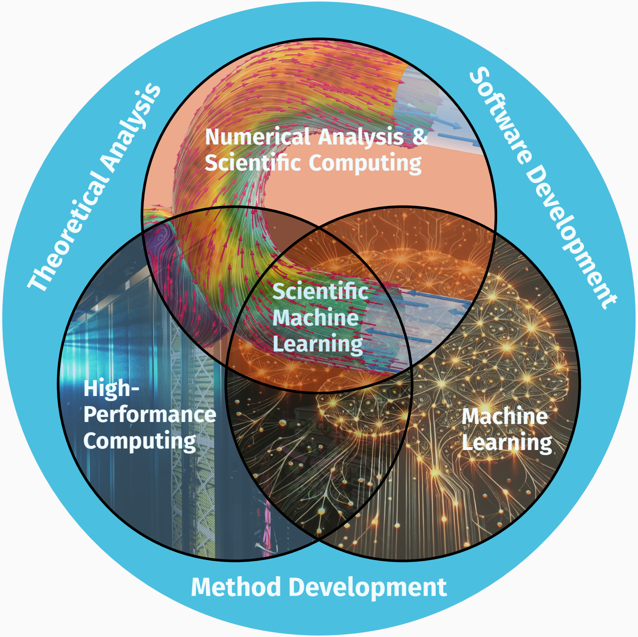
Domain Decomposition Methods for Scientific Computing and Machine Learning

Alexander Heinlein[1]

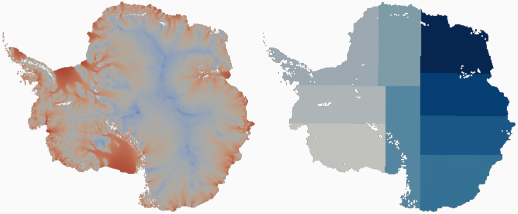
Diagnostics & Data Science Seminar, ASML, March 19, 2025

[1]Delft University of Technology

SCaLA – Scalable Scientific Computing and Learning Algorithms



Domain Decomposition Methods



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

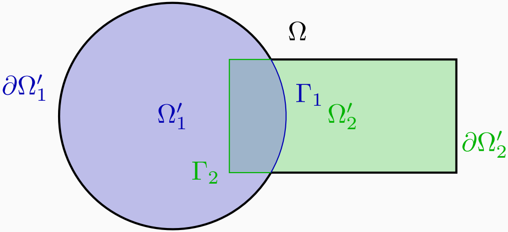
Historical remarks: The **alternating Schwarz method** is the earliest **domain decomposition method (DDM)**, which has been invented by **H. A. Schwarz** and published in **1870**:

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

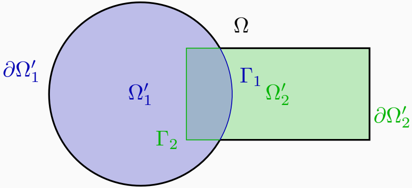
Idea

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

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



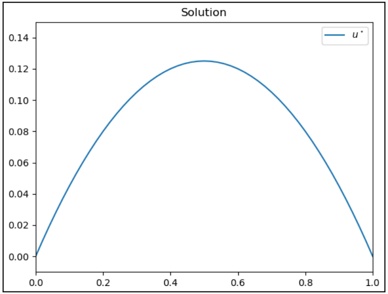
For the sake of simplicity, instead of the two-dimensional geometry,



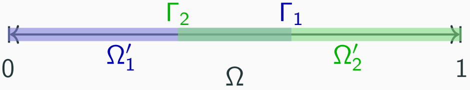
we consider the **one-dimensional Poisson equation**

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

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



Overlapping domain decomposition:



The Alternating Schwarz Algorithm – 1D Laplace Results

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

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

We perform an **alternating Schwarz iteration**:

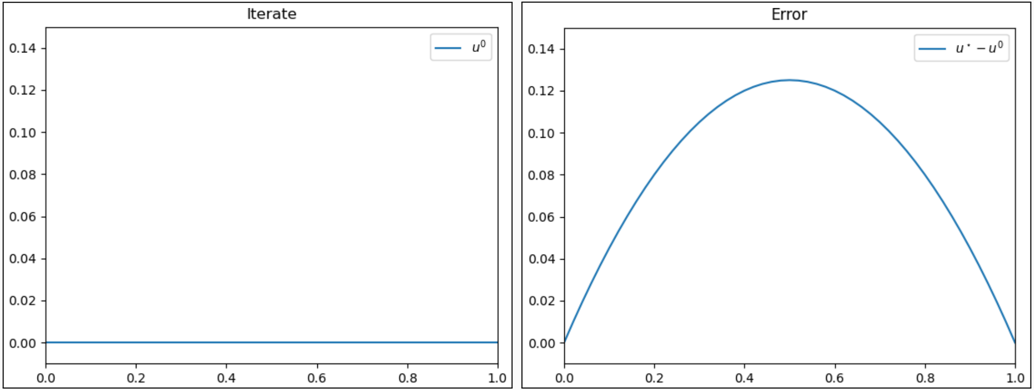


Figure 1: Iterate (left) and error (right) in iteration 0.

The Alternating Schwarz Algorithm – 1D Laplace Results

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

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

We perform an **alternating Schwarz iteration**:

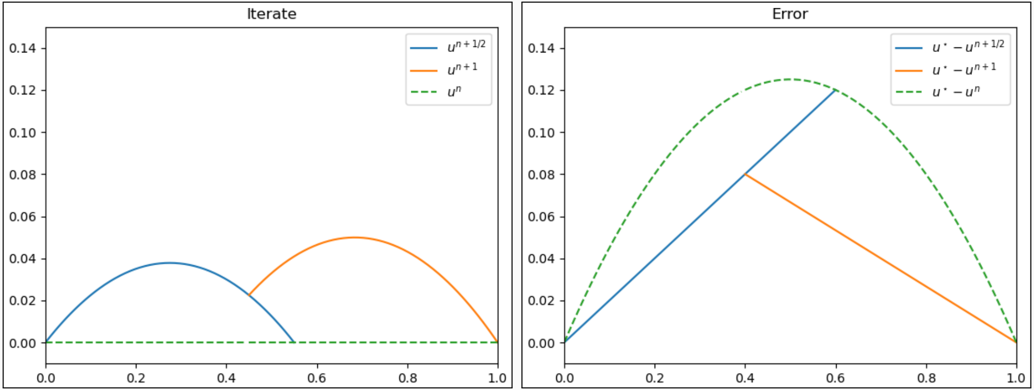


Figure 1: Iterate (left) and error (right) in iteration 1.

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

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

We perform an **alternating Schwarz iteration**:

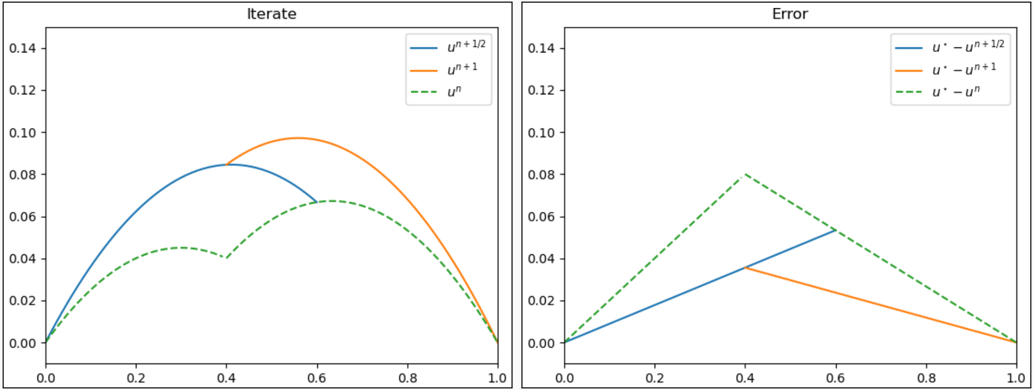


Figure 1: Iterate (left) and error (right) in iteration 2.

The Alternating Schwarz Algorithm – 1D Laplace Results

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

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

We perform an **alternating Schwarz iteration**:

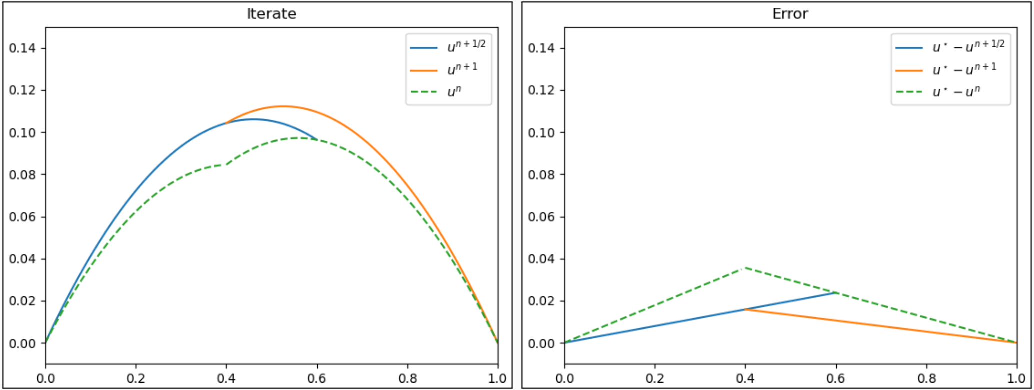


Figure 1: Iterate (left) and error (right) in iteration 3.

