Sparsity Pattern: Dynamic Pattern Finding

- Start with thin approximate pattern $\mathcal{J}_k$ for M_k
- Compute optimal column $M_{k,\text{opt}}(\mathcal{J}_k)$ by least squares

Sparsity Pattern: Dynamic Pattern Finding

- Start with thin approximate pattern $\mathcal{J}_k$ for M_k
- Compute optimal column $M_{k,\text{opt}}(\mathcal{J}_k)$ by least squares
- Find new entry j for M_k such that $M_{k,\text{opt}}(\mathcal{J}_k) + \lambda e_j$ has smaller residual in the Frobenius norm.

$$\begin{aligned}\min_{M_k \in \mathcal{P}_k} \left\| A(M_k + \lambda e_j) - e_k \right\|_2^2 &= \min_{M_k \in \mathcal{P}_k} \left\| (AM_k - e_k) + \lambda A e_j \right\|_2^2 = \\ &= \min \left(\|r_k\|_2^2 + 2\lambda (r_k^T A_j) + \lambda^2 \|A_j\|_2^2 \right)\end{aligned}$$

For each possible index candidate j compute

$$\lambda_j = -\frac{r_k^T A_j}{\|A_j\|_2^2}$$

Sparsity Pattern: Dynamic Pattern Finding

- Start with thin approximate pattern $\mathcal{J}_k$ for M_k
- Compute optimal column $M_{k,\text{opt}}(\mathcal{J}_k)$ by least squares
- Find new entry j for M_k such that $M_{k,\text{opt}}(\mathcal{J}_k) + \lambda e_j$ has smaller residual in the Frobenius norm.

$$\begin{aligned}\min_{M_k \in \mathcal{P}_k} \left\| A(M_k + \lambda e_j) - e_k \right\|_2^2 &= \min_{M_k \in \mathcal{P}_k} \left\| (AM_k - e_k) + \lambda A e_j \right\|_2^2 = \\ &= \min \left(\|r_k\|_2^2 + 2\lambda (r_k^T A_j) + \lambda^2 \|A_j\|_2^2 \right)\end{aligned}$$

For each possible index candidate j compute

$$\lambda_j = -\frac{r_k^T A_j}{\|A_j\|_2^2}$$

Choose index j with $r_k^T A_j \neq 0$ and $j = \arg\min(\lambda_j)$ since

$$\min = \|r_k\|_2^2 - \frac{(r_k^T A_j)^2}{\|A_j\|_2^2}.$$

Error Estimates

- Assume that for all columns holds

$$\|r_k\| = \|AM_k - e_k\| < \epsilon$$

- Then

$$\begin{aligned} \|AM - I\|_F &\leq \sqrt{n}\epsilon \\ \|AM - I\|_2 &\leq \sqrt{n}\epsilon \\ \|AM - I\|_1 &\leq \sqrt{p}\epsilon \end{aligned}$$

with p being the maximum number of nonzero entries in r_k, $k = 1, \ldots, n$.

Overview Following SPAI Variants

1. Block SPAI
2. Iterative SPAI
3. Factorized SPAI (FSPAI)
4. Modified SPAI (MSPAI)

Further Variants of SPAI

1. Block SPAI

- Partition the matrix M in small blocks (e.g., 2×2 or 3×3) and apply the Frobenius norm minimization with blockwise pattern.
- Advantages:
  - Underlying block structure will also appear in the pattern of $A^{-1} \rightarrow$ improved pattern
  - Block operations are more efficient
  - Less least squares problems to solve

Further Variants of SPAI

1. Block SPAI

- Partition the matrix M in small blocks (e.g., 2×2 or 3×3) and apply the Frobenius norm minimization with blockwise pattern.
- Advantages:
  - Underlying block structure will also appear in the pattern of $A^{-1} \rightarrow$ improved pattern
  - Block operations are more efficient
  - Less least squares problems to solve

2. Iterative SPAI

- Start with pattern of $A \rightarrow M_1$
- Construct M_2 relative to new matrix AM_1
- Construct M_3 relative to new matrix AM_1M_2
- ...
- Advantage: cheaper (but inferior approximation)

3. Factorized SPAI, FSPAI

- Parallel Preconditioner for SPD matrices
- Approximate the inverse Cholesky factor of $A = L_A^T L_A$

$$A^{-1} = (L_A^T L_A)^{-1} = L_A^{-1} L_A^{-T} \approx L L^T$$

$$L_A^{-1} \approx L \qquad \rightarrow \qquad \min \| L_A L - I \|_F$$

- Normal equations give $A(\mathcal{J}_k, \mathcal{J}_k) L(\mathcal{J}_k) = \alpha e_k$

3. Factorized SPAI, FSPAI

- Parallel Preconditioner for SPD matrices
- Approximate the inverse Cholesky factor of $A = L_A^T L_A$

$$A^{-1} = (L_A^T L_A)^{-1} = L_A^{-1} L_A^{-T} \approx L L^T$$

$$L_A^{-1} \approx L \qquad \rightarrow \qquad \min \| L_A L - I \|_F$$

- Normal equations give $A(\mathcal{J}_k, \mathcal{J}_k) L(\mathcal{J}_k) = \alpha e_k$

4. Modified SPAI, MSPAI: Combining Targeting and Probing in classical SPAI formulation.

Targeting (for MSPAI)

$$\min_{M \in \mathcal{P}} \|AM - T\|_F$$

- Assume, T is a good sparse preconditioner for A. Improve T by computing M and solving the above Frobenius norm minimization.

