A Guided Tour of Modern Domain Decomposition

From Schwarz iterations to robust preconditioners, HPC, and what comes next

Victorita Dolean

with: P. Jolivet, F. Nataf, P.-H. Tournier

Sevilla, 2026

High Performance Scientific Computing, CASA
Eindhoven University of Technology

Outline

Part I: classical domain decomposition 60 min

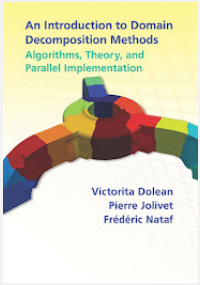
1. Why domain decomposition?
2. From Schwarz to additive Schwarz
3. The algebraic picture
4. Schwarz as a Krylov preconditioner
5. Scalability and robust coarse spaces
6. Hard regimes and HPC implementations

Part II: scientific machine learning 30 min

7. Locality: one structure, two vocabularies
8. Neural PDE solvers and their limits
9. FBPINNs and the level above
10. From optimisation to linear algebra
11. The converse direction and open questions

Based on **Dolean**, Jolivet, Nataf, Tournier, A guided tour of modern domain decomposition, arXiv:2605.24982, and on joint work with A. Heinlein, S. Mishra, B. Moseley.

Where all of this is written down



The book

Dolean, P. Jolivet, F. Nataf,
An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation, SIAM, 2015.

The classical material of the next hour, with the proofs and the FreeFem++ code.

The updated version: a book chapter, on arXiv

Dolean, P. Jolivet, F. Nataf, P.-H. Tournier,
A Guided Tour of Modern Domain Decomposition: From Schwarz Iterations to Robust Preconditioners and HPC Implementations, 2026.

This carries the latest developments: robust coarse spaces, indefinite and high-frequency regimes, and high-performance implementations.

arXiv:2605.24982 github.com/vicdolean/domain-decomposition-notes

Why domain decomposition?

Divide and conquer: an old idea, a modern necessity



> "*Divide* each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution."
>
> René Descartes, Discours de la méthode, 1637

Divide and conquer: an old idea, a modern necessity



"Divide each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution."

René Descartes, Discours de la méthode, 1637



"If you were plowing a field, which would you rather use: two strong oxen or 1024 chickens?"

Seymour Cray

Divide and conquer: an old idea, a modern necessity





"Divide each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution."

René Descartes, Discours de la méthode, 1637

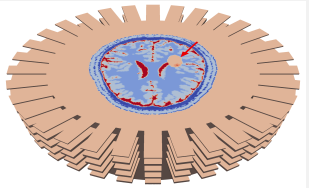
"If you were plowing a field, which would you rather use: two strong oxen or 1024 chickens?"

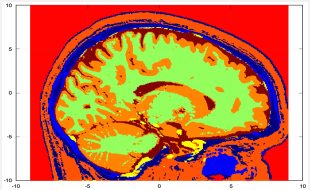
Seymour Cray

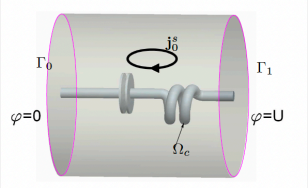
Today we plough with **millions of chickens**.

Dividing is easy. The art is making the pieces agree on the whole.

EM wave propagation in heterogeneous media







Maxwell's equations

Reconstruct the permittivity ε

$$\nabla \times (\mu^{-1} \nabla \times \mathbf{E}) - \omega^2 \varepsilon \mathbf{E} = \mathbf{J}$$

- $\mathbf{E}$ is the electric field
- $\mu > 0$ is the magnetic permeability
- $\varepsilon > 0$ is the electric permittivity.
- ω is the frequency

Challenges

- **High frequency**: solution highly oscillatory $\rightsquigarrow$ pollution effect, large linear systems.
- **Low frequency**: near singular operator with a huge near kernel.

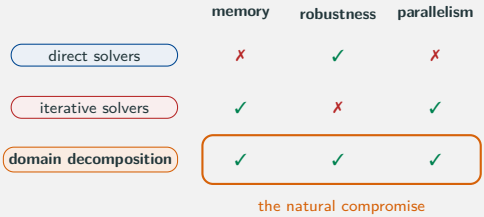
Aim

The linear system inherits the properties of the PDE $\rightsquigarrow$ design a **robust** solver.

$Au = b$? Landscape of linear solvers

Iterative Methods

- Fixed-point: Jacobi, Gauss-Seidel, SSOR
- Krylov methods:
  - Conjugate Gradient (Stiefel–Hestenes)
  - GMRES (Y. Saad), QMR (R. Freund)
  - MinRes, BiCGSTAB (van der Vorst)



Direct Solvers

- **MUMPS** (J.Y. L'Excellent)
- **SuperLU** (Demmel et al), **PastiX**
- **UMFPACK**, **PARDISO** (O. Schenk)

"Hybrid methods (DDM)

- Multigrid: Brandt, Ruge-Stüben, Falgout, McCormick, Ruhe, Notay ...
- Domain decomposition (DDM): Widlund, Farhat, Mandel, Lions ...

Natural parallel compromise between robustness and scalability

Sparse Gaussian Elimination: Complexity and Practical Limits

Asymptotic Complexity for Structured PDE Matrices

Method	1D ($d = 1$)	2D ($d = 2$)	3D ($d = 3$)
Dense matrix	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Band structure exploited	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{7/3})$
Sparse (e.g. nested dissection)	$\mathcal{O}(n)$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^2)$

Practical Limits (2025)

- Sparse direct solvers handle up to $n \sim 10^7$ in 2D and $n \sim 10^5$ in 3D on modern hardware.
- Fill-in in 3D limits scalability: memory and factorization time dominate.
- Hybrid methods (e.g. domain decomposition or multilevel solvers) are preferred for large-scale problems.

Takeaway

Direct solvers are robust but hit a memory wall in 3D. Iterative solvers scale, but need a good preconditioner. **Domain decomposition builds that preconditioner.**

Need & Opportunities for massively parallel computing

The Rise of Parallel Machines

Parallel computing is now accessible to everyone:

- Laptops (Apple Mx, Linux, Windows): 4–12 cores
- Desktops/Workstations: 16–128 cores
- Lab Clusters: ∼300 cores
- University HPC Clusters: ∼10,000 cores
- Cloud Infrastructures (AWS, Azure): elastic, on-demand compute
- National Supercomputers: $> 100,000$ cores, e.g., Fugaku, LUMI, Frontier

All fields of science and engineering are affected.

Hardware Trends (2025)

- ARM-based SoCs: **A64FX** (Fugaku), **Apple M1/M2**, **Graviton3**
- Scalable Vector Extensions (SVE) enable efficient HPC+AI workloads
- High Bandwidth Memory (HBM) improves memory-bound performance
- Heterogeneous systems: CPUs + GPUs + AI accelerators

Software Evolution

- **Languages**: Julia, Rust, Python (MPI4Py), offer easy parallelism
- **Libraries**: PETSc, HPDDM, ScaLAPACK (linear algebra); DUNE, OpenFOAM (PDEs)
- **Standards**: MPI + OpenMP + OpenACC remain critical

From Schwarz to additive Schwarz

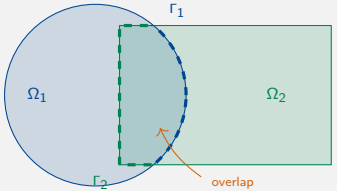
The original Schwarz method (H. A. Schwarz, 1870)

Solves $-\Delta u = f$ in Ω with $u = 0$ on $\partial\Omega$ using overlapping subdomains:

Iteration Scheme: Schwarz Alternating Method

Given u_1^n, u_2^n, compute:

$$
\begin{aligned}
-\Delta u_1^{n+1} &= f \quad \text{in } \Omega_1 \\
u_1^{n+1} &= 0 \quad \text{on } \partial\Omega_1 \cap \partial\Omega \\
u_1^{n+1} &= u_2^n \quad \text{on } \partial\Omega_1 \cap \overline{\Omega}_2 \\
-\Delta u_2^{n+1} &= f \quad \text{in } \Omega_2 \\
u_2^{n+1} &= 0 \quad \text{on } \partial\Omega_2 \cap \partial\Omega \\
u_2^{n+1} &= u_1^{n+1} \quad \text{on } \partial\Omega_2 \cap \overline{\Omega}_1
\end{aligned}
$$



$\Omega = \Omega_1 \cup \Omega_2$: Schwarz's own example, a disk and a rectangle

Each solve needs a boundary value only the other subdomain knows.

The First Domain Decomposition Method

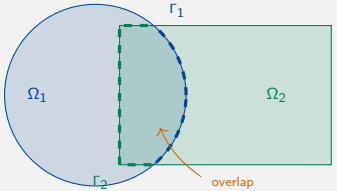
The original Schwarz method (H. A. Schwarz, 1870)

Solves $-\Delta u = f$ in Ω with $u = 0$ on $\partial\Omega$ using overlapping subdomains:

Iteration Scheme: Schwarz Alternating Method

Given u_1^n, u_2^n, compute:

$$
\begin{aligned}
-\Delta u_1^{n+1} &= f \quad \text{in } \Omega_1 \\
u_1^{n+1} &= 0 \quad \text{on } \partial\Omega_1 \cap \partial\Omega \\
u_1^{n+1} &= u_2^n \quad \text{on } \partial\Omega_1 \cap \overline{\Omega}_2 \\
-\Delta u_2^{n+1} &= f \quad \text{in } \Omega_2 \\
u_2^{n+1} &= 0 \quad \text{on } \partial\Omega_2 \cap \partial\Omega \\
u_2^{n+1} &= u_1^{n+1} \quad \text{on } \partial\Omega_2 \cap \overline{\Omega}_1
\end{aligned}
$$



$\Omega = \Omega_1 \cup \Omega_2$: Schwarz's own example, a disk and a rectangle

Each solve needs a boundary value only the other subdomain knows.

What is good, and what is not

- ✓ Naturally parallel and it needs only a local solver.
- ✗ Convergence is slow, and it **stops entirely without overlap**.
- The parallel version is the **Jacobi-Schwarz method (JSM)**.

Continuous ASM and RAS – Key Ingredients

Local-to-Global Strategy

We solve on local functions u_i supported on Ω_i and reconstruct the global function u.

Extension Operators

Each E_i extends a local function $w_i : \Omega_i \to \mathbb{R}$ to $E_i(w_i) : \Omega \to \mathbb{R}$ by zero outside Ω_i.

Partition of Unity

Let $\chi_i : \Omega_i \to \mathbb{R}$ satisfy:

$$\chi_i \geq 0, \quad \chi_i(x) = 0 \text{ on } \partial\Omega_i, \quad \text{and} \quad w(x) = \sum_{i=1}^{2} E_i(\chi_i \, w|_{\Omega_i}).$$

Iteration

Given $u^n \approx u$, compute u^{n+1} by solving local problems and combining them using E_i and χ_i.

Local Subproblems (for $i = 1, 2$)

At each iteration n, solve:

$$\begin{aligned} -\Delta u_i^{n+1} &= f \quad \text{in } \Omega_i \\ u_i^{n+1} &= 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega \\ u_i^{n+1} &= u^n \quad \text{on } \partial\Omega_i \cap \overline{\Omega}_{3-i} \end{aligned}$$

Pros & cons

- RAS: faster convergence, non-symmetric
- ASM: easier analysis, symmetric
- Both methods are **parallel**, overlap-dependent

Gluing the solutions: two strategies

Restricted Additive Schwarz (RAS)

$$u^{n+1} = \sum_{i=1}^{2} E_i(\chi_i \, u_i^{n+1})$$

Uses partition of unity for overlap control.

Additive Schwarz (ASM)

$$u^{n+1} = \sum_{i=1}^{2} E_i(u_i^{n+1})$$

Pure sum of extensions (no weighting).