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

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

We perform an **alternating Schwarz iteration**:

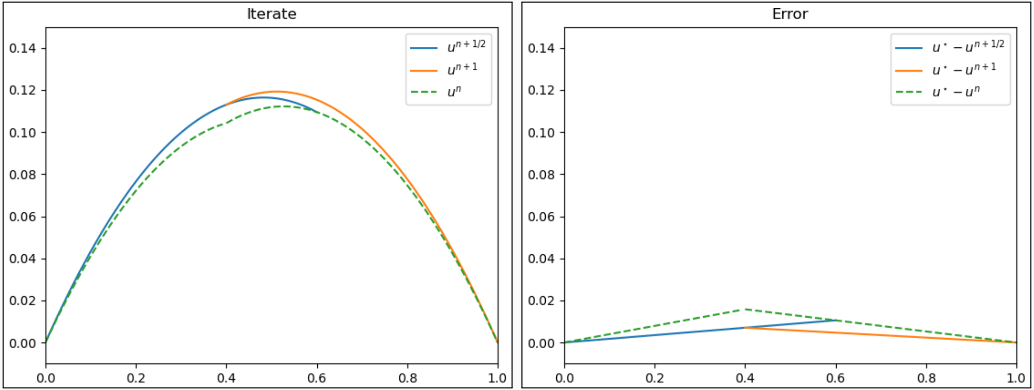


Figure 1: Iterate (left) and error (right) in iteration 4.

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

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

We perform an **alternating Schwarz iteration**:

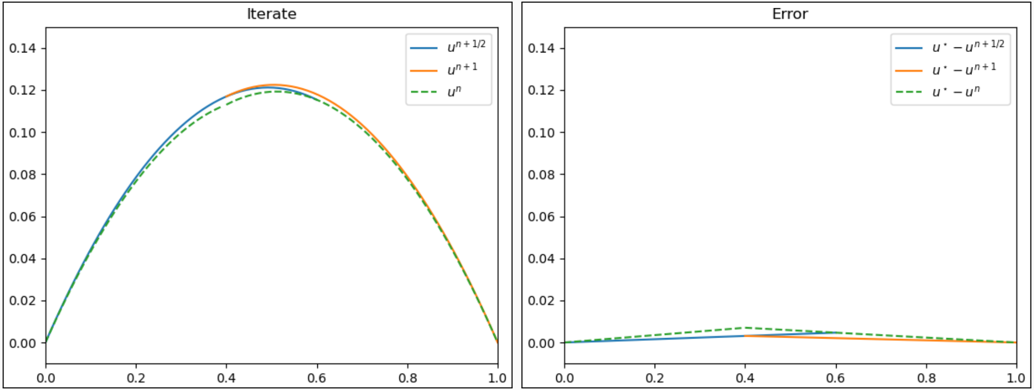


Figure 1: Iterate (left) and error (right) in iteration 5.

Solvers for Partial Different Equations

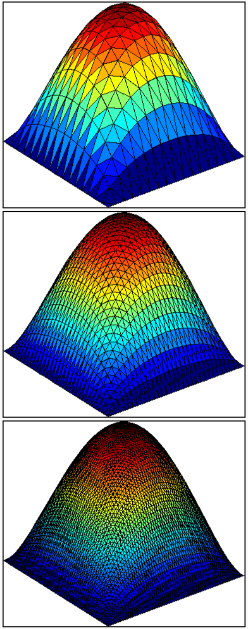
Consider a **diffusion model problem**:

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

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

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

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



Direct solvers

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

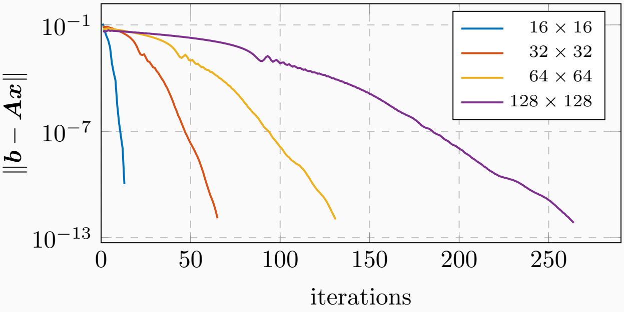
Iterative solvers

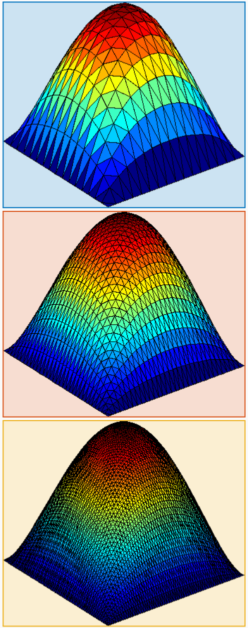
Iterative solvers are efficient for solving **sparse systems**, however, the **convergence rate depends on the spectral properties of $\boldsymbol{K}$.**

Consider a **diffusion model problem**:

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

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

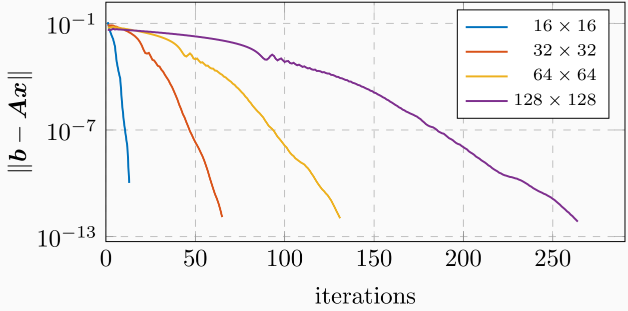


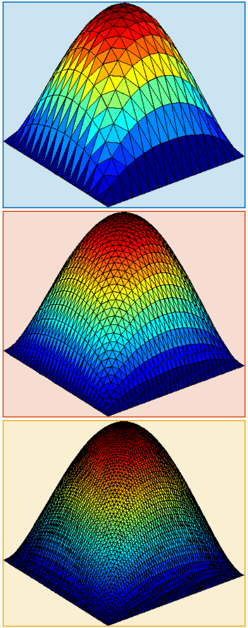


Consider a **diffusion model problem**:

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

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





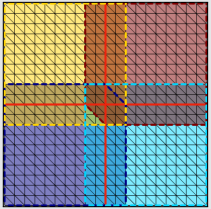
$\Rightarrow$ Introduce a preconditioner $\boldsymbol{M}^{-1} \approx \boldsymbol{K}^{-1}$ to **improve convergence**:

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

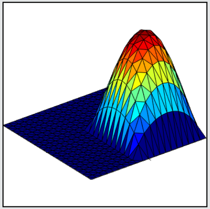
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



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

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

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

Condition number estimate:

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

with subdomain size H and overlap width δ.

Lagrangian coarse space

Coarse triangulation

Coarse solution

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

$$\boldsymbol{M}_{\text{OS-2}}^{-1}\boldsymbol{K} = \underbrace{\Phi \boldsymbol{K}_0^{-1} \Phi^{\top} \boldsymbol{K}}_{\text{coarse level – global}} + \underbrace{\sum\nolimits_{i=1}^{N} \boldsymbol{R}_i^{\top} \boldsymbol{K}_i^{-1} \boldsymbol{R}_i \boldsymbol{K}}_{\text{first level – local}},$$

where Φ contains the coarse basis functions and $\boldsymbol{K}_0 := \Phi^{\top} \boldsymbol{K} \Phi$; cf., e.g., **Toselli, Widlund (2005)**.
The construction of a Lagrangian coarse basis requires a coarse triangulation.

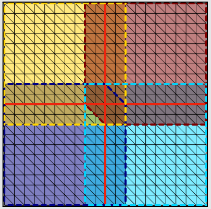
Condition number estimate:

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

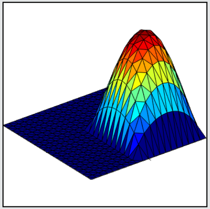
Two-Level Schwarz Preconditioners

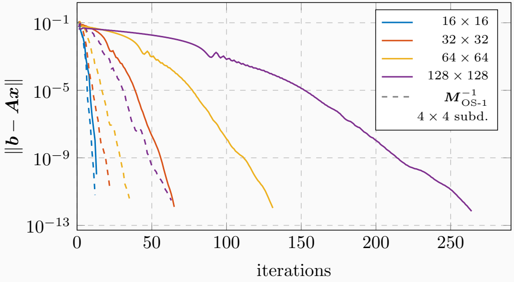
One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem





Lagrangian coarse space

Coarse triangulation

Coarse solution

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

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

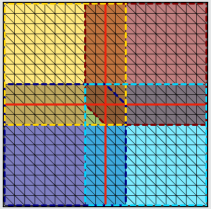
where Φ contains the coarse basis functions and $K_0 := \Phi^\top K \Phi$; cf., e.g., **Toselli, Widlund (2005)**. The construction of a Lagrangian coarse basis requires a coarse triangulation.

Condition number estimate:

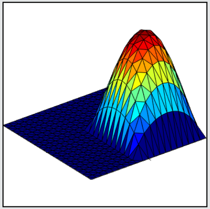
$$\kappa\left(M_{\text{OS-2}}^{-1} K\right) \leq C \left(1 + \frac{H}{\delta}\right)$$

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

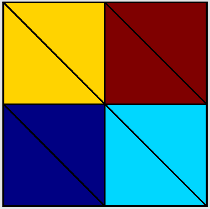
Condition number estimate:

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

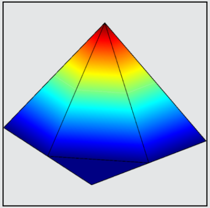
with subdomain size H and overlap width δ.

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.

