Preconditioning and domain decomposition methods

Alena Kopaničáková, Carola Kruse, Ronan Guivarch

Iterative methods for linear algebra

Outline

- Introduction/Motivation
- Preconditioning

- Alternating Schwarz methods
- Overlapping Additive/Multiplicative Schwarz methods
- Scalability and coarse space

- Discussion on the implementation aspects
- Discussion regarding the links with multigrid methods

Literature

- Yousef Saad
  Iterative methods for sparse linear systems
  Society for Industrial and Applied Mathematics, 2003

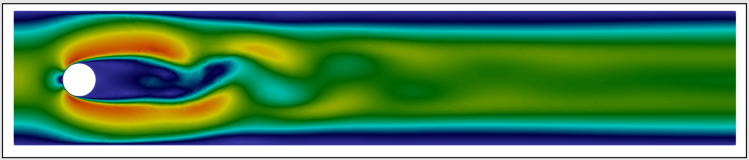
- Andrea Toselli and Olof Widlund
  Domain decomposition methods-algorithms and theory
  Springer Science & Business Media, 2006

- Dolean, Jolivet, Nataf
  An Introduction to Domain Decomposition Methods
  Society for Industrial and Applied Mathematics, Philadelphia, 2015

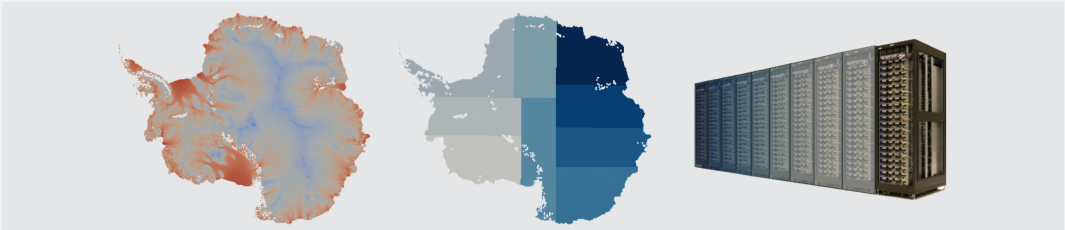
- Jinchao Xu
  Iterative methods by space decomposition and subspace correction
  SIAM review 34(4), pp.581-613, 1992

Goal

Solve complex physical phenomena



Efficiency requires distribution of computational work



Model problem: Let $\Omega \subset \Omega^d$, $d = 2, 3$ be polygonal or polyhedral domain. Our model problem is stationary diffusion problem

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega_D \end{aligned}$$

Linear system of equations:
After performing finite-element, or finite-difference discretization, we obtain the following system to be solved

$$Au = f,$$

where $A \in \mathbb{R}^{n \times n}$, $f \in \mathbb{R}^n$
The exact solution is given by $u^* = A^{-1} f$

Matrix properties:

- symmetric
- positive definite
- sparse

Construction of iterative methods:

Let $M \in \mathbb{R}^{n\times n}$ be invertible. We consider the following splitting:

$$Mu + (A - M)u = f$$

and define an iteration

$$Mu^{(k+1)} + (A - M)u^{(k)} = f$$

This is equivalent to

$$u^{(k+1)} = u^{(k)} + M^{-1}\left(f - Au^{(k)}\right) = u^{(k)} + M^{-1}r^{(k)},$$

where $r^{(k)}$ is the residual in the k-th iteration step. Introducing a relaxation parameter $\alpha \in \mathbb{R}$ to control convergence, we obtain the **(stationary) Richardson method**:

$$u^{(k+1)} = u^{(k)} + \alpha M^{-1}r^{(k)} = \underbrace{\left(I - \alpha M^{-1}A\right)}_{:=B} u^{(k)} + \underbrace{\alpha M^{-1}f}_{:=c}.$$

(Stationary) Richardson method:

$$u^{(k+1)} = u^{(k)} + \alpha M^{-1} r^{(k)} = \underbrace{\left(I - \alpha M^{-1} A\right)}_{:=B} u^{(k)} + \underbrace{\alpha M^{-1} f}_{:=c}.$$

Theorem

The iteration scheme converges if and only if

$$\rho(B) < 1,$$

where $\rho(B) := \max_{\lambda \in \sigma(B)} |\lambda|$ is the **spectral radius** and
$\sigma(B) := \{\lambda \in \mathbb{C} : \lambda \text{ eigenvalue of } B\}$ is the **spectrum** of B.

Theorem

Let all eigenvalues of $M^{-1}A$ be real and satisfy $0 < \lambda_1 \leq \cdots \leq \lambda_n$. Then, the stationary Richardson method converges $\Leftrightarrow 0 < \alpha < \dfrac{2}{\lambda_n}$. The convergence is optimal for $\alpha = \alpha_{\text{opt}} = \dfrac{2}{\lambda_1 + \lambda_n}$ with

$$\rho_{\text{opt}} = \min_{\alpha} \left(\rho(B_\alpha)\right) = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = \frac{\kappa - 1}{\kappa + 1},$$

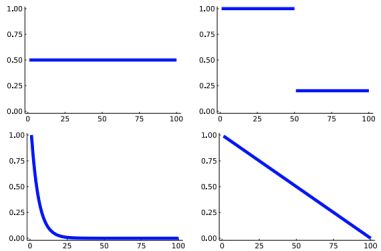
where $\kappa := \kappa(M^{-1}A) = \dfrac{\lambda_n}{\lambda_1}$ is the **condition number** of $M^{-1}A$.

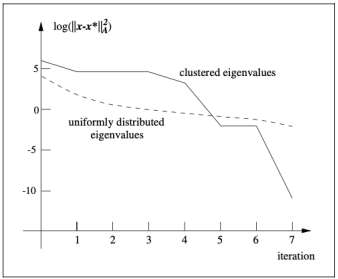
Spectrum and condition number

- Condition number can be used to provide upper bound on convergence - often not sharp
- Condition number for our model problem: $\kappa(A) \leq C\frac{1}{h^2}$, where h denotes the mesh size

  $\Longrightarrow$ obtaining more accurate solution is difficult, as we have larger problem, and worse condition number

- !!! We might encounter different convergence for matrices with the same $\kappa(A)$
- Having clustered eigenvalues often leads to faster convergence





Since the matrix M can be chosen arbitrarily, the condition number and therefore the convergence can be influenced by the choice of M.
The matrix M, or equivalently M^{-1}, is also called a preconditioner.

Requirements for a good preconditioner:

- M^{-1} is a **good approximation** of A^{-1}, that is,

$$M^{-1}A \approx I.$$

As a result

$$\kappa\left(M^{-1}A\right) = \frac{\lambda_{\max}(M^{-1}A)}{\lambda_{\min}(M^{-1}A)} \approx 1$$

$\Rightarrow$ **Rapid convergence!**

Requirements for a good preconditioner:

- **Reliability**: The method should converge to the solution for any initial guess
- **Optimality:** The method can give a solution with $O(n)$ computational cost
- **Efficiency:** The method can give a solution in "reasonably short" wall time
- **Scalability:** The method can scale well on modern parallel architectures
- **Usability:** The method can be implemented and used relatively easily

Requirements for a good preconditioner:

- The condition number of $\kappa(M^{-1}A)$ should be bounded independently of the mesh size
- The **computation or application of M^{-1} should be possible with reasonable costs**
  Remark: It is not necessary to compute M^{-1} explicitly since, in our iterative scheme, it only has to be applicable to a vector

Preconditioner M^{-1} might be:

- Explicit: M^{-1} is known, e.g., incomplete Cholesky, ILU, SPAI, ...
- Implicit: The application of M^{-1} to a vector is known and it has to be computationally efficient, e.g., MG, DD methods, ...

Beyond preconditioning stationary Richardson:
Let $A \in \mathbb{R}^{n\times n}$ be **symmetric positive definite (SPD)**. Then, the following equivalence holds:

$$Au = f \quad \Leftrightarrow \quad x = \arg\min \Phi(x) := \frac{1}{2}u^T Au - u^T f$$

In the application of the preconditioned Richardson method

$$u^{(k+1)} = u^{(k)} + \alpha^{(k)} M^{-1} r^{(k)},$$

we can optimize the error $e^{(k+1)} = u^* - u^{(k+1)}$ in the A-norm:

$$\min_{\alpha^{(k)}} \left\| e^{(k+1)}(\alpha^{(k)}) \right\|_A^2 = \min_{\alpha^{(k)}} \left(Ae^{(k+1)}, e^{(k+1)} \right).$$

This gives us $\alpha^{(k)} = \dfrac{\left(y^{(k)}, r^{(k)}\right)}{\left(Ay^{(k)}, y^{(k)}\right)}$ with $y^{(k)} = M^{-1}r^{(k)} = M^{-1}Ae^{(k)}$

Remark on preconditioning of Krylov methods

- Instead of preconditioning just Richardson, we can also precondition Krylov methods, e.g., CG, GMRES, ...

- Preconditioner must have suitable properties for a given Krylov method, e.g., CG requires SPD preconditioner ...

Preconditioning approaches:

- **Left:** $M^{-1}Au = M^{-1}f$

- **Right:** $AM^{-1}x = f$, with $u = M^{-1}x$

Symmetrization:
In some cases, it might be possible to apply symmetrization algorithm:

$$u^{(k+1/2)} = u^{(k)} + M^{-1}r^{(k)}$$
$$u^{(k+1)} = u^{(k+1/2)} + M^{-T}r^{(k+1/2)}$$

Algorithm 1: (P)CG Method

Result: Approximate solution of the linear equation system $Ax = b$
Given: Initial guess $x^{(0)} \in \mathbb{R}^n$, tol. $\varepsilon > 0$

- $r^{(0)} := b - Ax^{(0)}$ (initial residual)
- $p^{(0)} := y^{(0)} := M^{-1}r^{(0)}$

while $\|r^{(k)}\| \geq \varepsilon \|r^{(0)}\|$ **do**

- $\alpha^{(k)} := \frac{(r^{(k)}, r^{(k)})}{(Ap^{(k)}, p^{(k)})}$
- $x^{(k+1)} := x^{(k)} + \alpha^{(k)} y^{(k)}$
- $r^{(k+1)} := r^{(k)} - \alpha^{(k)} Ap^{(k)}$
- $y^{(k+1)} := M^{-1} r^{(k+1)}$
- $\beta^{(k)} := \frac{(y^{(k+1)}, Ap^{(k)})}{(p^{(k)}, Ap^{(k)})}$
- $p^{(k+1)} := r^{(k+1)} - \beta^{(k)} p^{(k)}$

end

Theorem

Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. Then, for any $x^{(0)} \in \mathbb{R}^n$, the (P)CG method converges to the solution x of the linear system $Ax = b$ in at most n steps.