Block Jacobi: Domain Partitioning

Index Partition

Split degrees of freedom into two index sets:

$$\mathcal{N}_1 := \{1, \ldots, m_s\}, \quad \mathcal{N}_2 := \{m_s + 1, \ldots, m\}$$

Define:

$$\mathbf{U}_1 := \mathbf{U}_{|\mathcal{N}_1}, \quad \mathbf{U}_2 := \mathbf{U}_{|\mathcal{N}_2}, \quad \mathbf{F}_1, \mathbf{F}_2 \text{ likewise.}$$

Block Form of A

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{pmatrix}$$

Block-Jacobi Method — Compact Matrix Form

Operators

- R_1, R_2: Restriction operators from $\mathcal{N}$ to $\mathcal{N}_1$, $\mathcal{N}_2$ respectively
- R_1^T, R_2^T: Extension operators
- $A_i = R_i A R_i^T$: local block matrix

Residual

$$\mathbf{r}^n = \mathbf{F} - A\mathbf{U}^n, \quad \mathbf{r}_i^n = \mathbf{r}^n_{|\mathcal{N}_i}$$

Compact block-Jacobi update

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \left(R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2 \right) \mathbf{r}^n$$

- Parallel update using local block solves
- A_1^{-1} and A_2^{-1} are local inverses (independent)
- Structure generalizes to N blocks trivially

Schwarz Methods as Block-Jacobi Algorithms (1D Case)

Problem Setup

Let $\Omega = (0,1)$ with Dirichlet BCs:

$$-\Delta u = f \text{ in } \Omega, \quad u(0) = u(1) = 0$$

Discretize with m internal nodes using 3-point finite differences:

$$A\mathbf{U} = \mathbf{F}, \quad A \in \mathbb{R}^{m \times m}$$

where:

$$A_{jj} = \tfrac{2}{h^2}, \quad A_{j,j\pm1} = -\tfrac{1}{h^2}, \quad h = \tfrac{1}{m+1}$$

Subdomain Decomposition

Overlap of width h:

$$\Omega_1 = (0, (m_s+1)h), \quad \Omega_2 = (m_s h, 1)$$

Jacobi–Schwarz Update on Ω_1

$$\begin{cases} -\dfrac{u_{1,j-1}^{n+1} - 2u_{1,j}^{n+1} + u_{1,j+1}^{n+1}}{h^2} = f_j, & 1 \le j \le m_s \\ u_{1,0}^{n+1} = 0 \\ u_{1,m_s+1}^{n+1} = u_{2,m_s+1}^{n} \end{cases}$$

Matrix Formulation

$$A_{11}\mathbf{U}_1^{n+1} + A_{12}\mathbf{U}_2^{n} = \mathbf{F}_1$$
$$A_{22}\mathbf{U}_2^{n+1} + A_{21}\mathbf{U}_1^{n} = \mathbf{F}_2$$

Schwarz = Block-Jacobi (Under Minimal Overlap)

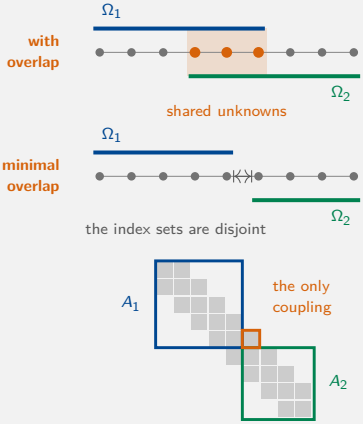
Extension Operators

$$E_1(U_1) = \begin{pmatrix} U_1 \\ 0 \end{pmatrix}, \quad E_2(U_2) = \begin{pmatrix} 0 \\ U_2 \end{pmatrix}$$

Partition of Unity

With overlap:

$$E_1(U_1)+E_2(U_2) = E_1(\chi_1 U_1)+E_2(\chi_2 U_2) = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$



Key insight

With **minimal overlap**, additive Schwarz, restricted additive Schwarz and Jacobi-Schwarz **all reduce to the same block-Jacobi method**.

Convergence Condition and Visualization

Interface Coupling

$$e_2^{n+1}(L_1) = e_2^n(L_1) \cdot \frac{l_2}{L_1} \cdot \frac{L - L_1}{L - l_2}$$

Observation

The current interface value depends linearly on the previous iterate.

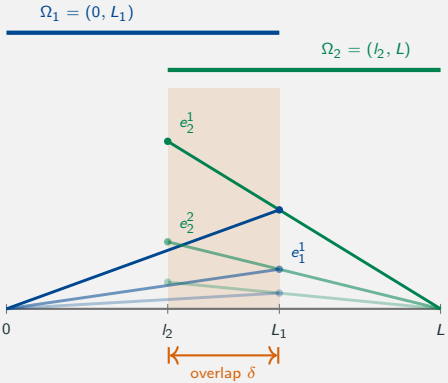
Interface Iteration

Let $\delta := L_1 - l_2$ (overlap):

$$e_2^{n+1}(L_1) = \frac{1 - \delta/(L - l_2)}{1 + \delta/l_2} \cdot e_2^n(L_1)$$

Convergence Condition

$\delta > 0$ is necessary and sufficient for convergence.



the error after each sweep

Each sweep multiplies the interface error by $\frac{L - L_1}{L - l_2} \cdot \frac{l_2}{L_1} < 1$. Set $\delta = 0$ and the factor becomes 1: nothing moves.

Convergence Factor and Iteration Behavior

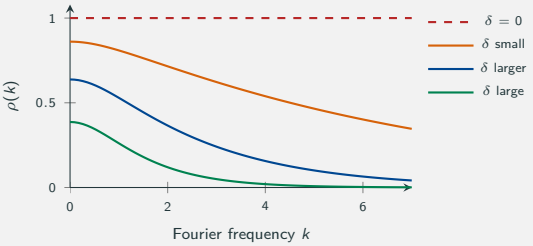
Interface Update and Factor

Matching at the interfaces gives:

$$\gamma_+^{n+1}(k) = \gamma_-^n(k)\, e^{-\lambda(k)\delta}$$
$$\gamma_-^{n+1}(k) = \gamma_+^n(k)\, e^{-\lambda(k)\delta}$$
$$\Rightarrow\ \gamma_\pm^{n+1}(k) = \rho(k)^2 \cdot \gamma_\pm^{n-1}(k)$$

Convergence factor

$$\rho(k;\eta,\delta) = e^{-\lambda(k)\delta}, \quad \lambda(k) = \sqrt{\eta + k^2}$$



$\rho(k;\eta,\delta) = e^{-\sqrt{\eta+k^2}\,\delta}$: higher frequencies always die faster.

Convergence insights

- $\Rightarrow$ **Uniform convergence:** $\rho(k) < e^{-\sqrt{\eta}\,\delta} < 1$
- $\Rightarrow$ **High-frequency error components decay rapidly**
- $\Rightarrow$ **No overlap ($\delta = 0$)** $\Rightarrow$ no decay ($\rho = 1$): stagnation

Summary of Schwarz Convergence Insights (1D and 2D)

1D Case: Iterative Affine Model

- Schwarz iterates satisfy linear error decay via interface transfer
- Convergence factor < 1 only if overlap $\delta > 0$
- No overlap $\Rightarrow$ method stagnates

2D Case: Fourier Mode Analysis

- Fourier transform yields exact decay rate per mode k
- **High-frequency modes** decay fastest
- Convergence rate: $\rho(k; \eta, \delta) = e^{-\sqrt{\eta + k^2}\,\delta}$

Takeaway

Overlap is essential. Without it, Schwarz-type methods stall.

With it, convergence improves with frequency.

The algebraic picture

From Continuous to Discrete: Domain Decomposition View

Continuous Level

- Domain: $\Omega = \bigcup_{i=1}^{N} \Omega_i$ (overlapping decomposition)
- Global function: $u : \Omega \to \mathbb{R}$
- Restriction: $u_i = u|_{\Omega_i}$
- Extension: $E_i(u_i) : \Omega \to \mathbb{R}$ (zero outside Ω_i)
- Partition of unity: functions $\chi_i : \Omega_i \to \mathbb{R}$ with

$$u = \sum_{i=1}^{N} E_i(\chi_i \, u|_{\Omega_i})$$

Discrete Level

- Degrees of freedom: $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$
- Global vector: $U \in \mathbb{R}^{\#\mathcal{N}}$
- Restriction: $R_i \in \{0,1\}^{\#\mathcal{N}_i \times \#\mathcal{N}}$
- Extension: R_i^T (transpose of R_i)
- Partition of unity: diagonal matrices D_i with positive entries s.t.:

$$I = \sum_{i=1}^{N} R_i^T D_i R_i$$

Restriction Operators in Finite Element Decomposition

Mesh-Based Domain Decomposition

- $\mathcal{T}_h$: global mesh of domain Ω
- $\mathcal{T}_{h,i}$: mesh of subdomain Ω_i
- V_h: global finite element space
- $V_{h,i}$: local FE space on $\mathcal{T}_{h,i}$
- u_h: global FE solution

Restriction Operator

Restriction of u_h to Ω_i:

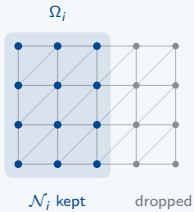
$$r_i(u_h) = u_h|_{\Omega_i}, \quad r_i : V_h \to V_{h,i}$$

Matrix form:

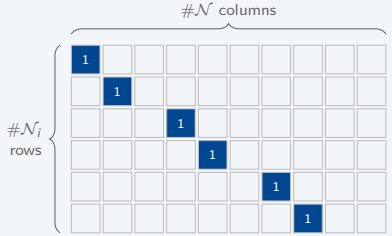
$$R_i : \mathbb{R}^{\#\mathcal{N}} \to \mathbb{R}^{\#\mathcal{N}_i}$$

(R_i is a Boolean selector)

Geometric interpretation



Algebraic form: the Boolean selector R_i



One 1 per row: R_i **picks** the local unknowns, R_i^T puts them back.

Operators

- Restriction: $R_i : \mathbb{R}^{\#\mathcal{N}} \to \mathbb{R}^{\#\mathcal{N}_i}$
- Prolongation (extension): $R_i^T : \mathbb{R}^{\#\mathcal{N}_i} \to \mathbb{R}^{\#\mathcal{N}}$
- Local matrices: $A_i = R_i A R_i^T$
- Partition of unity:

$$D_i \in \mathbb{R}^{\#\mathcal{N}_i \times \#\mathcal{N}_i} \text{ diagonal, s.t. } \sum_{i=1}^{N} R_i^T D_i R_i = I$$

Purpose

These ingredients allow expressing local computations and gluing into a global update.

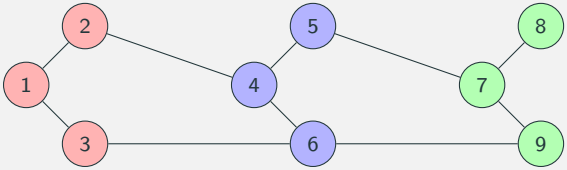
Multi-Domain Partitioning: General Procedure

Graph-Based Partitioning (METIS, SCOTCH)

- From matrix A, construct a graph G:
  - Nodes $\leftrightarrow$ degrees of freedom
  - Edges: $A_{ij} \neq 0$
- Symmetrize G if needed
- Apply partitioner to divide $\mathcal{N}$ into N subdomains

Partitioning Goals

- Load balancing: equal work per subdomain
- Minimal communication: few cross edges

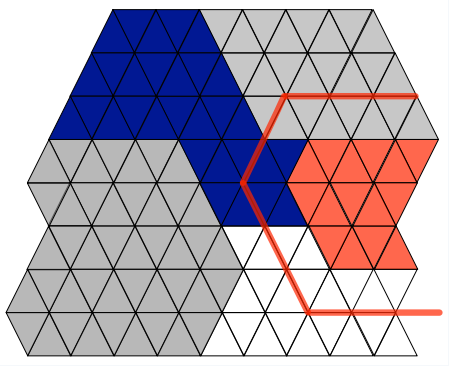


Graph G partitioned into 3 subdomains

Multi-D Algebraic Finite Element Decomposition

Mesh and Overlap

- Let $\mathcal{T}_h$: mesh of Ω
- Each $\mathcal{T}_{h,i}$ gives overlapping Ω_i



FE Basis and Index Partition

Let $\{\phi_k\}_{k\in\mathcal{N}}$ be basis functions:

$$\mathcal{N}_i := \{k : \mathrm{supp}(\phi_k) \cap \Omega_i \neq \emptyset\}$$

Multiplicity per node:

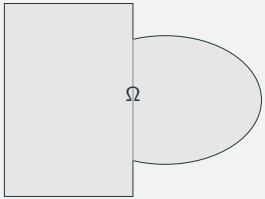
$$\mu_k := \#\{j : \mathrm{supp}(\phi_k) \cap \Omega_j \neq \emptyset\}$$

Algebraic Partition of Unity

$$(D_i)_{kk} = \frac{1}{\mu_k}, \quad k \in \mathcal{N}_i$$

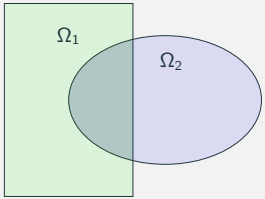
Summary: Additive Schwarz

Let the discretised Poisson problem: $A\mathbf{U} = \mathbf{F} \in \mathbb{R}^n$.