Targeting (for MSPAI)

$$\min_{M \in \mathcal{P}} \|AM - T\|_F$$

- Assume, T is a good sparse preconditioner for A. Improve T by computing M and solving the above Frobenius norm minimization.
- Application: Assume that A is given by two parts, e.g. an advection part and a diffusion part.

  We can choose T as Laplacian relative to the diffusion part (easy to solve) and then we add M for improving T relative to the advection part.

Probing (for MSPAI)

- Find preconditioner of special form (tridiag, band) for preconditioning a matrix that is not given explicitly, but only by its action on certain (probing-)vectors, e.g.

$$e^T S = f^T.$$

  Example: S is Schur complement or general matrix.

- Choose, e.g. $e = (1, 1, \ldots, 1)^T$, $e = (1, -1, 1, -1, \ldots)^T$, $\ldots$

Probing (for MSPAI)

- Find preconditioner of special form (tridiag, band) for preconditioning a matrix that is not given explicitly, but only by its action on certain (probing-)vectors, e.g.

$$e^T S = f^T.$$

  Example: S is Schur complement or general matrix.

- Choose, e.g. $e = (1, 1, \ldots, 1)^T$, $e = (1, -1, 1, -1, \ldots)^T$, $\ldots$

- Disadvantage: Can use only very special pattern for M and special probing vectors e.
  Example: tridiagonal probing

4. Modified SPAI: SPAI with Targeting and Probing

Generalize Frobenius norm minimization to

$$\min_{M\in\mathcal{P}} \|CM - B\|_F = \min_{M\in\mathcal{P}} \left\| \begin{pmatrix} C_0 \\ \rho u^T \end{pmatrix} M - \begin{pmatrix} B_0 \\ \rho v^T \end{pmatrix} \right\|_F$$

For example: original SPAI extended by an additional norm minimization to deliver especially good results on vector e:

$$\min_{M\in\mathcal{P}} \|CM - B\|_F = \min_{M\in\mathcal{P}} \left\| \begin{pmatrix} A \\ \rho e^T A \end{pmatrix} M - \begin{pmatrix} I \\ \rho e^T \end{pmatrix} \right\|_F$$

SPAI for Triangular Matrices

Solve $Lx = b$ with L triangular. Goal: Improve Jacobi iteration!

SPAI: $\min \|L\Lambda - I\|_F$ with pattern of Λ like L.

SPAI for Triangular Matrices

Solve $Lx = b$ with L triangular. Goal: Improve Jacobi iteration!

SPAI: $\min \|L\Lambda - I\|_F$ with pattern of Λ like L.

Modification: Sparse approximate block inverse.

Replace rectangular Least Squares problem in SPAI $L(I, J)\Lambda_j = e_j$ by square linear system $L(J, J)\Lambda_j = e_j$.

SPAI for Triangular Matrices

Solve $Lx = b$ with L triangular. Goal: Improve Jacobi iteration!

SPAI: $\min \|L\Lambda - I\|_F$ with pattern of Λ like L.

Modification: Sparse approximate block inverse.

Replace rectangular Least Squares problem in SPAI $L(I, J)\Lambda_j = e_j$ by square linear system $L(J, J)\Lambda_j = e_j$.

Stationary iterative method based on preconditioner Λ:

$$b = L\Lambda\Lambda^{-1}x = L\Lambda y$$

$$diag(L\Lambda)y^{k+1} = b - (L\Lambda)_0 y^k$$

Solution $x = \Lambda y$

Convergence again guaranteed, faster than Jacobi, fully parallel!

Hybrid SPAI - Gauss-Seidel

First compute SPAI-like preconditioner via $\min \|AM - I\|_F$.

Replace $b = Ax$ by $b = A(MM^{-1})x = (AM)\tilde{x} = \tilde{A}\tilde{x}$.

Apply Gauss-Seidel splitting on $\tilde{A} = D + L_0 + U_0 = L + U_0$.

Hybrid SPAI - Gauss-Seidel

First compute SPAI-like preconditioner via $\min \|AM - I\|_F$.

Replace $b = Ax$ by $b = A(MM^{-1})x = (AM)\tilde{x} = \tilde{A}\tilde{x}$.

Apply Gauss-Seidel splitting on $\tilde{A} = D + L_0 + U_0 = L + U_0$.

In every iteration step we have to solve triangular system in L.

Apply sparse approximate block inverse preconditioner for L.

Hybrid SPAI - Gauss-Seidel

First compute SPAI-like preconditioner via $\min \|AM - I\|_F$.

Replace $b = Ax$ by $b = A(MM^{-1})x = (AM)\tilde{x} = \tilde{A}\tilde{x}$.

Apply Gauss-Seidel splitting on $\tilde{A} = D + L_0 + U_0 = L + U_0$.

In every iteration step we have to solve triangular system in L.

Apply sparse approximate block inverse preconditioner for L.

Everything fully parallel.

