Localization via Partition of Unity Functions

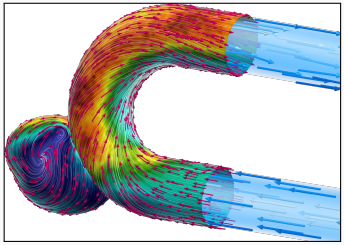
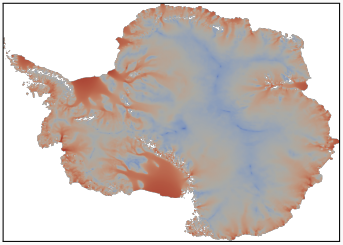
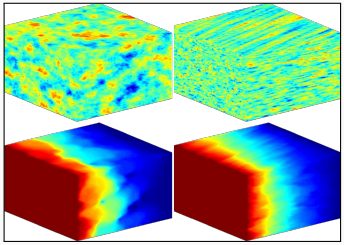
Coarse Spaces for Domain Decomposition Preconditioners and Multilevel Architectures for Neural Networks

Alexander Heinlein[1]

KAUST International Research Conference on Multi-Grid and Multi-Scale Methods in Computational Science, IMG 2025, KAUST, Saudi Arabia, February 3-5, 2025

[1]Delft University of Technology

Numerical Analysis and Machine Learning

Numerical methods

Based on physical models

+ Robust and generalizable
− Require availability of mathematical models

Machine learning models

Driven by data

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

Scientific machine learning (SciML)

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

numerical methods **improve** machine learning techniques

machine learning techniques **assist** numerical methods

Outline

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

   Based on joint work with

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

2. Multilevel domain decomposition-based architectures for physics-informed neural networks

   Based on joint work with

Damien Beecroft	(University of Washington)
Victorita Dolean	(Eindhoven University of Technology)
Amanda A. Howard and **Panos Stinis**	(Pacific Northwest National Laboratory)
Ben Moseley	(Imperial College London)
Siddhartha Mishra	(ETH Zürich)

The FROSch Package – Algebraic and Parallel Schwarz Preconditioners in Trilinos

Solvers for Partial Different Equations

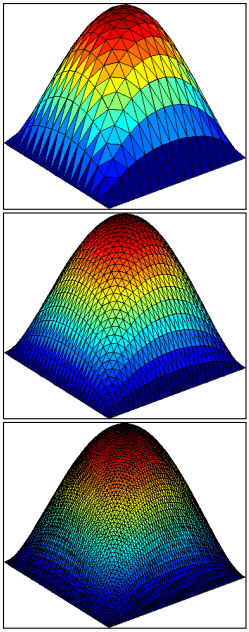
Consider a **diffusion model problem**:

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

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

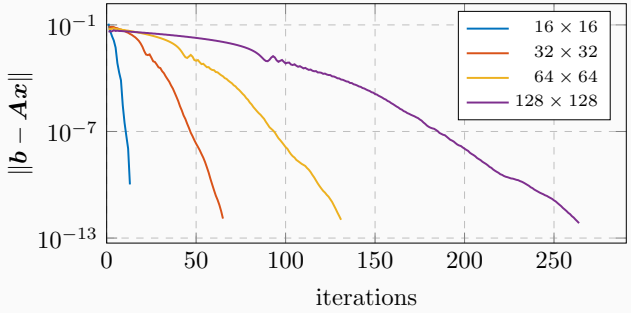
Iterative solvers are efficient for solving **sparse systems**, however, the **convergence rate depends on the spectral properties of K.**

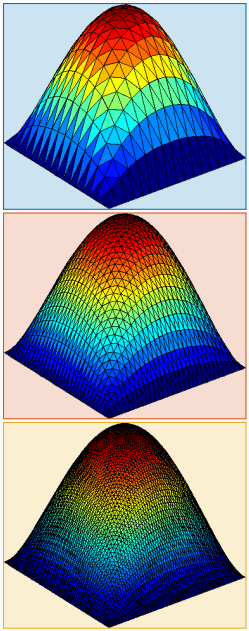
Solvers for Partial Different Equations

Consider a **diffusion model problem**:

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

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



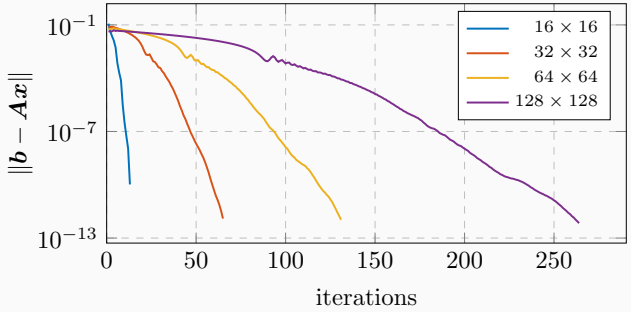


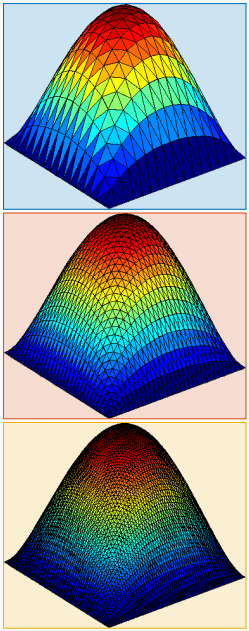
Solvers for Partial Different Equations

Consider a **diffusion model problem**:

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

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





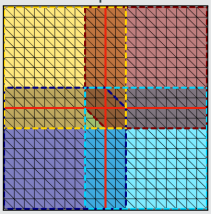
$\Rightarrow$ Introduce a preconditioner $\boldsymbol{M}^{-1} \approx \boldsymbol{K}^{-1}$ to **improve convergence**:

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

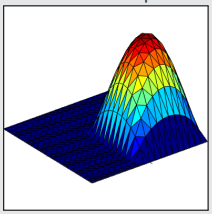
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



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

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

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

Condition number estimate:

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

with subdomain size H and overlap width δ.

