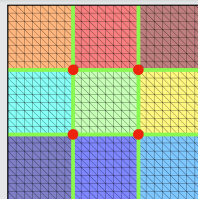
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

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

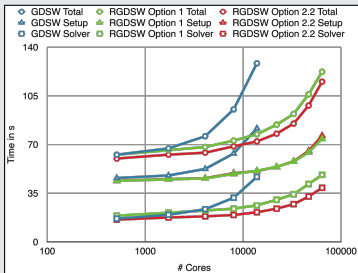
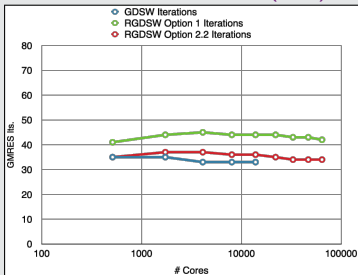


Piecewise linear interface partition of unity functions and a **structured domain decomposition**.

Weak Scalability up to 64k MPI Ranks / 1.7 b Unknowns (3D Poisson; Juqueen)

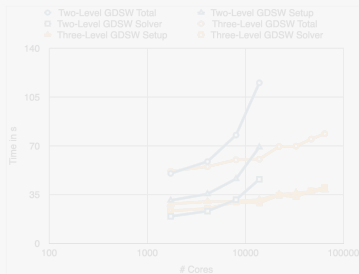
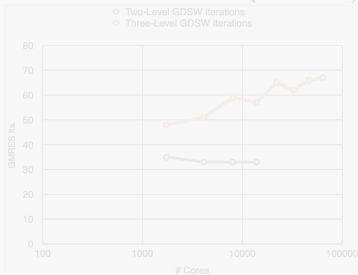
GDSW vs RGDSW (reduced dimension)

Heinlein, Klawonn, Rheinbach, Widlund (2019).



Two-level vs three-level GDSW

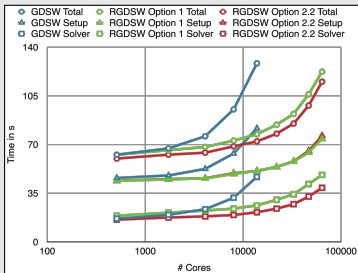
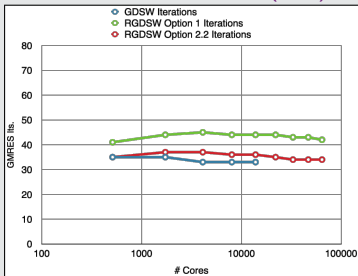
Heinlein, Klawonn, Rheinbach, Röver (2019, 2020).



Weak Scalability up to 64k MPI Ranks / 1.7 b Unknowns (3D Poisson; Juqueen)

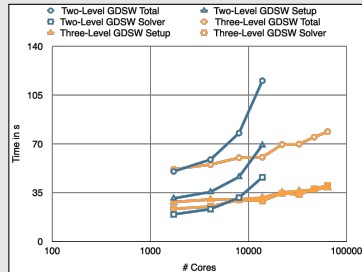
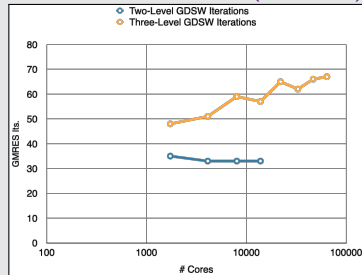
GDSW vs RGDSW (reduced dimension)

Heinlein, Klawonn, Rheinbach, Widlund (2019).



Two-level vs three-level GDSW

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



Inexact Extension Solvers – Ifpack2

$$M_{OS-2}^{-1}A = \Phi A_0^{-1} \Phi^T A + \sum_{i=1}^N R_i^T A_i^{-1} R_i A$$

3D Laplacian; 512 MPI ranks = 512 ($= 8 \times 8 \times 8$) subdomains; $H/\delta = 10$; RGDSW coarse space.

		subdomain solver						
		direct solver	ILU(k)		symm. Gauß–Seidel		Chebyshev polyn.	
			k = 2	k = 3	5 sweeps	10 sweeps	p = 6	p = 8
$H/h = 20$, $\approx 14\,k$ dofs per rank	iter	26	33	30	31	28	34	31
	setup time	1.89 s	0.97 s	1.01 s	0.89 s	0.91 s	0.73 s	0.71 s
	apply time	0.39 s	0.27 s	0.31 s	0.31 s	0.35 s	0.30 s	0.30 s
	prec. time	2.28 s	1.24 s	1.32 s	1.20 s	1.26 s	1.03 s	1.01 s
$H/h = 40$, $\approx 105\,k$ dofs per rank	iter	30	55	46	52	41	59	51
	setup time	12.09 s	6.14 s	6.26 s	5.74 s	5.89 s	5.55 s	5.64 s
	apply time	4.21 s	1.84 s	1.96 s	2.66 s	3.28 s	2.52 s	2.47 s
	prec. time	16.30 s	7.98 s	8.22 s	8.40 s	9.18 s	8.16 s	8.11 s
$H/h = 60$, $\approx 350\,k$ dofs per rank	iter	OOM	81	64	76	56	88	74
	setup time	-	47.29 s	47.87 s	45.14 s	45.08 s	45.44 s	45.49 s
	apply time	-	10.79 s	9.98 s	13.00 s	16.16 s	11.95 s	12.09 s
	prec. time	-	58.08 s	57.85 s	58.15 s	61.25 s	57.39 s	57.59 s

INTEL MKL PARDISO; ILU / symmetric Gauß–Seidel / Chebyshev polynomials from IFPACK2.