Improved convergence compared to plain SPAI or Gauss-Seidel

Outlook: Further Research

1. Apply MSPAI with the probing feature to multigrid or regularization problems (smoother, preconditioner).
   In connection with domain decomposition:
   - In the small domains use MSPAI as smoother
   - Use MSPAI as preconditioner for Schur complement

2. Find good block pattern
   - that reflects the physical connections
   - combines columns with very similar pattern to one LS problem

3. Factorized SPAI for indefinite matrices
   - Use a priori permutations (perfect matching)
   - and blocking

   to avoid breakdowns.

Available Code

- SPAI, MSPAI, FSPAI available in parallel version (MPI, C/C++)
  - SPAI (University of Basel):
    `http://www.computational.unibas.ch/software/spai/`
  - SPAI, MSPAI (TUM):
    `http://www5.in.tum.de/wiki/index.php/MSPAI`
  - FSPAI (TUM):
    `http://www5.in.tum.de/wiki/index.php/FSPAI`

    Publications on SPAI, MSPAI, FSPAI available on these websites!

Available Code

- SPAI, MSPAI, FSPAI available in parallel version (MPI, C/C++)
  - SPAI (University of Basel):
    `http://www.computational.unibas.ch/software/spai/`
  - SPAI, MSPAI (TUM):
    `http://www5.in.tum.de/wiki/index.php/MSPAI`
  - FSPAI (TUM):
    `http://www5.in.tum.de/wiki/index.php/FSPAI`

    Publications on SPAI, MSPAI, FSPAI available on these websites!
- Block SPAI also available for GPU and shared memory (OpenMP, C/C++).
- High priority: Parallel triangular solves.

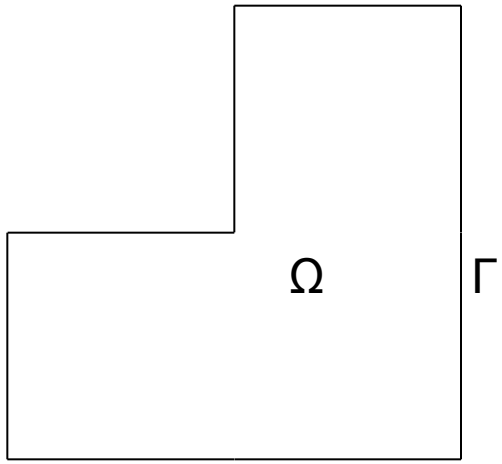
6. Domain Decomposition

Consider elliptic PDE on region Ω with boundary Γ, e.g. with Dirichlet boundary conditions.
Example:

$$\Delta u = u_{xx} + u_{yy} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \qquad in \quad \Omega$$

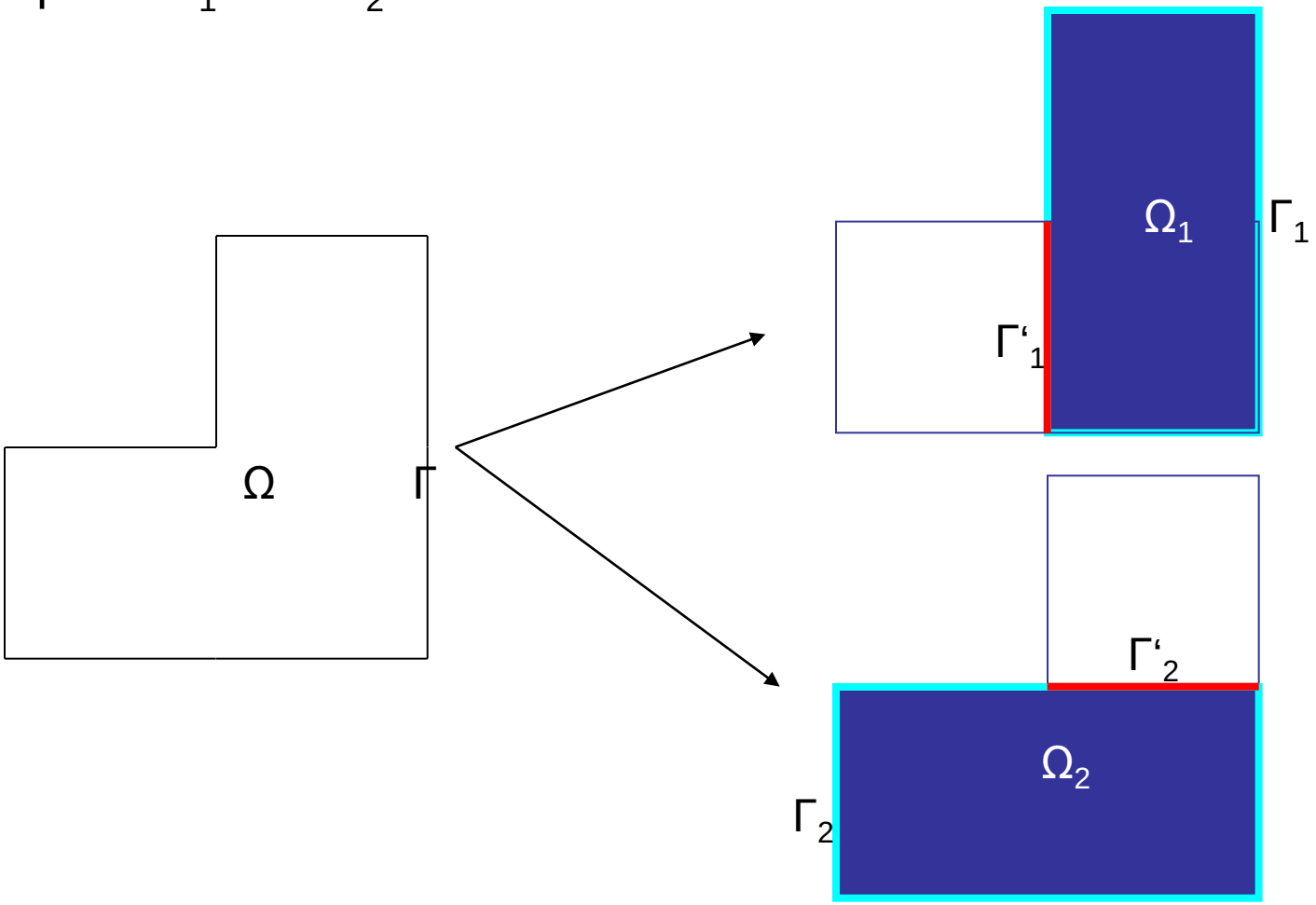
$$u(x,y)\big|_{\Gamma} = g(x,y) \qquad in \quad \Gamma$$