Theorem

Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite. Then the (P)CG method converges and the following error estimate holds:

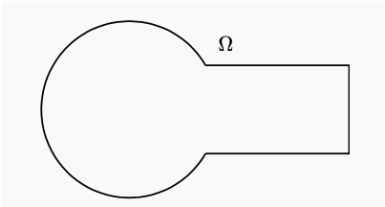
$$\|e^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\kappa(M^{-1}A)} - 1}{\sqrt{\kappa(M^{-1}A)} + 1} \right)^k \|e^{(0)}\|_A,$$

where $\kappa(M^{-1}A)$ is the condition number of the preconditioned system

Continuous DD methods

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

- Schwarz used the algorithm to establish the **existence of harmonic functions** with prescribed boundary values on **regions with non-smooth boundaries**.
- At the core of Schwarz's work is a proof that **this iterative method converges in the maximum norm at a geometric rate**.

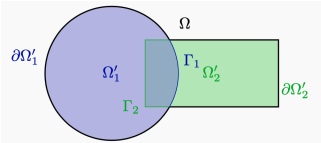


Classical "doorknob" geometry

Overlapping Domain Decomposition

We solve:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega. \end{aligned}$$



We decompose Ω into two overlapping subdomains $\Omega_1', \Omega_2' \subset \Omega$ with

$$\begin{aligned} \Omega &= \Omega_1' \cup \Omega_2', \\ \Gamma_1 &= \partial\Omega_1' \setminus \partial\Omega, \\ \Gamma_2 &= \partial\Omega_2' \setminus \partial\Omega. \end{aligned}$$

The region $\Omega_1' \cap \Omega_2'$ is denoted as the overlapping region of the overlapping domain decomposition. This region is essential for the convergence of the following Schwarz algorithms.

The Schwarz Alternating method

Given an initial guess $u^{(0)}$ which satisfies the boundary condition

$$u^{(0)} = g \quad \text{on } \partial\Omega.$$

We perform the following fixed-point iteration, solving alternately two Dirichlet problems:

$$(D_1) \quad \begin{cases} -\Delta u_1^{(k)} = f & \text{in } \Omega_1, \\ u_1^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_1, \\ u_1^{(k)} = u^{(k-1)} & \text{on } \Gamma_1, \end{cases}$$

$$(D_2) \quad \begin{cases} -\Delta u_2^{(k)} = f & \text{in } \Omega_2, \\ u_2^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_2, \\ u_2^{(k)} = u^{(k)} & \text{on } \Gamma_2 \end{cases}$$

We obtain continuous iterates which satisfy the PDE within Ω_1' and Ω_2'.

Error equation

Let $\{u_1^{(k)}\}_{k\geq 1}$, $\{u_2^{(k)}\}_{k\geq 1}$ be sequences of approximations generated by the Schwarz method.

To study the convergence of $u_1^{(k)} \to u|_{\Omega_1}$ and $u_2^{(k)} \to u|_{\Omega_2}$, it is sufficient to analyze how the quantities

$$e_1^{(k)} := u|_{\Omega_1} - u_1^{(k)}, \quad e_2^{(k)} := u|_{\Omega_2} - u_2^{(k)},$$

converge to zero $e_j^{(k)} \to 0$, $j = 1, 2$.

In our setting, due to linearity, $e_j^{(k)}$ satisfies

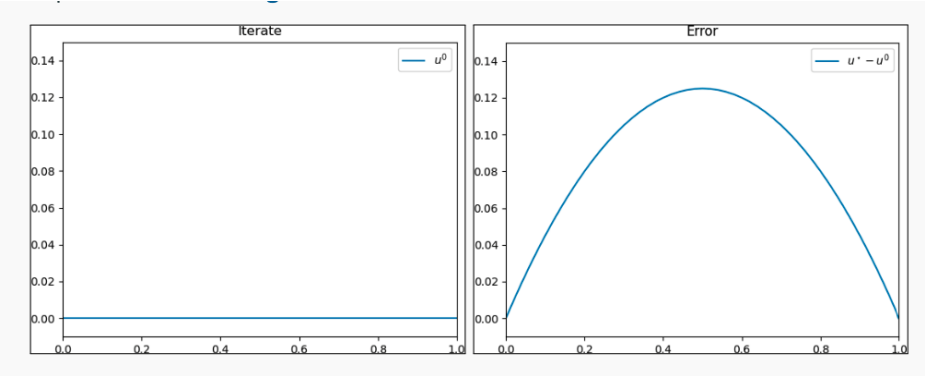
$$\begin{aligned} -\Delta e_1^{(k)} &= 0 && \text{in } \Omega_1, \\ e_1^{(k)} &= 0 && \text{on } \partial\Omega \cap \bar{\Omega}_1, \\ e_1^{(k)} &= e_2^{(k-1)} && \text{on } \Gamma_1, \end{aligned} \qquad \begin{aligned} -\Delta e_2^{(k)} &= 0 && \text{in } \Omega_2, \\ e_2^{(k)} &= 0 && \text{on } \partial\Omega \cap \bar{\Omega}_2, \\ e_2^{(k)} &= e_1^{(k)} && \text{on } \Gamma_2. \end{aligned}$$

Error equation

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

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

We perform an **alternating Schwarz iteration**:



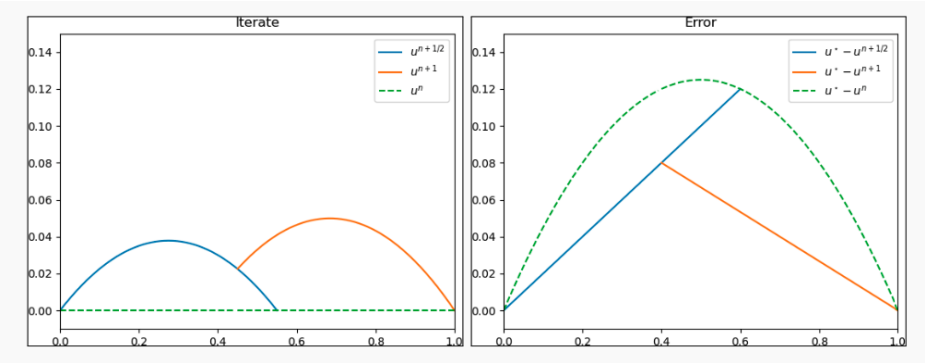
Iterate (left) and error (right) in iteration 0.

Error equation

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

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

We perform an **alternating Schwarz iteration**:



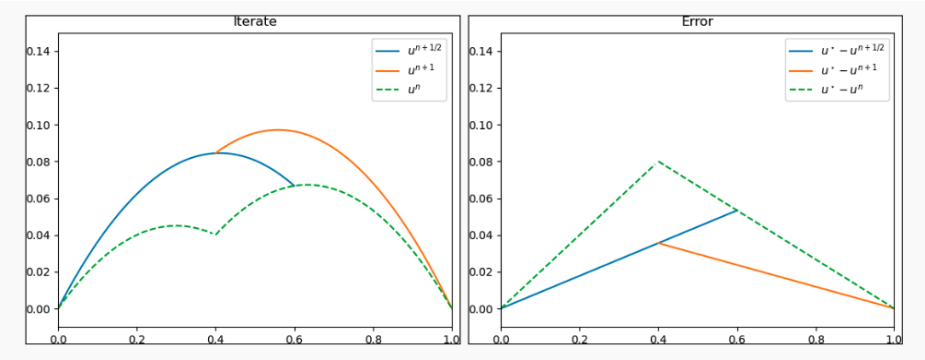
Iterate (left) and error (right) in iteration 1.

Error equation

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

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

We perform an **alternating Schwarz iteration**:



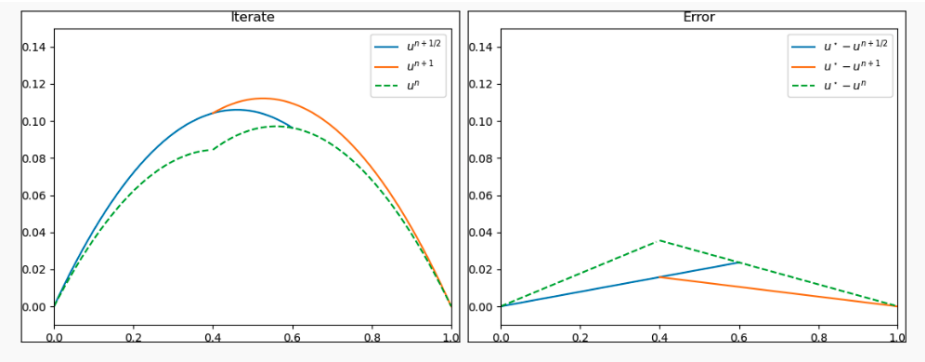
Iterate (left) and error (right) in iteration 2.

Error equation

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

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

We perform an **alternating Schwarz iteration**:



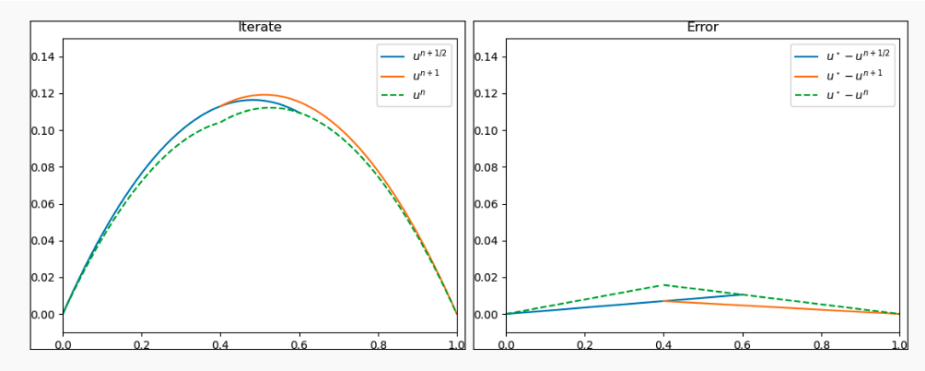
Iterate (left) and error (right) in iteration 3.

Error equation

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

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

We perform an **alternating Schwarz iteration**:



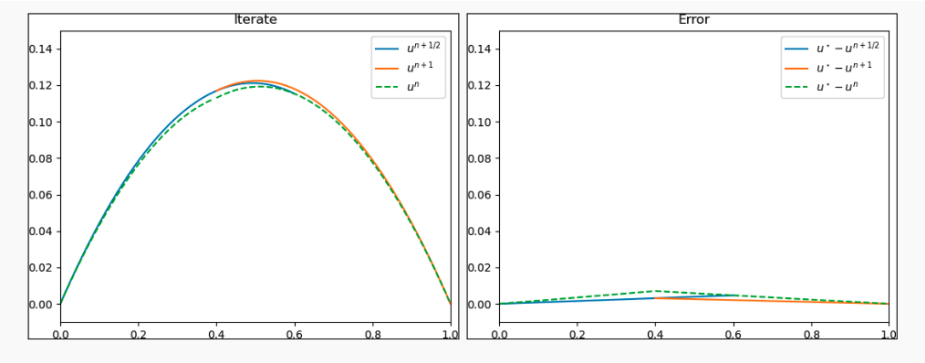
Iterate (left) and error (right) in iteration 4.