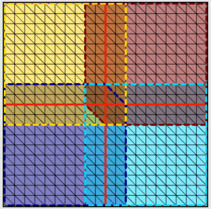
Condition number estimate:

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

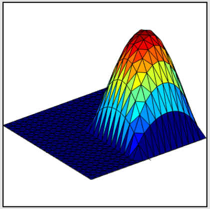
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$

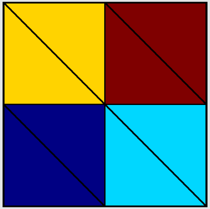


Solution of local problem

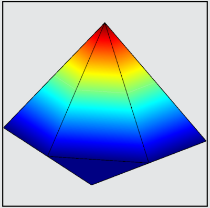


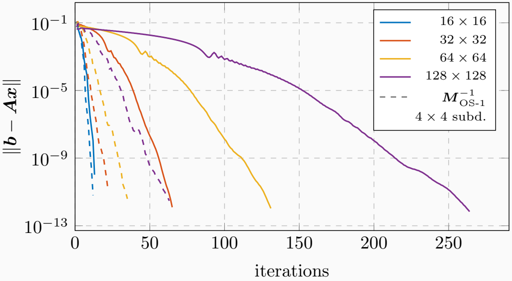
Lagrangian coarse space

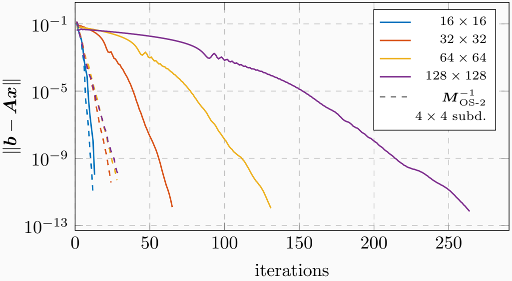
Coarse triangulation



Coarse solution



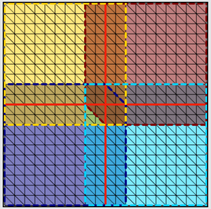




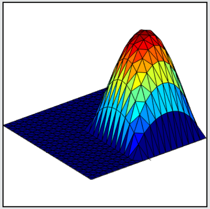
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$

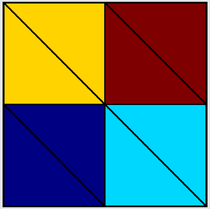


Solution of local problem

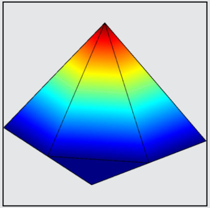


Lagrangian coarse space

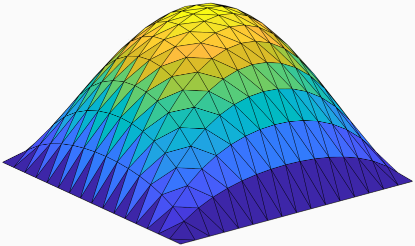
Coarse triangulation

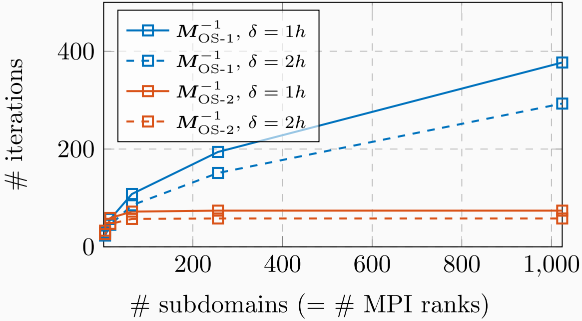


Coarse solution

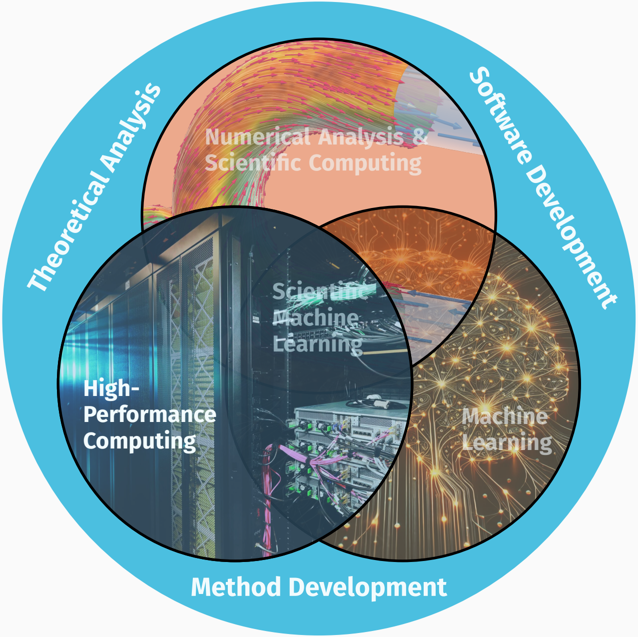


Diffusion model problem in two dimensions,
$H/h = 100$





SCaLA – Scalable Scientific Computing and Learning Algorithms



FROSch (Fast and Robust Overlapping Schwarz) Framework in Trilinos

Software

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

Methodology

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

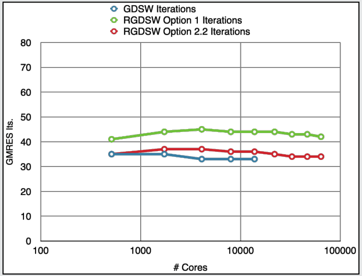
Team (active)

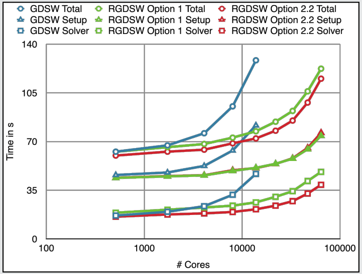
- Filipe Cumaru (TU Delft)
- Kyrill Ho (UCologne)
- Jascha Knepper (UCologne)
- Friederike Röver (TUBAF)
- Lea Saßmannshausen (UCologne)
- Alexander Heinlein (TU Delft)
- Axel Klawonn (UCologne)
- Siva Rajamanickam (SNL)
- Oliver Rheinbach (TUBAF)
- Ichitaro Yamazaki (SNL)

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

GDSW vs RGDSW (reduced dimension)

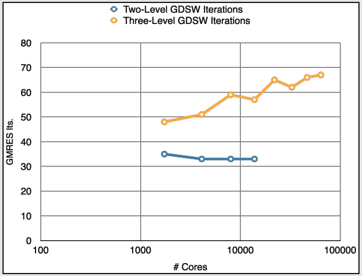
Heinlein, Klawonn, Rheinbach, Widlund (2019).

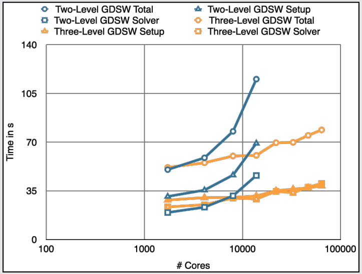




Two-level vs three-level GDSW

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

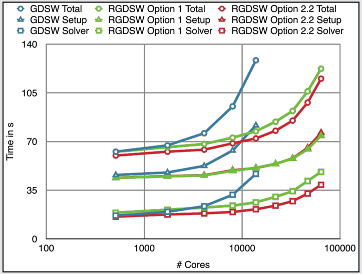




Weak Scalability up to $64\,\mathrm{k}$ MPI Ranks / $1.7\,\mathrm{b}$ Unknowns (3D Poisson; Juqueen)

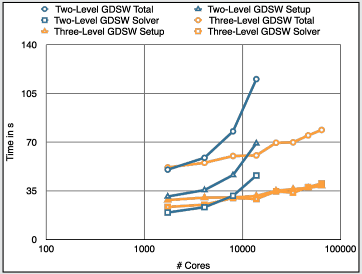
GDSW vs RGDSW (reduced dimension)

Heinlein, Klawonn, Rheinbach, Widlund (2019).

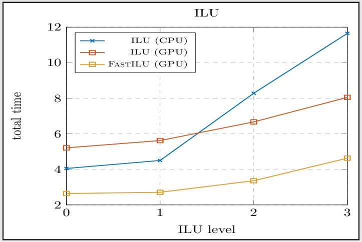


Two-level vs three-level GDSW

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



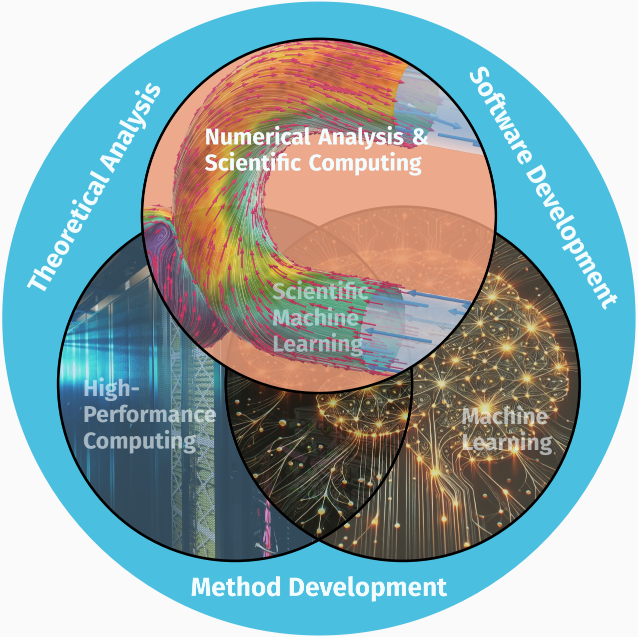
Inexact subdomain solvers & GPUs



→ **Speedup** possible via **inexact subdomain solvers & GPUs** using Kokkos and KokkosKernels.

