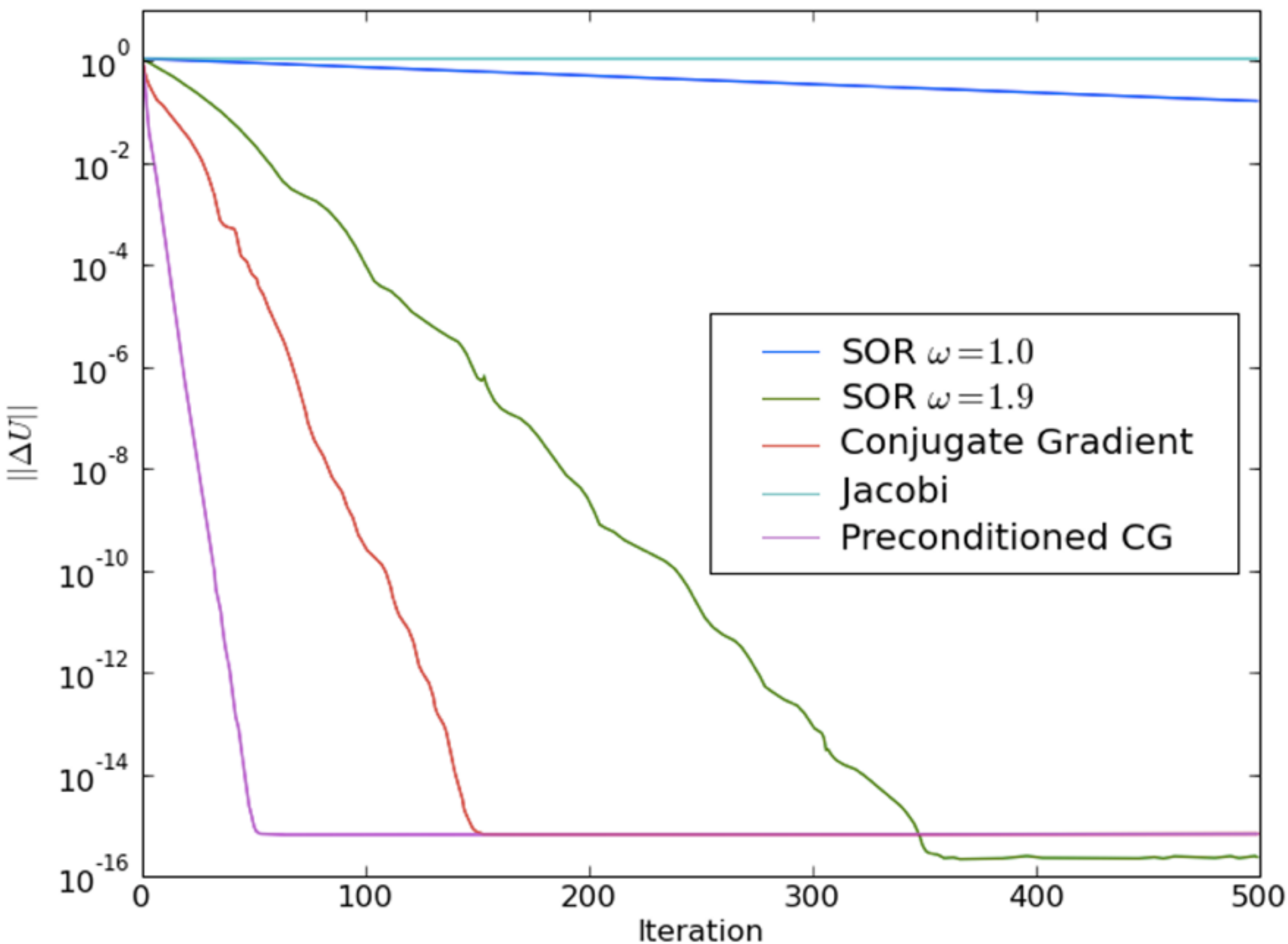
05/09	Class 9	Dense direct solvers	Understand the principle of LU decomposition and the optimization and parallelization techniques that lead to the LINPACK benchmark.
05/12	Class 10	Dense eigensolvers	Determine eigenvalues and eigenvectors and understand the fast algorithms for diagonalization and orthonormalization.
05/16	Class 11	Sparse direct solvers	Understand reordering in AMD and nested dissection, and fast algorithms such as skyline and multifrontal methods.
05/19	Class 12	Sparse iterative solvers	Understand the notion of positive definiteness, condition number, and the difference between Jacobi, CG, and GMRES.
05/23	Class 13	Preconditioners	Understand how preconditioning affects the condition number and spectral radius, and how that affects the CG method.
05/26	Class 14	Multigrid methods	Understand the role of smoothers, restriction, and prolongation in the V-cycle.
05/30	Class 15	Fast multipole methods, H-matrices	Understand the concept of multipole expansion and low-rank approximation, and the role of the tree structure.