Error equation

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

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

We perform an **alternating Schwarz iteration**:



Iterate (left) and error (right) in iteration 5.

Sequential Nature of the Alternating Schwarz Algorithm

The alternating Schwarz algorithm is sequential because each local boundary value problem depends on the values from the previous half-step:

$$(D_1) \quad \begin{cases} -\Delta u_1^{(k)} = f & \text{in } \Omega_1, \\ u_1^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_1, \\ u_1^{(k)} = u^{(k-1)} & \text{on } \Gamma_1, \end{cases}$$

$$(D_2) \quad \begin{cases} -\Delta u_2^{(k)} = f & \text{in } \Omega_2, \\ u_2^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_2, \\ u_2^{(k)} = u^{(k)} & \text{on } \Gamma_2 \end{cases}$$

The Parallel Schwarz Algorithm

Given an initial guess $u^{(0)} =: u_1^{(0)} := u_2^{(0)}$ which satisfies the boundary condition $u^{(0)} = 0$ on $\partial\Omega$. Then, we perform the following fixed-point iteration, solving two Dirichlet problems:

$$(D_1) \quad \begin{cases} -\Delta u_1^{(k)} = f & \text{in } \Omega_1, \\ u_1^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_1, \\ u_1^{(k)} = u^{(k-1)} & \text{on } \Gamma_1, \end{cases}$$

$$(D_2) \quad \begin{cases} -\Delta u_2^{(k)} = f & \text{in } \Omega_2, \\ u_2^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_2, \\ u_2^{(k)} = u^{(k-1)} & \text{on } \Gamma_2 \end{cases}$$

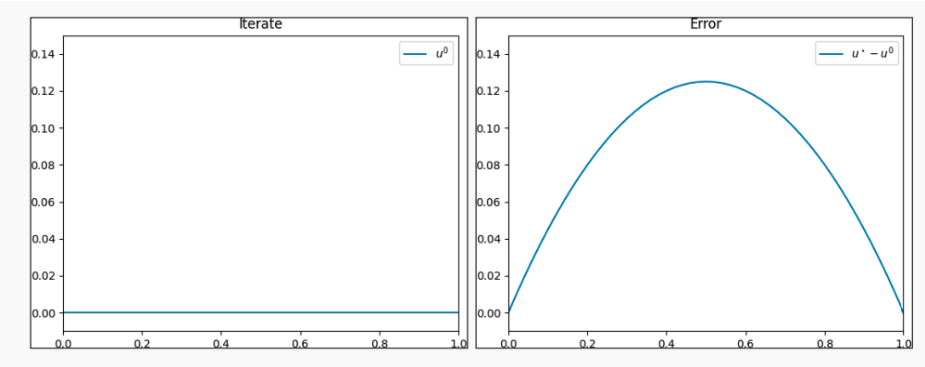
Note: The costs for a single iteration are the same as in the alternating case. However, in parallel computing, we could solve both problems at the same time.

Parallel Schwarz Iteration

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

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

We perform a **parallel Schwarz iteration**:



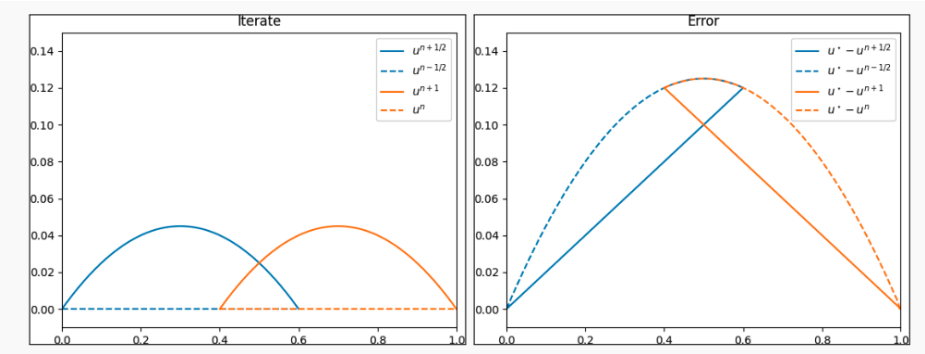
Iterate (left) and error (right) in iteration 0.

Parallel Schwarz Iteration

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

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

We perform a **parallel Schwarz iteration**:



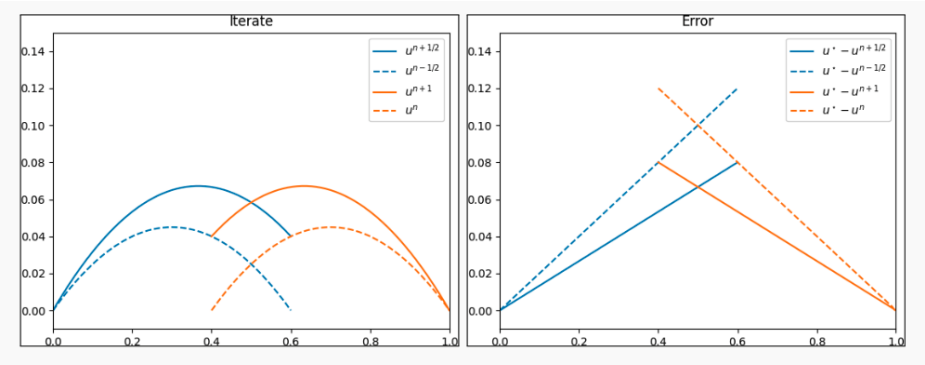
Iterate (left) and error (right) in iteration 1.

Parallel Schwarz Iteration

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

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

We perform a **parallel Schwarz iteration**:



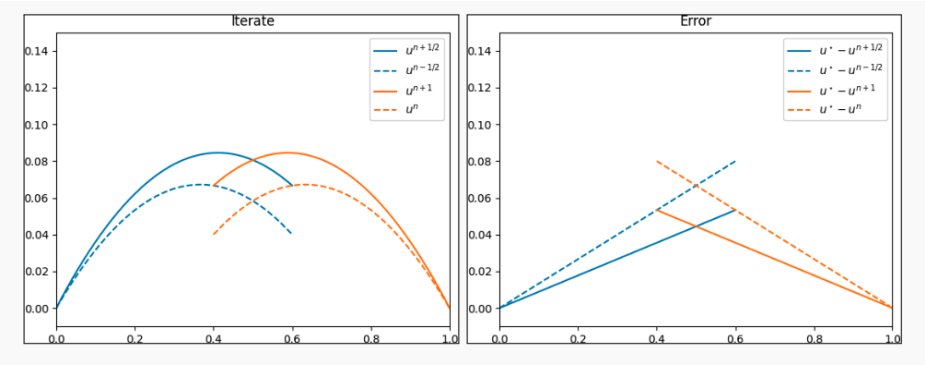
Iterate (left) and error (right) in iteration 2.

Parallel Schwarz Iteration

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

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

We perform a **parallel Schwarz iteration**:



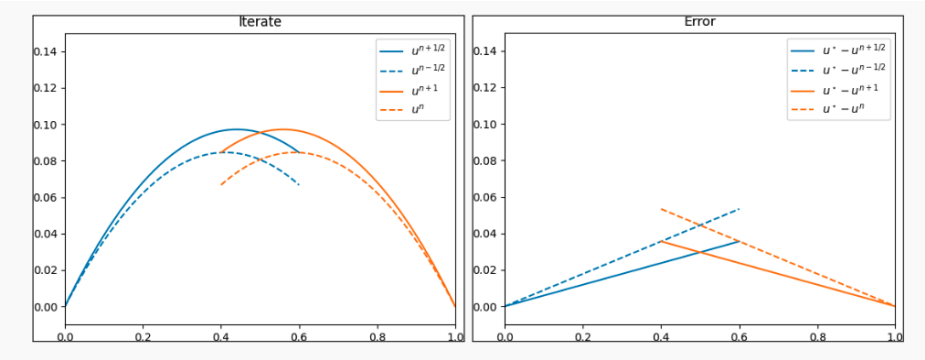
Iterate (left) and error (right) in iteration 3.

Parallel Schwarz Iteration

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

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

We perform a **parallel Schwarz iteration**:



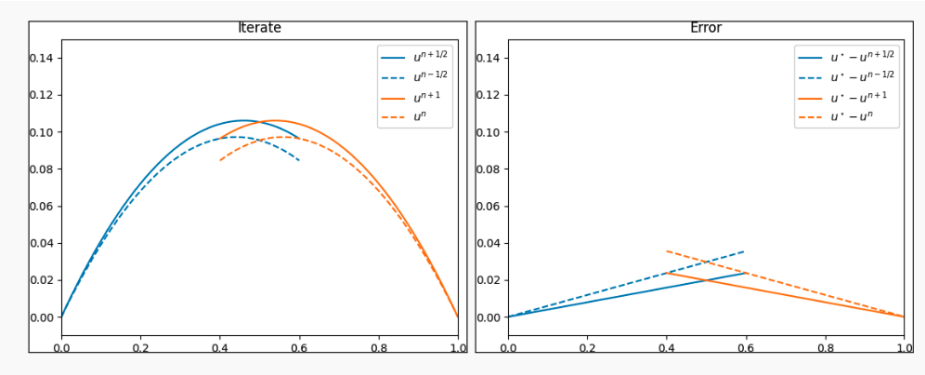
Iterate (left) and error (right) in iteration 4.

Parallel Schwarz Iteration

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

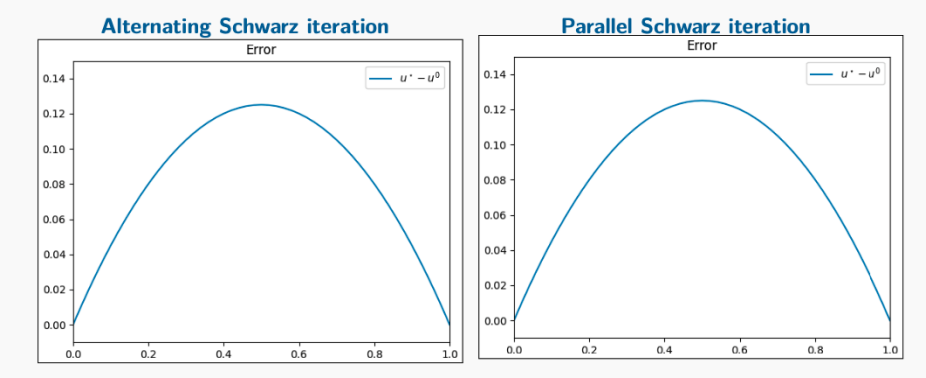
$$-u'' = 1, \quad \text{in } [0,1], \quad u(0) = u(1) = 0.$$

We perform a **parallel Schwarz iteration**:



Iterate (left) and error (right) in iteration 5.