Parallel computations on dual-socket Intel Xeon Platinum machine at Sandia National Laboratories (Blake).

Inexact Extension Solvers – Ifpack2

$$M_{OS-2}^{-1}A = \Phi A_0^{-1} \Phi^T A + \sum_{i=1}^N R_i^T A_i^{-1} R_i A$$

3D Laplacian; 512 MPI ranks = 512 ($= 8 \times 8 \times 8$) subdomains; $H/\delta = 10$; RGDSW coarse space.

		subdomain solver						
		direct solver	ILU(k)		symm. Gauß–Seidel		Chebyshev polyn.	
			k = 2	k = 3	5 sweeps	10 sweeps	p = 6	p = 8
$H/h = 20$, ≈ 14 k dofs per rank	iter	26	33	30	31	28	34	31
	setup time	1.89 s	0.97 s	1.01 s	0.89 s	0.91 s	0.73 s	0.71 s
	apply time	0.39 s	0.27 s	0.31 s	0.31 s	0.35 s	0.30 s	0.30 s
	prec. time	2.28 s	1.24 s	1.32 s	1.20 s	1.26 s	1.03 s	1.01 s
$H/h = 40$, ≈ 105 k dofs per rank	iter	30	55	46	52	41	59	51
	setup time	12.09 s	6.14 s	6.26 s	5.74 s	5.89 s	5.55 s	5.64 s
	apply time	4.21 s	1.84 s	1.96 s	2.66 s	3.28 s	2.52 s	2.47 s
	prec. time	16.30 s	7.98 s	8.22 s	8.40 s	9.18 s	8.16 s	8.11 s
$H/h = 60$, ≈ 350 k dofs per rank	iter	OOM	81	64	76	56	88	74
	setup time	-	47.29 s	47.87 s	45.14 s	45.08 s	45.44 s	45.49 s
	apply time	-	10.79 s	9.98 s	13.00 s	16.16 s	11.95 s	12.09 s
	prec. time	-	58.08 s	57.85 s	58.15 s	61.25 s	57.39 s	57.59 s

INTEL MKL PARDISO; ILU / symmetric Gauß–Seidel / Chebyshev polynomials from IFPACK2.

Parallel computations on dual-socket Intel Xeon Platinum machine at Sandia National Laboratories (Blake).

Three-Dimensional Linear Elasticity – ILU Subdomain Solver

ILU level		0	1	2	3
setup					
CPU	No	1.5	1.9	3.0	4.8
	ND	1.6	2.6	4.4	7.4
GPU	KK(No)	1.4	1.5	1.8	2.4
	KK(ND)	1.7	2.0	2.9	5.2
	Fast(No)	1.5	1.6	2.1	3.2
	Fast(ND)	1.5	1.7	2.5	4.5
speedup		1.0×	1.2×	1.4×	1.5×
solve					
CPU	No	2.55 (158)	3.60 (112)	5.28 (99)	6.85 (88)
	ND	4.17 (227)	5.36 (134)	6.61 (105)	7.68 (88)
GPU	KK(No)	3.81 (158)	4.12 (112)	4.77 (99)	5.65 (88)
	KK(ND)	2.89 (227)	4.27 (134)	5.57 (105)	6.36 (88)
	Fast(No)	1.14 (173)	1.11 (141)	1.26 (134)	1.43 (126)
	Fast(ND)	1.49 (227)	1.15 (137)	1.10 (109)	1.22 (100)
speedup		2.2×	3.2×	4.3×	4.8×

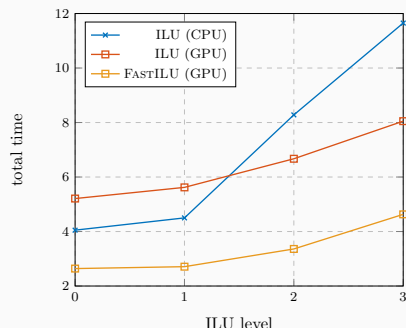
Computations on Summit (OLCF):
42 IBM Power9 CPU cores and 6 NVIDIA
V100 GPUs per node.