Preconditioning



Sparse iterative solvers

SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | SciPy v0.17.1 Reference Guide

Sparse linear algebra (scipy.sparse.linalg)

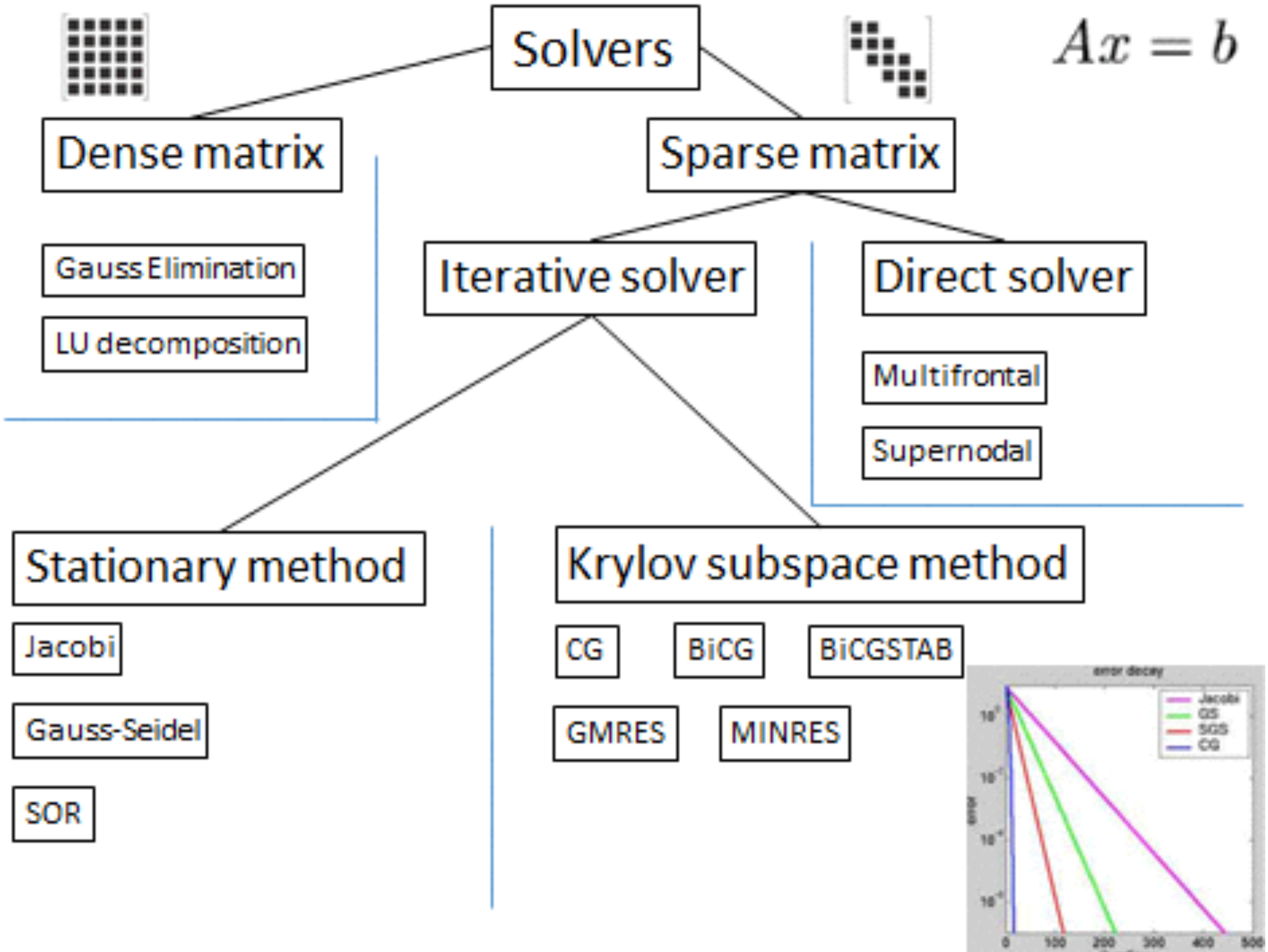
Solving linear problems

Direct methods for linear equation systems:

spsolve(A, b[, permc_spec, use_umfpack])	Solve the sparse linear system Ax=b, where b may be a vector or a matrix.
factorized(A)	Return a fuction for solving a sparse linear system, with A pre-factorized.
MatrixRankWarning	
use_solver(**kwargs)	Select default sparse direct solver to be used.

Iterative methods for linear equation systems:

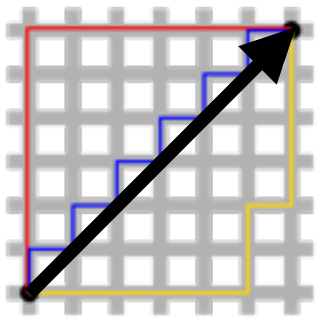
bicg(A, b[, x0, tol, maxiter, xtype, M, ...])	Use BIConjugate Gradient iteration to solve A x = b
bicgstab(A, b[, x0, tol, maxiter, xtype, M, ...])	Use BIConjugate Gradient STABilized iteration to solve A x = b
cg(A, b[, x0, tol, maxiter, xtype, M, callback])	Use Conjugate Gradient iteration to solve A x = b
cgs(A, b[, x0, tol, maxiter, xtype, M, callback])	Use Conjugate Gradient Squared iteration to solve A x = b
gmres(A, b[, x0, tol, restart, maxiter, ...])	Use Generalized Minimal RESidual iteration to solve A x = b.
lgmres(A, b[, x0, tol, maxiter, M, ...])	Solve a matrix equation using the LGMRES algorithm.
minres(A, b[, x0, shift, tol, maxiter, ...])	Use MINimum RESidual iteration to solve Ax=b
qmr(A, b[, x0, tol, maxiter, xtype, M1, M2, ...])	Use Quasi-Minimal Residual iteration to solve A x = b



Solvers
$Ax = b$
Dense matrix
Sparse matrix
Gauss Elimination
LU decomposition
Iterative solver
Direct solver
Multifrontal
Supernodal
Stationary method
Krylov subspace method
Jacobi
Gauss-Seidel
SOR
CG
BiCG
BiCGSTAB
GMRES
MINRES
error decay
Jacobi
GS
SGS
CG