Parallel solution?

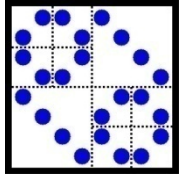


Overlapping DD

Partition region Ω in two regions Ω_1 and Ω_2 , with new boundaries Γ_1 and Γ_2 which are partially given by old boundary Γ and some unknown parts Γ'_1 and Γ'_2 :



Overlapping DD II



We can discretize and solve the given PDE on Ω_1 with boundary Γ_1 , but we need the values of u(x,y) on the new artificial boundary Γ'_1.

In a first step we can assume any values for Γ'_1 , e.g. u(x,y)=0.

Then we can solve the linear system relative to region Ω_1 .

The same can be done in parallel for Ω_2.

The values of the resulting solution on Γ'_2 can be used, to compute in the next step the solution for region Ω_2 with boundary Γ_2.

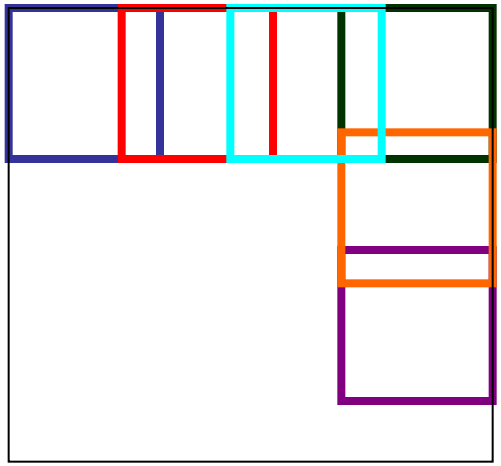
In the same way we can use the resulting solution on Γ'_1, to compute the solution for region Ω_1 with boundary Γ_1.

So we can generate solutions on the partial regions which provide us with approximate values for the unknown boundary of the other partial solution.

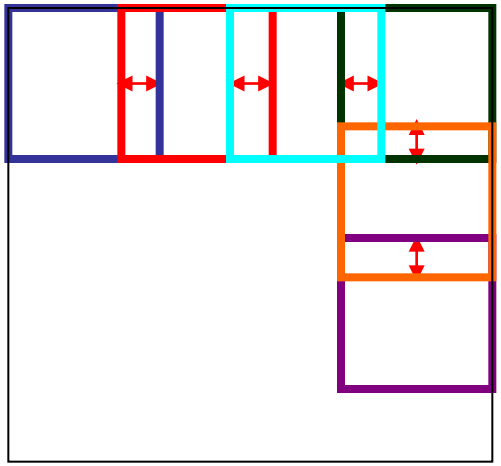
The sequence of solutions converges in each region against the solution on Ω.

Overlapping DD III

First step:
Solve in parallel the PDE
on all small subdomains
with certain boundary cond.



Second step:
Exchange boundary values



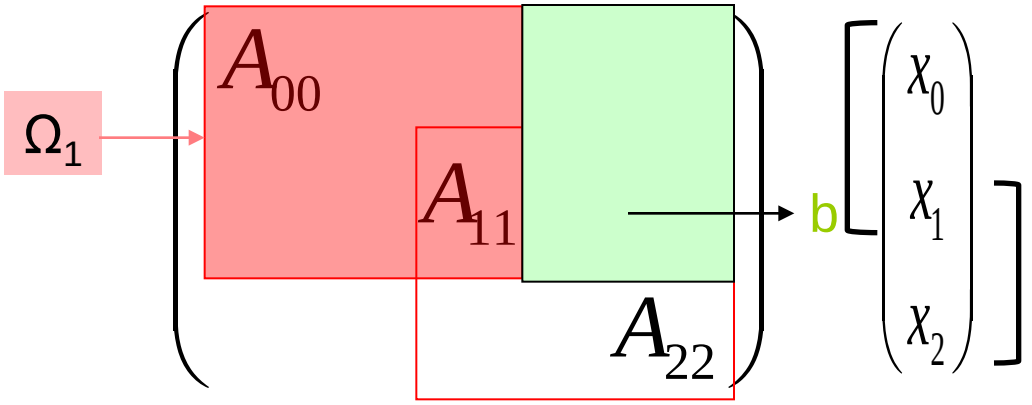
Repeat until convergence.