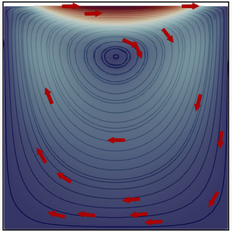
Cf. Yamazaki, Heinlein, Rajamanickam (2023).

SCaLA – Scalable Scientific Computing and Learning Algorithms

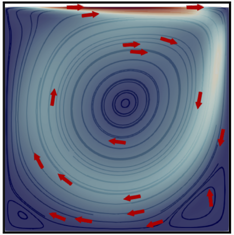


Consider the discrete saddle point problem

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



Stokes flow



Navier–Stokes flow

Monolithic GDSW preconditioner

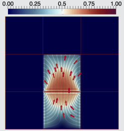
We construct a **monolithic GDSW preconditioner**

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

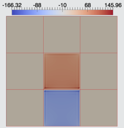
with block matrices $\mathcal{A}_0 = \phi^\top \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^\top$, local pressure projections $\overline{\mathcal{P}}_i$, and

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

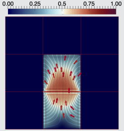
Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



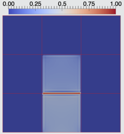
Φ_{u,u_0}



Φ_{p,u_0}



Φ_{u,p_0}



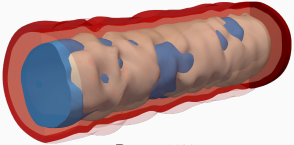
Φ_{p,p_0} .

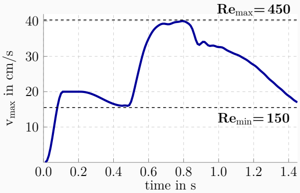
Related work:

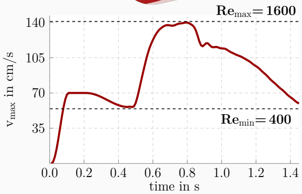
- Original work on monolithic Schwarz preconditioners: **Klawonn and Pavarino (1998, 2000)**
- Other publications on monolithic Schwarz preconditioners: e.g., **Hwang and Cai (2006)**, **Barker and Cai (2010)**, **Wu and Cai (2014)**, and the presentation **Dohrmann (2010)** at the *Workshop on Adaptive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

Results for Blood Flow Simulations

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





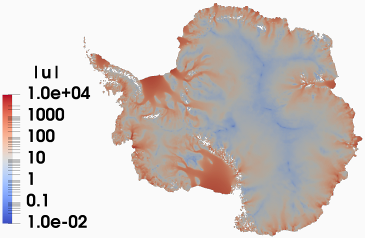


prec.	# MPI ranks	16	64	256
Monolithic RGDSW (FROSCH)	avg. #its.	33	31	30
	setup	4 825 s	1 422 s	701 s
	solve	3 198 s	1 004 s	463 s
	total	8 023 s	2 426 s	**1 164 s**
SIMPLE RGDSW (TEKO & FROSCH)	avg. #its.	82	82	87
	setup	3 046 s	824 s	428 s
	solve	4 679 s	1 533 s	801 s
	total	**7 725 s**	**2 357 s**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

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

Albany

https://github.com/SNLComputation/Albany



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

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

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

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

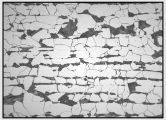
Computations performed on Cori (NERSC).

Heinlein, Perego, Rajamanickam (2022)

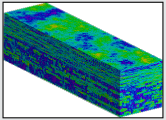
Spectral Extension-Based Coarse Spaces for Schwarz Preconditioners

Highly heterogeneous problems . . .

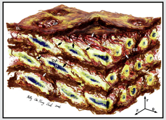
. . . appear in most areas of modern science and engineering:



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 $tol_{\mathcal{F}}$:

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

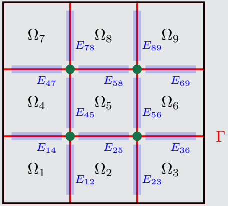
C does not depend on h, H, or the coefficients.
OS-ACMS & adaptive GDSW (AGDSW) (**Heinlein, Klawonn, Knepper, Rheinbach (2018, 2018, 2019)**).

Local eigenvalue problems

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

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

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

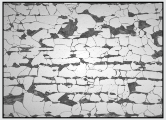


with **Schur complements** S_{EE}, S_{FF} with **Neumann boundary conditions** and **submatrices** K_{EE}, K_{FF} of K. We select eigenfunctions corresponding to **eigenvalues below tolerances** $tol_{\mathcal{E}}$ and $tol_{\mathcal{F}}$.

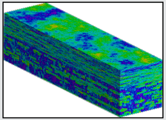
$\rightarrow$ The corresponding coarse basis functions are **energy-minimizing extensions** into the interior of the subdomains.

Highly heterogeneous problems ...

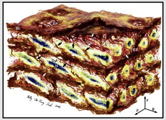
...appear in most areas of modern science and engineering:



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)**.

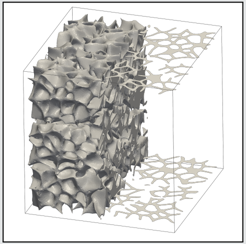
Spectral coarse spaces

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

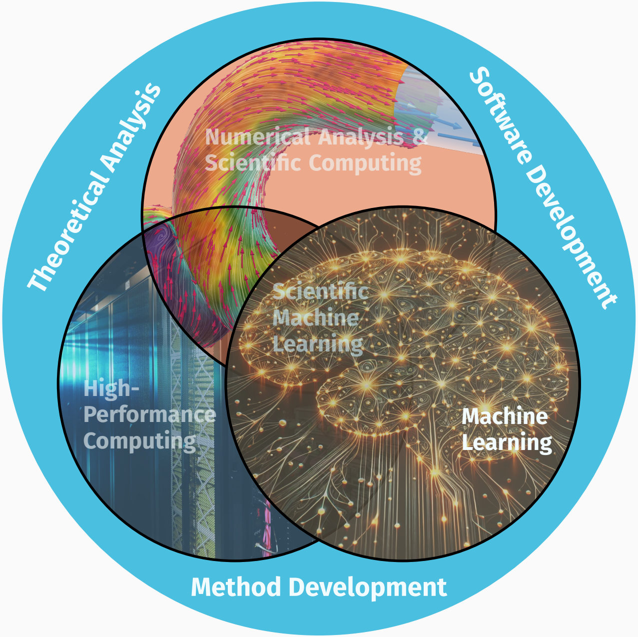


Solid phase: $\alpha = 10^6$; **transparent phase:** $\alpha = 1$; 100 subdomains

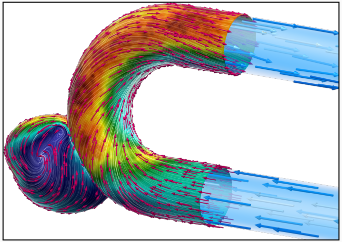
V_0	$tol_{\mathcal{E}}$	$tol_{\mathcal{F}}$	it.	κ	dim V_0	dim V_0/ dof
V_{GDSW}	—	—	**565**	$1.3 \cdot 10^6$	1 601	0.27 %
V_{AGDSW}	0.05	0.05	**60**	30.2	1 968	0.33 %
$V_{\mathrm{OS-ACMS}}$	0.001	0.001	**57**	30.3	690	0.12 %

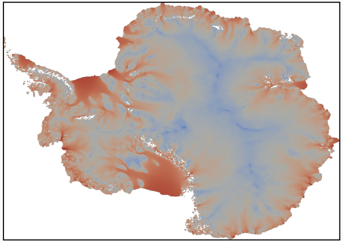
Cf. **Heinlein, Klawonn, Knepper, Rheinbach (2018, 2019)**.

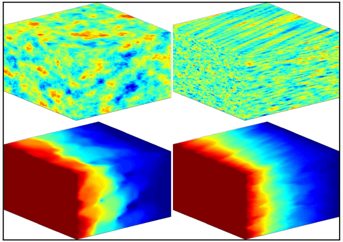
SCaLA – Scalable Scientific Computing and Learning Algorithms



Numerical Analysis and Machine Learning







Numerical methods

Based on physical models

- \+ Robust and generalizable
- − Require availability of mathematical models

Machine learning models

Driven by data

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

Scientific machine learning (SciML)

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

numerical methods **improve** machine learning techniques

machine learning techniques **assist** numerical methods

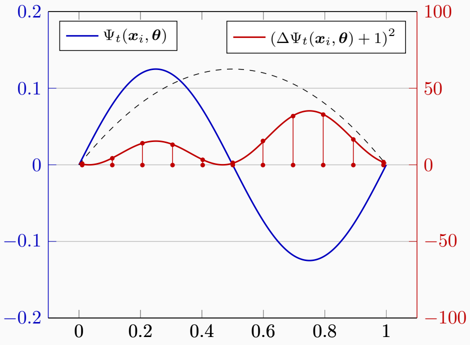
Physics-Informed Neural Networks (PINNs) – Idea

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

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

via a **collocation approach**:

$$\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}_i} \left(\Delta \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}) + 1\right)^2$$



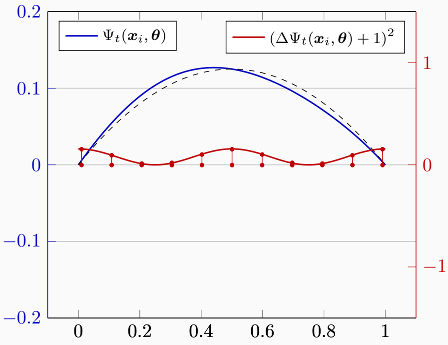
$(\Delta \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}) + 1)^2 >> 0$

Boundary conditions ...