Norm and Condition Number

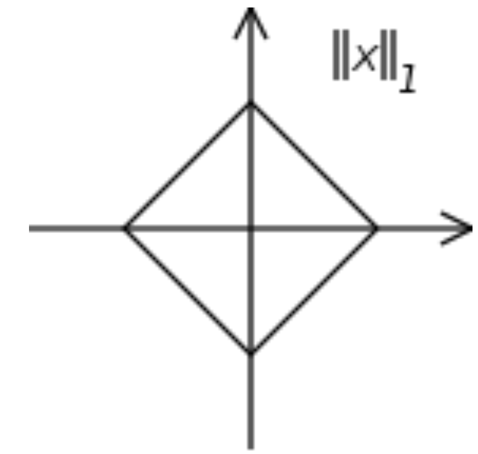
$||\mathbf{x}|| \leq 1$

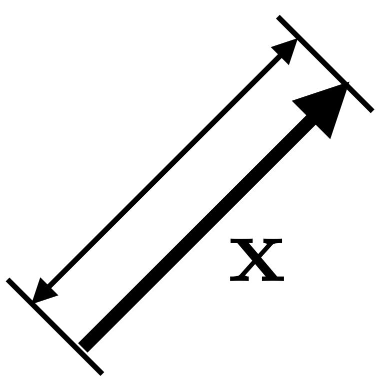


$$||\mathbf{x}||_1 = \sum_{i=1}^{n} |\mathbf{x}_i|$$

Manhattan norm

$\mathcal{L}_1$ norm

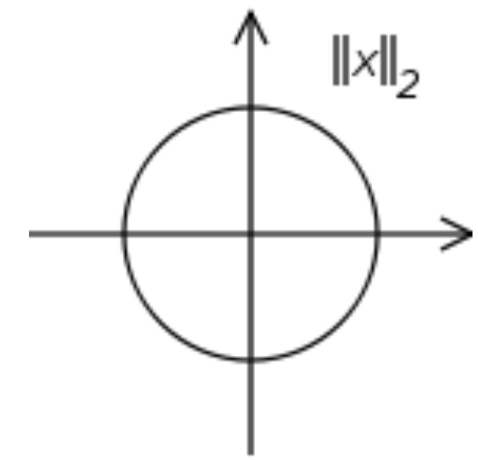


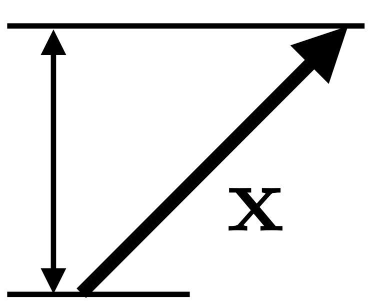


$$||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} \mathbf{x}_i^2}$$

Euclidean norm

$\mathcal{L}_2$ norm

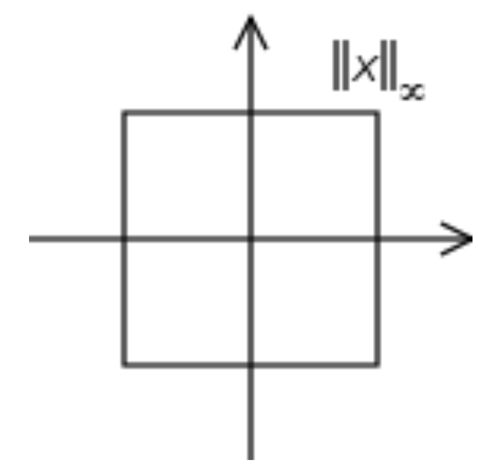




$$||\mathbf{x}||_\infty = \max_{1 \leq i \leq n} |\mathbf{x}_i|$$

Maximum norm

$\mathcal{L}_\infty$ norm



Norm and Condition Number

Schatten norm

$$||A||_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |A_{ij}|$$

1-norm

$$||A||_2 = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |A_{ij}^2|}$$

2-norm

$$||A||_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |A_{ij}|$$

Infinity norm

Operator norm

$$||A||_p = \max_{\mathbf{x} \neq 0} \frac{||A\mathbf{x}||_p}{||\mathbf{x}||_p}$$

Condition number

$$cond(A) = ||A|| \; ||A^{-1}||$$

Norm and Condition Number

Condition number

$$cond(A) = ||A||\ ||A^{-1}|| = \frac{\sigma_{max}}{\sigma_{min}} = \frac{|\lambda_{max}|}{|\lambda_{min}|}$$

if $AA^* = A^*A$

Spectral radius

$$\rho(A) = \max_{1 \le i \le n} |\lambda_i|$$

The following lemma shows a simple yet useful upper bound for the spectral radius of a matrix:

Lemma. Let $A \in \mathbf{C}^{n \times n}$ with spectral radius $\rho(A)$ and a consistent matrix norm $||\cdot||$; then, for each $k \in \mathbf{N}$:

$$\rho(A) \le \|A^k\|^{\frac{1}{k}}.$$

Proof: Let $(\mathbf{v}, \lambda)$ be an eigenvector-eigenvalue pair for a matrix *A*. By the sub-multiplicative property of the matrix norm, we get:

$$|\lambda|^k \|\mathbf{v}\| = \|\lambda^k \mathbf{v}\| = \|A^k \mathbf{v}\| \le \|A^k\| \cdot \|\mathbf{v}\|$$