Summary: Additive Schwarz

Let the discretised Poisson problem: $A\mathbf{U} = \mathbf{F} \in \mathbb{R}^n$. Given a decomposition of $[\![1; n]\!]$, $(\mathcal{N}_1, \mathcal{N}_2)$, define: the restriction operator R_i from $\mathbb{R}^{[\![1;n]\!]}$ into $\mathbb{R}^{\mathcal{N}_i}$, R_i^T as the extension by 0 from $\mathbb{R}^{\mathcal{N}_i}$ into $\mathbb{R}^{[\![1;n]\!]}$.

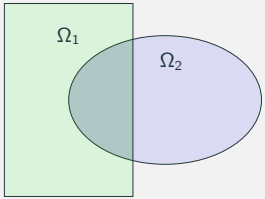


Summary: Additive Schwarz

Let the discretised Poisson problem: $A\mathbf{U} = \mathbf{F} \in \mathbb{R}^n$. Given a decomposition of $[\![1; n]\!]$, $(\mathcal{N}_1, \mathcal{N}_2)$, define: the restriction operator R_i from $\mathbb{R}^{[\![1;n]\!]}$ into $\mathbb{R}^{\mathcal{N}_i}$, R_i^T as the extension by 0 from $\mathbb{R}^{\mathcal{N}_i}$ into $\mathbb{R}^{[\![1;n]\!]}$.
Find $\mathbf{U}^m \longrightarrow \mathbf{U}^{m+1}$ by solving concurrently:

$$\mathbf{U}_j^{m+1} = \mathbf{U}_j^m + A_j^{-1} R_j(\mathbf{F} - A\mathbf{U}^m),\ j = 1, 2$$

where $\mathbf{U}_i^m = R_i\mathbf{U}^m$ and $A_i := R_i A R_i^T$.



Algebraic RAS Formulation: Equivalence and Use

RAS as Global Iteration

The RAS method updates the global iterate using local solves and partition weights:

$$\mathbf{U}^{n+1} = \mathbf{U}^n + M_{\text{RAS}}^{-1}\mathbf{r}^n, \quad \mathbf{r}^n := \mathbf{F} - A\mathbf{U}^n$$

Local to Global Representation

The current iterate $\mathbf{U}^n$ is reconstructed from local components:

$$\mathbf{U}^n = R_1^T D_1 \mathbf{U}_1^n + R_2^T D_2 \mathbf{U}_2^n$$

Each $\mathbf{U}_i^n$ solves a local problem over an overlapping subdomain.

Krylov Usage

The operator M_{RAS}^{-1} serves as a preconditioner in Krylov subspace methods such as GMRES, BiCGStab, and others. This improves convergence for non-symmetric and ill-conditioned problems.

Preconditioners Summary

- **RAS (Restricted Additive Schwarz):**

$$M_{\text{RAS}}^{-1} = \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i$$

- **ASM (Additive Schwarz Method):**

$$M_{\text{ASM}}^{-1} = \sum_{i=1}^{N} R_i^T A_i^{-1} R_i$$

- **SORAS (Symmetrized Overlapping RAS):**

$$M_{\text{SORAS}}^{-1} = \sum_{i=1}^{N} R_i^T D_i B_i^{-1} D_i R_i$$

Theory vs. performance

RAS is often more efficient, but ASM is more amenable to theoretical analysis.

Schwarz as a Krylov preconditioner

Fixed Point Iteration

Given a linear system $A\mathbf{x} = \mathbf{b}$, we apply an iterative fixed-point method:

$$\mathbf{x}^{n+1} = \mathbf{x}^n + B^{-1}(\mathbf{b} - A\mathbf{x}^n)$$

a fixed point of $\mathbf{x} \mapsto \mathbf{x} + B^{-1}(\mathbf{b} - A\mathbf{x})$.

With $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}^0$ and $C = B^{-1}A$:

$$\mathbf{x}^n = \sum_{i=0}^{n} (I - C)^i B^{-1}\mathbf{r}_0 + \mathbf{x}^0$$

Convergence condition: Spectral radius $\rho(I - C) < 1$.

Krylov Perspective

In Krylov methods, we solve the preconditioned system:

$$C\mathbf{x} = B^{-1}\mathbf{b}, \quad \text{with } C = B^{-1}A$$

Given an initial guess $\mathbf{x}^0$, the residual is:

$$\mathbf{r}^0 = B^{-1}\mathbf{b} - C\mathbf{x}^0$$

We define $\mathbf{y} := \mathbf{x} - \mathbf{x}^0$ and solve:

$$C\mathbf{y} = \mathbf{r}^0$$

Polynomial inversion insight

There is a polynomial $\mathcal{P}$ of degree below N with $C^{-1} = \mathcal{P}(C)$: the foundation of Krylov methods.

GMRES Iteration Principle

Find the best approximation in a Krylov subspace:

$$\mathbf{y}^n = \arg \min_{\mathbf{w} \in \mathcal{K}^n(C, \mathbf{r}^0)} \|C\mathbf{w} - \mathbf{r}^0\|_2$$

Preconditioned space:

$$\mathcal{K}^n(C, B^{-1}\mathbf{r}_0) = \text{span}\{B^{-1}\mathbf{r}_0, CB^{-1}\mathbf{r}_0, \ldots\}$$

Why Krylov?

Krylov methods generate optimally weighted iterates at minimal cost (especially for small n), whereas fixed-point schemes rely on static weights.

Schwarz Preconditioners

Schwarz methods serve as powerful preconditioners:

- **RAS (Restricted Additive Schwarz)**: Used with GMRES or BiCGStab

$$B^{-1} = M_{\text{RAS}}^{-1} = \sum_{i=1}^{N} R_i^T D_i (R_i A R_i^T)^{-1} R_i$$

- **ASM (Additive Schwarz Method)**: Used with CG

$$B^{-1} = M_{\text{ASM}}^{-1} = \sum_{i=1}^{N} R_i^T (R_i A R_i^T)^{-1} R_i$$

Preconditioned CG with ASM: Efficiency and Convergence

Theoretical convergence guarantee

When using the Additive Schwarz preconditioner M_{ASM}^{-1} in Conjugate Gradient:

$$\|\mathbf{x} - \mathbf{x}_m\|_{M_{\text{ASM}}^{-1}A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|\mathbf{x} - \mathbf{x}_0\|_{M_{\text{ASM}}^{-1}A}$$

where $\kappa = \text{cond}(M_{\text{ASM}}^{-1}A)$ is the condition number of the preconditioned matrix.

Implication

A well-chosen M_{ASM}^{-1} significantly reduces κ, accelerating convergence of CG.

PCG Iteration with ASM Preconditioner

for $i = 1, 2, \ldots$

$\rho_{i-1} = (\mathbf{r}_{i-1}, M_{\text{ASM}}^{-1}\mathbf{r}_{i-1})$

if $i = 1$

$\mathbf{p}_1 = M_{\text{ASM}}^{-1}\mathbf{r}_0$

else

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

$\mathbf{p}_i = M_{\text{ASM}}^{-1}\mathbf{r}_{i-1} + \beta_{i-1}\mathbf{p}_{i-1}$

end if

$\mathbf{q}_i = A\mathbf{p}_i$

$\alpha_i = \rho_{i-1}/(\mathbf{p}_i, \mathbf{q}_i)$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\mathbf{p}_i$

$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i\mathbf{q}_i$

end for

Why we put Schwarz inside a Krylov method

Two ways to use the same M^{-1}

As a **fixed-point** iteration:

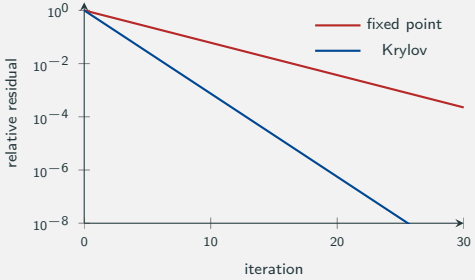
$$\mathbf{x}^{n+1} = \mathbf{x}^n + M^{-1}(\mathbf{b} - A\mathbf{x}^n)$$

or as a **preconditioner** for CG or GMRES on

$$M^{-1}A\,\mathbf{x} = M^{-1}\mathbf{b}.$$

What changes

The cost of one iteration is identical: N local solves, all in parallel. The Krylov method simply **reuses the whole history** instead of the last iterate.



Schematic. The gap is why nobody runs Schwarz as a stationary iteration any more.

Takeaway

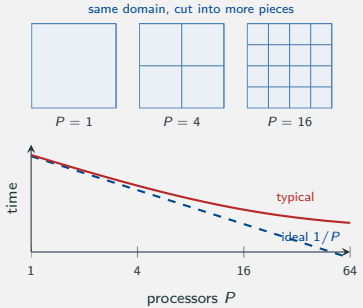
In practice M_{AS}^{-1} is never the solver. It is the **preconditioner**, and Krylov does the rest.

Scalability and robust coarse spaces

Many cores: what are we actually measuring?

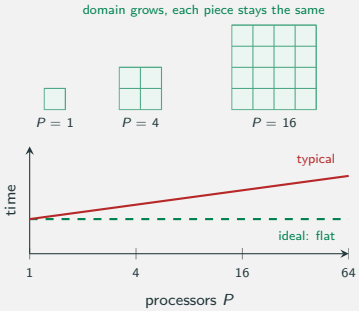
Strong scalability (Amdahl)

The **total problem is fixed**. Add processors and ask how much **faster** it gets.



Weak scalability (Gustafson)

The **work per processor is fixed**. Grow the problem with the machine and ask whether the time **stays flat**.



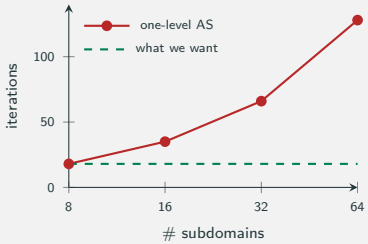
Takeaway

Per iteration, a domain decomposition method parallelises by construction. Both curves are decided by the **iteration count**.

One-level Schwarz is not scalable

Measured iteration counts

# subdomains	8	16	32	64
AS iterations	18	35	66	128



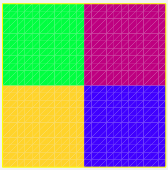
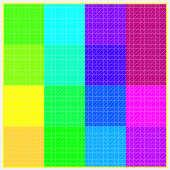
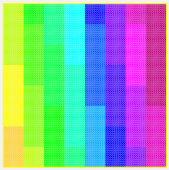
- The count grows **linearly** with the number of subdomains.
- Doubling the machine does not halve the time.

Takeaway

Locality alone does not scale. Information has to cross the whole domain in **one** step, and only a **coarse space** can do that.

Why the Algorithm is Not Scalable?

- We have that $\lambda_{\max}(M_{AS}^{-1}A) \leq \mathcal{N}_c \ll N$
- But $\lambda_{\min}(M_{AS}^{-1}A)$ decreases as N increases.
- $\Rightarrow$ Condition number grows with N, breaking weak scalability.