**Yamazaki, Heinlein,
Rajamanickam (2023)**

ILU variants

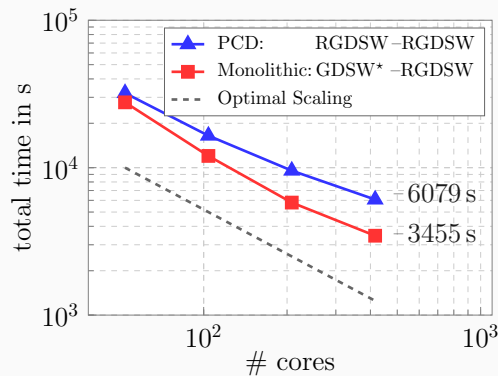
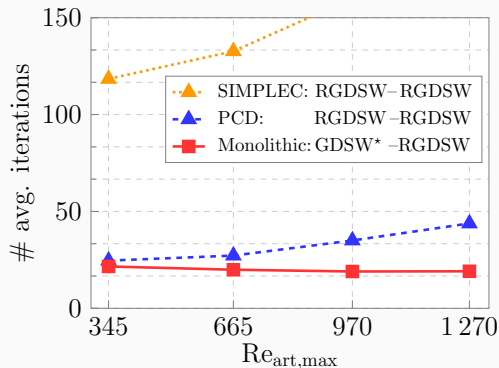
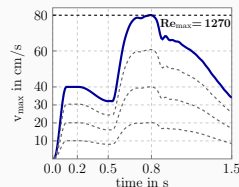
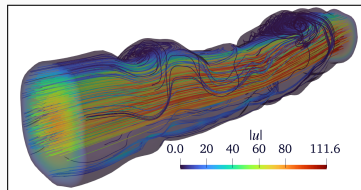
- KOKKOSKERNELS ILU (KK)
- Iterative FASTILU (Fast); cf. **Chow, Patel (2015)** and **Boman, Patel, Chow, Rajamanickam (2016)**

No reordering (**No**) and nested dissection (**ND**)



Results for Blood Flow Simulations

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



Cf. [Heinlein, Klawonn, Knepper, Saßmannshausen \(arXiv 2025\)](#)

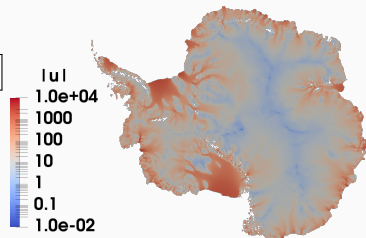


<https://github.com/SNLComputation/Albany>

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)**.



MPI ranks	Antarctica (velocity)			Greenland (multiphysics vel. & temperature)		
	4 km resolution, 20 layers, 35 m dofs			1-10 km resolution, 20 layers, 69 m dofs		
	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
1024	43.3 (11)	9.18 s	5.85 s	53.0 (29)	8.68 s	4.22 s
2048	41.4 (11)	4.15 s	2.63 s	62.2 (86)	4.47 s	4.23 s
4096	41.2 (11)	1.66 s	1.49 s	68.9 (40)	2.52 s	2.86 s
8192	40.2 (11)	1.26 s	1.06 s	-	-	-

Computations performed on Cori (NERSC).

Heinlein, Perego, Rajamanickam (2022)

Domain Decomposition for Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) – Idea

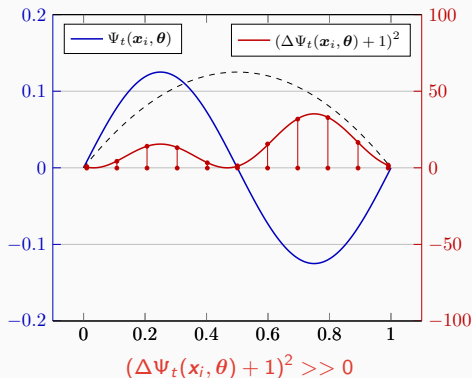
In **Lagaris et al. (1998)**, the authors solve the **boundary value problem**

$$-\Delta \Psi_t(\mathbf{x}, \theta) = 1 \text{ on } [0, 1],$$

$$\Psi_t(0, \theta) = \Psi_t(1, \theta) = 0,$$

via a **collocation approach**:

$$\min_{\theta} \sum_{x_i} (\Delta \Psi_t(x_i, \theta) + 1)^2$$

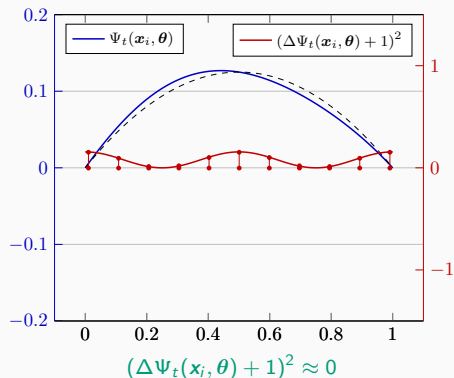


Boundary conditions ...