Lagrangian coarse space

Coarse triangulation

Coarse solution

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

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

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

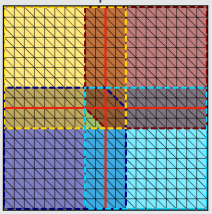
Condition number estimate:

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

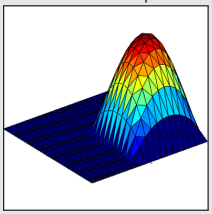
Two-Level Schwarz Preconditioners

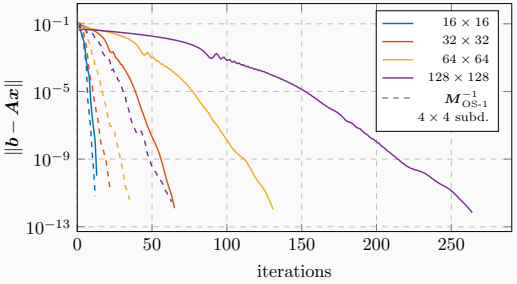
One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem





Lagrangian coarse space

Coarse triangulation

Coarse solution

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

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

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

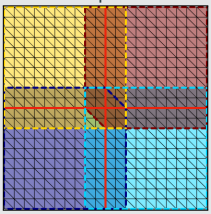
Condition number estimate:

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

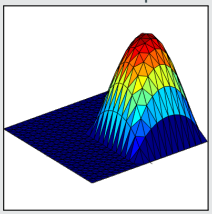
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$



Solution of local problem



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

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

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

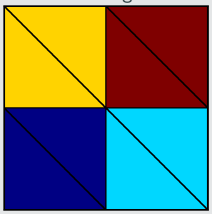
Condition number estimate:

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

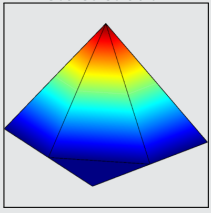
with subdomain size H and overlap width δ.

Lagrangian coarse space

Coarse triangulation



Coarse solution



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

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

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

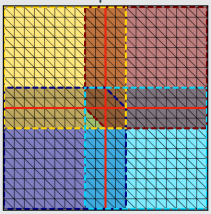
Condition number estimate:

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

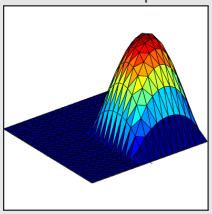
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$

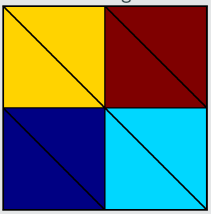


Solution of local problem

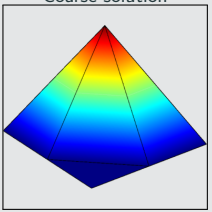


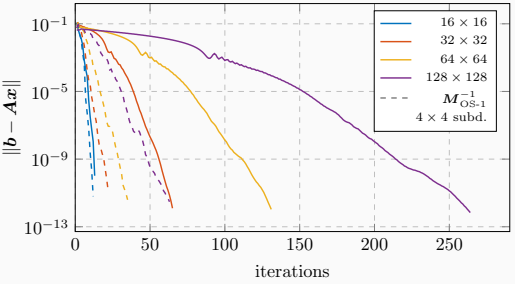
Lagrangian coarse space

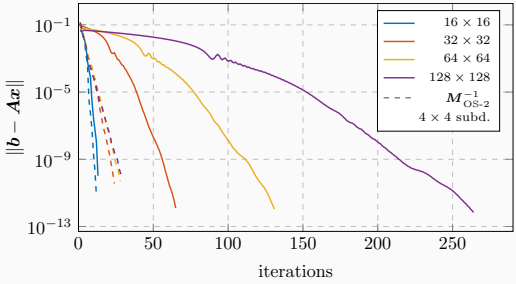
Coarse triangulation



Coarse solution



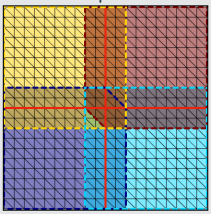




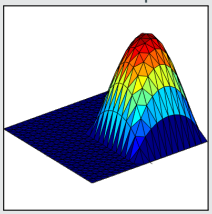
Two-Level Schwarz Preconditioners

One-level Schwarz preconditioner

Overlap $\delta = 1h$

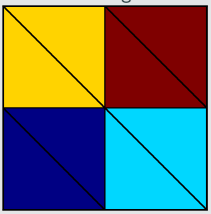


Solution of local problem

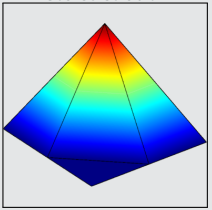


Lagrangian coarse space

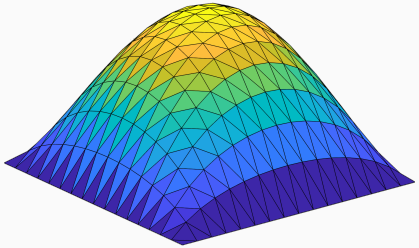
Coarse triangulation

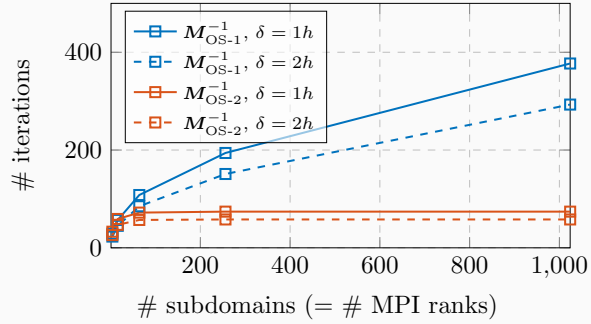


Coarse solution

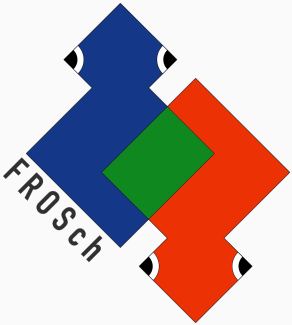


Diffusion model problem in two dimensions,
$H/h = 100$





FROSch (Fast and Robust Overlapping Schwarz) Framework in Trilinos



TU Delft

Sandia National Laboratories

Universität zu Köln

TU Bergakademie Freiberg

TUBAF
Die Ressourcenuniversität.
Seit 1765.

Software

- Object-oriented C++ domain decomposition solver framework with MPI-based distributed memory parallelization
- Part of Trilinos with support for both parallel linear algebra packages Epetra and Tpetra
- Node-level parallelization and performance portability on CPU and GPU architectures through Kokkos and KokkosKernels
- Accessible through unified Trilinos solver interface Stratimikos

Methodology

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

Team (active)

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

Partition of Unity

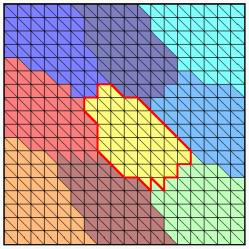
The **energy-minimizing extension** $v_i = H_{\partial\Omega_i \to \Omega_i}(v_{i,\partial\Omega_i})$ solves