Poisson Problem $-\Delta u = f$ with 20×20 discretisation and 2-layer overlap

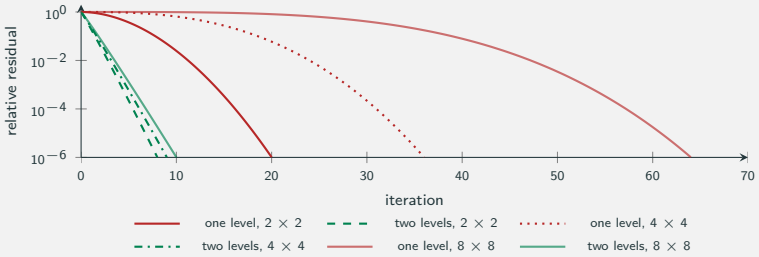
# Subdomains	2×2	4×4	8×8
Iterations	20	36	64

Root Cause of Stagnation

- Stagnation is caused by a few small eigenvalues in the spectrum of the preconditioned system.
- These small eigenvalues arise due to the **lack of global information exchange** in the preconditioner.

Classical Remedy

Introduce a **coarse problem** that couples all subdomains and enables global information exchange.



One level: the residual sits on a **plateau** that gets longer the more subdomains there are across the domain, and only then falls (20, 36, 64 iterations). Two levels: no plateau. Schematic, drawn to match the measured counts.

Information travels one subdomain per iteration

One level

$$M_1^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j$$

Local solves, fully parallel, but a subdomain only ever hears from its **neighbours**.

$$\kappa(M_1^{-1}A) \lesssim \frac{1}{H^2}\left(1 + \frac{H}{\delta}\right)$$

one level: local solves only



Information travels one subdomain per iteration

One level

$$M_1^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j$$

Local solves, fully parallel, but a subdomain only ever hears from its **neighbours**.

$$\kappa(M_1^{-1}A) \lesssim \frac{1}{H^2}\left(1 + \frac{H}{\delta}\right)$$

one level: local solves only



Information travels one subdomain per iteration

One level

$$M_1^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j$$

Local solves, fully parallel, but a subdomain only ever hears from its **neighbours**.

$$\kappa(M_1^{-1}A) \lesssim \frac{1}{H^2}\left(1 + \frac{H}{\delta}\right)$$

one level: local solves only



Information travels one subdomain per iteration

One level

$$M_1^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j$$

Local solves, fully parallel, but a subdomain only ever hears from its **neighbours**.

$$\kappa(M_1^{-1} A) \lesssim \frac{1}{H^2}\left(1 + \frac{H}{\delta}\right)$$

one level: local solves only



crossing N subdomains needs $\mathcal{O}(N)$ iterations:
add cores $\Rightarrow$ add iterations

Information travels one subdomain per iteration

One level

$$M_1^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j$$

Local solves, fully parallel, but a subdomain only ever hears from its **neighbours**.

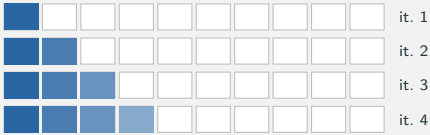
$$\kappa(M_1^{-1}A) \lesssim \frac{1}{H^2}\left(1 + \frac{H}{\delta}\right)$$

Two levels

$$M_2^{-1} = M_1^{-1} + Z(Z^* A Z)^{-1} Z^*$$

$$\kappa(M_2^{-1}A) \lesssim 1 + \frac{H}{\delta} \quad \text{independent of } N$$

one level: local solves only



crossing N subdomains needs $\mathcal{O}(N)$ iterations:
add cores $\Rightarrow$ add iterations

two levels: add a coarse space



Takeaway

A coarse space is a **bird's-eye view** that complements the **street-level detail** of the local solvers.

Targeting slow convergence modes

Assume we have identified the slow modes of the preconditioned system:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}$$

Examples of slow modes

- Constant functions in the **null space of the Laplace operator**.
- **Rigid body motions** in linear elasticity problems.

Notation

Let Z be the rectangular matrix whose columns span these slow modes.

Adding a Coarse Space: the Galerkin Correction

Correction by Minimization

We seek to correct an approximate solution $\mathbf{y}$ with $Z\beta$:

$$\min_{\beta} \|A(\mathbf{y} + Z\beta) - \mathbf{b}\|_{A^{-1}}$$

Adding a Coarse Space: the Galerkin Correction

Correction by Minimization

We seek to correct an approximate solution $\mathbf{y}$ with $Z\beta$:

$$\min_{\beta} \|A(\mathbf{y} + Z\beta) - \mathbf{b}\|_{A^{-1}}$$

Equivalent Formulation

$$\min_{\beta \in \mathbb{R}^{n_c}} \; 2(A\mathbf{y} - \mathbf{b}, Z\beta) + (AZ\beta, Z\beta)$$

Adding a Coarse Space: the Galerkin Correction

Correction by Minimization

We seek to correct an approximate solution $\mathbf{y}$ with $Z\beta$:

$$\min_{\beta} \|A(\mathbf{y} + Z\beta) - \mathbf{b}\|_{A^{-1}}$$

Equivalent Formulation

$$\min_{\beta \in \mathbb{R}^{n_c}} \; 2(A\mathbf{y} - \mathbf{b}, Z\beta) + (AZ\beta, Z\beta)$$

Optimal Coarse Coefficients

$$\beta = (Z^T AZ)^{-1} Z^T (\mathbf{b} - A\mathbf{y})$$

Adding a Coarse Space: the Galerkin Correction

Correction by Minimization

We seek to correct an approximate solution $\mathbf{y}$ with $Z\beta$:

$$\min_{\beta} \|A(\mathbf{y} + Z\beta) - \mathbf{b}\|_{A^{-1}}$$

Equivalent Formulation

$$\min_{\beta \in \mathbb{R}^{n_c}} \; 2(A\mathbf{y} - \mathbf{b}, Z\beta) + (AZ\beta, Z\beta)$$

Optimal Coarse Coefficients

$$\beta = (Z^T AZ)^{-1} Z^T (\mathbf{b} - A\mathbf{y})$$

$$\Rightarrow \quad Z\beta = Z(Z^T AZ)^{-1} Z^T \underbrace{(\mathbf{b} - A\mathbf{y})}_{\mathbf{r}}$$

Galerkin Correction

This term is known as the Galerkin correction.

Coarse Space Correction

Let $R_0 := Z^T$ and $\mathbf{r} = \mathbf{b} - A\mathbf{y}$.

Then the coarse correction is:

$$Z\beta = R_0^T\beta = R_0^T(R_0AR_0^T)^{-1}R_0\,\mathbf{r}$$

A Two-Level Schwarz Preconditioner

Coarse Space Correction

Let $R_0 := Z^T$ and $\mathbf{r} = \mathbf{b} - A\mathbf{y}$.

Then the coarse correction is:

$$Z\beta = R_0^T\beta = R_0^T(R_0AR_0^T)^{-1}R_0\,\mathbf{r}$$

Two-Level Preconditioner Definition

$$M_{AS,2}^{-1} := \underbrace{R_0^T(R_0AR_0^T)^{-1}R_0}_{\text{Coarse problem}} + \underbrace{\sum_{i=1}^{N} R_i^T(R_iAR_i^T)^{-1}R_i}_{M_{AS}^{-1}}$$

A Two-Level Schwarz Preconditioner

Coarse Space Correction

Let $R_0 := Z^T$ and $\mathbf{r} = \mathbf{b} - A\mathbf{y}$.

Then the coarse correction is:

$$Z\beta = R_0^T\beta = R_0^T(R_0AR_0^T)^{-1}R_0\,\mathbf{r}$$

Two-Level Preconditioner Definition

$$M_{AS,2}^{-1} := \underbrace{R_0^T(R_0AR_0^T)^{-1}R_0}_{\text{Coarse problem}} + \underbrace{\sum_{i=1}^{N} R_i^T(R_iAR_i^T)^{-1}R_i}_{M_{AS}^{-1}}$$

Remarks

- The structure of $M_{AS,2}^{-1}$ mimics the one-level method.
- The choice of coarse basis R_0 (or Z) is not unique.
- The coarse problem is small ($O(n_C \times n_C)$); the added cost is negligible.

Theoretical Convergence Result

Theorem (Widlund and Dryja)

If $M_{AS,2}^{-1}$ is the two-level additive Schwarz preconditioner, then

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

where $H =$ size of subdomains and $\delta =$ overlap size.

Theoretical Convergence Result

Theorem (Widlund and Dryja)

If $M_{AS,2}^{-1}$ is the two-level additive Schwarz preconditioner, then

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

where $H =$ size of subdomains and $\delta =$ overlap size.

Numerical validation

# Subdomains	8	16	32	64
AS (1-level)	18	35	66	128
AS + Nicolaides	20	27	28	27

Theoretical Convergence Result

Theorem (Widlund and Dryja)

If $M_{AS,2}^{-1}$ is the two-level additive Schwarz preconditioner, then

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

where $H =$ size of subdomains and $\delta =$ overlap size.

Numerical validation

# Subdomains	8	16	32	64
AS (1-level)	18	35	66	128
AS + Nicolaides	20	27	28	27

Limitation

Fails for highly heterogeneous problems.

$\Rightarrow$ **We need larger and possibly adaptive coarse spaces.**

Motivation

Context

Solving large discretized PDE systems with strongly heterogeneous coefficients:

- High contrast
- Multiscale structure

Example

Darcy pressure equation with P^1 finite elements:

$$\mathbf{Au} = \mathbf{f}, \quad \text{cond}(\mathbf{A}) \sim \frac{\alpha_{\max}}{\alpha_{\min}} \cdot h^{-2}$$

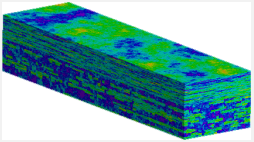
Goal: robust solvers

Iterative methods that remain effective:

- As mesh size $h \to 0$
- For high coefficient contrast

Applications

- Flow in layered / stochastic media
- Structural mechanics
- Electromagnetics



GenEO: Adaptive Spectral Coarse Space

What is GenEO?

GenEO constructs an **adaptive coarse space** for:

- Highly heterogeneous **Darcy problems**
- Compressible **elasticity problems**

It uses local spectral analysis to capture slow-to-converge error modes.

Why?

To accelerate convergence of iterative solvers by improving global information flow.

Local Generalized Eigenproblem (EVP)

For each subdomain j, find:

$$A_j^{\text{Neu}} \phi_{j,k} = \lambda_{j,k} D_j R_j A R_j^T D_j \cdot \phi_{j,k}$$

where:

$$\phi_{j,k} \in \mathbb{R}^{\mathcal{N}_j}, \quad \lambda_{j,k} \geq 0$$

Mode Selection (via Threshold τ)

Choose eigenmodes per subdomain with:

$$\lambda_{j,k} \leq \tau \quad \Rightarrow \quad Z := \left(R_j^T D_j \, \phi_{j,k} \right)$$

- $\lambda_{j,k} = 0 \Rightarrow$ **Nicolaides coarse space** (zero-energy modes)
- Smaller $\tau \rightsquigarrow$ fewer, high-impact modes
- Adaptivity to material contrast and geometry

Theoretical guarantee (Spillane et al., 2014)

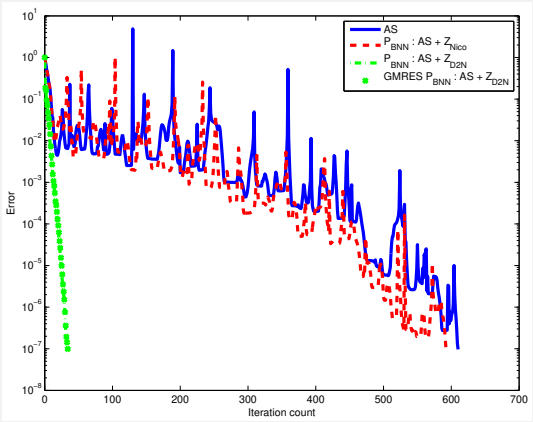
Under mild assumptions, for all j with $0 < \mu_{j,m_{j+1}} < \infty$:

$$\kappa(M_{AS,2}^{-1} A) \leq (1 + k_0) \left[2 + k_0 (2k_0 + 1)(1 + 1/\tau) \right]$$

Recommended threshold:

$$\tau := \left(\max_j \frac{H_j}{\delta_j} \right)^{-1}$$

Convergence with and without Spectral Coarse Spaces

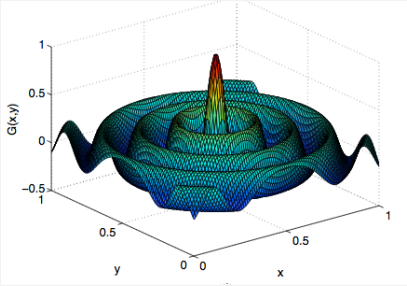


Key takeaway

The GenEO coarse space drastically improves convergence and makes iteration robust w.r.t the heterogeneity.