... can be **enforced explicitly** via the ansatz:

$$\Psi_t(\mathbf{x}, \theta) = A(\mathbf{x}) + F(\mathbf{x}, \text{NN}(\mathbf{x}, \theta))$$

- A satisfies the boundary conditions
- F does not contribute to the boundary conditions



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**:

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

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

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

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

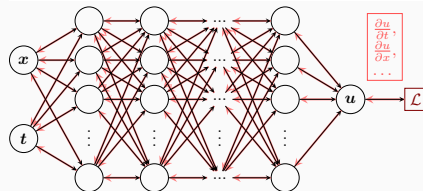
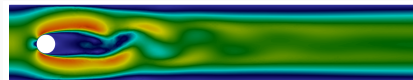
See also **Dissanayake and Phan-Thien (1994)**; **Lagaris et al. (1998)**.

Advantages

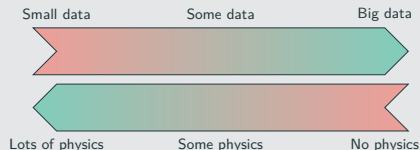
- **"Meshfree"**
- **Small data**
- **Generalization properties**
- **High-dimensional problems**
- **Inverse and parameterized problems**

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

Error Estimate & Spectral Bias

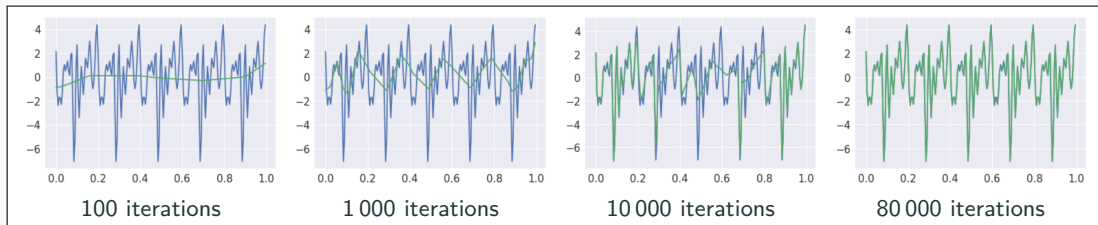
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

$$\varepsilon_G \leq C_{\text{PDE}} \varepsilon_{\mathcal{T}} + C_{\text{PDE}} C_{\text{quad}}^{1/p} N^{-\alpha/p}$$

- $\varepsilon_G = \varepsilon_G(\mathbf{X}, \theta) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **general. error** (V Sobolev space, $\mathbf{X}$ training data set)
- $\varepsilon_{\mathcal{T}}$ **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**”*



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

Related works: Cao et al. (2021), Wang, et al. (2022), Hong et al. (arXiv 2022), Xu et al. (2024), ...

Scaling of PINNs for a Simple ODE Problem

Solve

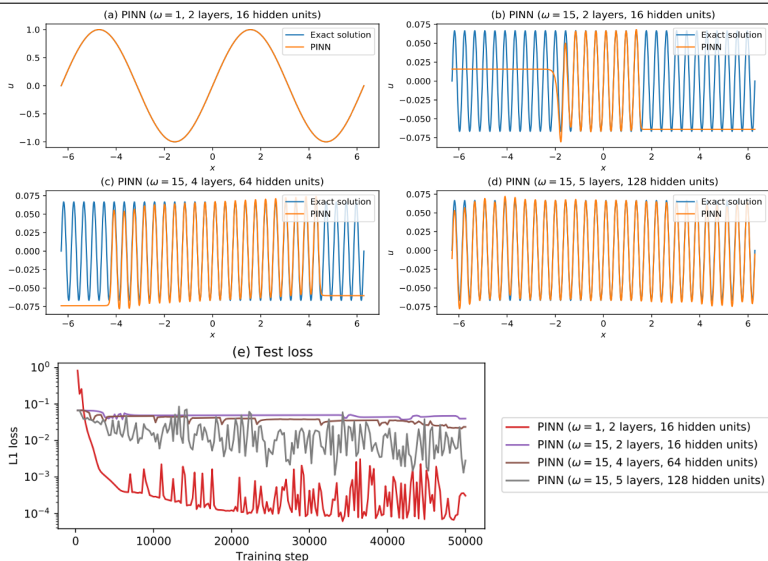
$$\begin{aligned}u' &= \cos(\omega x), \\ u(0) &= 0,\end{aligned}$$

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

Scaling of PINNs for a Simple ODE Problem

Solve

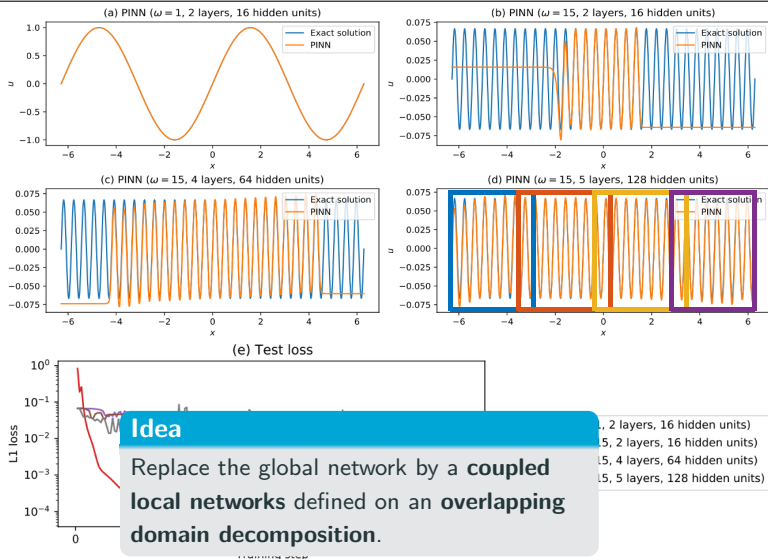
$$\begin{aligned}u' &= \cos(\omega x), \\ u(0) &= 0,\end{aligned}$$