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

$$\Psi_t(\boldsymbol{x}, \boldsymbol{\theta}) = A(\boldsymbol{x}) + F(\boldsymbol{x}, \mathrm{NN}(\boldsymbol{x}, \boldsymbol{\theta}))$$

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



$(\Delta \Psi_t(\boldsymbol{x}_i, \boldsymbol{\theta}) + 1)^2 \approx 0$

Physics-Informed Neural Networks (PINNs)

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

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

PINNs use a **hybrid loss function**:

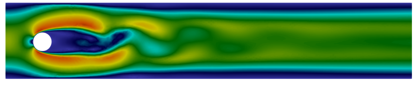
$$\mathcal{L}(\boldsymbol{\theta}) = \omega_{\text{data}} \mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{PDE}} \mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}),$$

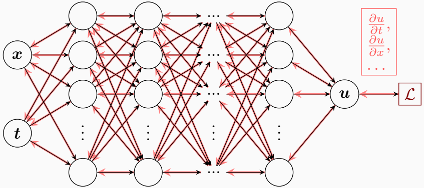
where ω_{data} and ω_{PDE} are **weights** and

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

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

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





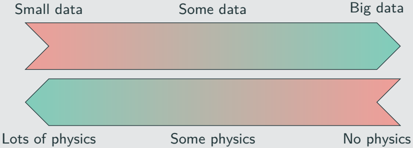
Advantages

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

Drawbacks

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

Hybrid loss



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

Error Estimate & Spectral Bias

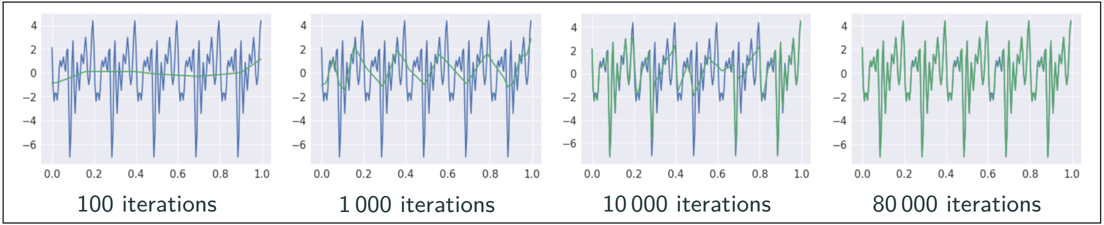
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

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

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

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



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

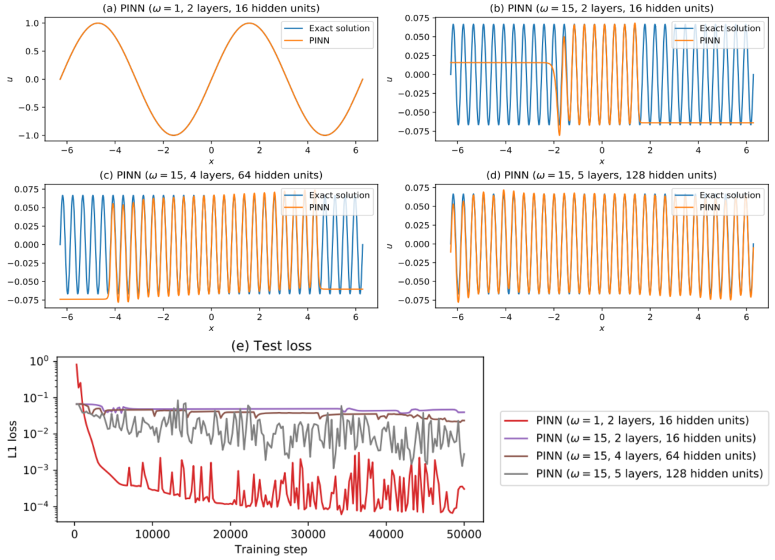
$$u' = \cos(\omega \boldsymbol{x}),$$
$$u(0) = 0,$$

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

Solve

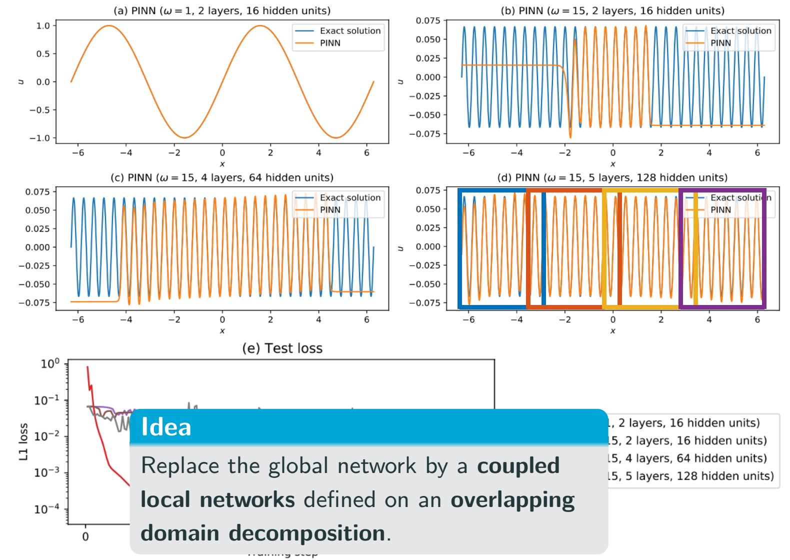
$$u' = \cos(\omega \boldsymbol{x}),$$
$$u(0) = 0,$$

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

Domain Decomposition Methods and Machine Learning – Literature

A **non-exhaustive literature overview**:

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

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

A. Heinlein, A. Klawonn, M. Lanser, J. Weber

Combining machine learning and domain decomposition methods for the solution of partial differential equations — A review

GAMM-Mitteilungen. 2021.

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

A. Klawonn, M. Lanser, J. Weber

Machine learning and domain decomposition methods – a survey

Computational Science and Engineering. 2024

Finite Basis Physics-Informed Neural Networks (FBPINNs)

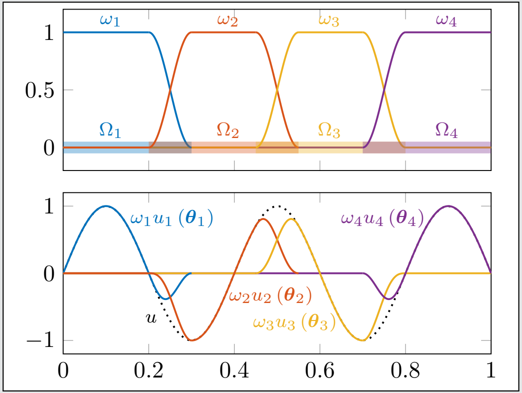
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

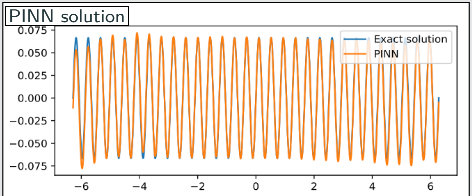
$$u(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J) = \sum_{j=1}^{J} \omega_j u_j\left(\boldsymbol{\theta}_j\right)$$

and the **loss function**

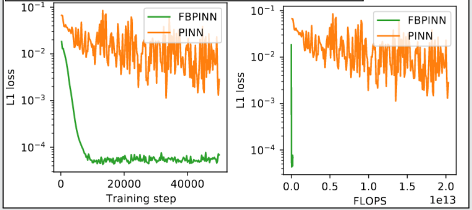
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(n[\sum_{\boldsymbol{x}_i \in \Omega_j} \omega_j u_j](\boldsymbol{x}_i, \boldsymbol{\theta}_j) - f(\boldsymbol{x}_i) \right)^2.$$



1D single-frequency problem



Moseley, Markham, Nissen-Meyer (2023)



Finite Basis Physics-Informed Neural Networks (FBPINNs)

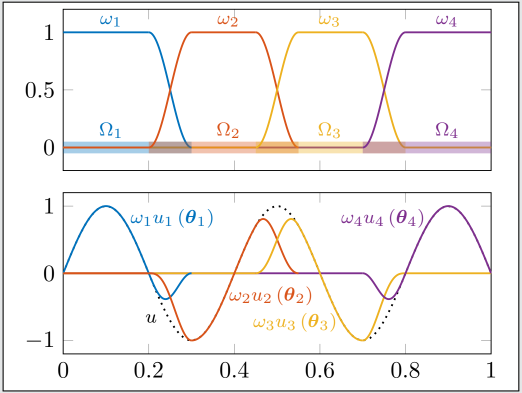
FBPINNs (Moseley, Markham, Nissen-Meyer (2023))

FBPINNs employ the **network architecture**

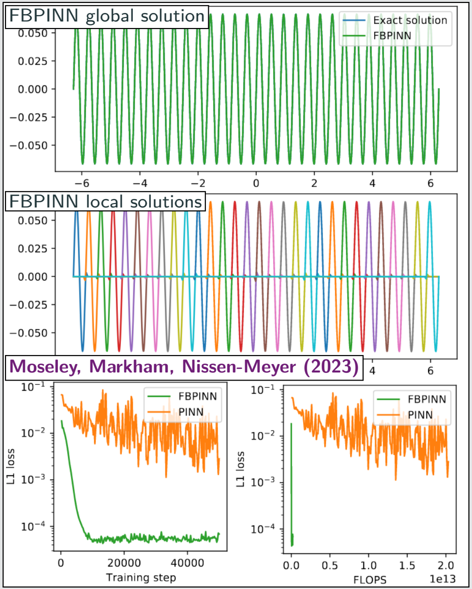
$$u(\theta_1, \ldots, \theta_J) = \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