Hard regimes and HPC implementations

Why the high-frequency problem is hard? (Accuracy and pollution)

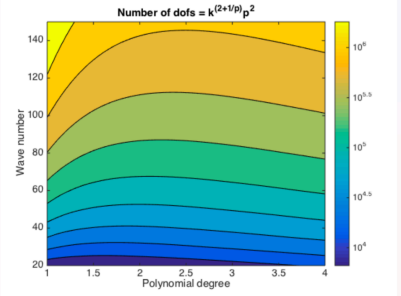


How to discretise well

- After discretisation **maximise** accuracy and **minimise** the number of degrees of freedom (#DoF)
- If $h\omega$ is kept constant the error increases with $\omega \rightsquigarrow$ **pollution error** [Babuska, Sauter, *SINUM*, 1997]
- FEM discretisations: for quasi-optimality we need [Melenk, Sauter, *SINUM*, 2011]

$$h^p \omega^{p+1} \lesssim 1$$

Why the high-frequency problem is hard? (Accuracy and pollution)



How to discretise well

- After discretisation **maximise** accuracy and **minimise** the number of degrees of freedom (#DoF)
- If $h\omega$ is kept constant the error increases with $\omega \rightsquigarrow$ **pollution error** [Babuska, Sauter, *SINUM*, 1997]
- FEM discretisations: for quasi-optimality we need [Melenk, Sauter, *SINUM*, 2011]

$$h^p \omega^{p+1} \lesssim 1$$

Consequences

- **High-frequency solution** u oscillates at a scale $1/\omega \Rightarrow h \sim \frac{1}{\omega} \rightsquigarrow$ large #DoF.
- **Pollution effect** requires $h \ll \frac{1}{\omega}$, $h \sim \omega^{-1-1/p}$, with p the finite element order $\rightsquigarrow$ even larger #DoF.
- Trade-off: **number of points per wavelength** (ppwl) $G = \frac{\lambda}{h} = \frac{2\pi}{\omega h}$ and polynomial degree $\rightsquigarrow$ **dispersion analysis** (measuring the ratio between the numerical and physical wave speeds).

Solve the preconditioned $A\mathbf{u} = \mathbf{b}$, i.e. $M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}$ by GMRES

The one-level preconditioner

$$M^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j, \text{ where}$$

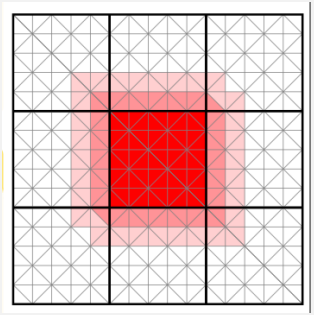
R_j	$\Omega \to \Omega_j$	**restriction operator**
R_j^T	$\Omega_j \to \Omega$	**prolongation operator**
D_j		**partition of unity matrix.**

Definition of the local matrices A_j $(k = \frac{\omega}{c})$

A_j is the stiffness matrix of the local **Robin problem**

$$\begin{aligned} (-\Delta - k^2)(u_j) &= f \quad \text{in } \Omega_j \\ \left(\frac{\partial}{\partial n_j} + ik\right)(u_j) &= 0 \quad \text{on } \partial\Omega_j \setminus \partial\Omega. \end{aligned}$$

$\Omega = \cup_{j=1}^{N} \Omega_j$



Solve the preconditioned $A\mathbf{u} = \mathbf{b}$, i.e. $M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}$ by GMRES

The one-level preconditioner

$$M^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j, \text{ where}$$

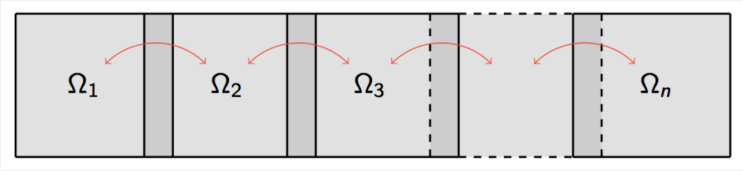
R_j	$\Omega \to \Omega_j$	**restriction operator**
R_j^T	$\Omega_j \to \Omega$	**prolongation operator**
D_j		**partition of unity matrix**.

Definition of the local matrices A_j ($k = \frac{\omega}{c}$)

A_j: stiffness matrix of the local **Robin problem**

$$\begin{aligned} (-\Delta - k^2)(u_j) &= f \quad \text{in } \Omega_j \\ \left(\tfrac{\partial}{\partial n_j} + ik\right)(u_j) &= 0 \quad \text{on } \partial\Omega_j \setminus \partial\Omega. \end{aligned}$$

One level is not enough (only neighbouring subdomains communicate)



Solve the preconditioned $A\mathbf{u} = \mathbf{b}$, i.e. $M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}$ by GMRES

The two-level (additive) preconditioner

$$M_2^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j + M_0^{-1}, \text{ where}$$

R_j	$\Omega \to \Omega_j$	**restriction operator**
R_j^T	$\Omega_j \to \Omega$	**prolongation operator**
D_j		**partition of unity matrix**.

Definition of the local matrices A_j ($k = \frac{\omega}{c}$)

A_j: stiffness matrix of the local **Robin problem**

$$\begin{aligned} (-\Delta - k^2)(u_j) &= f \quad \text{in } \Omega_j \\ \left(\tfrac{\partial}{\partial n_j} + ik\right)(u_j) &= 0 \quad \text{on } \partial\Omega_j \setminus \partial\Omega. \end{aligned}$$

How to add a second level or coarse information Z

$M_0^{-1} = ZE^{-1}Z^*$	is the coarse space correction
$Z, E = Z^*AZ$	matrix spanning the coarse space and the coarse matrix

Remark: Hybrid variants of the preconditioner are also possible.

An example of coarse space for Helmholtz

How to choose the coarse information Z? [Graham, Spence, Vainikko, *Math. Comp.*, 2017]

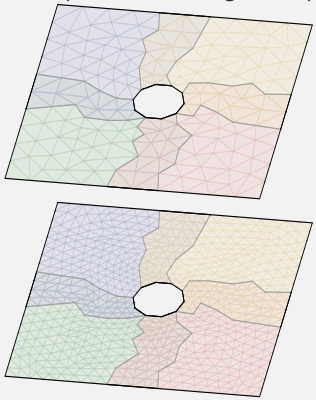
The grid coarse space (Grid CS)

- is based on a **geometrical** coarse mesh of diameter H_{coarse}
- R_0^T interpolation matrix from the fine to the coarse grid
- $Z = R_0^T$ matrix spanning the coarse space
- $E = Z^T A Z$ stiffness matrix on the coarse grid

Theory for absorptive problem: $-\Delta - (k^2 + i\xi)$

- For scalability and robustness w.r.t to the frequency we need $H_{\text{coarse}} \sim k^{-\alpha}$, $0 < \alpha <= 1$.
- $|\xi| \sim k^2$ and $\delta \sim H_{\text{coarse}}$, then weighted GMRES will converge with the number of iterations **independent of the wavenumber**.

Is the grid CS optimal for heterogeneous problems?



Bonazzoli, Dolean, Graham, Spence, Tournier. Domain decomposition preconditioning for the high-frequency time-harmonic Maxwell [...] Math. Comp., 2019.

Summary of available results

Two-level DD preconditioner

Solve $A\mathbf{u} = \mathbf{b}$, i.e. $M_2^{-1}A\mathbf{u} = M_2^{-1}\mathbf{b}$ by GMRES. DD preconditioner based on N domains of diameter $\sim H$.

$$M_2^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j + Z(Z^* A Z)^{-1} Z^*$$

Different CS according to the choice of Z:

- **Grid CS**: $Z = R_0^T$ with R_0^T **interpolation matrix** from the fine to the coarse grid.
- **DtN CS** : solve $\mathrm{DtN}_{\Omega_j}(u^l_{\Gamma_j}) = \lambda^l u^l_{\Gamma_j}$, Z is formed from **local harmonic extensions** ($\mathcal{H}$) of modes, **weighted** (D_j) and **extended globally** $R_j^T D_j \mathcal{H} u^l_{\Gamma_j}$
- **Δ-GenEO** (**$\mathcal{H}_k$-GenEO**): solve $L_j \mathbf{u}_j^l = \lambda^l D_j L_j D_j \mathbf{u}_j^l$ ($\widetilde{B}_j \mathbf{u}_j^l = \lambda^l D_j B_{j,k} D_j \mathbf{u}_j^l$), Z is formed from **weighted** (D_j) and **extended globally** modes $R_j^T D_j \mathbf{u}^l_{\Gamma_j}$

Summary of available results

Two-level DD preconditioner

Solve $A\mathbf{u} = \mathbf{b}$, i.e. $M_2^{-1}A\mathbf{u} = M_2^{-1}\mathbf{b}$ by GMRES. DD preconditioner based on N domains of diameter $\sim H$.

$$M_2^{-1} = \sum_{j=1}^{N} R_j^T D_j A_j^{-1} R_j + Z(Z^* A Z)^{-1} Z^*$$

Different CS according to the choice of Z:

- **Grid CS**: $Z = R_0^T$ with R_0^T **interpolation matrix** from the fine to the coarse grid.
- **DtN CS** : solve $\mathrm{DtN}_{\Omega_j}(u_{\Gamma_j}^l) = \lambda^l u_{\Gamma_j}^l$, Z is formed from **local harmonic extensions** $(\mathcal{H})$ of modes, **weighted** (D_j) and **extended globally** $R_j^T D_j \mathcal{H} u_{\Gamma_j}^l$
- **Δ-GenEO** (**$\mathcal{H}_k$-GenEO**): solve $L_j \mathbf{u}_j^l = \lambda^l D_j L_j D_j \mathbf{u}_j^l$ $(\widetilde{B}_j \mathbf{u}_j^l = \lambda^l D_j B_{j,k} D_j \mathbf{u}_j^l)$, Z is formed from **weighted** (D_j) and **extended globally** modes $R_j^T D_j \mathbf{u}_{\Gamma_j}^l$

Advantages and available results

- ⊕ **Grid CS**: Theoretical/numerical results absorptive problem, robustness for $H \sim k^{-\alpha}, 0 < \alpha \leq 1$.
- ⊕ **Δ-GenEO** : Theoretical/numerical results and robustness for **mild heterogeneities** and **low frequencies**.
- ⊕ **$\mathcal{H}_k$-GenEO** : Theoretical/numerical results and robustness for **high frequencies** in **the indefinite case**.



Bootland, Dolean, Graham, Ma, Scheichl. Overlapping Schwarz methods with GenEO coarse spaces for indefinite and nonself-adjoint problems, IMAJNA, 2022.

Strong scalability in two and three dimensions (with FreeFem++ and HPDDM)

Stokes problem with automatic mesh partition. Driven cavity problem

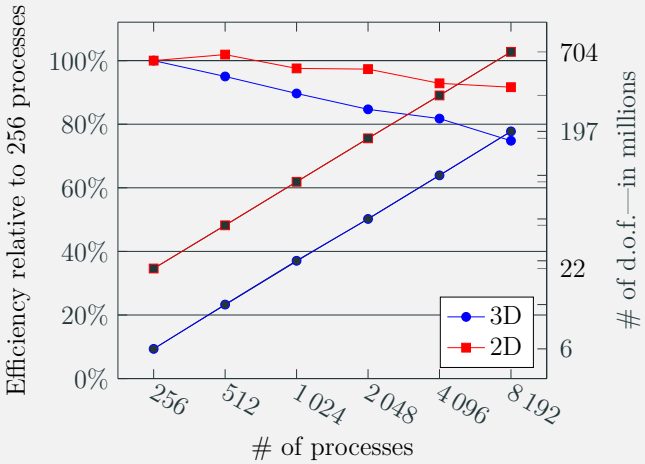
	N	Factorization	Deflation	Solution	# of it.	Total	# of d.o.f.
3D	1 024	79.2 s	229.0 s	76.3 s	45	387.5 s	$50.63 \cdot 10^6$
	2 048	29.5 s	76.5 s	34.8 s	42	143.9 s	
	4 096	11.1 s	45.8 s	19.8 s	42	80.9 s	
	8 192	4.7 s	26.1 s	14.9 s	41	56.8 s	
2D	1 024	5.2 s	37.9 s	51.5 s	51	95.6 s	$100.13 \cdot 10^6$
	2 048	2.4 s	19.3 s	22.1 s	42	44.5 s	
	4 096	1.1 s	10.4 s	10.2 s	35	22.6 s	
	8 192	0.5 s	4.6 s	6.9 s	38	12.7 s	

Peak performance: 50 millions d.o.f's in 3D in 57 sec.