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

A non-exhaustive literature overview:

- **ML for adaptive BDDC, FETI-DP, AGDSW:** H., Klawonn, Lanser, Weber (2019, 2020, 2021, 2021, 2021, 2022); Klawonn, Lanser, Weber (2024, 2025)
- **cPINNs, XPINNs:** Jagtap, Kharazmi, Karniadakis (2020); Jagtap, Karniadakis (2020)
- **Classical Schwarz iteration for PINNs or DeepRitz::** Li, Tang, Wu, and Liao (2019); Li, Xiang, Xu (2020); Mercier, Gratton, Boudier (arXiv 2021); Dolean, H., Mercier, Gratton (acc. 2025); Li, Wang, Cui, Xiang, Xu (2023); Sun, Xu, Yi (arXiv 2023, 2024); Kim, Yang (2023, 2024, 2024)
- **FBPINNs, FBKANs:** Moseley, Markham, Nissen-Meyer (2023); Dolean, H., Mishra, Moseley (2024, 2024); H., Howard, Beecroft, Stinis (2025); Howard, Jacob, Murphy, H., Stinis (arXiv 2024)
- **DD for randomized NNs:** Dong, Li (2021); Dang, Wang (2024); Sun, Dong, Wang (2024); Sun, Wang (2024); Chen, Chi, E, Yang (2022); Shang, H., Mishra, Wang (2025); Anderson, Dolean, Moseley, Pestana, (arXiv 2024); van Beek, Dolean, Moseley (arxiv 2025)
- **DD for Neural Operators and Surrogate Models:** H., Howard, Beecroft, Stinis (2025); Ramezankhani, Parekh, Deodhar, Birru (arXiv 2024); Wu, Kovachki, Liu (arXiv 2025); Pelzer, Verburg, H., Schulte (arXiv 2025); Klaes, Klawonn, Kubicki, Lanser, Nakajima, Shimokawabe, Weber (arXiv 2025); Howard, H., Stinis (in prep.)
- **DD for CNNs:** Gu, Zhang, Liu, Cai (2022); Lee, Park, Lee (2022); Klawonn, Lanser, Weber (2024); Verburg, H., Cyr (2025)

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



A. Klawonn, M. Lanser, J. Weber

Machine learning, domain decomposition methods – a survey

Computational Science and Engineering. 2024

Finite Basis Physics-Informed Neural Networks (FBPINNs)

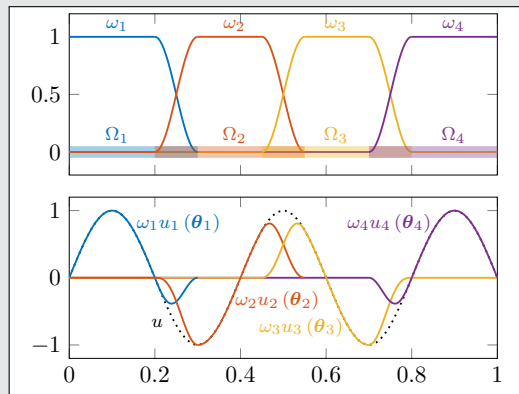
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

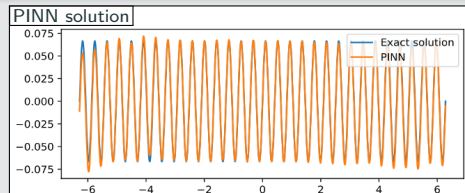
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

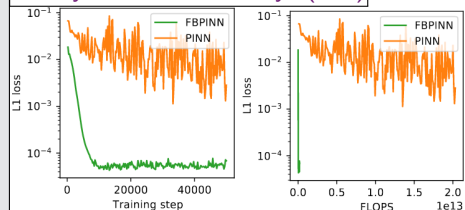
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(n \left[\sum_{\mathbf{x}_i \in \Omega_j} \omega_j u_j(\mathbf{x}_i, \theta_j) - f(\mathbf{x}_i) \right]^2 \right)$$



1D single-frequency problem



Moseley, Markham, Nissen-Meyer (2023)



Finite Basis Physics-Informed Neural Networks (FBPINNs)