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

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

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

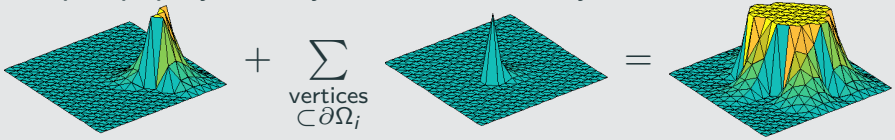
$$\sum_i \varphi_i = \mathbb{1}_{\partial\Omega_i} \Rightarrow \sum_i E_{\partial\Omega_i \to \Omega_i}(\varphi_i) = \mathbb{1}_{\Omega_i}$$



Null space property

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

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



Algebraicity of the energy-minimizing extension

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

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

Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

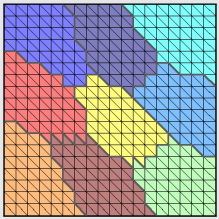
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

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

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

Nonoverlapping DD



Overlapping domain decomposition

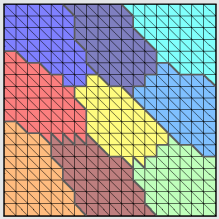
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

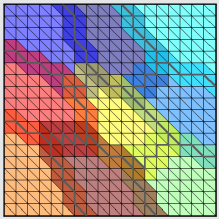
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

Nonoverlapping DD



Overlap $\delta = 1h$



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

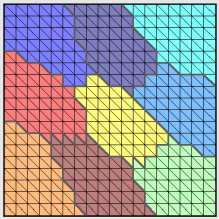
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

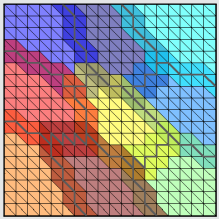
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

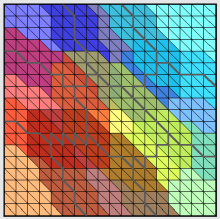
Nonoverlapping DD



Overlap $\delta = 1h$



Overlap $\delta = 2h$



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

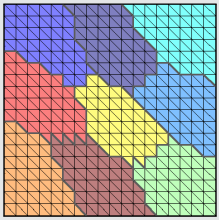
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

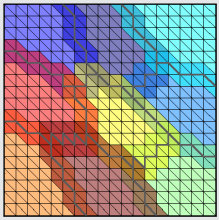
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

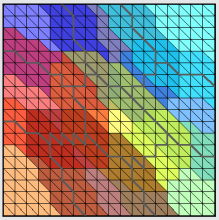
Nonoverlapping DD



Overlap $\delta = 1h$

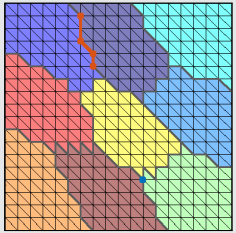


Overlap $\delta = 2h$



Coarse space

1. Interface components



Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

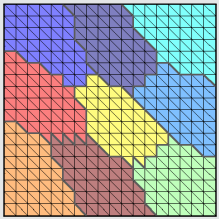
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

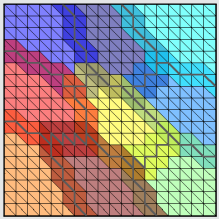
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

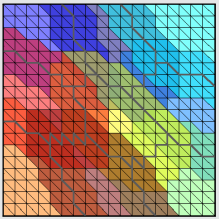
Nonoverlapping DD



Overlap $\delta = 1h$

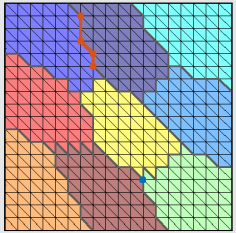


Overlap $\delta = 2h$

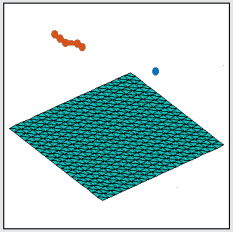


Coarse space

1. Interface components



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



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

Algorithmic Framework for FROSch Preconditioners

Overlapping domain decomposition

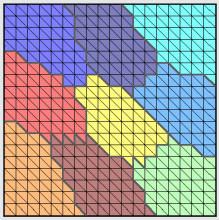
The **overlapping subdomains** are constructed by **recursively adding layers of elements** via the sparsity pattern of $\boldsymbol{K}$.

The corresponding matrices

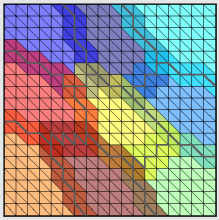
$$\boldsymbol{K}_i = \boldsymbol{R}_i \boldsymbol{K} \boldsymbol{R}_i^T$$

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

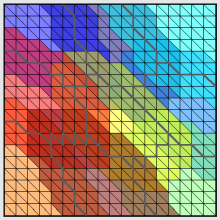
Nonoverlapping DD



Overlap $\delta = 1h$

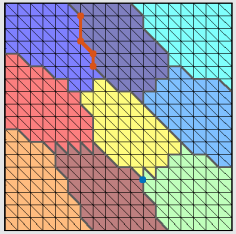


Overlap $\delta = 2h$

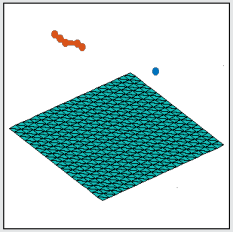


Coarse space

1. Interface components

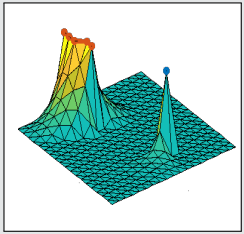


2. Interface basis (partition of unity $\times$ null space)



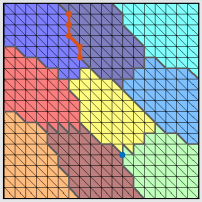
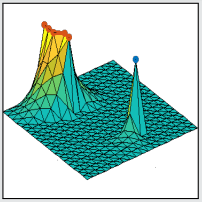
For **scalar elliptic problems**, the **null space** consists only of **constant functions**.