and since $\mathbf{v} \neq 0$ we have

$$|\lambda|^k \le \|A^k\|$$

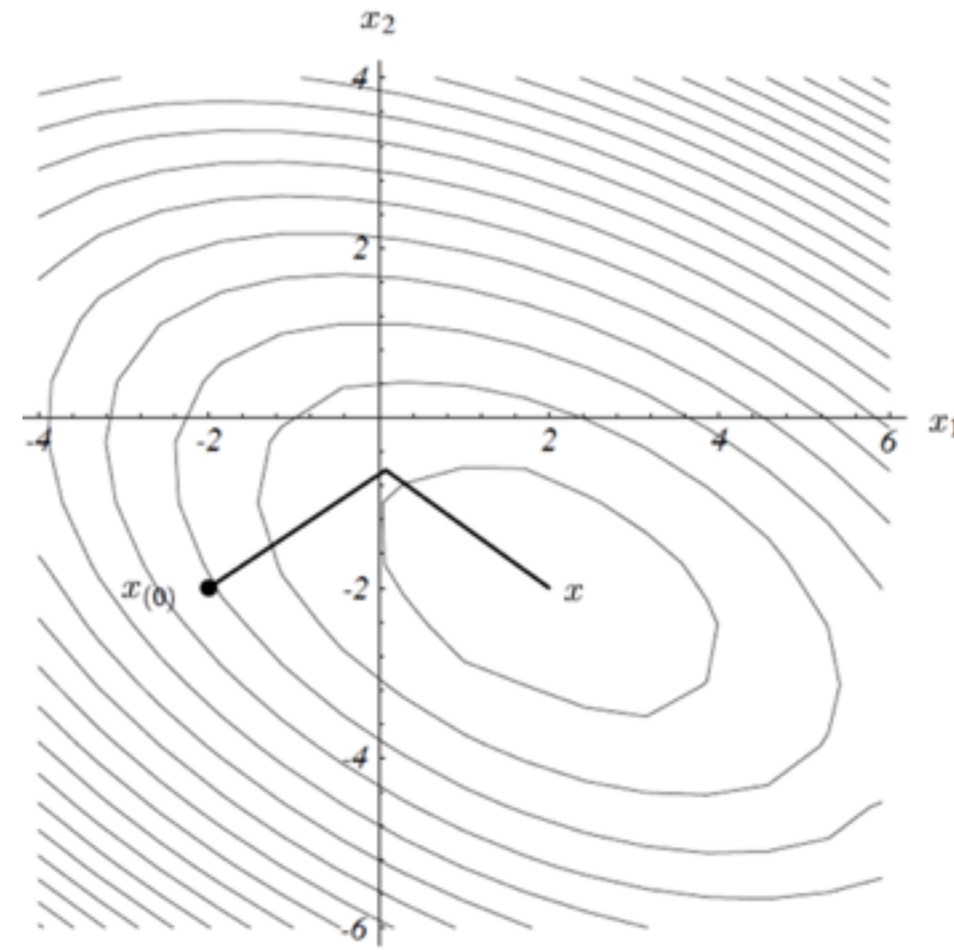
and therefore

$$\rho(A) \le \|A^k\|^{\frac{1}{k}}.$$

https://en.wikipedia.org/wiki/Spectral_radius

Convergence rate of Conjugate Gradient

minimize $\quad f(x) = \frac{1}{2}x^T A x - b^T x$



Optimal value

$$f(x^\star) = -\frac{1}{2}b^T A^{-1} b = -\frac{1}{2}\|x^\star\|_A^2$$

Suboptimality at x

$$f(x) - f^\star = \frac{1}{2}\|x - x^\star\|_A^2$$

Relative error measure

$$\tau = \frac{f(x) - f^\star}{f(0) - f^\star} = \frac{\|x - x^\star\|_A^2}{\|x^\star\|_A^2}$$

Convergence rate of Conjugate Gradient

Error after k steps

- $x^{(k)} \in \mathcal{K}_k = \mathrm{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}$, so $x^{(k)}$ can be expressed as

$$x^{(k)} = \sum_{i=1}^{k} c_i A^{i-1} b = p(A)b$$

  where $p(\lambda) = \sum_{i=1}^{k} c_i \lambda^{i-1}$ is some polynomial of degree $k-1$ or less

- $x^{(k)}$ minimizes $f(x)$ over $\mathcal{K}_k$; hence

$$2(f(x^{(k)}) - f^\star) = \inf_{x \in \mathcal{K}_k} \|x - x^\star\|_A^2 = \inf_{\deg p < k} \left\|(p(A) - A^{-1})b\right\|_A^2$$

we now use the eigenvalue decomposition of A to bound this quantity

Convergence rate of Conjugate Gradient

- eigenvalue decomposition of A

$$A = Q\Lambda Q^T = \sum_{i=1}^{n} \lambda_i q_i q_i^T \qquad (Q^T Q = I, \quad \Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_n))$$