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



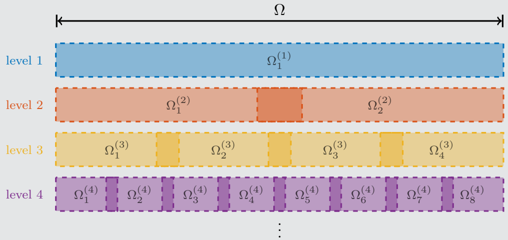
1D single-frequency problem



Multi-Level FBPINNs

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**

$$u\left(\boldsymbol{\theta}_1^{(1)},\ldots,\boldsymbol{\theta}_{J^{(L)}}^{(L)}\right) = \sum\nolimits_{l=1}^{L} \sum\nolimits_{i=1}^{N^{(l)}} \omega_j^{(l)} u_j^{(l)}\left(\boldsymbol{\theta}_j^{(l)}\right)$$

and the **loss function**

$$\mathcal{L} = \frac{1}{N}\sum\nolimits_{i=1}^{N} \left(n[\sum\nolimits_{\boldsymbol{x}_i\in\Omega_j^{(l)}} \omega_j^{(l)} u_j^{(l)}](\boldsymbol{x}_i,\boldsymbol{\theta}_j^{(l)}) - f(\boldsymbol{x}_i)\right)^2.$$

Multi-Frequency Problem

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

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

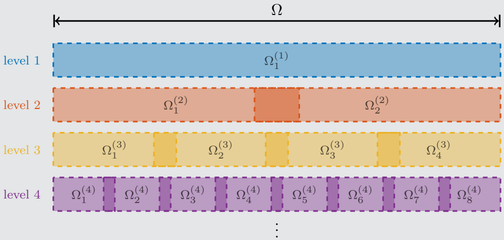
with $\omega_i = 2^i$.

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

Multi-Level FBPINNs

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**

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

and the **loss function**

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

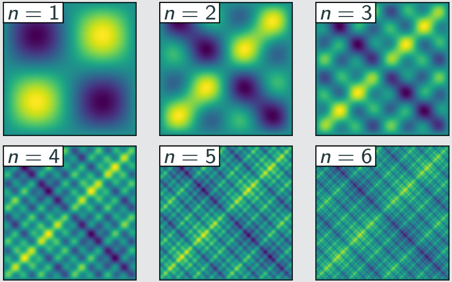
Multi-Frequency Problem

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

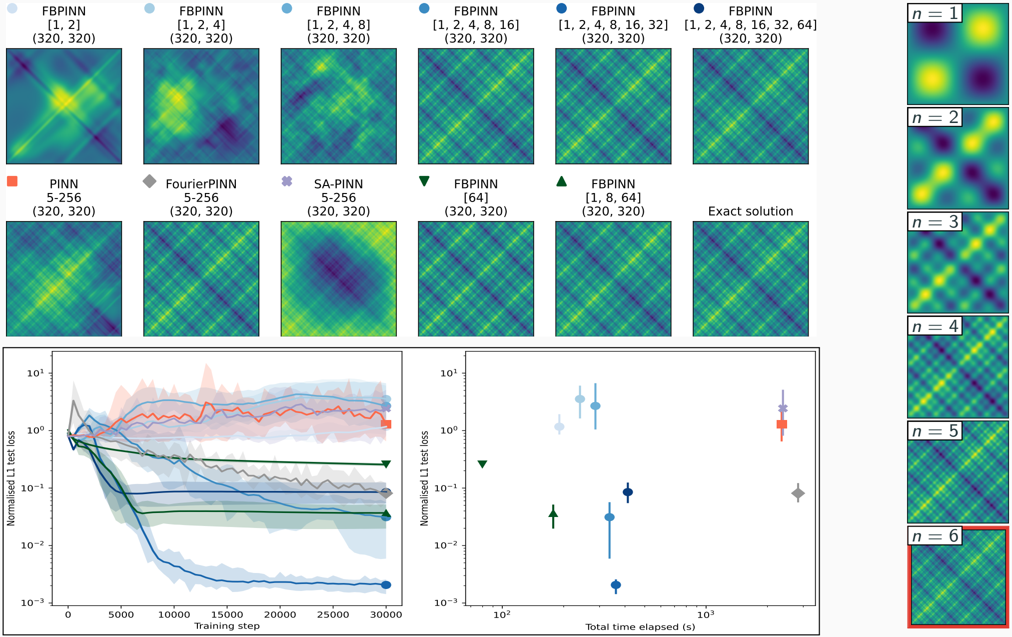
$$-\Delta u = 2 \sum_{i=1}^{n} (\omega_i \pi)^2 \sin(\omega_i \pi x) \sin(\omega_i \pi y) \quad \text{in } \Omega,$$
$$u = 0 \quad \text{on } \partial\Omega,$$

with $\omega_i = 2^i$.

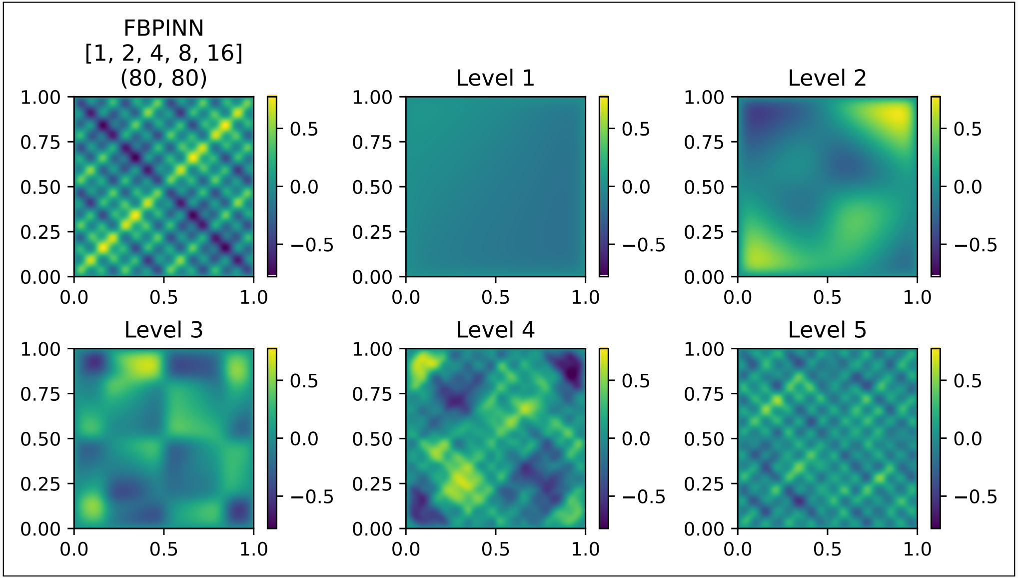
For increasing values of n, we obtain the **analytical solutions**:



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

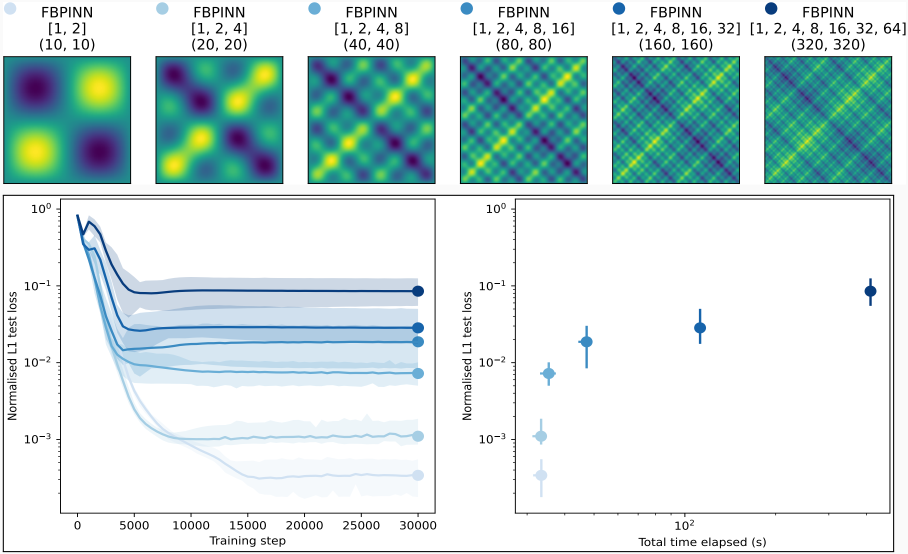


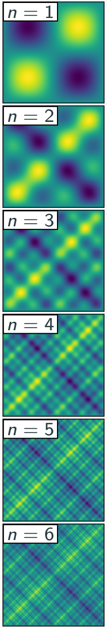
Multi-Frequency Problem – What the FBPINN Learns



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

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





→ Details and results for the Helmholtz equation can be found in **Dolean, Heinlein, Mishra, Moseley (2024)**.