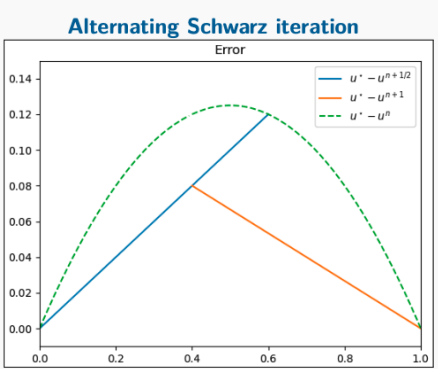
Comparison of the Two Methods



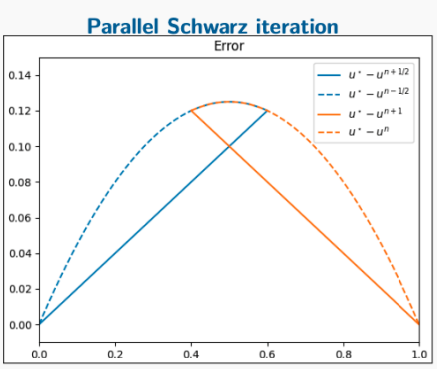
Error in iteration 0.

Error in iteration 0.

Comparison of the Two Methods

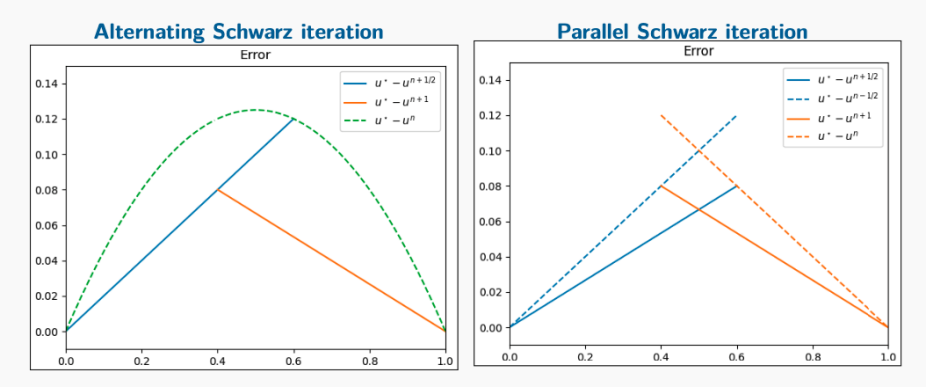


Error in iteration 1.



Error in iteration 1.

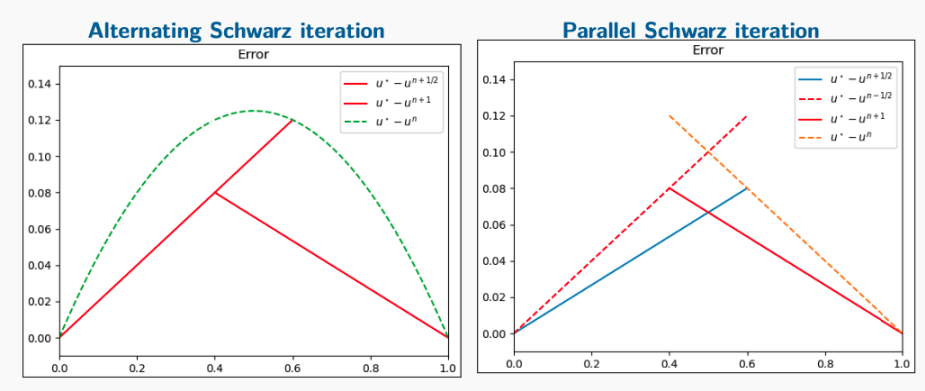
Comparison of the Two Methods



Error in iteration 1.

Error in iteration 2.

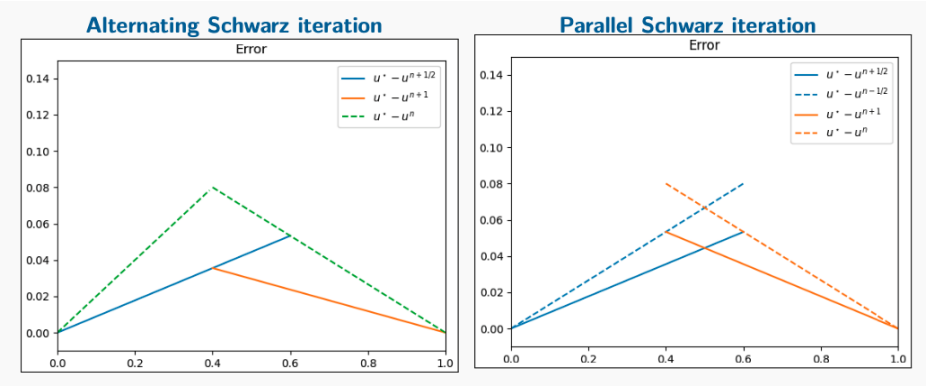
Comparison of the Two Methods



Error in iteration 1.

Error in iteration 2.

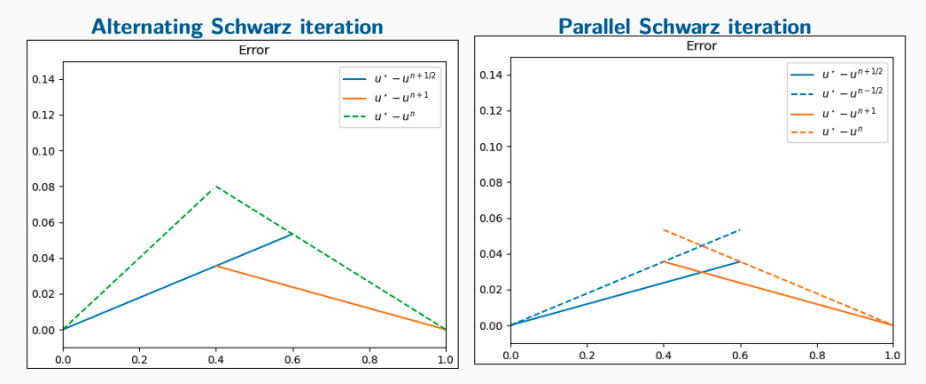
Comparison of the Two Methods



Error in iteration 2.

Error in iteration 3.

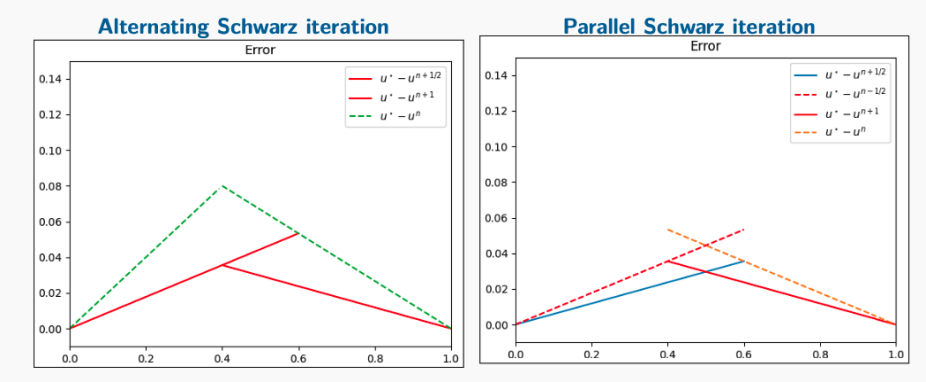
Comparison of the Two Methods



Error in iteration 2.

Error in iteration 4.

Comparison of the Two Methods



Error in iteration 2.

Error in iteration 4.

Define $v_{2,1}^{(k)} := e_2^{(k)}(\Gamma_1)$ and $v_{1,2}^{(k)} := e_1^{(k)}(\Gamma_2)$.

Then,

$$e_1^{(k)}(x) = v_{2,1}^{(k-1)} \frac{x-a}{\Gamma_1 - a} \quad \text{and} \quad e_2^{(k)}(x) = v_{1,2}^{(k-1)} \frac{b-x}{b-\Gamma_2}.$$

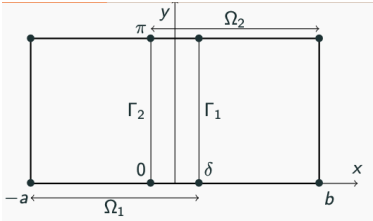
Further,

$$v_{1,2}^{(k)} := e_1^{(k)}(\Gamma_2) = v_{2,1}^{(k-1)} \frac{\Gamma_2 - a}{\Gamma_1 - a} = e_2^{(k-1)}(\Gamma_1) \frac{\Gamma_2 - a}{\Gamma_1 - a} = v_{1,2}^{(k-2)} \underbrace{\frac{b-\Gamma_1}{b-\Gamma_2} \frac{\Gamma_2 - a}{\Gamma_1 - a}}_{\rho}.$$

$$\Rightarrow v_{1,2}^{(k)} = \rho v_{1,2}^{(k-2)} \quad \text{with } \rho < 1 \quad \Rightarrow \text{the Schwarz iterates converge to zero!}$$

- **Conclusion:** The larger the overlap $(\Gamma_1 - \Gamma_2)$, the faster is the contraction!
- **Remark:** The analysis holds for any right-hand side f and boundary condition g (sufficient to look at the error equation $e_j^{(k)} := u|_{\Omega_j} - u_j^{(k)}$).

Convergence Analysis in 2D



$$
\begin{aligned}
-\Delta e_1^{(k)} &= 0 && \text{in } \Omega_1, & -\Delta e_2^{(k)} &= 0 && \text{in } \Omega_2,\\
e_1^{(k)} &= 0 && \text{on } \partial\Omega \cap \bar{\Omega}_1, & e_2^{(k)} &= 0 && \text{on } \partial\Omega \cap \bar{\Omega}_2,\\
e_1^{(k)} &= e_2^{(k-1)} && \text{on } \Gamma_1, & e_2^{(k)} &= e_1^{(k-1)} && \text{on } \Gamma_2.
\end{aligned}
$$

- Expand solutions in Fourier sine series:

$$e_j^{(k)}(x,y) = \sum_{j=1}^{\infty} \hat{e}_j^{(k)}(x,K)\sin(K\pi y).$$

- Substitute into $-\Delta e = 0$, gives

$$\sum_{j=1}^{\infty} (\partial_{xx} - K^2)\hat{e}_j^{(k)}(x,K)\sin(K\pi y) = 0.$$

- Due to orthogonality, we can analyze Schwarz algorithm frequency-by-frequency

(Chaouqui et al, *On the scalability of classical one-level domain decomposition methods*, 2018)
Please, see also the details provided in the supplementary material

$$(\partial_{xx} - K^2)\hat{e}_1^{(k)}(x,K) = 0, x \in (-a,\delta),$$
$$\hat{e}_1^{(k)}(-a,K) = 0,$$
$$\hat{e}_1^{(k)}(\delta,K) = \hat{e}_2^{(k-1)}(\delta,K),$$

$$(\partial_{xx} - K^2)\hat{e}_2^{(k)}(x,K) = 0, x \in (0,b),$$
$$\hat{e}_2^{(k)}(b,K) = 0,$$
$$\hat{e}_2^{(k)}(0,K) = \hat{e}_1^{(k-1)}(0,K)$$