For getting better interior boundary values:
Solve whole problem on coarse mesh and interpolate

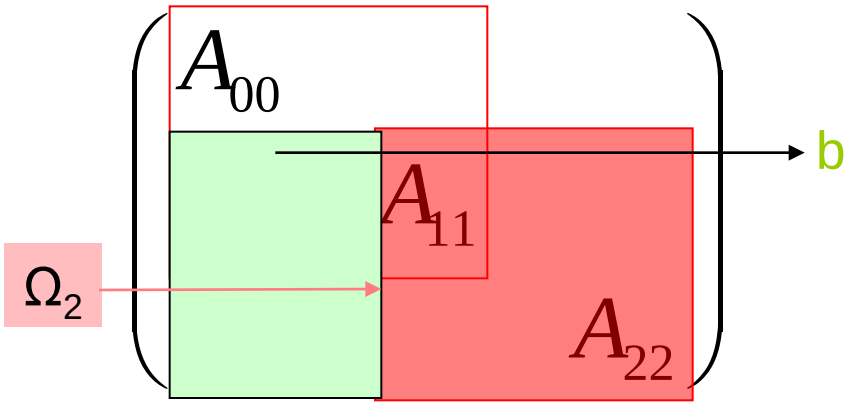
Overlapping DD IV

Matrix representation of overlapping DD:

First step:



Second step:

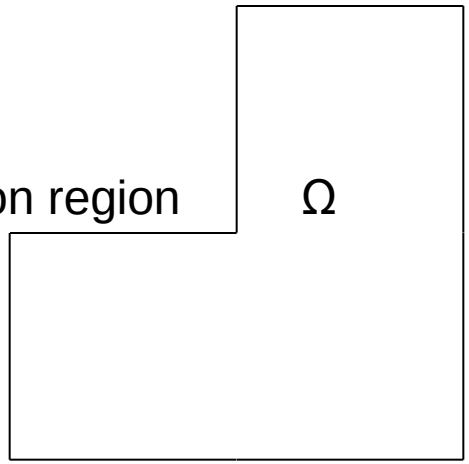


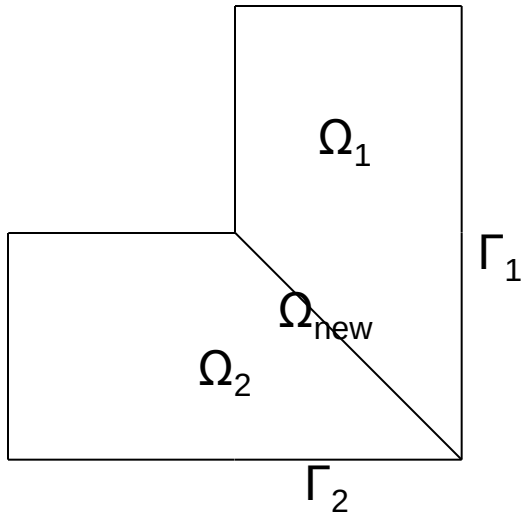
Repeat until convergence.

Green parts are related to the other domain and we asssume to know the related components in the vector x of unknowns.
They are moved to the right-hand-side b.

Nonoverlapping DD

Partition region Ω Γ in the form





Discretization of the original problem with numbering of the unknowns relative to the partitioning given by Ω_1 and Ω_2 leads to a linear system with a matrix in dissection form:

Nonoverlapping DD II

$$f = Au = \begin{pmatrix} A_1 & 0 & F_1 \\ 0 & A_2 & F_2 \\ G_1 & G_2 & A_3 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

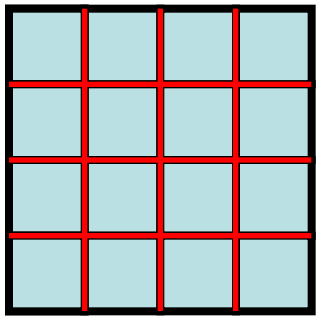
A_3 is the so called interface matrix.

We can solve Au=f iteratively with pcg and preconditioner: $\begin{pmatrix} A_1^{-1} & 0 & 0 \\ 0 & A_2^{-1} & 0 \\ 0 & 0 & M \end{pmatrix}$

Here, for M we can use the identity or an approximate inverse for the Schur complement.

Hence, we reduce the original problem to two partial subproblems and one interface Schur complement system.

Nonoverlapping DD III



Leads to 16 block matrices on the diagonal

$A_1, \ldots, A_{16}$ and Schur complement S.