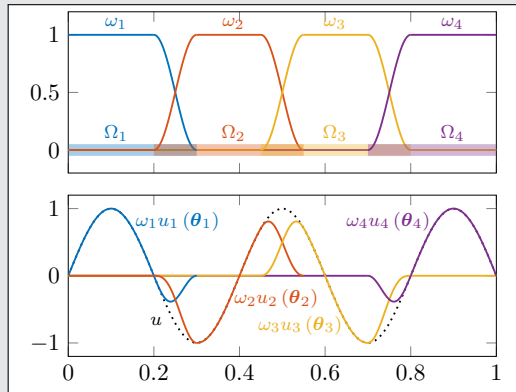
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

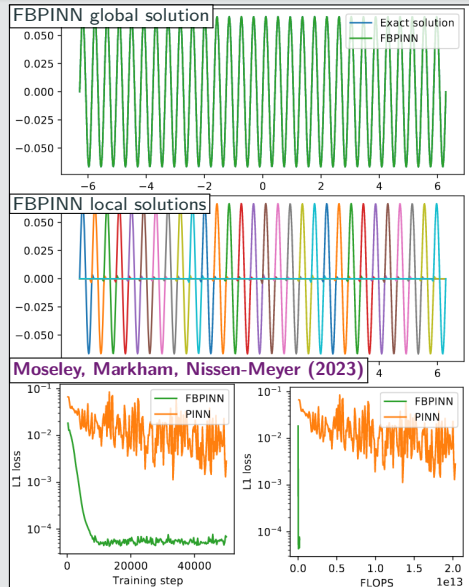
$$u(\theta_1, \dots, \theta_J) = \sum_{j=1}^J \omega_j u_j(\theta_j)$$

and the **loss function**

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(n \left[\sum_{\mathbf{x}_i \in \Omega_j} \omega_j u_j(\mathbf{x}_i, \theta_j) - f(\mathbf{x}_i) \right]^2 \right)$$

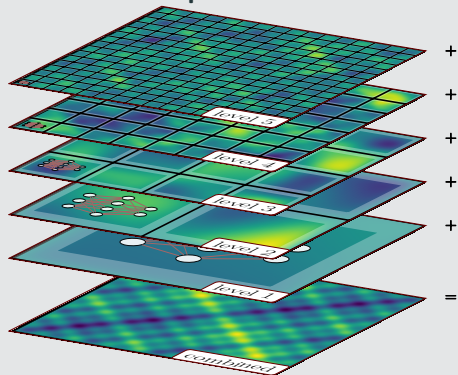


1D single-frequency problem



Multi-level FBPINNs (ML-FBPINNs)

ML-FBPINNs (Dolean, Heinlein, Mishra, Moseley (2024)) are based on a hierarchy of domain decompositions:

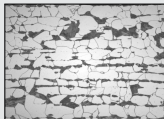


This yields the **network architecture**:

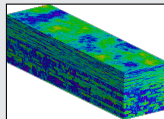
$$u(\theta_1^{(1)}, \dots, \theta_{j^{(L)}}^{(L)}) = \sum_{l=1}^L \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)})$$

Multiscale problems ...

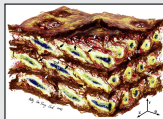
... appear in most areas of modern science and engineering:



Dual-phase steel;
fig. courtesy of
J. Schröder.



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



Arterial walls;
cf. **O'Connell et al.**
(2008).

Multi-frequency problem

Consider the **multi-frequency Laplace problem**

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

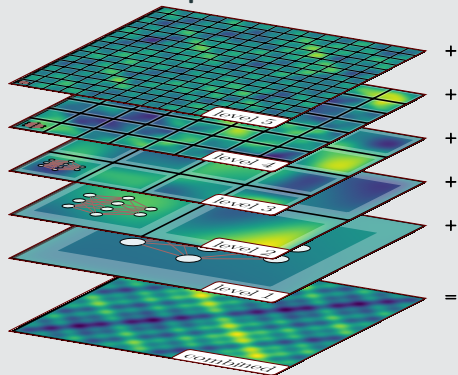
with homogeneous Dirichlet boundary conditions and $\omega_j = 2^j$.

For increasing values of n , we obtain the solutions:



Multi-level FBPINNs (ML-FBPINNs)

ML-FBPINNs (Dolean, Heinlein, Mishra, Moseley (2024)) are based on a **hierarchy of domain decompositions**:



This yields the **network architecture**:

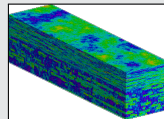
$$u(\theta_1^{(1)}, \dots, \theta_{j^{(L)}}^{(L)}) = \sum_{l=1}^L \sum_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}(\theta_j^{(l)})$$

Multiscale problems ...

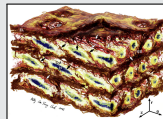
... appear in most areas of modern science and engineering:



Dual-phase steel;
fig. courtesy of
J. Schröder.



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



Arterial walls;
cf. **O'Connell et al.**
(2008).