IBM/Blue Gene Q machine with 1.6 GHz Power A2 processors. Hours provided by an IDRIS-GENCI project.

Weak scalability for heterogeneous elasticity (with FreeFem++ and HPDDM)

Rubber Steel sandwich with automatic mesh partition



(a) Timings of various simulations

Locality: one structure, two vocabularies

Same formula, two communities

ML (1991): mixture of local experts (MoE)

$$y(x) = \sum_{i=1}^{E} g_i(x)\, f_i(x;\theta_i), \qquad g_i \geq 0, \quad \sum_i g_i(x) = 1$$

- f_i: small expert network
- $g = \mathrm{softmax}(Wx)$: **learned** gate
- motive: specialisation; today, many parameters at a modest computing cost

NA (1997): partition of unity (PoU) method

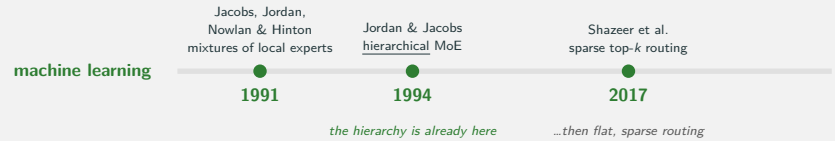
$$u(x) = \sum_{j=1}^{J} \varphi_j(x)\, u_j(x), \qquad \varphi_j \geq 0, \quad \sum_j \varphi_j \equiv 1$$

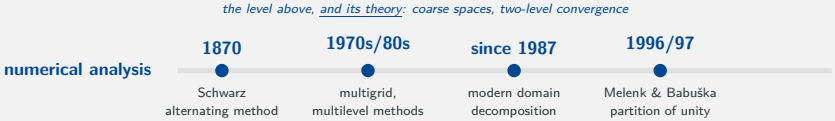
- $u_j \in V_j$: local approximation space on Ω_j
- φ_j: **prescribed**, smooth, compactly supported
- motive: local approximation, a priori error bounds

Both are partitions of unity over local approximants.
One community went on to build trillion-parameter models; the other built better approximation spaces.

Jacobs, Jordan, Nowlan, Hinton, Adaptive mixtures of local experts, Neural Comput. 1991 ▪ Melenk, Babuška, The partition of unity finite element method, CMAME 1996

Two lineages, one construction

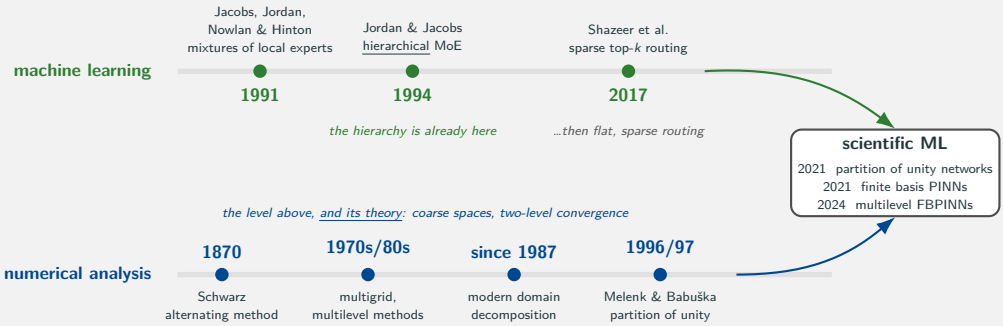




2017: the 1991 formula, but the gate keeps only the k largest weights, so each input visits only a few experts. This is what made mixtures of experts usable in today's large models.

Jordan, Jacobs, Hierarchical mixtures of experts, Neural Comput. 1994 ▪ Shazeer et al., Outrageously large neural networks, ICLR 2017 ▪ Moseley et al. 2023

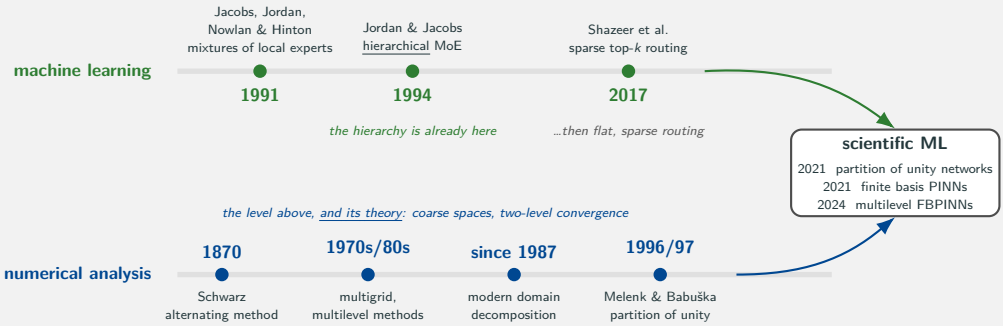
Two lineages, one construction



2017: the 1991 formula, but the gate keeps only the k largest weights, so each input visits only a few experts. This is what made mixtures of experts usable in today's large models.

Jordan, Jacobs, Hierarchical mixtures of experts, Neural Comput. 1994 ▪ Shazeer et al., Outrageously large neural networks, ICLR 2017 ▪ Moseley et al. 2023

Two lineages, one construction



2017: the 1991 formula, but the gate keeps only the k largest weights, so each input visits only a few experts. This is what made mixtures of experts usable in today's large models.

The two lineages met the same object **from opposite ends**: NA already held the answer ML would need years later.

Jordan, Jacobs, Hierarchical mixtures of experts, Neural Comput. 1994 ▪ Shazeer et al., Outrageously large neural networks, ICLR 2017 ▪ Moseley et al. 2023

One structure, different vocabularies



optional, yet it decides whether you scale

	Numerical analysis / DD	Scientific ML	Machine learning
decompose	subdomains (overlapping or not)	subdomains, windows	tokens routed to experts
local approximation	local solve	local subnetwork	expert network
recombine	partition of unity	PU window functions	gate / router, top-k
level above	coarse space, multigrid	multilevel FBPINN	shared experts, global tokens

One structure, different vocabularies



optional, yet it decides whether you scale

	Numerical analysis / DD	Scientific ML	Machine learning
decompose	subdomains (overlapping or not)	subdomains, windows	tokens routed to experts
local approximation	local solve	local subnetwork	expert network
recombine	partition of unity	PU window functions	gate / router, top-k
level above	coarse space, multigrid	multilevel FBPINN	shared experts, global tokens
partition is	geometric, prescribed, smooth	geometric, smooth	learned, data-dependent, hard
why you do it	parallelism, preconditioning	multiscale features, training	more parameters, same computing cost

Same shape, different design choices. The real differences and the research questions lie in the last rows.

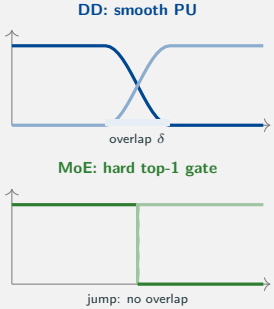
The interesting part is where the analogy breaks

1. Smooth vs. hard

PU / DD theory needs smooth φ_j ($\|\nabla\varphi_j\| \sim 1/\delta$). Sparse top-$k$ routing is **discontinuous**: router instability, abrupt reassignment.

2. Prescribed vs. learned

A geometric partition exists before you start, so a coarse space can be built on it. A learned partition **moves during training**.



3. Different motives

DD decomposes to parallelise and precondition; MoE decomposes to afford more parameters at the same computing cost.

4. Load balancing

DD: graph partitioners (METIS, ...). MoE: an auxiliary **penalty loss** against "routing collapse".

Takeaway

Each box is an open problem where numerical analysis has tools: stability of non-smooth partitions, adaptive coarse spaces, partitioning with guarantees.

Neural PDE solvers and their limits

Physics-Informed Neural Networks (PINNs)

Raissi, Perdikaris and Karniadakis revisited the approach of Lagaris et al. and called it **physics-informed neural networks (PINNs)**. For

$$\mathcal{N}[u](\boldsymbol{x}, t) \quad = \quad f(\boldsymbol{x}, t), \qquad (\boldsymbol{x}, t) \in [0, T] \times \Omega \subset \mathbb{R}^d.$$

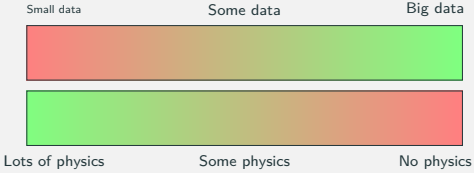
the novelty is a **hybrid loss**:

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

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

$$\mathcal{L}_{\text{data}} \quad = \quad \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(u(x_i, t_i) - u_i \right)^2,$$

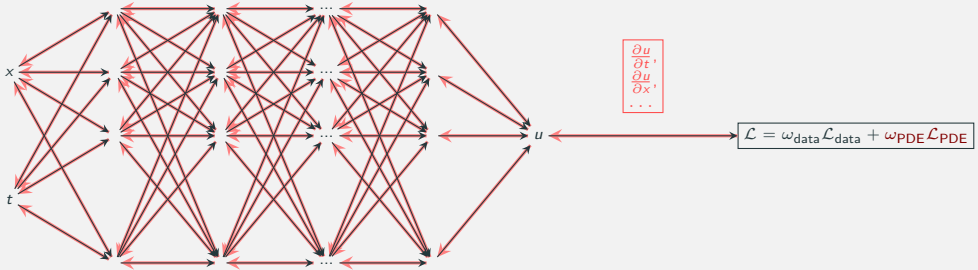
$$\mathcal{L}_{\text{PDE}} \quad = \quad \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{data}}} \left(\mathcal{N}[u](\boldsymbol{x}, t) - f(\boldsymbol{x}, t) \right)^2.$$



- **Known solution values** can be included in $\mathcal{L}_{\text{data}}$.
- **IC and BC** are also included in $\mathcal{L}_{\text{data}}$

Lagaris, Likas, Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE TNN 1998 ▪ Raissi, Perdikaris, Karniadakis, Physics-informed neural networks, JCP 2019

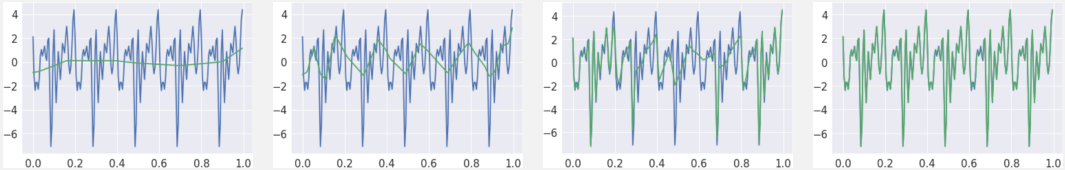
Sketch of the PINN approach by Raissi et al.



Lagaris, Likas, Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE TNN 1998 ▪ Raissi, Perdikaris, Karniadakis, Physics-informed neural networks, JCP 2019

Scaling Issues in Neural Network Training

- **Spectral bias:** Neural networks prioritise learning lower frequency functions first irrespective of their amplitude.



Rahaman, N., et al, *On the spectral bias of neural networks*, **ICML (2019)**
Hong, Siegel, Tan, Xu, *On the Activation Function Dependence of the Spectral Bias of Neural Networks*, **arXiv (2022)**

- Solving solutions on large domains and/or with multiscale features potentially requires **very large neural networks**.
- Training may **not sufficiently reduce the loss** or take **large numbers of iterations**.
- Significant **increase on the computational work**

When PINNs fail to train?