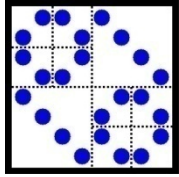
$$A = \begin{pmatrix} A_1 & & & F_1 \\ & \ddots & & \vdots \\ & & A_{16} & F_{16} \\ G_1 & \cdots & G_{16} & A_{17} \end{pmatrix}, \qquad S = A_{17} - G_1 A_1^{-1} F_1 - \cdots - G_{16} A_{16}^{-1} F_{16}$$

Overlapping easy parallel
But slow convergence

Solve small problems e.g. with Multigrid in parallel.

Precondition Schur complement e.g. with MSPAI

Nonoverlapping harder parallel, but more influence on convergence in S.



Multigrid

Starting point: Solve Partial Differential Equation, e.g.

$$-u_{xx} - u_{yy} = f(x, y)$$

with boundary conditions, e.g. Dirichlet BC.

Discretization leads to system of linear equations, resp. matrix A.

A is sparse, (structured,) and ill-conditioned.
Looking for O(n) solver.

Direct solver $O(n^2)$ or $O(n \log(n))$
PCG also > O(n) because matrix is ill-conditioned

Idea

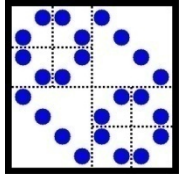
Vector of unknowns:

(1) Project the fine discretization on a coarser (smaller) problem

(2) Solve the coarse matrix (smaller matrix)

(3) Project the coarse solution back on fine grid

Problems: - only possible for smooth vector
without high oscillatory components
- backprojection introduces (high-oscillatory)
errors that have to be removed, too.



Observation:

For typical PDE matrices high-oscillatory vectors are related to the subspace to large eigenvalues and are removed e.g. by the stationary Gauss-Seidel iteration:

$|| I - M^{-1}A ||$ small for eigenvectors to large eigenvalues!

To solve Ax=f:

(1) Apply a few steps Gauss-Seidel smoothing steps → x_a
Residual equation $A(x_a+x)=f$ → $Ax = f - Ax_a = r$

(2) Project (restrict) r, resp. A on coarse grid by mean value
Solve $A_c\, x_c = r_c$

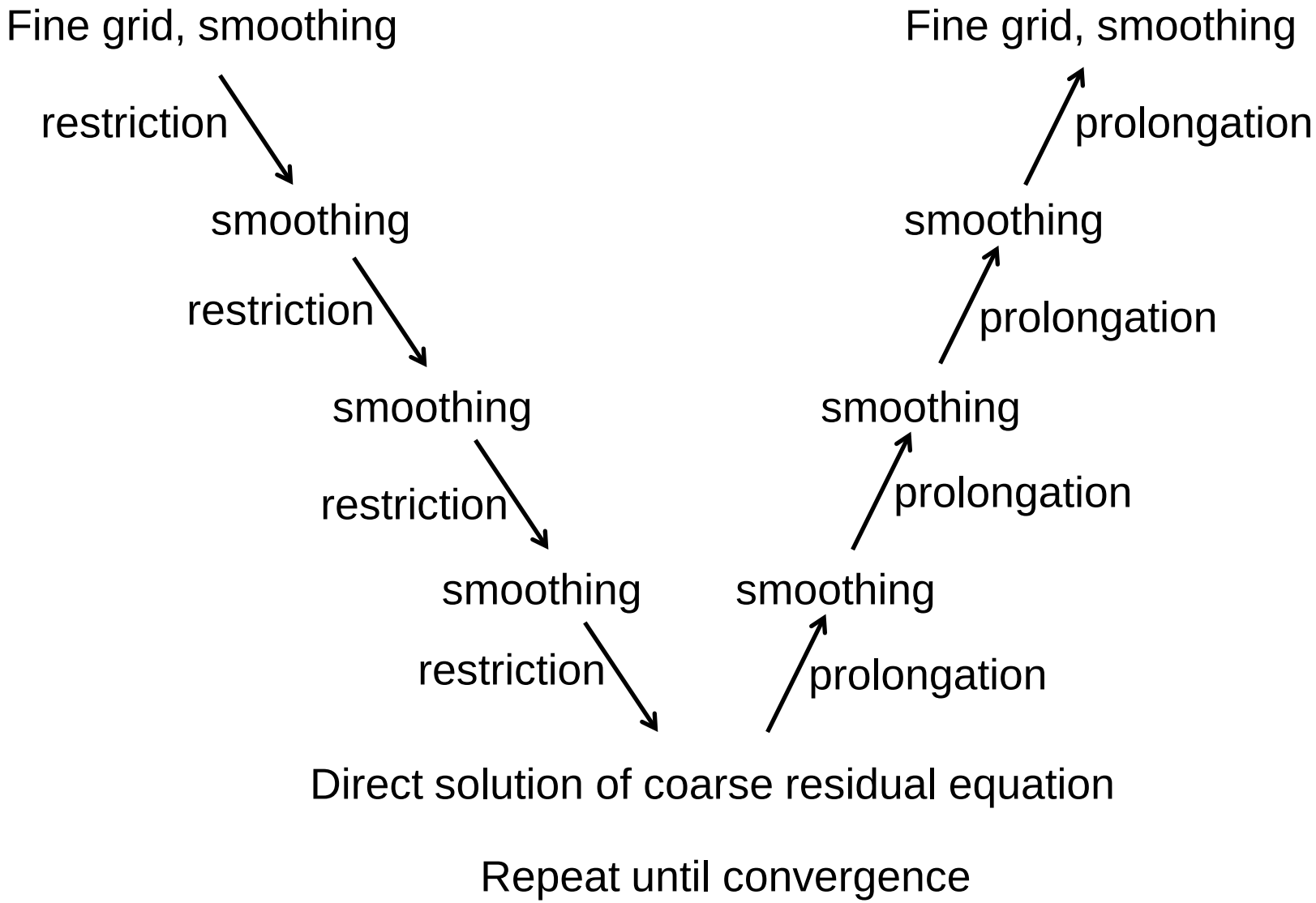
(3) Project (prolongate) x_c back on fine grid x by interpolation
x_c → x → new approximate solution $x_a + x$