PINNs for Time-Dependent Problems

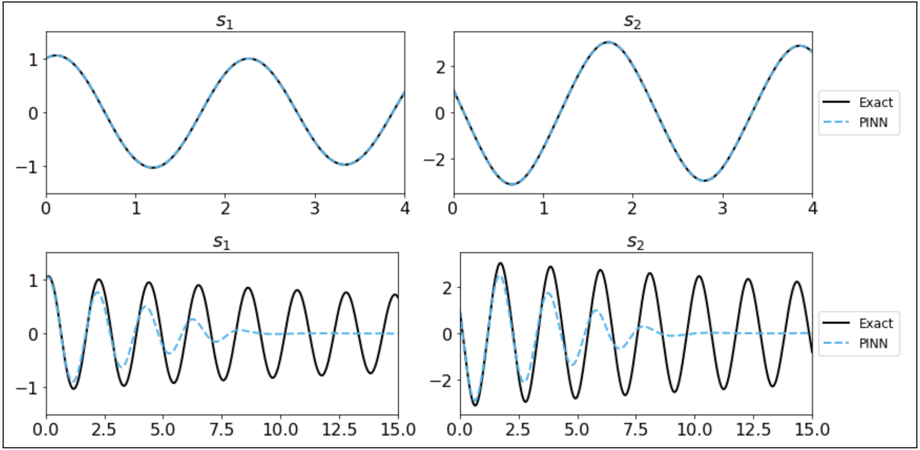
We investigate the performance of PINNs for **time-dependent problems**. Therefore, consider the simple **pedulum problem**:

$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1).$$

Problem parameters

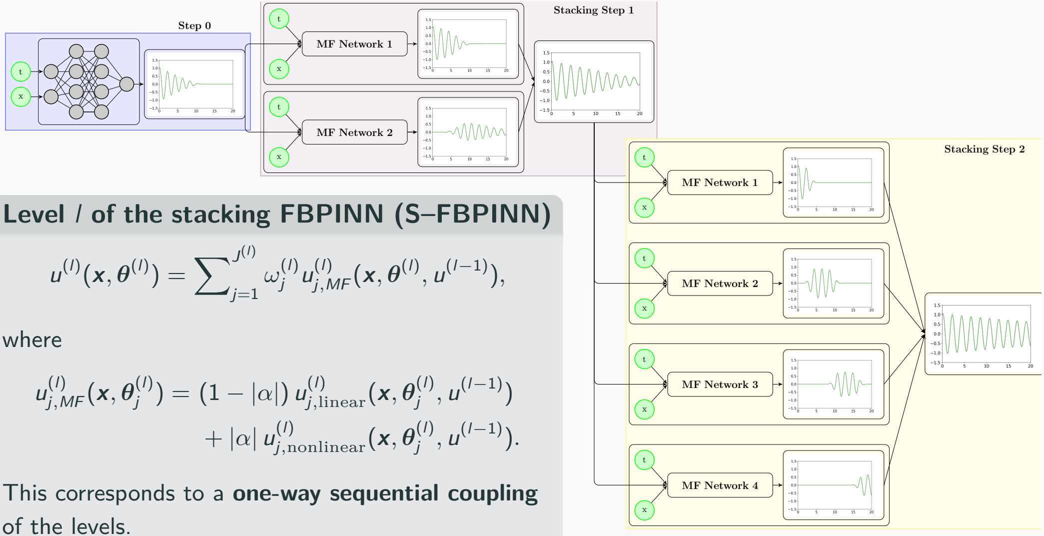
$m = L = 1,\ b = 0.05,$
$g = 9.81$

- **Top:** $T = 4$
- **Bottom:** $T = 20$



Multifidelity Stacking FBPINNs

In **Heinlein, Howard, Beecroft, and Stinis (acc. 2024 / arXiv:2401.07888)**, we **combine stacking multifidelity PINNs with FBPINNs** by using an FBPINN model in each stacking step.



Level l of the stacking FBPINN (S–FBPINN)

$$u^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}^{(l)}) = \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} u_{j,MF}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}^{(l)}, u^{(l-1)}),$$

where

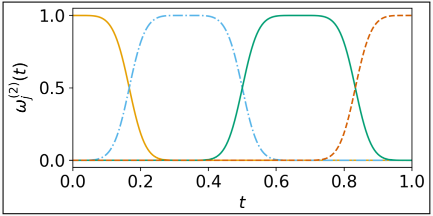
$$u_{j,MF}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}) = (1 - |\alpha|)\, u_{j,\text{linear}}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}, u^{(l-1)}) + |\alpha|\, u_{j,\text{nonlinear}}^{(l)}(\boldsymbol{x}, \boldsymbol{\theta}_j^{(l)}, u^{(l-1)}).$$

This corresponds to a **one-way sequential coupling** of the levels.

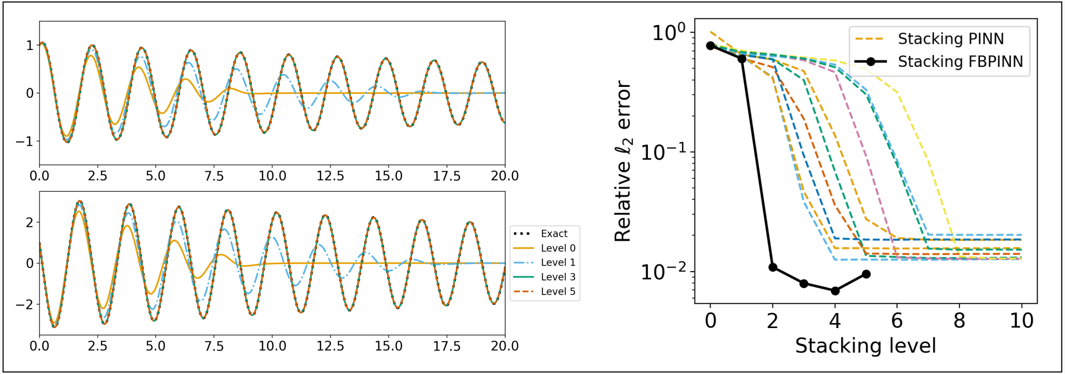
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1)$$

with $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$.



Exemplary partition of unity in time



Multifidelity Stacking FBPINNs – Pendulum Problem

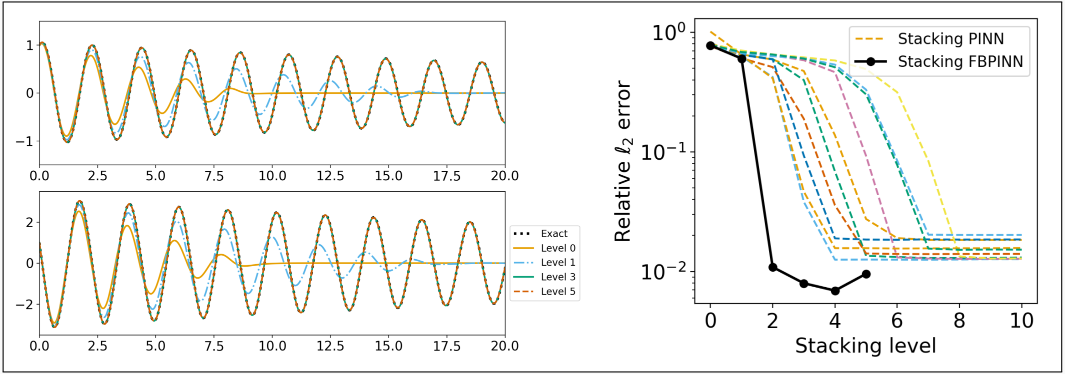
First, we consider a **pedulum problem** and **compare the stacking multifidelity PINN and FBPINN** approaches:

$$\frac{ds_1}{dt} = s_2,$$
$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1)$$

with $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$.

Model details:

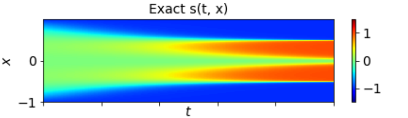
method	arch.	# levels	# params	error
S–PINN	5x50, 1x20	4	63 018	0.0125
S–FBPINN	3x32, 1x 4	2	34 570	0.0074



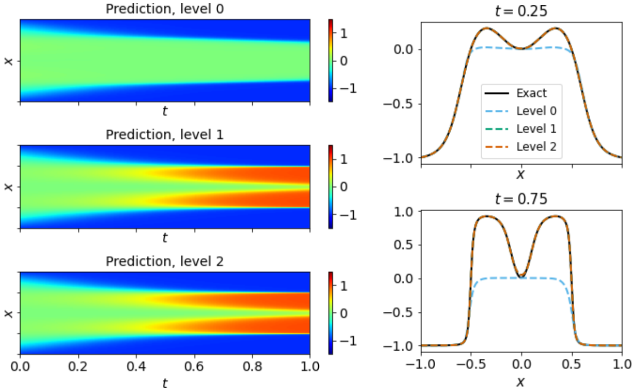
Multifidelity Stacking FBPINNs – Allen–Cahn Equation

Finally, we consider the **Allen–Cahn equation**, describing phase separation in multi-component alloy systems:

$$
\begin{aligned}
s_t - 0.0001 s_{xx} + 5s^3 - 5s &= 0, && t \in (0,1], x \in [-1,1], \\
s(x,0) &= x^2 \cos(\pi x), && x \in [-1,1], \\
s(x,t) &= s(-x,t), && t \in [0,1], x = -1, x = 1, \\
s_x(x,t) &= s_x(-x,t), && t \in [0,1], x = -1, x = 1.
\end{aligned}
$$



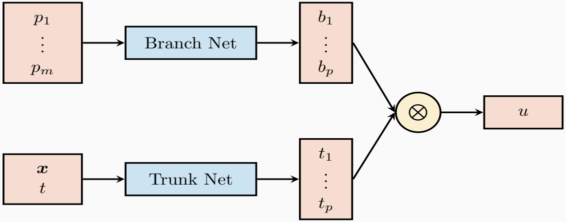
method	arch.	# params	error
PINN	0 levels 6x50	12 951	0.499
S–FBPINN	2 levels 5x32, 1x20	39 627	0.00594



PINN **gets stuck** at **fixed point of the of dynamical system**; cf. **Rohrhofer et al. (2023)**.