3. Extension



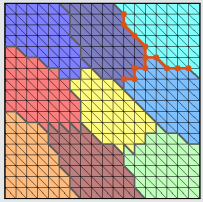
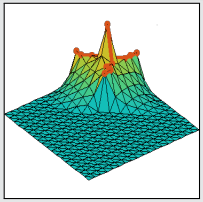
Examples of FROSch Coarse Spaces

GDSW (Generalized Dryja–Smith–Widlund)

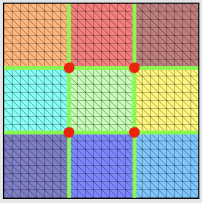
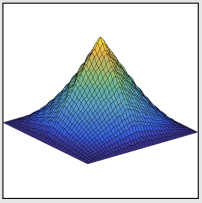
- Dohrmann, Klawonn, Widlund (2008)
- Dohrmann, Widlund (2009, 2010, 2012)

RGDSW (Reduced dimension GDSW)

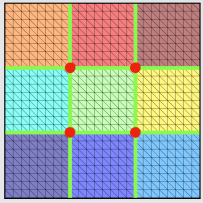
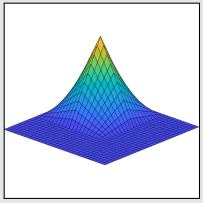
- Dohrmann, Widlund (2017)
- H., Klawonn, Knepper, Rheinbach, Widlund (2022)

MsFEM (Multiscale Finite Element Method)

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

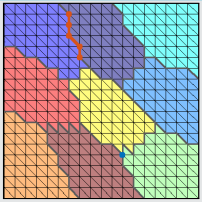
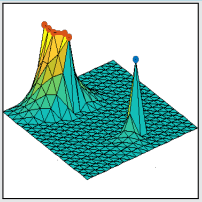
Q1 Lagrangian / piecewise bilinear

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

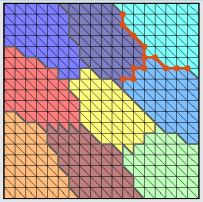
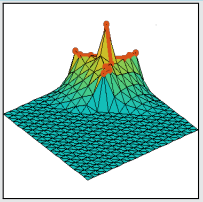
Examples of FROSch Coarse Spaces

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)

For elliptic model problems, the **condition number of the (R)GDSW two-level Schwarz operator** is bounded by

$$\kappa\left(\boldsymbol{M}_{(\mathrm{R})\mathrm{GDSW}}^{-1}\boldsymbol{K}\right) \leq C\left(1+\frac{H}{\delta}\right)\left(1+\log\left(\frac{H}{h}\right)\right)^{\alpha},$$

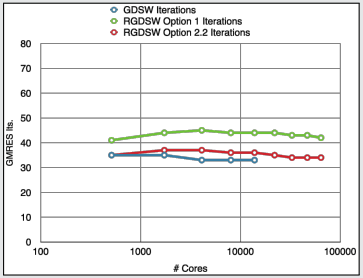
where

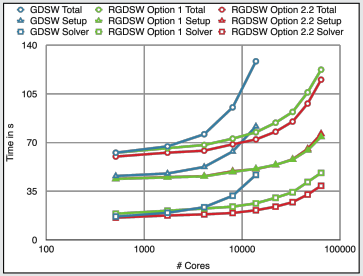
C	constant (does not depend on h, H, or δ),
H	subdomain diameter,
h	element size,
δ	width of the overlap,
$\alpha \in \{0,1,2\}$	power (depends on problem dimension, regularity of the subdomains, and variant of the algorithm).

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).





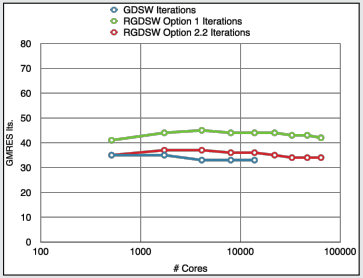
Two-level vs three-level GDSW

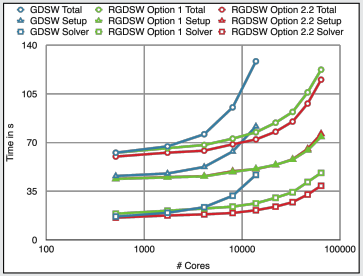
Heinlein, Klawonn, Rheinbach, Röver (2019, 2020).

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

GDSW vs RGDSW (reduced dimension)

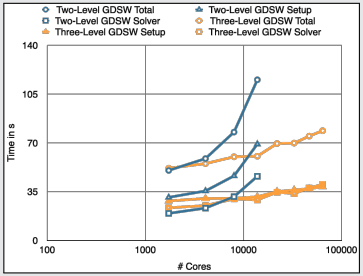
Heinlein, Klawonn, Rheinbach, Widlund (2019).





Two-level vs three-level GDSW

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



Monolithic (R)GDSW Preconditioners for CFD Simulations

Consider the discrete saddle point problem

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

Monolithic GDSW preconditioner

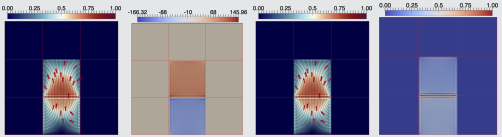
We construct a **monolithic GDSW preconditioner**

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

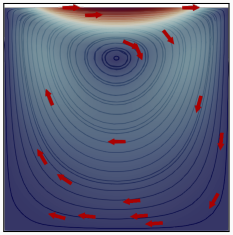
with block matrices $\mathcal{A}_0 = \phi^\top \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^\top$, local pressure projections $\overline{\mathcal{P}}_i$, and

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

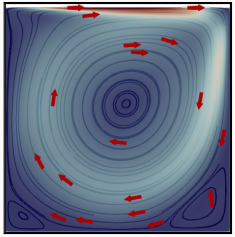
Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



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



Stokes flow

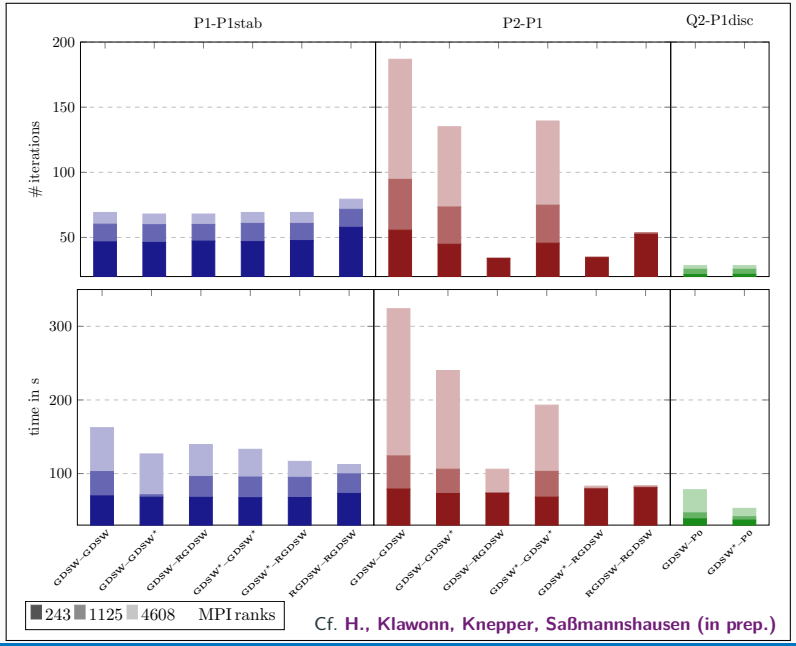


Navier–Stokes flow

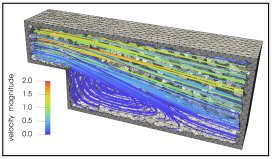
Related work:

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

Balancing the Velocity and Pressure Coarse Spaces

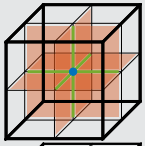


Cf. **H., Klawonn, Knepper, Saßmannshausen (in prep.)**

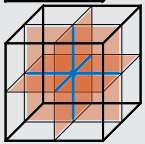


Varying the POU

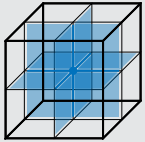
GDSW:



GDSW*:

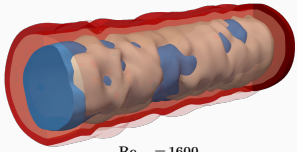


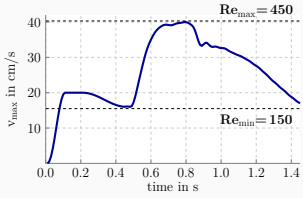
RGDSW:

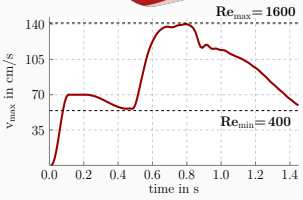


Results for Blood Flow Simulations

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





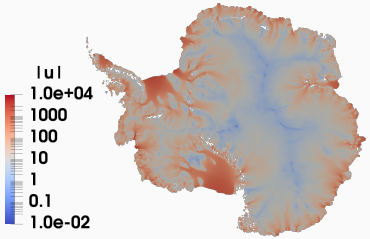


prec.	# MPI ranks	16	64	256
Monolithic RGDSW (FROSch)	avg. #its.	33	31	30
	setup	4 825 s	1 422 s	701 s
	solve	3 198 s	1 004 s	463 s
	total	8 023 s	2 426 s	**1 164 s**
SIMPLE RGDSW (Teko & FROSch)	avg. #its.	82	82	87
	setup	3 046 s	824 s	428 s
	solve	4 679 s	1 533 s	801 s
	total	**7 725 s**	**2 357 s**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

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

FROSch Preconditioners for Land Ice Simulations

https://github.com/SNLComputation/Albany



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

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

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

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

Computations performed on Cori (NERSC).

Heinlein, Perego, Rajamanickam (2022)

Linear & Nonlinear Preconditioning

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

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

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

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

Linear preconditioning

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

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

Goal:

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

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

Nonlinear preconditioning

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

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

Goals:

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

Linear & Nonlinear Preconditioning

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

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

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

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

Linear preconditioning

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

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

Goal:

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

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

Nonlinear preconditioning

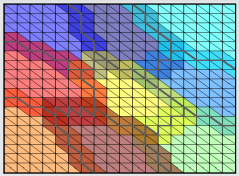
In nonlinear preconditioning, we **improve the convergence speed of the nonlinear solver** by constructing a **nonlinear operator** G and solve the nonlinear system

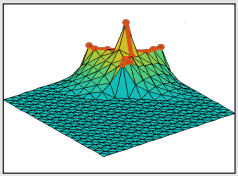
$$(\boldsymbol{G} \circ \boldsymbol{F})(\boldsymbol{u}) = 0.$$

Goals:

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

ASPEN & ASPIN methods





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

$$F(u) = 0$$

is reformulated as the **equivalent problem**

$$\sum_{i=1}^{N} P_i T_i(u) = 0$$

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

$$R_i F(u - P_i T_i(u)) = 0.$$

R_i restriction; P_i prolongation.

Results for p-Laplacian model problem

p-Laplacian model problem

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

with $\alpha \Delta_p u := \operatorname{div}(\alpha |\nabla u|^{p-2} \nabla u)$.

$p = 4$; $H/h = 16$; overlap $\delta = 1$

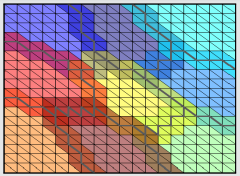
N	RASPEN solver	nonlin. outer it.	nonlin. inner it. (avg.)	lin. GMRES it. (sum)
9	NK-RAS	18	-	272
	RASPEN	5	25.2	89
25	NK-RAS	19	-	488
	RASPEN	6	28.3	172
49	NK-RAS	20	-	691
	RASPEN	6	27.3	232

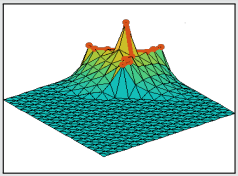
⇒ **Improved nonlinear convergence**, but **no scalability** in the linear iterations.

Cf. **Heinlein, Lanser (2020)**.

Nonlinear Schwarz Methods

Two-level ASPEN & ASPIN methods





In **two-level additive Schwarz preconditioned (in)exact Newton (ASPEN/ASPIN)** in **Heinlein, Lanser (2020)**, we consider

$$R_0^T T_0(u) + \sum_{i=1}^{N} P_i T_i(u) = 0$$

with

- corrections $T_i(u)$ as in the one-level case and
- the correction $T_0(u)$ given by a **nonlinear problem** in the **coarse space**

$$R_0 F(u - R_0^T T_0(u)) = 0;$$

via a **Galerkin projection**.

Results for p-Laplacian model problem

1-lvl One-level RASPEN
2-lvl A **Additive** two-level RASPEN
2-lvl M **Multiplicative** two-level RASPEN

RGDSW coarse space: extensions computed using the tangent from the first linearization.

$p = 4$; $H/h = 16$; overlap $\delta = 1$

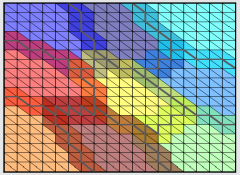
N	RASPEN solver	nonlin. outer it.	nonlin. inner it. (avg.)	nonlin. coarse it.	lin. GMRES it. (sum)
9	1-lvl	5	25.2	-	89
	2-lvl A	6	33.4	27	93
	2-lvl M	4	17.1	29	52
49	1-lvl	6	27.3	-	232
	2-lvl A	6	29.2	28	137
	2-lvl M	4	12.6	29	80

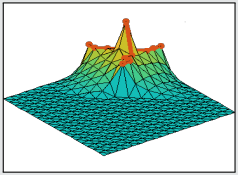
⇒ **Improved nonlinear convergence and scalability.**

Cf. **Heinlein, Lanser (2020)**.