Perdikaris, P. et al, *When and why PINNs fail to train: A neural tangent kernel perspective*, 2022.

Neural tangent kernel (NTK) theory in a nutshell

- Write the gradient descent method at the continuous level ($\mathcal{L}$ is the loss function)

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla\mathcal{L}(\boldsymbol{\theta}),\ \mathcal{L}(\boldsymbol{\theta}), := \omega_b \sum \mathcal{R}(\mathbf{x}_b, \boldsymbol{\theta}(t))^2 + \omega_r \sum \mathcal{R}(\mathbf{x}_r, \boldsymbol{\theta}(t))^2$$

- Residual vectors in the collocation points obey an ODE

$$\frac{d}{dt}\begin{bmatrix} \mathcal{R}(\mathbf{x}_b, \boldsymbol{\theta}(t)) \\ \mathcal{R}(\mathbf{x}_r, \boldsymbol{\theta}(t)) \end{bmatrix} = -K(t)\begin{bmatrix} \mathcal{R}(\mathbf{x}_b, \boldsymbol{\theta}(t)) \\ \mathcal{R}(\mathbf{x}_r, \boldsymbol{\theta}(t)) \end{bmatrix}$$

- The Neural Tangent Kernel $K(t) \to K^*$ for infinitely wide and shallow networks
- Spectral properties if K^* explain the speed of training.

"To provide further insight, we analyze the training dynamics of fully-connected PINNs through the lens of their NTK and show that not only they suffer from spectral bias, but they also exhibit a discrepancy in the convergence rate among the different loss components contributing to the total training error"

A Motivation for Domain Decomposition Approaches

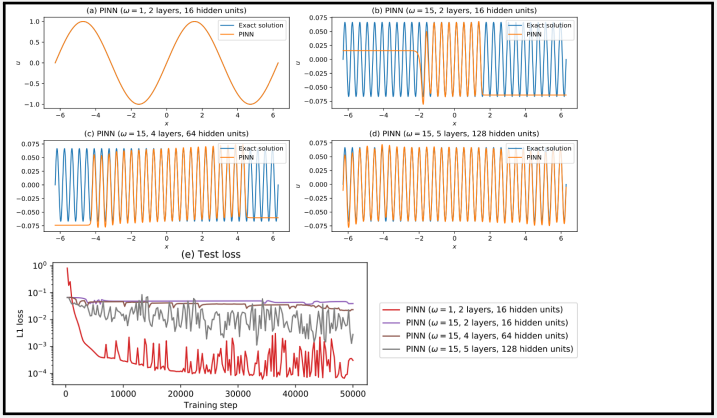
Solve

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

for different values of ω.

Scaling issues

- Size of the computational domain
- Size of frequencies



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

Moseley, Markham, Nissen-Meyer, Finite basis physics-informed neural networks, arXiv:2107.07871, 2021

FBPINNs and the level above

Finite Basis Physics-Informed Neural Networks (FBPINNs)

In the **finite basis physics informed neural network (FBPINNs) method** introduced in **Moseley, Markham, and Nissen-Meyer (arXiv 2021)**, we solve the boundary value problem

$$\begin{aligned} \mathcal{N}[u](\boldsymbol{x}) &= f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \subset \mathbb{R}^d, \\ \mathcal{B}_k[u](\boldsymbol{x}) &= g_k(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma_k \subset \partial\Omega. \end{aligned}$$

using neural networks, we employ the **PINN** approach and **enforce the boundary conditions using a constraining operator**, similar to **Lagaris et al. (1998)**.

Weak enforcement of boundary conditions

Loss function

$$\mathcal{L}(\boldsymbol{\theta}) = w_I \mathcal{L}_{\text{PDE}} + w_B \mathcal{L}_{\text{BC}},$$

where

$$\begin{aligned} \mathcal{L}_{\text{PDE}}(\boldsymbol{\theta}) &= \frac{1}{N_I} \sum_{i=1}^{N_I} (\mathcal{N}[u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i))^2, \\ \mathcal{L}_{\text{BC}}(\boldsymbol{\theta}) &= \frac{1}{N_{\text{B}}} \sum_{i=1}^{N_{\text{B}}} (\mathcal{B}_k[u](\mathbf{x}_i, \boldsymbol{\theta}) - g_k(\mathbf{x}_i))^2. \end{aligned}$$

Hard enforcement of boundary conditions

Loss function

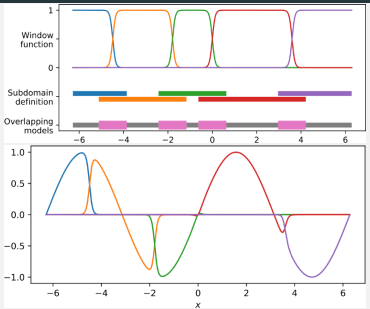
$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{N_I} (\mathcal{N}[\mathcal{C}u](\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i))^2,$$

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

$\rightarrow$ Often **improves training performance**

Moseley, Markham, Nissen-Meyer, Finite basis physics-informed neural networks, arXiv:2107.07871, 2021

FBPINNs – Overlapping Domain Decomposition



- Domain decomposition: $\Omega = \cup_{j=1}^{J} \Omega_j$
- Collocation points (global): $\{x_i\}_{i=1}^{N}$
- Overlapping/interior parts Ω_j° and Ω_j^{int}
- Local solutions u_j, window functions w_j
- Global solution $\mathcal{C}u = \mathcal{C} \sum_{j, x_i \in \Omega_j} \omega_j u_j$

Global loss function

$$\mathcal{L}(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_J) = \underbrace{\frac{1}{N} \sum_{j=1}^{J} \sum_{\mathbf{x}_i \in X_j^\circ} \left(\mathcal{N}[\mathcal{C} \sum_{l, \mathbf{x}_i \in X_l} \omega_l u_l](\mathbf{x}_i, \boldsymbol{\theta}_l) - f(\mathbf{x}_i) \right)^2}_{=: \mathcal{L}^\circ(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_J)} + \underbrace{\frac{1}{N} \sum_{j=1}^{J} \sum_{\mathbf{x}_i \in X_j^{int}} \left(\mathcal{N}[\mathcal{C} \sum_{l, \mathbf{x}_i \in X_l} \omega_l u_l](\mathbf{x}_i, \boldsymbol{\theta}_l) - f(\mathbf{x}_i) \right)^2}_{=: \mathcal{L}^{int}(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_J)}.$$

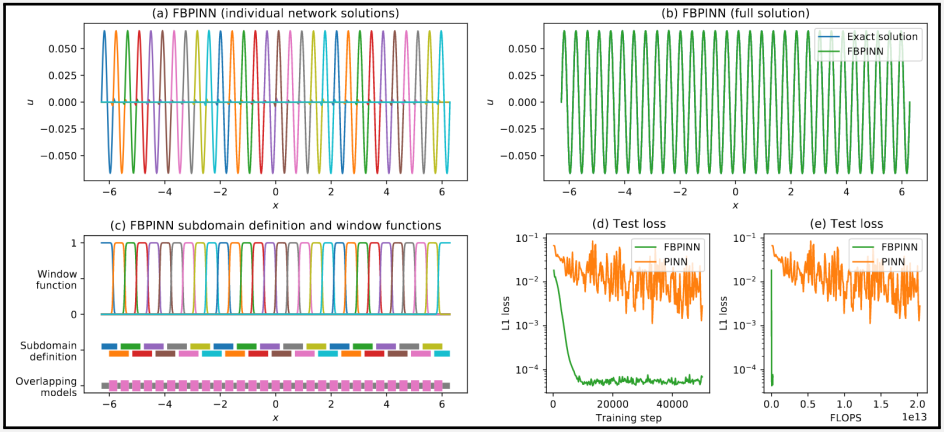
Since $X_i^{int} \cap X_j^{int} = \emptyset$ for $i \neq j$,

$$\mathcal{L}^{int}(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_J) = \frac{1}{N} \sum_{j=1}^{J} \sum_{\mathbf{x}_i \in X_j^{int}} (\mathcal{N}[\mathcal{C} \omega_j u_j](\mathbf{x}_i, \boldsymbol{\theta}_j) - f(\mathbf{x}_i))^2$$

- The subdomains can be split active (trained in parallel) and inactive (fixed)
- This corresponds to classical parallel (all active) or multiplicative (one active at a time) Schwarz methods

Moseley, Markham, Nissen-Meyer, Finite basis physics-informed neural networks, arXiv:2107.07871, 2021

Numerical Results – Comparison of PINNs and FBPINNs



Moseley, Markham, Nissen-Meyer, Finite basis physics-informed neural networks, arXiv:2107.07871, 2021

Step 4 restored: two-level and multilevel FBPINNs

Two-level architecture

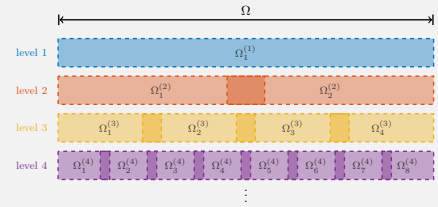
$$u = \mathcal{C}\Big[u_0(x;\theta_0) + \sum_l \omega_l(x)\, u_l(x;\theta_l)\Big]$$

- small **global network** u_0: low frequencies
- local networks: fine-scale, oscillatory content

Multilevel

$$u = \mathcal{C}\Big[\sum_{\ell=1}^{L}\sum_l \omega_l^{(\ell)}(x)\, u_l^{(\ell)}(x;\theta_l^{(\ell)})\Big]$$

a hierarchy of partitions of unity, i.e. a neural multilevel method.



coarse + local models

Same lesson as in DD

One level does not scale. The level above brings it back.

Dolean, Heinlein, Mishra, Moseley, Multilevel domain decomposition-based architectures for PINNs, CMAME (2024)

Multilevel FBPINN Algorithm

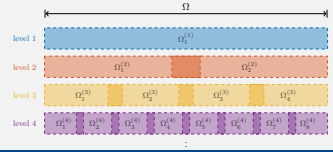
L-level network architecture

Let L **overlapping domain decompositions** $\Omega = \cup_{j=1}^{J^{(l)}} \Omega_j^{(l)}$ and **window functions** $\omega_j^{(l)}$ with

$$\sum_{j=1}^{J^{(l)}} \omega_j^{(l)} \equiv 1, \ \mathrm{supp}(\omega_j^{(l)}) \subset \Omega_j^{(l)}.$$

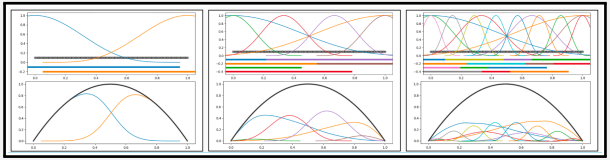
Then the multilevel architecture is

$$u(\mathbf{x}, \boldsymbol{\theta}) = \mathcal{C} \sum_{l=1}^{L} \sum_{j=1}^{J^{(l)}} \omega_j^{(l)} u_j^{(l)}(\mathbf{x}, \boldsymbol{\theta}_j^{(l)})$$

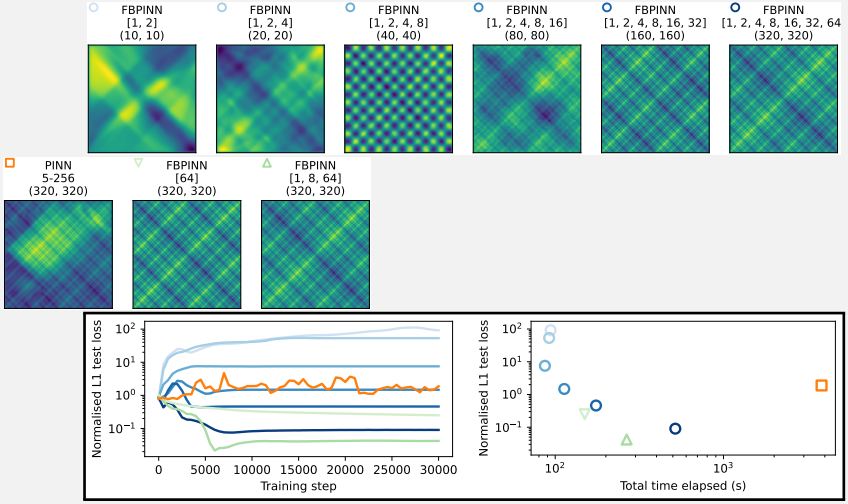


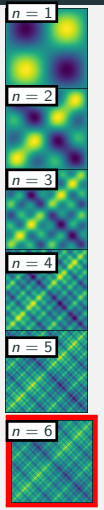
Loss function

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



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





Dolean, Heinlein, Mishra, Moseley, Multilevel domain decomposition-based architectures for PINNs, arXiv:2306.05486, 2023