Deep Operator Networks (DeepONets / DONs)

Neural operators learn operators between function spaces using neural networks. Here, we learn the **solution operator** of a initial-boundary value problem parametrized with $p_1, \ldots, p_m$ using **DeepONets** as introduced in **Lu et al. (2021)**.



Single-layer case

The DeepONet architecture is based on the **single-layer case** analyzed in **Chen and Chen (1995)**. In particular, the authors show **universal approximation properties for continuous operators**.

The architecture is based on the following ansatz for presenting the parametrized solution

$$u_{(p_1,\ldots,p_m)}(\boldsymbol{x}, t) = \sum\nolimits_{i=1}^{p} \underbrace{b_i(p_1, \ldots, p_m)}_{\text{branch}} \cdot \underbrace{t_i(\boldsymbol{x}, t)}_{\text{trunk}}$$

Physics-informed DeepONets

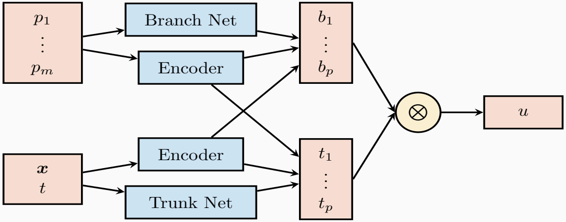
DeepONets are **compatible with the PINN approach** but **physics-informed DeepONets (PI-DeepONets)** are challenging to train.

Other operator learning approaches

- **FNOs**: **Li et al. (2021)**
- **PCA-Net**: **Bhattacharya et al. (2021)**
- **Random features**: **Nelsen and Stuart (2021)**
- **CNOs**: **Raonić et al. (2023)**

Deep Operator Networks (DeepONets / DONs)

Neural operators learn operators between function spaces using neural networks. Here, we learn the **solution operator** of a initial-boundary value problem parametrized with $p_1, \ldots, p_m$ using **DeepONets** as introduced in **Lu et al. (2021)**.



Modified architecture

In our numerical experiments, we employ the **modified DeepONet architecture** introduced in **Wang, Wang, and Perdikaris (2022)**.

The architecture is based on the following ansatz for presenting the parametrized solution

$$u_{(p_1,\ldots,p_m)}(\boldsymbol{x}, t) = \sum_{i=1}^{p} \underbrace{b_i(p_1, \ldots, p_m)}_{\text{branch}} \cdot \underbrace{t_i(\boldsymbol{x}, t)}_{\text{trunk}}$$

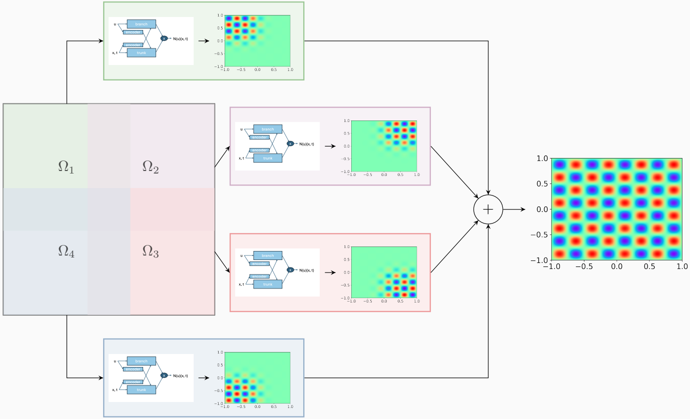
Physics-informed DeepONets

DeepONets are **compatible with the PINN approach** but **physics-informed DeepONets (PI-DeepONets)** are challenging to train.

Other operator learning approaches

- **FNOs**: **Li et al. (2021)**
- **PCA-Net**: **Bhattacharya et al. (2021)**
- **Random features**: **Nelsen and Stuart (2021)**
- **CNOs**: **Raonić et al. (2023)**

Finite Basis DeepONets (FBDONs)



Howard, Heinlein, Stinis (in prep.)

Variants:

Shared-trunk FBDONs (ST-FBDONs)

The trunk net learns spatio-temporal basis functions. In ST-FBDONs, we use the **same trunk network for all subdomains**.

Stacking FBDONs

Combination of the **stacking multifidelity approach** with FBDONs.

Heinlein, Howard, Beecroft, Stinis (acc. 2024/arXiv:2401.07888)

Wave equation

$$\frac{d^2 s}{dt^2} = 2\frac{d^2 s}{dx^2}, \qquad (x,t) \in [0,1]^2$$

$$s_t(x,0) = 0, x \in [0,1], \quad s(0,t) = s(1,t) = 0,$$

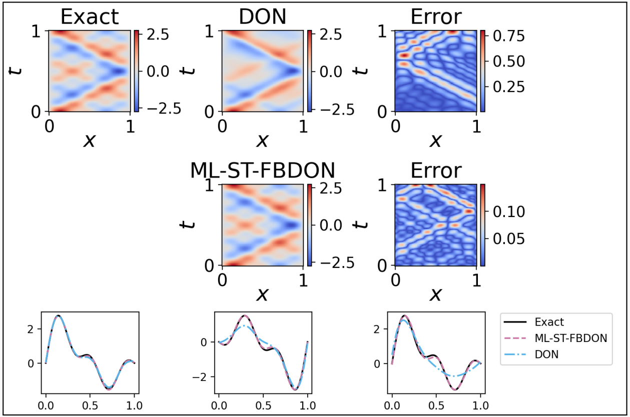
Solution: $s(x,t) = \sum_{n=1}^{5} b_n \sin(n\pi x)\cos(n\pi\sqrt{2}t)$

Parametrization

Initial conditions for s parametrized by $b = (b_1, \ldots, b_5)$ (normally distributed):

$$s(x,0) = \sum_{n=1}^{5} b_n \sin(n\pi x) \quad x \in [0,1]$$

Training on 1 000 random configurations.

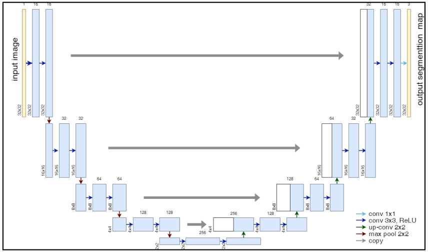


Mean rel. l_2 error on 100 config.	
DeepONet	0.30 ± 0.11
ML-ST-FBDON ([1, 4, 8, 16] subd.)	0.05 ± 0.03
ML-FBDON ([1, 4, 8, 16] subd.)	0.08 ± 0.04

→ Sharing the trunk network does not only save in the number of parameters but even yields **better performance**

Cf. **Howard, Heinlein, Stinis (in prep.)**

Domain Decomposition-Based U-Net Architecture

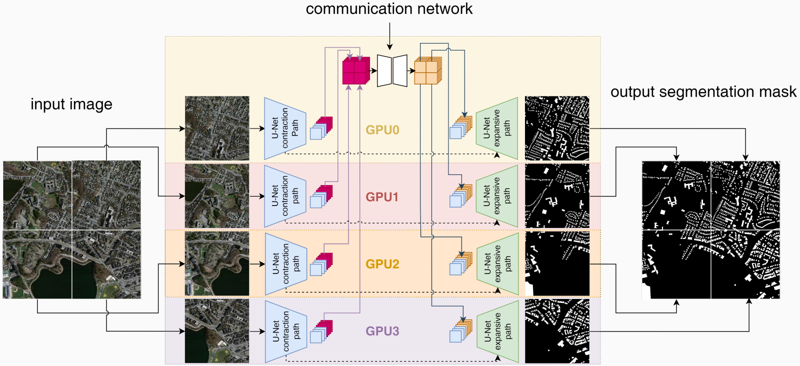


name	mem. feature maps		mem. weights	
	# of values	MB	# of values	MB
input block	268 M	**1 024.0**	38 848	**0.148**
encoder blocks	314 M	**1 320**	18 M	**72**
decoder blocks	754 M	**3880**	12 M	**47**
output block	3.1 M	**12.0**	195	**0.001**

Most memory in the **U-Net** is used by **feature maps**, **not weights**

→ **Decompose feature maps** to **distribute memory consumption**.

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



4TU.AMI – SRI "Bridging Numerical Analysis and Machine Learning"

UNIVERSITY OF TWENTE.


Christoph Brune


Silke Glas


Matthias Schlottbom

TU Delft Delft University of Technology


Alexander Heinlein


Matthias Möller


Deepesh Toshniwal

4TU.AMI

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY


Victorita Dolean


Olga Mula


Wil Schilders


Jemima Tabeart


Karen Veroy-Grepl

WAGENINGEN UNIVERSITY & RESEARCH


Xiaodong Cheng

CWI Research Semester: Synergies in Numerical Linear Algebra and Machine Learning

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

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

Centrum Wiskunde & Informatica

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

FROSch

- FROSch is based on the **Schwarz framework** and **energy-minimizing coarse spaces**, which provide **numerical scalability** using **only algebraic information** for a **variety of applications**

Multilevel neural network architectures

- **Domain decomposition-based architectures** improve the scalability of PINNs to **large domains / high frequencies**, keeping the complexity of the local networks low.
- As classical domain decomposition methods, **one-level FBPINNs** are not scalable to large numbers of subdomains; **multilevel FBPINNs** enable scalability.
- The multilevel FBPINN approach can also be **extended to operator learning**.

Thank you for your attention!

Topical Activity Group

Scientific Machine Learning