Nonlinear Schwarz Methods

Three-level ASPEN & ASPIN methods





In **three-level additive Schwarz preconditioned (in)exact Newton (ASPEN/ASPIN)**, we build the preconditioner **recursively** and consider

$$P_{00}^T T_{00}(u) + \sum_{j=1}^{M} P_{j0}^T T_{j0}(u) + \sum_{i=1}^{N} P_i T_i(u) = 0$$

with

level	correction	nonlinear problem
1	$T_0(u)$	$R_i F(u - P_i^T T_0(u)) = 0$
2	$T_{j0}(u)$	$R_{j0} F(u - P_{j0} T_{j0}(u)) = 0$
3	$T_{00}(u)$	$R_{j0} F(u - P_{00} T_{00}(u)) = 0$

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

Results for p-Laplacian model problem

	additive		multiplicative	
	2-level	3-level	2-level	3-level
linear	M_A^{-1}	M_{AA}^{-1}	M_M^{-1}	M_{MM}^{-1}
nonlinear	$\mathcal{F}_A$	$\mathcal{F}_{AA}$	$\mathcal{F}_M$	$\mathcal{F}_{MM}$

$p = 4$; 16^2 subd.; 4^2 subr.; $H/h = 8$; overlap $\delta = 1$

RASPEN solver	nonlin. outer it.	nonlin. subd. it. (avg./min/max)	nonlin. subr. it.	lin. coarse it.	lin. GMRES it. (sum)
$\mathcal{F}_A$	**6**	23/16/46	-	36	90
$\mathcal{F}_M$	**5**	11/10/25	-	34	60
M_A^{-1}	24	-	-	-	381
M_M^{-1}	24	-	-	-	335
$\mathcal{F}_{AA}$	**6**	25/17/47	25/20/39	35	108
$\mathcal{F}_{MM}$	**7**	17/15/40	19/16/33	34	92
M_{AA}^{-1}	24	-	-	-	396
M_{MM}^{-1}	24	-	-	-	338

Multilevel domain decomposition-based architectures for physics-informed neural networks

Physics-Informed Neural Networks (PINNs)

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

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

PINNs use a **hybrid loss function**:

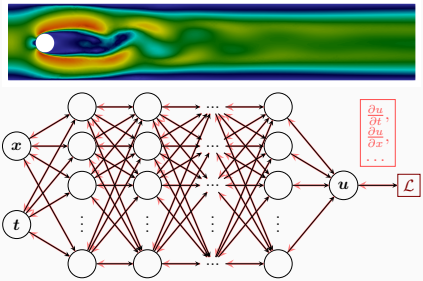
$$\mathcal{L}(\boldsymbol{\theta}) = \omega_{\text{data}}\mathcal{L}_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{PDE}}\mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}),$$

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

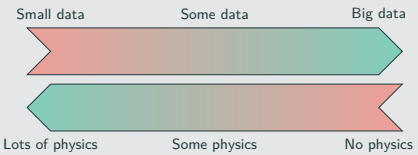
$$\mathcal{L}_{\text{data}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(u(\hat{\boldsymbol{x}}_i, \boldsymbol{\theta}) - u_i\right)^2,$$

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

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



Hybrid loss



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

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

Error Estimate & Spectral Bias

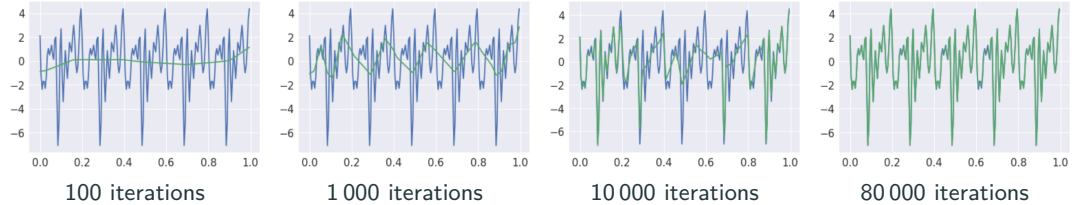
Estimate of the generalization error (Mishra and Molinaro (2022))

The generalization error (or total error) satisfies

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

- $\mathcal{E}_G = \mathcal{E}_G(\boldsymbol{X}, \boldsymbol{\theta}) := \|\mathbf{u} - \mathbf{u}^*\|_V$ **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), . . .**

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)

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

FBPINNs use the **network architecture**

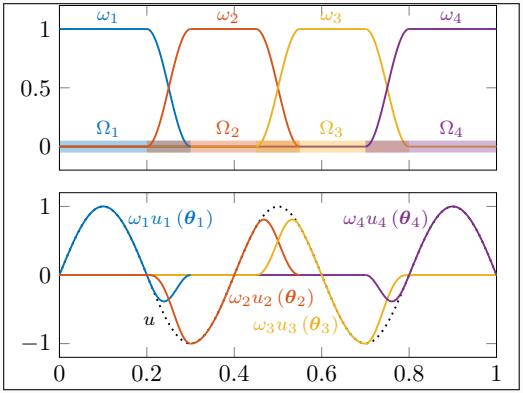
$$u(\theta_1, \dots, \theta_J) = \mathcal{C} \sum_{j=1}^{J} \omega_j u_j(\theta_j)$$

and the **loss function**

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

Here:

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



Hard enf. of boundary conditions

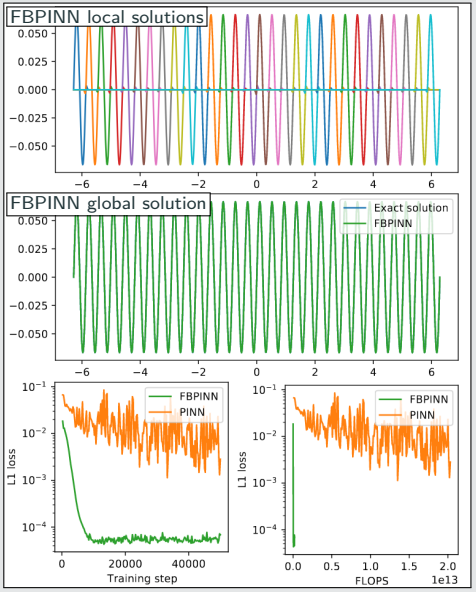
Loss function

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

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

Numerical Results for FBPINNs

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



Scalability of FBPINNs

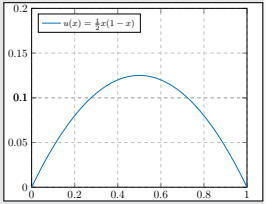
Consider the **simple boundary value problem**