(4) Improve approximate solution by Gauss-Seidel steps

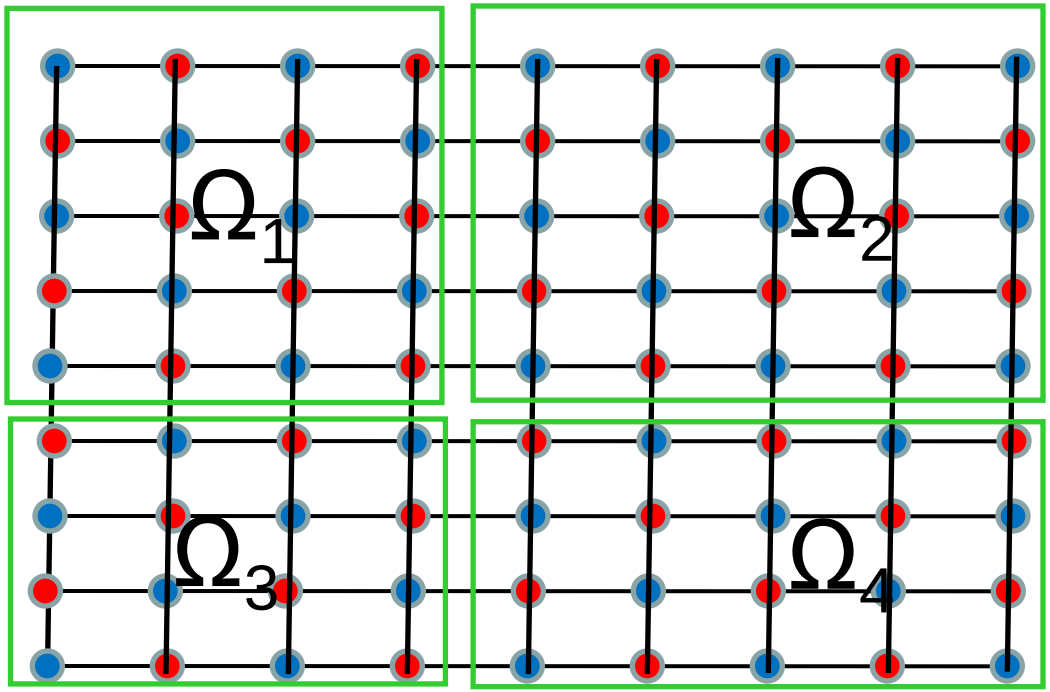
Repeat until convergence

Apply also recursively for solving coarse equations

V-Cycle

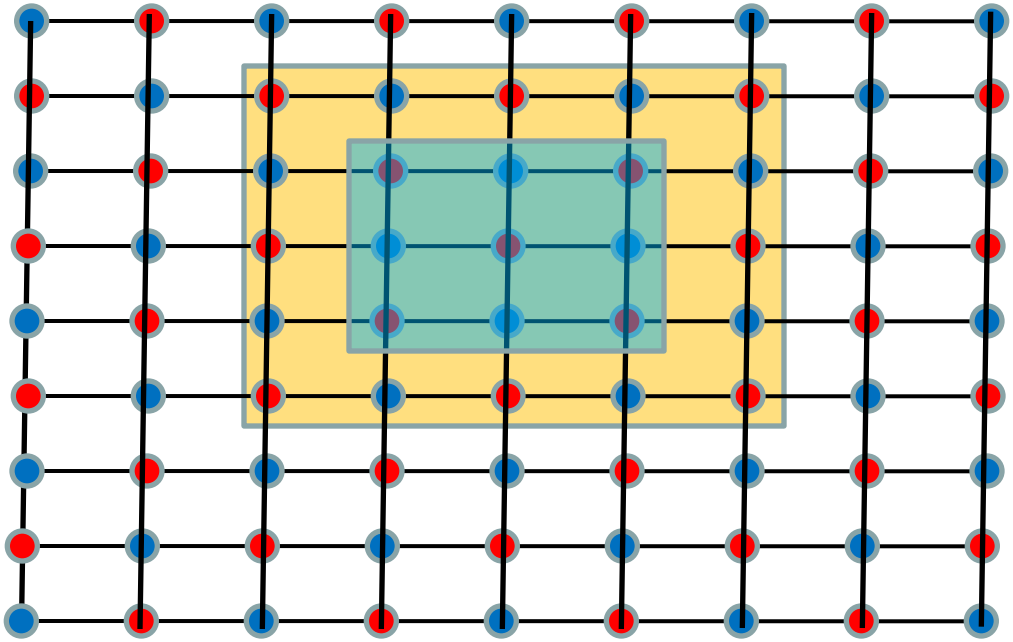

Fine grid, smoothing
restriction
smoothing
restriction
smoothing
restriction
smoothing
restriction
Direct solution of coarse residual equation
prolongation
smoothing
prolongation
smoothing
prolongation
smoothing
prolongation
Fine grid, smoothing
Repeat until convergence

Parallel Aspects



Each processor p_r has data for domain Ω_r and does projections and smoothing for its components.
Needs data from neighboring processor for projection and GS.
Load balancing!