- Using the first and second subdomain equations, coefficients are:

$$\hat{e}_1^{(k)}(x,K) = A^{(k)}(K)\sinh(\pi K(a+x)),$$
$$\hat{e}_2^{(k)}(x,K) = B^{(k)}(K)\sinh(\pi K(b-x))$$

- Using the third equation, we obtain

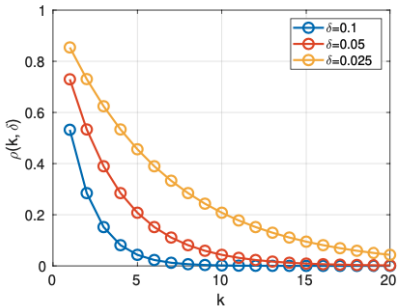
$$A^{(k)}(K) = \underbrace{\frac{\sinh(K\pi a)\sinh(K\pi(b-\delta))}{\sinh(K\pi b)\sinh(K\pi(a+\delta))}}_{\rho(K)} A^{(k-2)}(K),$$

Convergence Rate of Schwarz Method

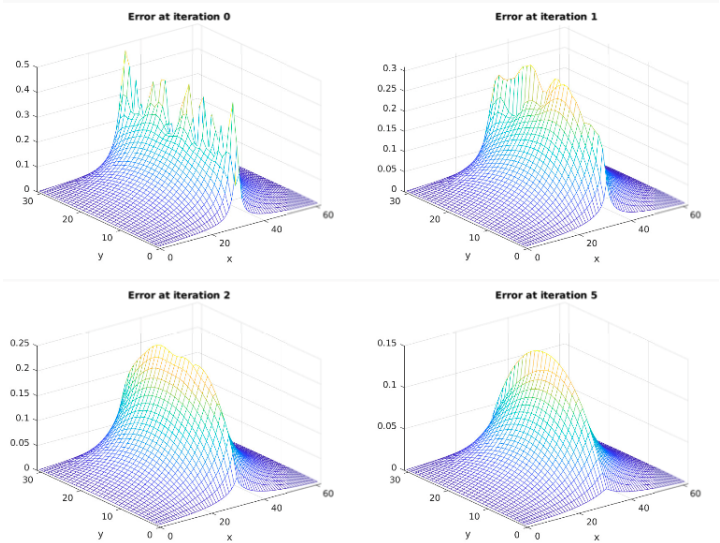
- Convergence factor for the Schwarz method:

$$\rho(K,\delta) := \frac{\sinh(K\pi a)\sinh(K\pi(b-\delta))}{\sinh(K\pi b)\sinh(K\pi(a+\delta))}$$

- The larger the overlap δ, the faster the convergence
- $\rho(K,0) = 1,\ \forall K \Rightarrow$ the Schwarz method **does not converge without overlap!**
- The Schwarz method is an excellent smoother (for $K_1 > K_2$, $\rho(K_1,\delta) >> \rho(K_2,\delta)$)

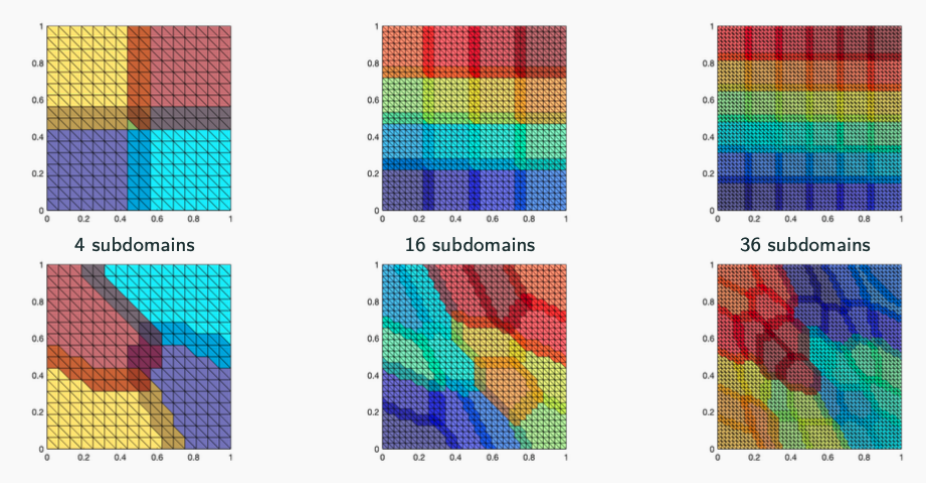


Vizualization of smoothing property



Extension to Multiple Subdomains

The alternating Schwarz method can be extended to multiple subdomains. Therefore, let Ω be decomposed into S overlapping subdomains $\Omega_i,\ i = 1, \ldots, S$, with overlap δ.



Parallel Schwarz Algorithm for S Subdomains

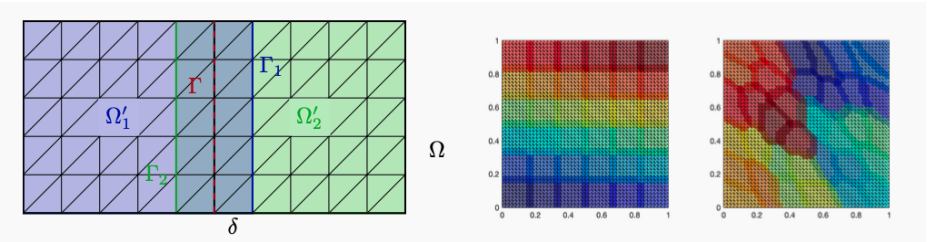
Given an initial guess $u^{(0)}$ which satisfies the boundary condition $u^{(0)} = 0$ on $\partial\Omega$. We perform the following fixed-point iteration, solving Dirichlet problems subdomain by subdomain:

$$(D_s) \quad \begin{cases} -\Delta u_s^{(k)} = f & \text{in } \Omega_s, \\ u_s^{(k)} = g & \text{on } \partial\Omega \cap \bar{\Omega}_s, \\ u_s^{(k)} = u^{(k-1)} & \text{on } \Gamma_s, \end{cases} \qquad s = 1, \ldots, S$$

An alternating variant can be derived in similar fashion.

Discrete Schwarz Preconditioners

Idea: Construct a preconditioner M^{-1} based on solving problems on overlapping subdomains, that is, based on Schwarz domain decomposition methods.



We will introduce the methods for general finite element spaces, but it can be helpful to view them with a specific finite element discretization (e.g., piecewise linear or bilinear finite elements) and type of overlapping domain decomposition in mind.

Space Decomposition

Let us consider finite element spaces $\{V_0, \ldots, V_S\}$ and corresponding interpolation operators (a.k.a. extension or prolongation operators)

$$R_s^T : V_s \to V,$$

for $s = 0, \ldots, S$, such that V can be decomposed in the following way:

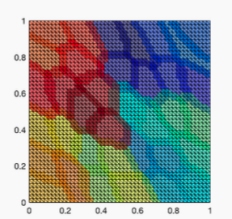
$$V = \sum_{s=1}^{S} R_s^T V_s$$

Remarks:

- The decomposition does not have to be a direct sum: the representation of an element of V in terms of components from the spaces V_s may not be unique.
- We denote the spaces $V_1, \ldots, V_S$ as **local spaces**.

Local Spaces (Specific Example)

As a specific example, we think of finite element spaces corresponding to local overlapping subdomains $\Omega'_1, \ldots, \Omega'_S$ with l layers of elements of overlap.



- V_s correspond to a local triangulation τ_s on Ω'_s
- All functions in V_s vanish on $\partial\Omega'_s$.

This gives rise to the natural extension and restriction operators

$$R_s^T : V_s \to V \quad \text{and} \quad R_s : V \to V_s,$$

for $s = 1, \ldots, S$.

For $v \in V$ and $v_s \in V_s$:

$$R_s v = \begin{cases} v & \text{for all interior nodes of } \Omega'_s, \\ 0 & \text{on } \partial\Omega'_s \end{cases}$$

$$R_s^T v_s = \begin{cases} v_s & \text{on } \Omega'_s, \\ 0 & \text{elsewhere} \end{cases}$$

Example without overlap

Let $S = \{1, 2, 3, 4, 5\}$ be an index set, which we **partition into**

$$S_1 = \{1, 2, 3\}, \quad S_2 = \{4, 5\}.$$

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Example with overlap

Let $S = \{1, 2, 3, 4, 5\}$ be an index set, which we **partition into**

$$S_1 = \{1, 2, 3, 4\}, \quad S_2 = \{3, 4, 5\}.$$

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The subspace correction:

- Let us look for an update δu that provides the "best" improvement for current approximation $u^{(k)}$

$$u^* = u^{(k)} - \delta u$$

- Construct δu by least-square minimization on some smaller space, i.e.,

$$\min_{\delta u \in \mathsf{span}(R_s)} \|\delta u - e^{(k)}\|$$

- Then $\delta u = R_s w$

$$R_s^T R_s w = R_s^T e^{(k)}$$

- Update from the projection method

$$u^{(k+1)} = u^{(k)} + R_s (R_s^T R_s)^{-1} R_s^T e^{(k)}$$

- To ensure, that we get quantity, we can compute we use $\|\cdot\|_A$, thus

$$\min_w \|e - R_s^T w\|_A^2 = (e - R_s^T w)^T A (e - R_s^T w)$$

Taking derivative wrt. w and setting it to zero, we obtain

$$-2R_s A(e - R_s^T w) = 0$$

or

$$R_s A R_s^T w = R_s A e$$

Corrections as projections

- Thus, the subdomain-correction is given by

$$e_s = R_s^T w = R_s^T (R_s A R_s^T)^{-1} R_s (f - A u^{(k)})$$

  In other words, the operator P_s returns the "closest" value to the error

- Let $M^{-1} = (R_s A R_s^T)^{-1}$, then $e_s = M_s^{-1} r$ is the vector closest to e in the subspace spanned by the rows of R_s

Corrections as projections

- Define operator P_s as

$$P_s = M_s^{-1} A$$

- Note,

$$R_s^T (R_s A R_s^T)^{-1} R_s r = M_s^{-1} r = M_s^{-1} A A^{-1} r = P_s e = e_s$$

  is a vector in the subspace spanned by the rows of R_s
- P_s is orthogonal projection (in A-inner product) onto subspace spanned by the rows of R_s, i.e., $P_s = P_s^2$ and when A is SPD, we have

$$(P_s x, y)_A = x^T P_s^T A y = x^T A M_s^{-1} A y = (x, P_s y)_A$$

$\implies$ correction $e_s = M_s^{-1} r$ is the vector closest to e in the subspace spanned by the rows of R_s

Discrete multiplicative Schwarz

A multiplicative Schwarz operator is given by

$$P_{\mathsf{MAS}} = (I - E_{\mathsf{MAS}})A,$$

where the error propagation operator is defined by

$$E_{\mathsf{MAS}} = (I - P_S)(I - P_{S-1}) \cdots (I - P_1),$$
$$E_{\mathsf{MAS}} = \prod_{s=1}^{S} (I - P_s),$$

where P_s is as before, i.e., $P_s = M_s^{-1} A$

Example: block Gauss-Seidel method

- Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix}$$

- Given initial guess $u^{(0)}$, the iterative update is

$$u^{(k+1)} = u^{(k)} + M^{-1} r^{(k)}$$

Iteration rule:

Given $u^{(0)}$, we can write:

$$u^{(k+1/2)} = u^{(k)} + P_1 e^{(k)}$$
$$u^{(k+1)} = u^{(k+1/2)} + P_2 e^{(k+1/2)}$$

or

$$u^{(k+1/2)} = u^{(k)} + \left(R_1^T (R_1 A R_1^T)^{-1} R_1\right) r^{(k)}$$
$$u^{(k+1)} = u^{(k+1/2)} + \left(R_2^T (R_2 A R_2^T)^{-1} R_2\right) r^{(k+1/2)}$$

Let $\{M_s = R_s^T (R_s A R_s^T)^{-1} R_s\}_{s \in [1,2]}$, then

$$u^{(k+1)} = u^{(k)} + (M_1 + M_2 - M_2 A M_1) r^{(k)}$$

which gives rise to the following error propagation

$$e^{(k+1)} = (I - M_2 A)(I - M_1 A) e^{(k)}$$

Properties of the Multiplicative Schwarz Operator

In general, the multiplicative Schwarz preconditioned system

$$P_{\text{MAS}} = M_{\text{ASM}}^{-1} A.$$

is generally not symmetric. Therefore, the multiplicative Schwarz operator is not well-suited for the application of the preconditioned conjugate gradient (PCG) method.
However, it can be used to precondition classical Richardson, or GMRES method
Alternative: Perform the symmetrization

Remark

Multiplicative coupling introduces **serialization** but typically leads to a **more rapid convergence** compared to additive coupling.

Additive Schwarz (AS) preconditioner:

$$M_{AS}^{-1} = \sum_{s=1}^{S} R_s^T (R_s A R_s^T)^{-1} R_s = \sum_{s=1}^{S} M_s^{-1}$$

- If the subdomains are disjoint, M_{AS}^{-1} is a block-Jacobi preconditioner, see example
- Doesn't converge as standalone solver, but OK as a preconditoner

Example: block Jacobi method

- Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}$$

- Given initial guess $u^{(0)}$, the iterative update is

$$u^{(k+1)} = u^{(k)} + M^{-1} r^{(k)}$$

Iteration rule:

Given $u^{(0)}$, we can write:

$$u^{(k+1)} = u^{(k)} + P_1 e^{(k)} + P_2 e^{(k)} = u^{(k)} + \sum_{s}^{2} P_s e^{(k)}$$

or

$$u^{(k+1)} = u^{(k)} + \left(R_1^T (R_1 A R_1^T)^{-1} R_1 + R_2^T (R_2 A R_2^T)^{-1} R_2 \right) r^{(k)}$$

Understanding the condition number of $P = \sum_{s=1}^{S} P_s = \sum_{s=1}^{S} M_s^{-1} A$, where M_s^{-1} is block Jacobi preconditioner

Assumption 1. For any decomposition $u = \sum_{s=1}^{S} u_s$, we have:

$$\sum_{s=1}^{S} \|u_s\|_A^2 = \sum_{s=1}^{S} (Au_s, u_s) \leq K_0 \|u\|_A^2 = K_0 (Au, u)$$

Assumption 2. Each operator P_s is an A-orthogonal projector, i.e.,

$$P_s^2 = P_s \quad \text{and} \quad (P_s u, v)_A = (u, P_s v)_A \text{ for all } u, v$$

This implies that P_i is self-adjoint in the A-inner product and has norm $\|P_s\|_A = 1$.

Understanding the condition number of $P = \sum_{s=1}^{S} P_s = \sum_{s=1}^{S} M_s^{-1} A$

Assumption 3. Straighten Cauchy-Schwarz inequality:

$$\sum_{i=1}^{p} (x_i, y_i) \leq \left(\sum_{i=1}^{p} (x_i, x_i) \right)^{1/2} \left(\sum_{i=1}^{p} (y_i, y_i) \right)^{1/2}$$

Upper Bound on $\lambda_{\max}(P)$

The largest eigenvalue of P satisfies:

$$\lambda_{\max}(P) \leq S,$$

where s is the number of subdomains.

Proof.
For any matrix norm, we have $\lambda_{\max}(P) \leq \|P\|$. In particular, if the A-norm is used, we have:

$$\lambda_{\max}(P) \leq \sum_{s=1}^{S} \|P_s\|_A$$

Each P_s is an A-orthogonal projector, so $\|P_s\|_A = 1$. Hence:

$$\lambda_{\max}(P) \leq S$$

Lower Bound on $\lambda_{\min}(P)$

The lowest eigenvalue of P satisfies:

$$\lambda_{\min}(P) \geq \frac{1}{K_0}$$

Proof. Let $u = \sum u_s$, and note:

$$(u,u)_A = \sum (u_s, u)_A = \sum (P_s u_s, u)_A = \sum (u_s, P_s u)_A$$

Using Cauchy-Schwarz, we get:

$$(u,u)_A \leq \left(\sum (u_s, u_s)_A \right)^{1/2} \left(\sum (P_s u, P_s u)_A \right)^{1/2}$$

Lower Bound on $\lambda_{\min}(P)$

By Assumption 1, this leads to

$$\|u\|_A^2 \leq K_0 \sum (P_s u, P_s u)_A$$

Since P_s is an A-orthogonal projector:

$$\sum (P_s u, P_s u)_A = (Pu, u)_A$$

Thus:

$$(Pu, u)_A \geq \frac{1}{K_0}(u, u)_A \Rightarrow \lambda_{\min}(P) \geq \frac{1}{K_0}$$

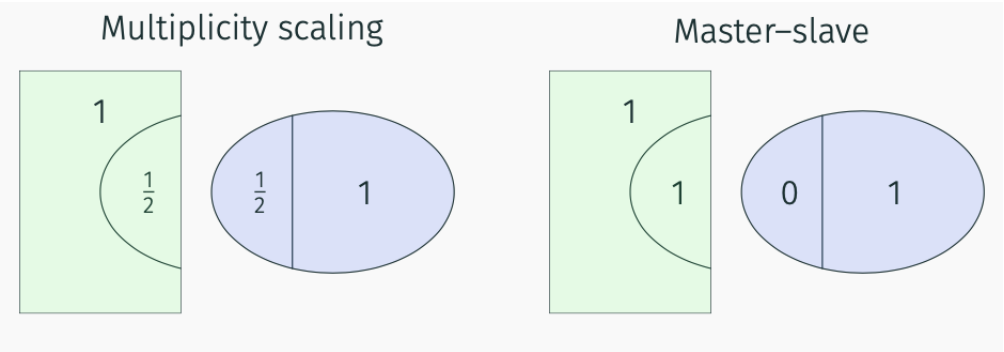
The condition number of preconditioned system is $\implies \kappa(P) = \frac{\lambda_{max}(P)}{\lambda_{min}(P)} \leq S K_0$

Standard ASM doesn't converge as standalone solver

Remedy: Restricted ASM (Cai, Sarkis, 1999)

Idea: Ensure partition of unity, i.e.,

$$\sum_{s=1}^{S} \tilde{R}_s^T R_s = R_s^T D_s R_s = I$$



Example with overlap

Let $S = \{1, 2, 3, 4, 5\}$ be an index set, which we **partition into**

$$S_1 = \{1, 2, 3, 4\}, \quad S_2 = \{3, 4, 5\}.$$

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \end{bmatrix}, \qquad D_2 = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Additive Schwarz (AS) preconditioner:

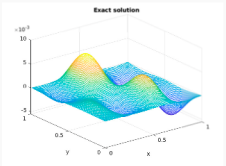
$$M_{AS}^{-1} = \sum_{s=1}^{S} R_s^T (R_s A R_s^T)^{-1} R_s$$

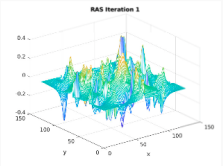
Restricted Additive Schwarz (RAS) preconditioner:

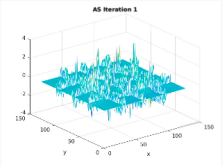
$$M_{RAS}^{-1} = \sum_{s=1}^{S} \tilde{R}_s^T (R_s A R_s^T)^{-1} R_s$$

Comparison between RAS and AS

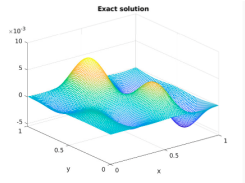
Property	AS	RAS
Works as stationary method	✗[1]	✓
Preserves symmetry	✓	✗
Condition number estimates	✓	✗
Convergence speed measured in # It.	✗	✓

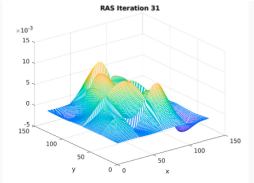


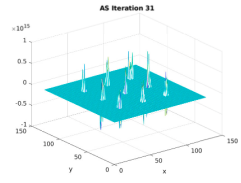




[1] Unless a damping parameter is suitable tuned







Remarks on the One-Level Additive Schwarz Preconditioners

- The major part of the computational work corresponds to the **solution of the completely local problems**. Communication is only necessary in the application of r, R_s, and R_s^T. However, these are typically relatively cheap:
  - A: parallel **matrix-vector multiplication**
  - R_s and R_s^T: **extraction or insertion of values** of a parallel distributed vector

- **Drawback:** With an increasing number of subdomains S, the subdomain size h_s will decrease. Then,

$$\kappa(M_{\mathsf{ASM}}^{-1}A) \leq C\frac{1}{h_s^2}\left(1+\frac{h_s}{\delta}\right) \to \infty$$

  This could only be avoided if overlap is enlarged at the same rate: $h_s\delta \geq C$.
  In practice, this means **enlarging the overlapping subdomains** and hence the size of the local problems $\Rightarrow$ computationally infeasible

Scalability of ASM and design of coarse spaces

Strong Scalability

An algorithm is said to be **strongly scalable** if, for a fixed **total problem size**, the elapsed time is inversely proportional to the number of cores.

Weak Scalability

An algorithm is said to be **weakly scalable** if, for a fixed problem size **per core**, the elapsed time is constant.

Introduction to Domain Decomposition Methods, Dolean et al., 2015

Strong Scalability

An algorithm is said to be **strongly scalable** if, for a fixed **total problem size**, the elapsed time is inversely proportional to the number of **subdomains**.

Weak Scalability

An algorithm is said to be **weakly scalable** if, for a fixed problem size **per subdomain**, the elapsed time is constant.