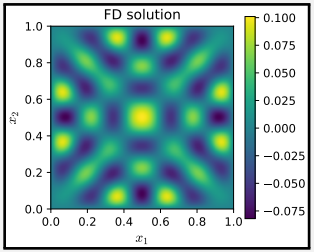
Multilevel FBPINNs – 2D Helmholtz Problem

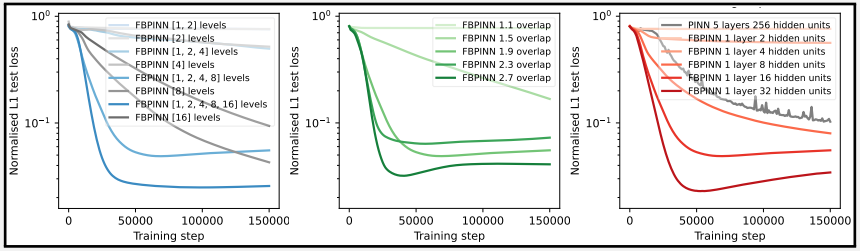
Let us consider the **two-dimensional Helmholtz boundary value problem**

$$\Delta u - k^2 u = f \quad \text{in } \Omega = [0,1]^2,$$
$$u = 0 \quad \text{on } \partial\Omega,$$
$$f(\mathbf{x}) = e^{-\frac{1}{2}(\|\mathbf{x}-0.5\|/\sigma)^2}.$$

with $k = 2^4\pi/1.6$ and $\sigma = 0.8/2^4$.

We compute a **reference solution** using **finite differences** with a 5-point stencil on a 320×320 grid.





Dolean, Heinlein, Mishra, Moseley, Multilevel domain decomposition-based architectures for PINNs, arXiv:2306.05486, 2023

From optimisation to linear algebra

Freeze the hidden layers: learning becomes least squares

Random-feature model

$$\hat{u}(x) = \sum_{k=1}^{m} a_k \, \sigma(w_k^\top x + b_k), \qquad (w_k, b_k) \text{ random, frozen}$$

- a network with **frozen hidden layers**; only the last layer is trained
- linear PDE + collocation $\Rightarrow$ $\min_a \|Ma - b\|_2$: no backpropagation loop

Random feature method for PDEs

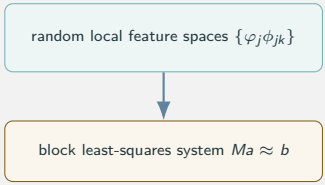
$$\hat{u}(x) = \sum_{j=1}^{J} \varphi_j(x) \sum_{k=1}^{K} a_{jk} \, \phi_{jk}(x; \theta_{jk})$$

This is the partition of unity method, with random local spaces.

Conceptual shift

nonlinear training $\rightsquigarrow$ linear algebra

$\rightsquigarrow$ solvers, preconditioners, guarantees



Takeaway

Learning = approximation space + linear solver.

Rahimi, Recht, Random features for kernel machines, 2007 ▪ Chen, Chi, E, Yang, The random feature method for solving PDEs, 2022

Multilevel random features for multiscale PDEs

Multilevel localisation

$$\hat{u}(x) = \sum_{\ell=1}^{L} \sum_{j} w_j^{(\ell)}(x)\, \hat{u}_j^{(\ell)}(x), \qquad \min_{a} \|Ma - b\|_2$$

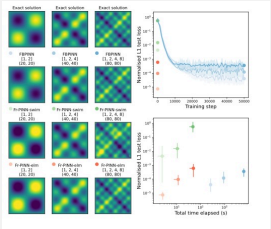
- coarse levels: low frequencies (the level above)
- fine levels: localised oscillations
- one least-squares solve, no gradient descent

Weak scaling

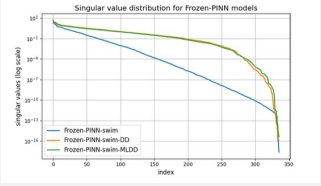
Harder multiscale problems: training time reduced by **120 to 184**$\times$ vs. gradient-trained networks.

Spectral evidence

DD flattens the singular-value spectrum of M; multilevel keeps more informative directions. **Multilevel = scaling and conditioning mechanism.**



weak scaling



singular values of M

Anderson, **Dolean**, Moseley, Pestana, ELM-FBPINN, MLCSE 2026 ▪ Dietrich, Yıldız, Datar, Rahma, **Dolean**, Fast multiscale PDE solvers, ICLR AI&PDE 2026

Local filtering + domain decomposition preconditioning

Step 1: local RRQR filtering

$$M_j \Pi_j = Q_j \begin{pmatrix} R_j & * \\ 0 & T_j \end{pmatrix} \implies \widehat{M}_j = Q_j R_j$$

remove redundant local features: a data-dependent sketch.

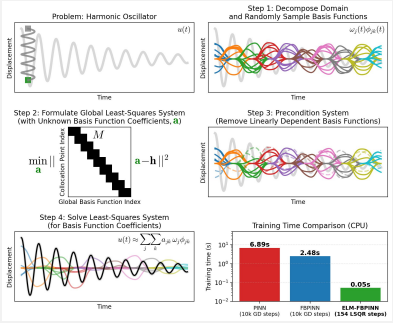
Step 2: additive Schwarz-type right preconditioner

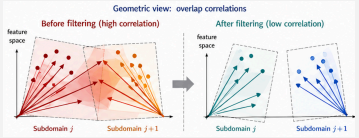
$$\widehat{S}^{-1} = \sum_{j=1}^{J} \widehat{W}_j^\top R_j^{-1} \widehat{W}_j, \qquad \min_z \|M\widehat{S}^{-1} z - b\|_2$$

Structure of the preconditioned system

$$Q^\top Q = I + E, \qquad E \text{ block tridiagonal}, \quad A_j = Q_{j+1,j}^\top Q_{j,j}$$

A_j = interaction of neighbouring subdomains through the overlap.





filtered local bases are less aligned in the overlap ⇒ smaller $\|A_j\|_2$

van Beek, **Dolean**, Moseley, Local feature filtering for domain decomposition random feature methods, CMAME (2026)

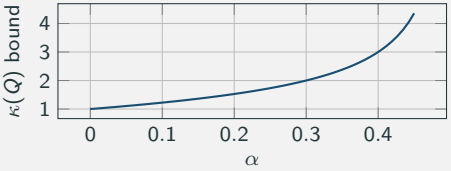
Conditioning: a deterministic bound and its typical behaviour

Theorem (deterministic)

If $\|A_j\|_2 \le \alpha$ for all j and $2\alpha < 1$,

$$\kappa(Q) \le \left(\frac{1+2\alpha}{1-2\alpha}\right)^{1/2},$$

independent of the number of subdomains J and of the total feature dimension.



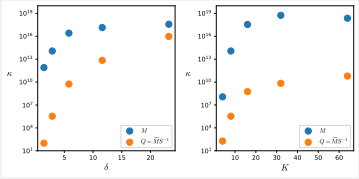
Conditioning: a deterministic bound and its typical behaviour

Theorem (deterministic)

If $\|A_j\|_2 \le \alpha$ for all j and $2\alpha < 1$,

$$\kappa(Q) \le \left(\frac{1+2\alpha}{1-2\alpha}\right)^{1/2},$$

independent of the number of subdomains J and of the total feature dimension.



κ grows with feature dimension K and overlap δ, as predicted

Random matrix view (typical case)

Filtered blocks $\approx$ independent random semi-orthogonal frames:

$$\mathbb{E}\|A_j\| \lesssim \varepsilon^2, \quad \varepsilon = \left(1 + \frac{K}{2n}\right)\frac{\sqrt{K}+\sqrt{\ell}}{\sqrt{n}}$$

K: features, ℓ: overlap, n: local collocation points.

Takeaway

Global conditioning is controlled by **local coupling**: a DD message, now for learned approximation spaces.

van Beek, **Dolean**, Moseley, Local feature filtering for domain decomposition random feature methods, CMAME 2026

The converse direction and open questions

The converse direction: ML enriching classical solvers

Learning the ingredients of DD / multigrid

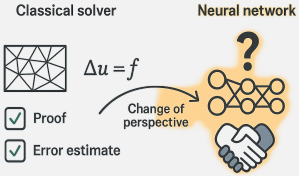
- where to put coarse constraints (adaptive FETI-DP)
- interpolation operators in AMG via graph networks
- optimised interface / transmission conditions

Hybrid solvers

- neural operators as smoothers / coarse solvers for the modes relaxation misses; spectral bias used constructively

Randomisation as structure

- sketching and randomised range finders for coarse spaces and local eigenproblems
- random features as cheap enrichment bases



The common thread

The structure (local + global, stability, convergence) comes from numerical analysis; the components can be learned or randomised.

Heinlein, Klawonn, Lanser, Weber, SISC 2019 ▪ Luz, Galun, Maron, Basri, Yavneh, ICML 2020 ▪ Taghibakhshi et al., NeurIPS 2022 ▪ Zhang, Kahana et al., Nat. Mach. Intell. 2024 ▪ Halko, Martinsson, Tropp, SIREV 2011

Open questions for our community

Learned partitions

What is the level above when the partition is data-dependent and moves during training?

Non-smooth PU

Stability and approximation theory for hard, top-k recombination.

Load balancing

Partitioners with guarantees vs. penalty losses.

Is attention the CS?

Transformers alternate local (MoE) and global (attention) steps: a **two-level method**?

Waves

Locality fails when the Green's function does not decay. The right level above for learned high-frequency solvers?

Benchmarks

Report strong/weak scaling and conditioning, not only the error at fixed size.



"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Turing, Computing Machinery and Intelligence, 1950

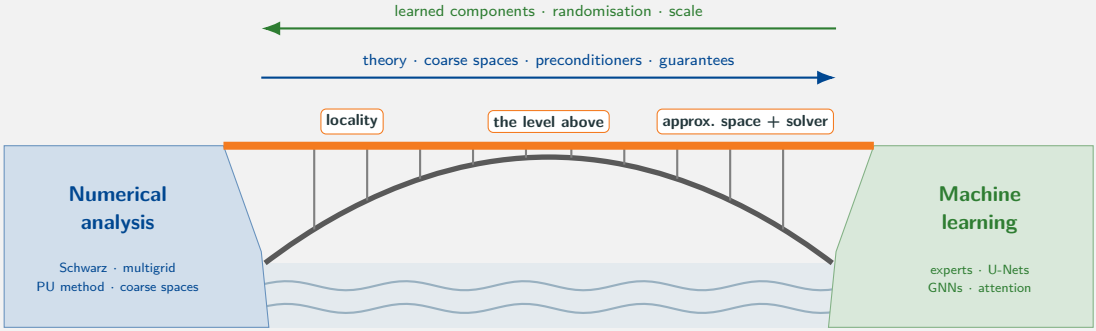
Take-home messages

1. Locality is one structure. Decompose, approximate locally, recombine: in DD, in SciML, in modern ML.

2. Local methods need a level above to scale. Numerical analysis has its theory; ML keeps rediscovering it empirically.

3. PDEs are where the two lineages meet. Multilevel FBPINNs and multilevel random features restore scalability, accuracy and conditioning.

4. Learning = approximation space + linear solver. Conditioning, preconditioning and coarse spaces are where our community is needed.



Thank you!

Divide. Approximate locally. Glue. **And don't forget the level above.**

"If you want to go fast, go alone. If you want to go far, go together."

with thanks to Heinlein, Mishra, Moseley, Anderson, Pestana, van Beek, Dietrich, Yıldız, Datar, Rahma, Graham, Scheichl, Nataf, ...