Ghost layers



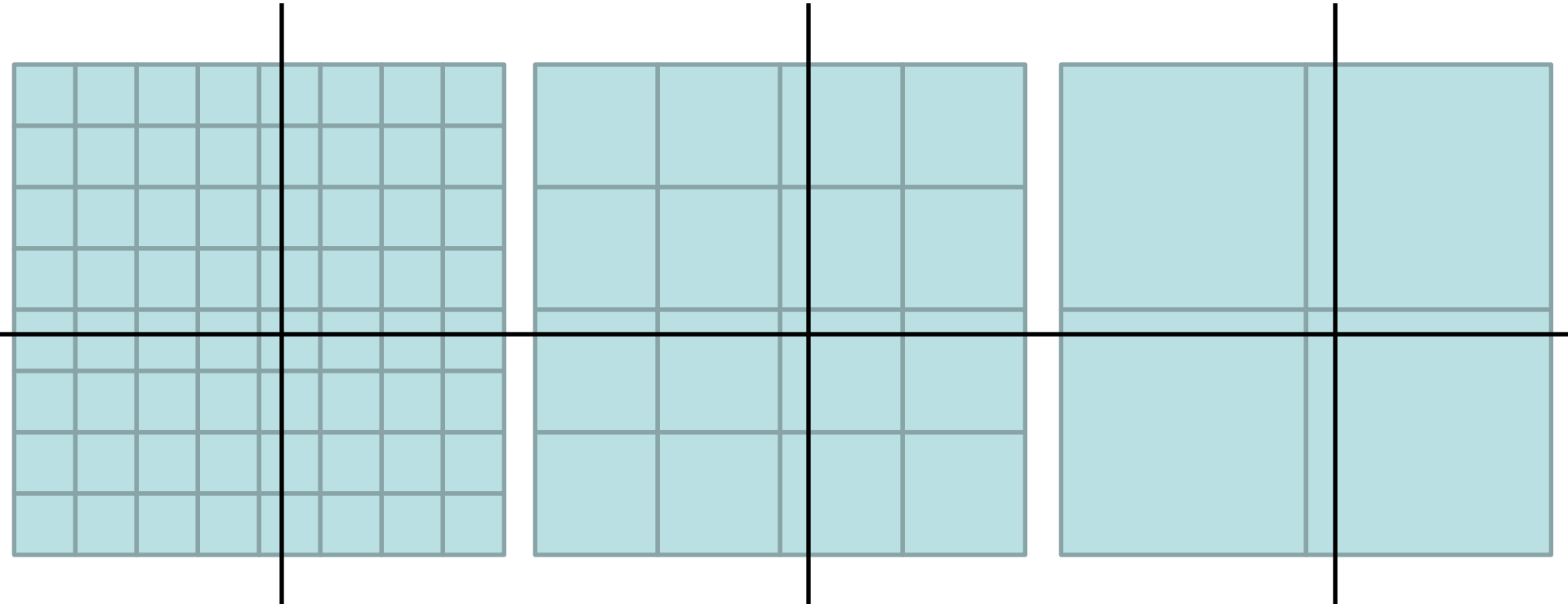
Jacobi smoothing:

(1) Each process performs Jacobi iteration (independently)

(2) Send messages to update ghost layers

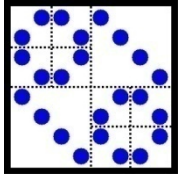
(3) Use communication/computation overlap for interior computations and exchange of boundary data

Restriction/Prolongation



Problems with work load: On coarse grids less computations, more communication!

Efficiency goes like 1/log(p) with the number of processors.



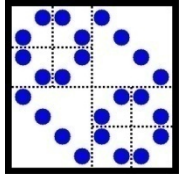
Modifications

Agglomeration: Coarse grid partitions are no longer aligned with the finer grid partitions (in order to avoid inefficiency on coarsest grids)

→ More communication in grid transfer, less in coarse grid solve.

Larger ghost layers can reduce the communication

Reduce number of V-cycle steps by using more powerful projections and smoothers or aggressive coarsening → less data transfer



Additive Multigrid

Consider the different levels at the same time,
Compute the related corrections in parallel and sum up the corrections.

$$x = x_0 + x_1 + \ldots + x_L$$

$$x := x + \alpha \sum_{l=0}^{L} A_l^{-1} P_l (f - Ax)$$

A_0
P_1, A_1
P_2, A_2
P_3, A_3
.
.
P_L, A_L

Hybrid conjugate gradient iteration with MG as preconditioner
→ Less V-cycles