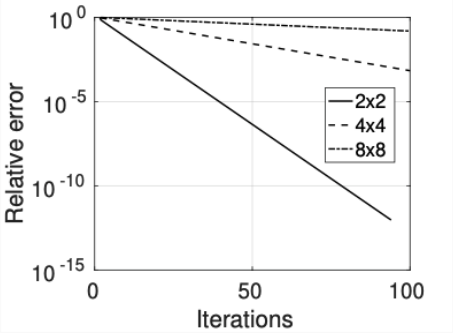
Notes:

- Strong scalability is generally unrealistic to achieve
- One-level domain decomposition methods are typically not weakly scalable

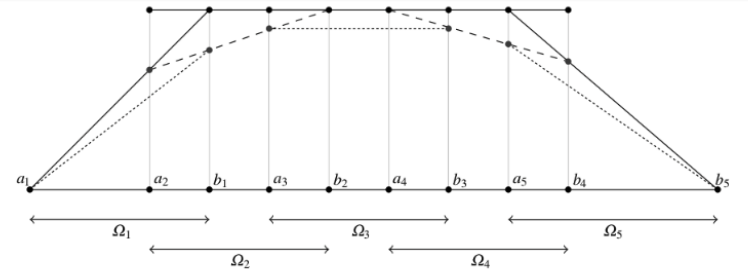
Introduction to Domain Decomposition Methods, Dolean et al., 2015

Convergence of the AS method

Convergence of the parallel Schwarz method applied to $-\Delta u = f$.



Intuition behind the lack of scalability



Generally, 1L-domain decomposition methods suffer the presence of "floating subdomain".

A subdomain Ω_j is said to be floating if $\partial\Omega_j \cap \partial\Omega = \emptyset$.

Idea: Incorporate global information transfer
Let us consider finite element spaces $\{V_0, \ldots, V_S\}$ and corresponding interpolation operators (a.k.a. extension or prolongation operators)

$$R_s^T : V_s \to V,$$

for $s = 0, \ldots, S$, such that V can be decomposed in the following way:

$$V = R_0^T V_0 + \sum_{s=1}^{S} R_s^T V_s$$

Remarks:

- The decomposition does not have to be a direct sum: the representation of an element of V in terms of components from the spaces V_s may not be unique.
- We denote the spaces $V_1, \ldots, V_S$ as **local spaces**.
- V_0 plays a special role: it is typically denoted as the so-called coarse space.

Incorporating global information transfer:

- Let us look for an update δu that provides the "best" improvement for current approximation $u^{(k)}$

$$u^* = u^{(k)} - \delta u$$

- Construct δu by least-square minimization on some smaller space, i.e.,

$$\min_{\delta u \in \mathsf{span}(R_0)} \|\delta u - e^{(k)}\|$$

- Then $\delta u = R_0 w$

$$R_0^T R_0 w = R_0^T e^{(k)}$$

- Update from the projection method

$$u^{(k+1)} = u^{(k)} + R_0 (R_0^T R_0)^{-1} R_0^T e^{(k)}$$

Idea: Use of A-orthogonal projection

- Instead of the standard Euclidean norm, we can employ A-norm $\implies$ employ A-orthogonal projection onto range of R_0
- Construct $\delta u \approx e^{(k)}$ by minimization in A-norm

$$\min_{\delta u \in \mathsf{span}(R_0)} \|\delta u - e^{(k)}\|_A$$

- Then $\delta u = R_0 w$

$$R_0^T A R_0 w = R_0^T A e^{(k)}$$

- Update from the projection method

$$u^{(k+1)} = u^{(k)} + R_0 (R_0^T A R_0)^{-1} R_0^T A e^{(k)}$$
$$u^{(k+1)} = u^{(k)} + R_0 (R_0^T A R_0)^{-1} R_0^T r^{(k)}$$

Two-level overlapping Schwarz preconditioner

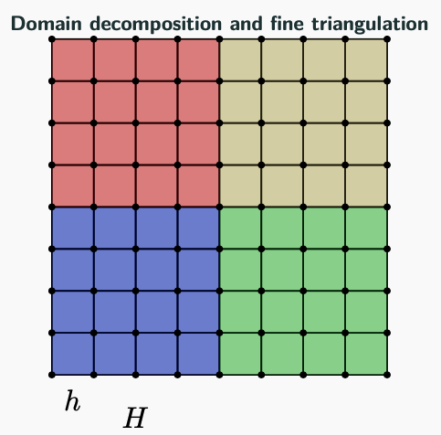
$$M_{\text{2AS-add}}^{-1} = R_0^T A_0^{-1} R_0 + \sum_{s=1}^{S} R_s^T A_s^{-1} R_s,$$

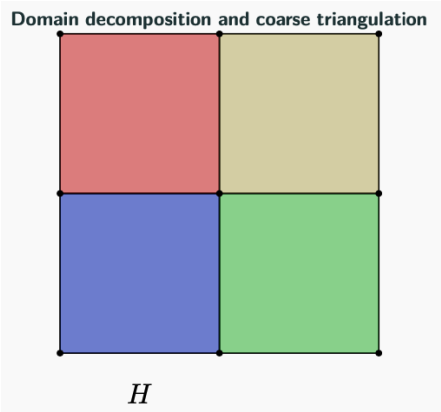
- The **computation of the first level** is the same as in the one-level case
- The **additional coarse problem** may be:
  - solved in parallel on multiple processes or
  - communicated to a single process first and then solved in serial.
- The application of R_0 and R_0^T generally involves communication, and $A_0 = R_0 A R_0^T$ requires **parallel matrix-matrix products**.

See paper by Susanne Brenner from 2000.

Coarse Space (Geometric)

Let the coarse space V_0 be some **finite element space** V_0 on a **coarse triangulation** τ_H with elements that **resolve the nonoverlapping subdomains** $\Omega_1, \ldots, \Omega_N$ corresponding to the overlapping subdomains $\Omega_1', \ldots, \Omega_N'$.





The coarse interpolation operator

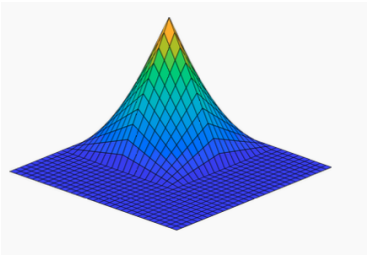
$$R_0^T : V_0 \to V$$

interpolates any function from the coarse space V_0 onto the finite element space V.

In particular, for a nodal finite element basis of V^h, the interpolant of a coarse function $v_0 \in V_0$ can easily be computed using a nodal interpolation, such that

$$R_0^T v_0 = \sum_{i=1}^{n} v_0(x_i)\varphi_i,$$

where $x_1, \ldots, x_n$ are the finite element nodes of V^h and $\varphi_1, \ldots, \varphi_n$ are the corresponding nodal basis functions.



Nicolaides Coarse Space

The Nicolaides approach is to use the kernel of the operator as a coarse space, and the s-th column of R_0 is defined as

$$R_0 := R_s^T D_s R_s \mathbf{1}, \quad 1 \leq s \leq S,$$

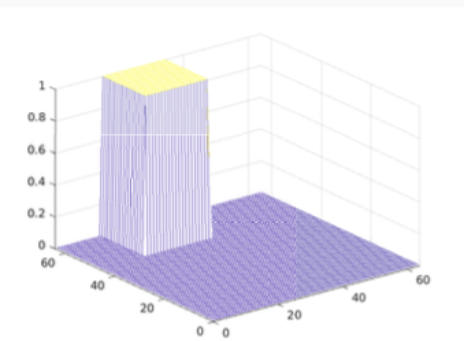
where $\mathbf{1}$ is a vector of all ones, and D_s are chosen so that we have a partition of unity:

$$\sum_{s=1}^{S} R_s^T D_s R_s = \mathbf{I}$$

Nicolaides - 1987

Choose R such that it represents a constant function over subdomain Ω_s and it is zero everywhere else

N. subdomains	4	16	64	128
RAS+Nicolaides	8	26	52	57



Coarse problem couples all subdomains ⇒ inter-subdomains communication!

Condition Number Bound of 2-level AS-add preconditioner

- The condition number bound of the additive two-level overlapping Schwarz operator (Nicolaides):

$$\kappa(M^{-1}_{\text{ASM-2l}}A) \leq C\left(1 + \frac{h_s}{\delta}\right)$$

- This differs significantly from the one-level case!

$\Rightarrow$ The additive two-level overlapping Schwarz method is **numerically (weakly) scalable**

Alternative coarse space construction methods

- **SHEM:** solves eigenvalue problems along the edges of the domain decompositions and extends harmonically.[1]
- **GenEO:** solves specific eigenvalue problems inside the subdomains.[2]
- **GDSW:** harmonic extension of the restriction of the null space of the Neumann matrix to the edges and vertices.[3]
- **Machine-learning**

[1] Gander, Loneland, *SHEM: an optimal coarse space for RAS and its multiscale approximation*, DDXVIII, 2016.
[2] Spillane, Dolean, Huaret, Nataf, Pechstein, Scheichl, *Abstract robust coarse spaces for systems of PDEs via generalized eigenvalues problems in the overlap*, Num. Mat., 2014.
[3] Dorhmann, Klawonn, Widlund, *A Family of Energy Minimizing Coarse Spaces for Overlapping Schwarz Preconditioners*, DXVII, 2008.

Hybrid two-level Schwarz Operators

Additive and multiplicative coupling of Schwarz operators can be combined, resulting in so-called hybrid Schwarz operators
For instance, we can consider

$$P_{\text{hy1}} = I - E_{\text{hy1}}, \quad E_{\text{hy1}} = \left(I - \sum_{s=1}^{S} P_s \right) (I - P_0) \left(I - \sum_{s=1}^{S} P_s \right),$$

where the local components are first coupled in an additive way ($\Longrightarrow$ parallel)
The coarse Schwarz operator is then coupled in a multiplicative way ($\Longrightarrow$ serial)

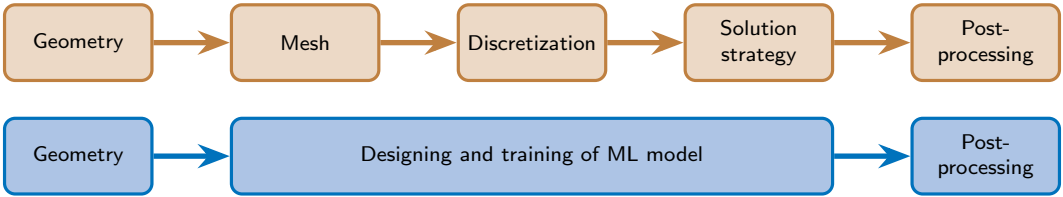
Hybrid two-level Schwarz Operator as 2-level MG with ASM smoothers

- Design of coarse-level is equivalent to AMG with aggregation strategy based on the subdomains decomposition
- Algorithm:

$$
\begin{aligned}
u^{(k+1/2)} &= \sum_{s=1}^{S} P_s r^{(k)} \quad \text{AS pre-smoothing step} \\
r^{(k+1/2)} &= f - A u^{(k+1/2)}, \\
r_c &= R_0 r^{(k+1/2)}, \\
u_c &= A_0^{-1} r_0, \\
u^{(k+1)} &= u^{(k+1/2)} + R_0^T u_c \\
r^{(k+1)} &= f - A u^{(k+1)}, \\
u^{(k+3/2)} &= \sum_{s=1}^{S} P_s r^{(k+1)} \quad \text{AS post-smoothing step}
\end{aligned}
$$

DD for machine-learning

Motivation behind using PINNs



"Classical" numerical methods

- High-fidelity solutions using advanced methods from numerics, scientific computing, optimization, ...
- Capture physics on multiple scales
- Encounter challenges in data integration

Scientific machine-learning (SciML)

- Exceptionally cheap low-fidelity surrogates using novel machine-learning (ML)
- Discover systems dynamics
- Seamless integration of data

What are Physics-Informed Neural Networks (PINNs)?