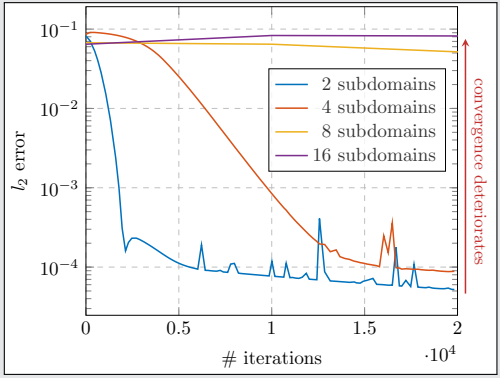
$$-u'' = 1 \text{ in } [0,1],$$

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

which has the **solution**

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

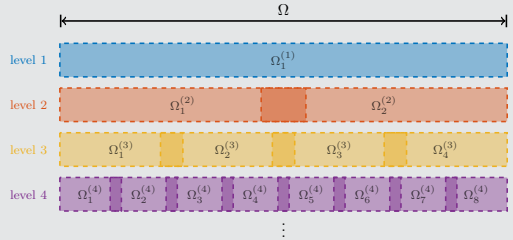




Numerical Results for FBPINNs

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**

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

and the **loss function**

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\Big(\eta[\sum_{\boldsymbol{x}_i\in\Omega_j^{(l)}} \omega_j^{(l)} u_j^{(l)}](\boldsymbol{x}_i,\theta_j^{(l)}) - f(\boldsymbol{x}_i)\Big)^2.$$

Multi-Frequency Problem

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

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

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

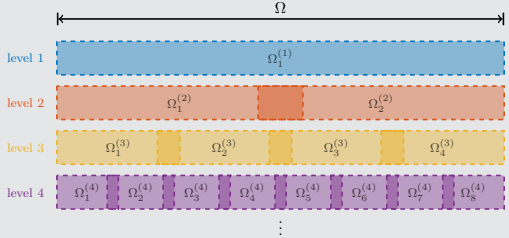
with $\omega_i = 2^i$.

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

Numerical Results for 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**

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

and the **loss function**

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

Multi-Frequency Problem

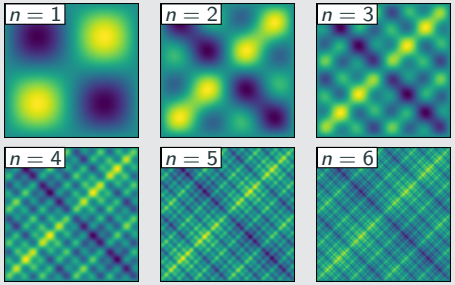
Let us now consider the **two-dimensional multi-frequency Laplace boundary value problem**

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

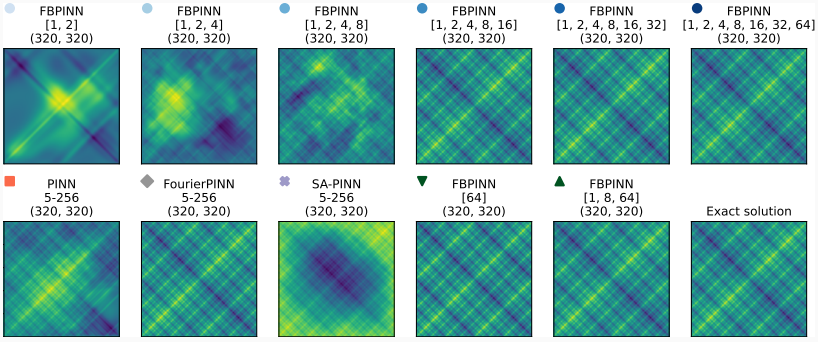
$$u = 0 \quad \text{on } \partial\Omega,$$

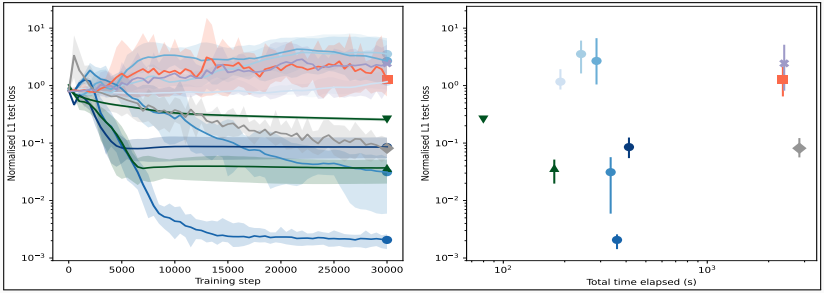
with $\omega_i = 2^i$.

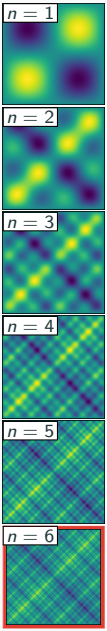
For increasing values of n, we obtain the **analytical solutions**:



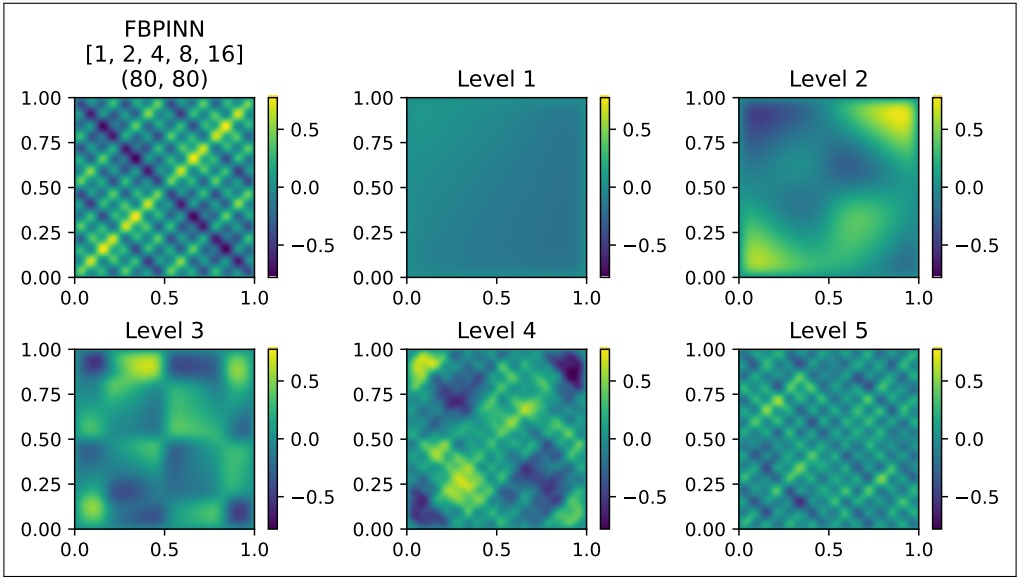
Multi-Level FBPINNs for a Multi-Frequency Problem – Strong Scaling