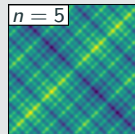
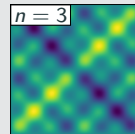
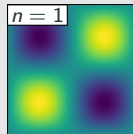
Multi-frequency problem

Consider the **multi-frequency Laplace problem**

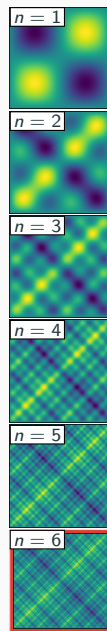
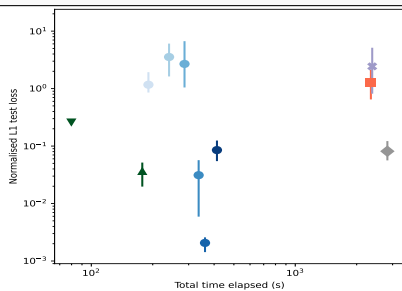
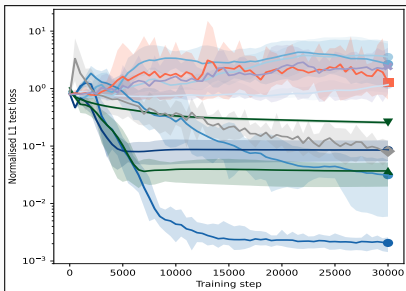
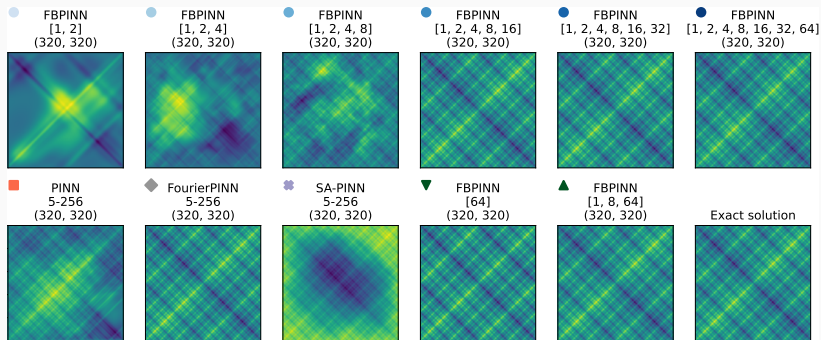
$$-\Delta u = 2 \sum_{i=1}^n (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y),$$

with homogeneous Dirichlet boundary conditions and $\omega_i = 2^i$.

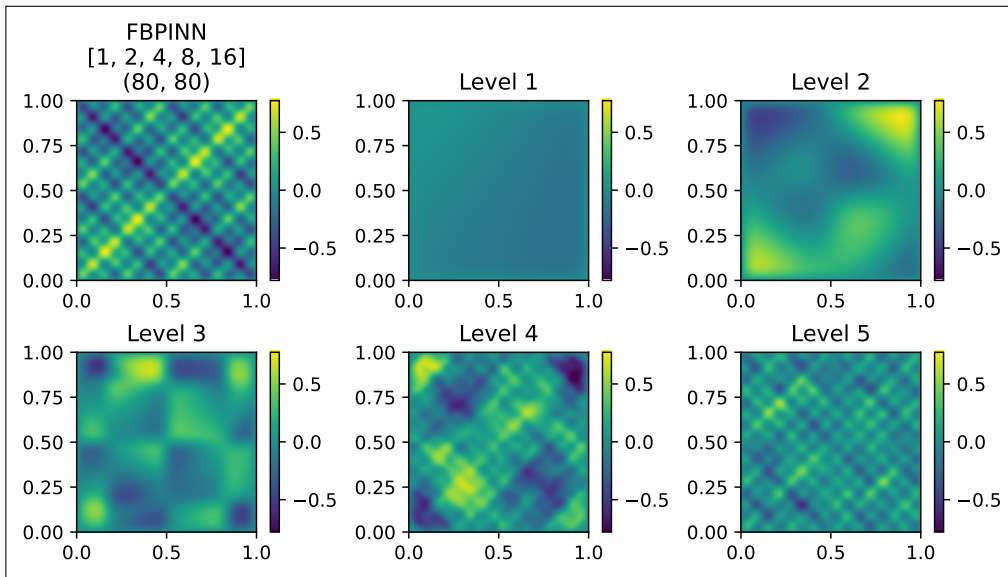
For increasing values of n , we obtain the **solutions**:



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



Multi-Frequency Problem – What the FBPINN Learns



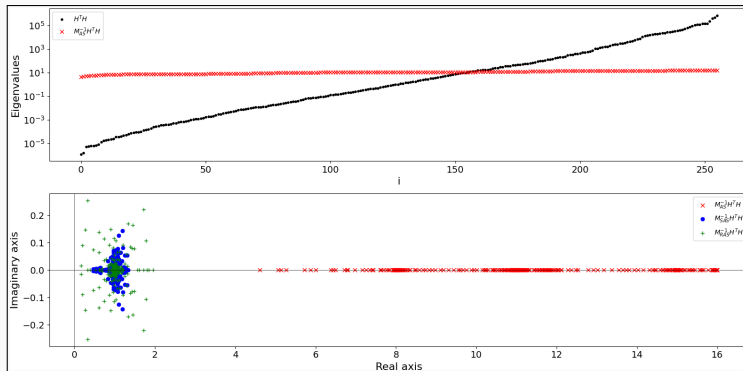
Cf. [Dolean, Heinlein, Mishra, Moseley \(2024\)](#).