- Neural network that approximates the solution of a PDE.
- Physics laws (e.g., PDEs) are enforced via the loss function.
- Integration of the data and physics
- "Mesh-free" (complex geo, high-dim. problems)
- Discovery of physics and model parameters
- Initial related work[4,5,6]

[4] Dissanayake, Phan-Thien. "Neural-network-based approximations for solving partial differential equations." 1994.

[5] Lagaris, Isaac E., Aristidis Likas, and Dimitrios I. Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations." 1998.

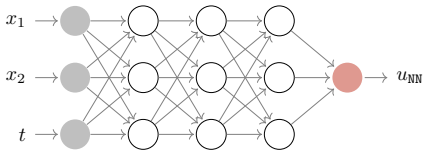
[6] Raissi et al. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." 2017.

Physics-informed neural networks (PINNs)[7]

For a fixed $\boldsymbol{\alpha} \in \boldsymbol{A}$, find $u : \Omega \to \mathbb{R}$, such that

$$\mathcal{P}(\boldsymbol{\alpha}, u(\boldsymbol{x})) = f(\boldsymbol{\alpha}, \boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega,$$
$$\mathcal{B}^k(\boldsymbol{\alpha}, u(\boldsymbol{x})) = g^k(\boldsymbol{\alpha}, \boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma^k \subset \partial\Omega, \quad \text{for } k = 1, 2, \ldots, n_\Gamma,$$

where $\mathcal{P}$ denotes a nonlinear differential operator, and $\{\mathcal{B}^k\}_{k=1}^{n_\Gamma}$ is a set of BC/IC operators



Goal: Approximate solution $u(\boldsymbol{x})$ using neural network, i.e., $u(\boldsymbol{x}) \approx u_{\mathtt{NN}} = \mathrm{DNN}(\boldsymbol{\theta}; \boldsymbol{x})$

[7] Raissi et al. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics. 2019.

PINNs: Hybrid loss minimization

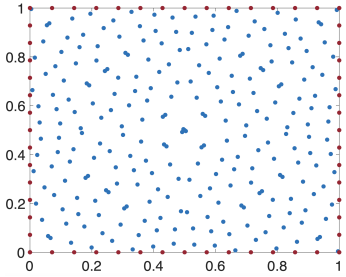
$$\text{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^n}\ \mathcal{L}(\boldsymbol{\theta}) := \underbrace{\mathcal{L}_{\text{int}}(\boldsymbol{\theta})}_{\text{interior loss}} + \underbrace{\mathcal{L}_{\text{bc}}(\boldsymbol{\theta})}_{\text{boundary loss}} + \underbrace{\mathcal{L}_{\text{dat}}(\boldsymbol{\theta})}_{\text{data loss}},$$

where

$$\mathcal{L}_{\text{int}}(\boldsymbol{\theta}) := \frac{w_{\text{int}}}{|\mathcal{D}_{\text{int}}|} \sum_{\boldsymbol{x}_j\in\mathcal{D}_{\text{int}}} \|\mathcal{P}(u_{\text{NN}}(\boldsymbol{\theta},\boldsymbol{x}_j)) - f(\boldsymbol{x}_j)\|^2,$$

$$\mathcal{L}_{\text{bc}}(\boldsymbol{\theta}) := \sum_{k=1}^{n_\Gamma} \left(\frac{w_{\text{bc}}}{|\mathcal{D}_{\text{bc}}^k|} \sum_{(\boldsymbol{x}_j^k,\boldsymbol{g}_j^k)\in\mathcal{D}_{\text{bc}}^k} \|(u_{\text{NN}}(\boldsymbol{\theta},\boldsymbol{x}_j^k) - g^k(\boldsymbol{x}_j^k)\|^2 \right)$$

$$\mathcal{L}_{\text{dat}}(\boldsymbol{\theta}) := \frac{w_{\text{dat}}}{|\mathcal{D}_{\text{dat}}|} \sum_{(\boldsymbol{x}_j,\boldsymbol{d}_j)\in\mathcal{D}_{\text{dat}}} \|(u_{\text{NN}}(\boldsymbol{\theta},\boldsymbol{x}_j) - d(\boldsymbol{x}_j)\|^2$$



Sampling techniques: Monte Carlo, quasi Monte Carlo (Sobol, Halton, ...), Latin Hybercube, adaptive strategies, ...

Example: Solving $-u''(x) = f(x)$

- Problem setup:
  - Domain: $x \in (0, 1)$
  - Dirichlet BCs specified by g function
  - No experiemental data available

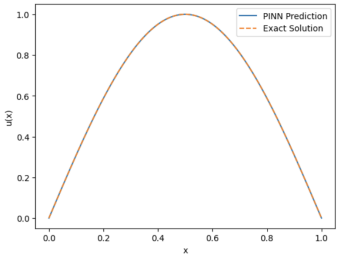
- Loss function:

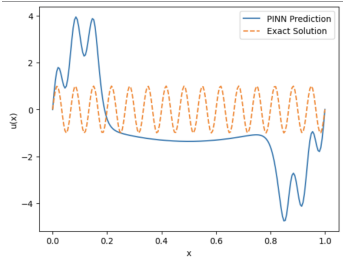
$$\mathcal{L} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left(-u''_\theta(x_f^i) - f(x_f^i)\right)^2 + \frac{1}{N_b} \sum_{j=1}^{N_b} \left(u_\theta(x_b^j) - g(x_b^j)\right)^2$$

- Training = Minimization of $\mathcal{L}$ using gradient-based optimization

Challenges with PINNs

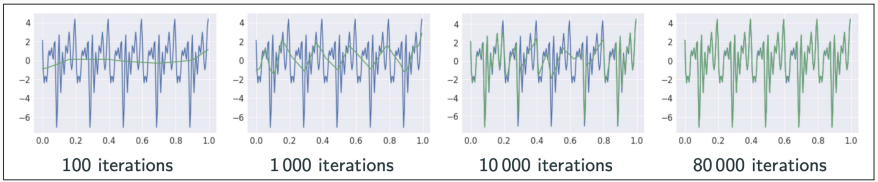
- Training instability and convergence issues
- Learning oscillatory functions
- Unstructured, and large domains and high-dimensional PDEs





- Poor performance in high dimensions
- Balancing multiple loss terms
- Choice of collocation points

Spectral bias [8]

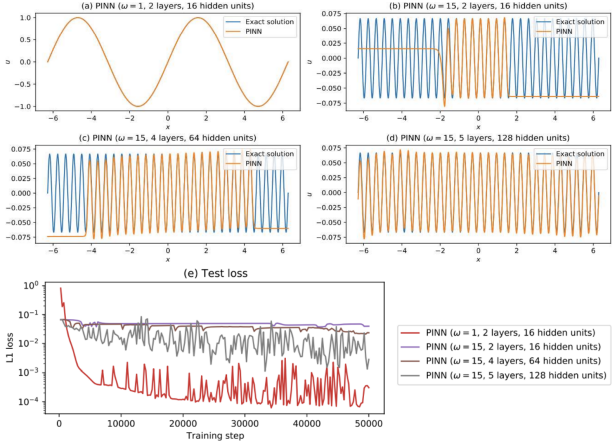


Rahaman et al., *On the spectral bias of neural networks*, **ICML (2019)**

- 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

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

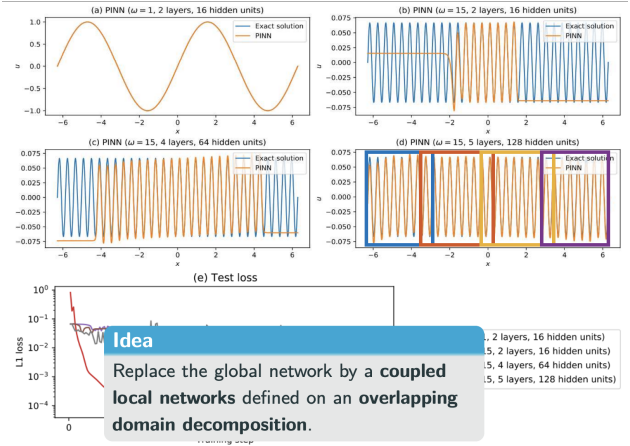
Motivation for DD-based approaches (Poisson problem) [9]



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

[9] Moseley et al. Finite basis Physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations.

Motivation for DD-based approaches (Poisson problem) [10]

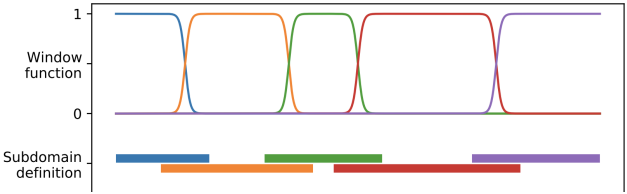


(a) 321 free parameters

(d) 66 433 free parameters

[10] Moseley et al. Finite basis Physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations.

FB-PINN: Domain-decomposition based PINN [11]



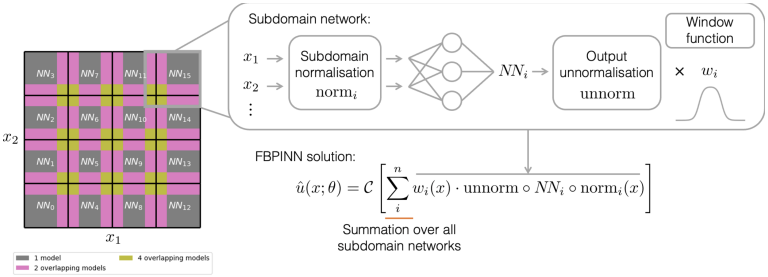
- Uses a collection of neural networks defined on overlapping subdomains
- The global solution is assembled using a partition of unity:

$$u(x) \approx \sum_{i=1}^{M} w_i(x) N_i(x; \theta_i)$$

- $w_i(x)$ are smooth weighting functions with compact support
- N_i are subnetwork approximations trained independently

[11] Moseley et al. Finite basis Physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations.

FB-PINN workflow



- Computational domain is subdivided by overlapping subdomains $\{\Omega_s\}$
- Each subdomain has associated sub-net
- Parallel training of subnetworks
- The global solution is smoothly blended using window functions $\{w_s(x)\}$

Advantages of FB-PINNs

- Scalable to large, high-dimensional domains
- Fully parallelizable architecture
- Modular: each subnetwork can be trained and updated independently
- Improves accuracy and convergence in stiff/multiscale problems