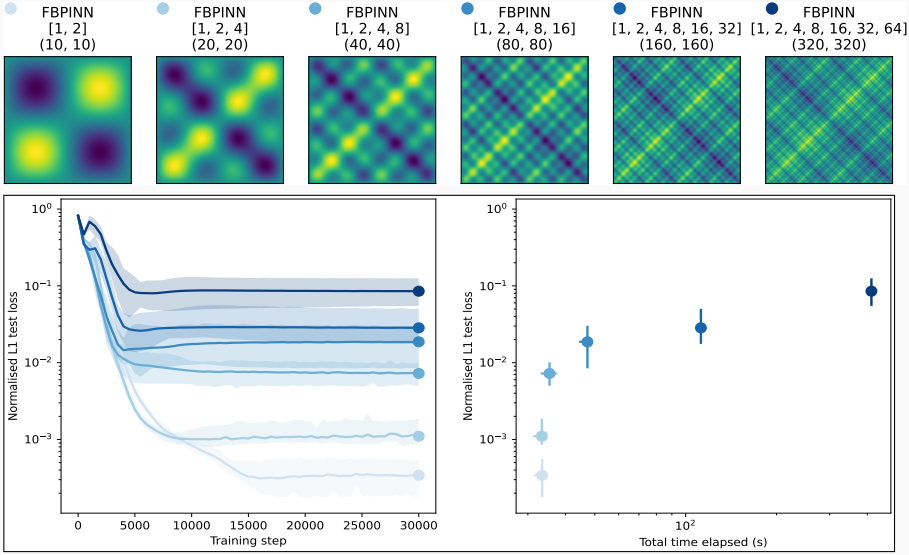


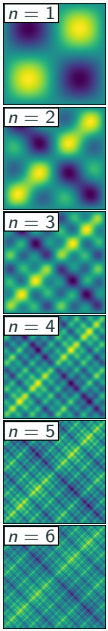
Multi-Frequency Problem – What the FBPINN Learns



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

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

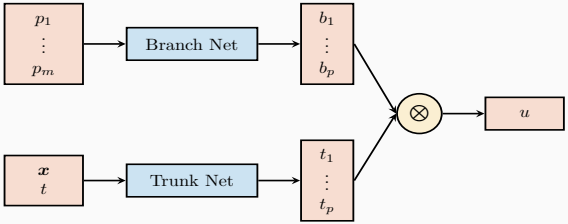




→ Details and results for the Helmholtz equation can be found in **Dolean, Heinlein, Mishra, Moseley (2024)**.

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 $p_1, \dots, p_m$ 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

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

Physics-informed DeepONets

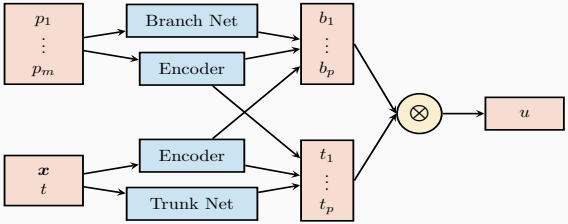
DeepONets are **compatible with the PINN approach** but **physics-informed DeepONets (PI-DeepONets)** are challenging to train.

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. (arXiv 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 $p_1, \dots, p_m$ 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

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

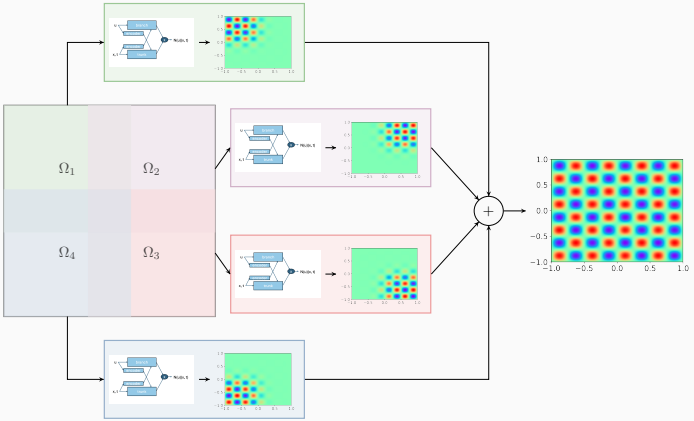
Physics-informed DeepONets

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. (arXiv 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)

FBDONs – Pendulum

Pendulum problem

$$\frac{ds_1}{dt} = s_2, \qquad t \in [0, T],$$
$$\frac{ds_2}{dt} = -\frac{b}{m} s_2 - \frac{g}{L} \sin(s_1), \qquad t \in [0, T],$$

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

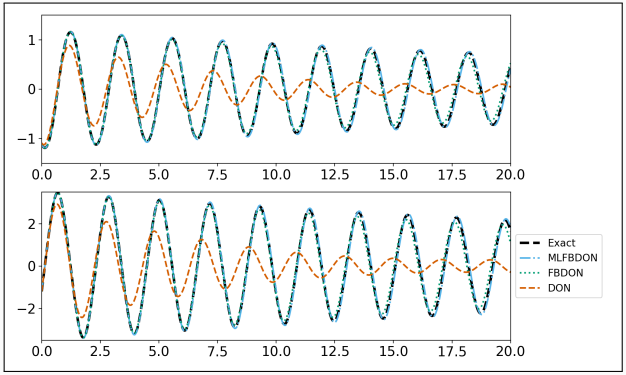
Parametrization

Initial conditions:

$$s_1(0) \in [-2, 2] \qquad s_2(0) \in [-1.2, 1.2]$$

$s_1(0)$ and $s_2(0)$ are the also inputs of the branch network.

Training on 50 k different configurations



Mean rel. l_2 error on 100 config.	
DeepONet	0.94
FBDON (32 subd.)	0.84
MLFBDON ($[1, 4, 8, 16, 32]$ subd.)	0.27

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

FBDONs – Wave Equation

Wave equation

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

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

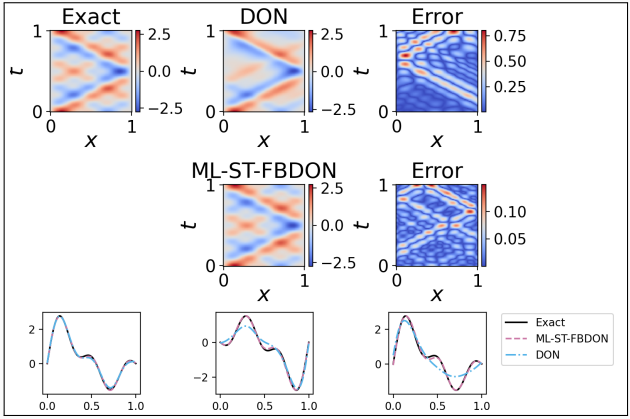
Solution: $s(x,t) = \sum_{n=1}^{5} b_n \sin(n\pi x)\cos(n\pi\sqrt{2}t)$

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.)**

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)

CWI
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 archictures

- **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