Randomized Neural Networks – Results for the Multi-Frequency Problem ($n=2$)

Methodology

- Randomized neural networks yield a quadratic problem
- Normal equations solved with Krylov solvers
- One-level Schwarz preconditioning
- Remove near-linear dependencies via singular value decomposition

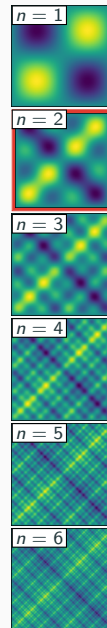
⇒ Improved convergence



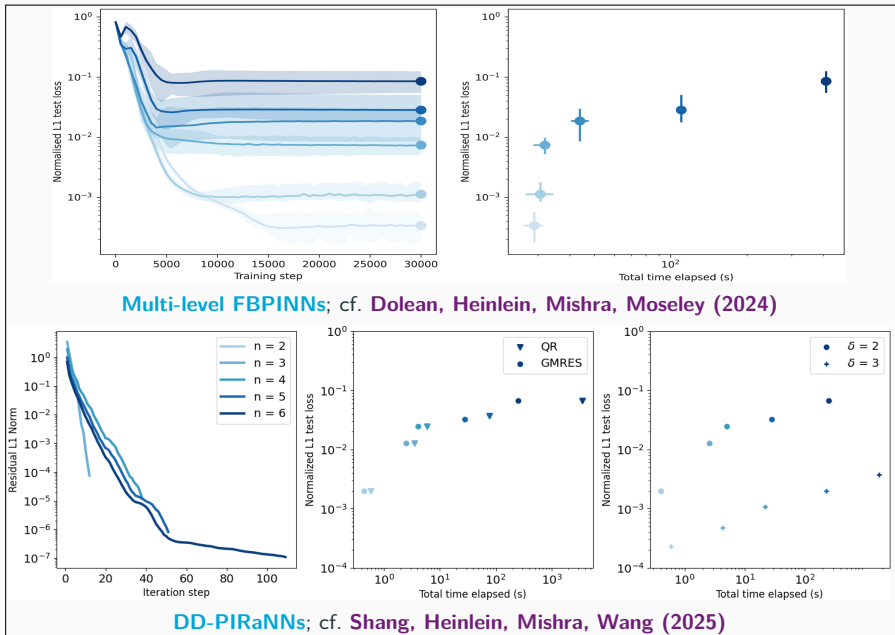
	$M^{-1} = I$		$M^{-1} = M_{AS}^{-1}$		$M^{-1} = M_{RAS}^{-1}$		$M^{-1} = M_{SAS}^{-1}$	
	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}	iter	e_{L^2}
CG	>2000	$1.95 \cdot 10^{-2}$	8	$5.03 \cdot 10^{-3}$	—	—	—	—
CGS	>2000	$2.63 \cdot 10^{-2}$	4	$5.04 \cdot 10^{-3}$	24	$5.03 \cdot 10^{-3}$	6	$5.04 \cdot 10^{-3}$
BICG	>2000	$1.03 \cdot 10^{-2}$	8	$5.08 \cdot 10^{-3}$	32	$5.05 \cdot 10^{-3}$	11	$5.09 \cdot 10^{-3}$
GMRES	>2000	$8.68 \cdot 10^{-2}$	13	$5.07 \cdot 10^{-3}$	31	$5.06 \cdot 10^{-3}$	11	$5.08 \cdot 10^{-3}$

4×4 DD; DoF=256; $N=1600$; $\theta^0 \in \mathcal{U}(-1, 1)$; stop.: $\|M^{-1} r^k\|_{L^2} / \|M^{-1} r^0\|_{L^2} \leq 10^{-5}$

Cf. Shang, Heinlein, Mishra, Wang (2025)



Results for the Multi-Frequency Problem



Advances of FROSch Preconditioners for Multiphysics and Multiscale Simulations

- Partition of unity-based coarse spaces yield **robust and scalable** two-/three-level Schwarz preconditioners, with scalability up to **more than 64k ranks / 1.7b DoF**.
- Inexact subdomain solvers (e.g., ILU) can **significantly speed up computing times**.
- Extension-based, monolithic coarse spaces stay **robust** for **saddle-point & multiphysics** (CFD, land ice) versus block/SIMPLE variants.

Domain Decomposition-Based Neural Network Architectures

- DD-based neural network architectures **scale PINNs** to **large domains / high-frequency problems** with **weakly coupled, small subnetworks**; multilevel hierarchies are needed for **multiscale problems**.
- Schwarz preconditioners & SVD (leveraging DD-induced sparsity) for **least squares** problems resulting from RaNNs **improve conditioning** and yield **fast convergence**.

Thank you for your attention!



Topical Activity
Group
Scientific Machine
Learning