- define $d = Q^T b$

expression on previous page simplifies to

$$\begin{aligned}
2(f(x^{(k)}) - f^\star) &= \inf_{\deg p < k} \left\| (p(A) - A^{-1}) b \right\|_A^2 \\
&= \inf_{\deg p < k} \left\| (p(\Lambda) - \Lambda^{-1})\, d \right\|_\Lambda^2 \\
&= \inf_{\deg p < k} \sum_{i=1}^{n} \frac{(\lambda_i p(\lambda_i) - 1)^2\, d_i^2}{\lambda_i} \\
&= \inf_{\deg q \le k,\ q(0)=1} \sum_{i=1}^{n} \frac{q(\lambda_i)^2\, d_i^2}{\lambda_i}
\end{aligned}$$

Convergence rate of Conjugate Gradient

Absolute error

$$\begin{aligned} f(x^{(k)}) - f^\star &\leq \left(\sum_{i=1}^{n} \frac{d_i^2}{2\lambda_i}\right) \inf_{\deg q \leq k,\ q(0)=1} \left(\max_{i=1,\ldots,n} q(\lambda_i)^2\right) \\ &= \frac{1}{2}\|x^\star\|_A^2 \inf_{\deg q \leq k,\ q(0)=1} \left(\max_{i=1,\ldots,n} q(\lambda_i)^2\right) \end{aligned}$$

(equality follows from $\sum_i d_i^2/\lambda_i = b^T A^{-1} b = \|x^\star\|_A^2$)

Relative error

$$\tau_k = \frac{\|x^{(k)} - x^\star\|_A^2}{\|x^\star\|_A^2} \leq \inf_{\deg q \leq k,\ q(0)=1} \left(\max_{i=1,\ldots,n} q(\lambda_i)^2\right)$$

Convergence rate of Conjugate Gradient

Convergence rate and spectrum of A

- if A has m distinct eigenvalues $\gamma_1, \ldots, \gamma_m$, CG terminates in m steps:

$$q(\lambda) = \frac{(-1)^m}{\gamma_1 \cdots \gamma_m}(\lambda - \gamma_1) \cdots (\lambda - \gamma_m)$$

  satisfies $\deg q = m$, $q(0) = 1$, $q(\lambda_1) = \cdots = q(\lambda_n) = 0$; therefore $\tau_m = 0$

- if eigenvalues are clustered in m groups, then τ_m is small

  can find $q(\lambda)$ of degree m, with $q(0) = 1$, that is small on spectrum

- if $x^\star$ is a linear combination of m eigenvectors, CG terminates in m steps

  take q of degree m with $q(\lambda_i) = 0$ where $d_i \neq 0$; then

$$\sum_{i=1}^{n} \frac{q(\lambda_i)^2 d_i^2}{\lambda_i} = 0$$

Preconditioner

- Main idea: Instead of solving

$$Ax = b$$

solve, using a nonsingular $m \times m$ *preconditioner* M,

$$M^{-1}Ax = M^{-1}b$$

which has the same solution x

- Convergence properties based on $M^{-1}A$ instead of A
- Trade-off between the cost of applying M^{-1} and the improvement of the convergence properties. Extreme cases:
  - $M = A$, perfect conditioning of $M^{-1}A = I$, but expensive M^{-1}
  - $M = I$, "do nothing" $M^{-1} = I$, but no improvement of $M^{-1}A = A$

http://persson.berkeley.edu/228A/Fall10/doc/lec41-2x3.pdf

Preconditioner

How to choose M ?

- M should be easy to invert
- M^{-1} should be close to A^{-1}

Given a stationary iterative method for $A\mathbf{u} = \mathbf{f}$,

$$M\mathbf{u}^{n+1} = (M - A)\mathbf{u}^n - \mathbf{f},$$

at convergence, the system

$$M\mathbf{u} = (M - A)\mathbf{u} - \mathbf{f} \quad \Longleftrightarrow \quad M^{-1}A\mathbf{u} = M^{-1}\mathbf{f}$$

is solved. Hence every station-nary iterative method gives raise to a preconditioner!

Example: Block Jacobi or Additive Schwarz without algebraic overlap

$$\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{pmatrix} \mathbf{u}_1^{n+1} \\ \mathbf{u}_2^{n+1} \end{pmatrix} = \begin{bmatrix} 0 & -A_{12} \\ -A_{21} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}_1^{n} \\ \mathbf{u}_2^{n} \end{pmatrix} + \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}$$

Preconditioner

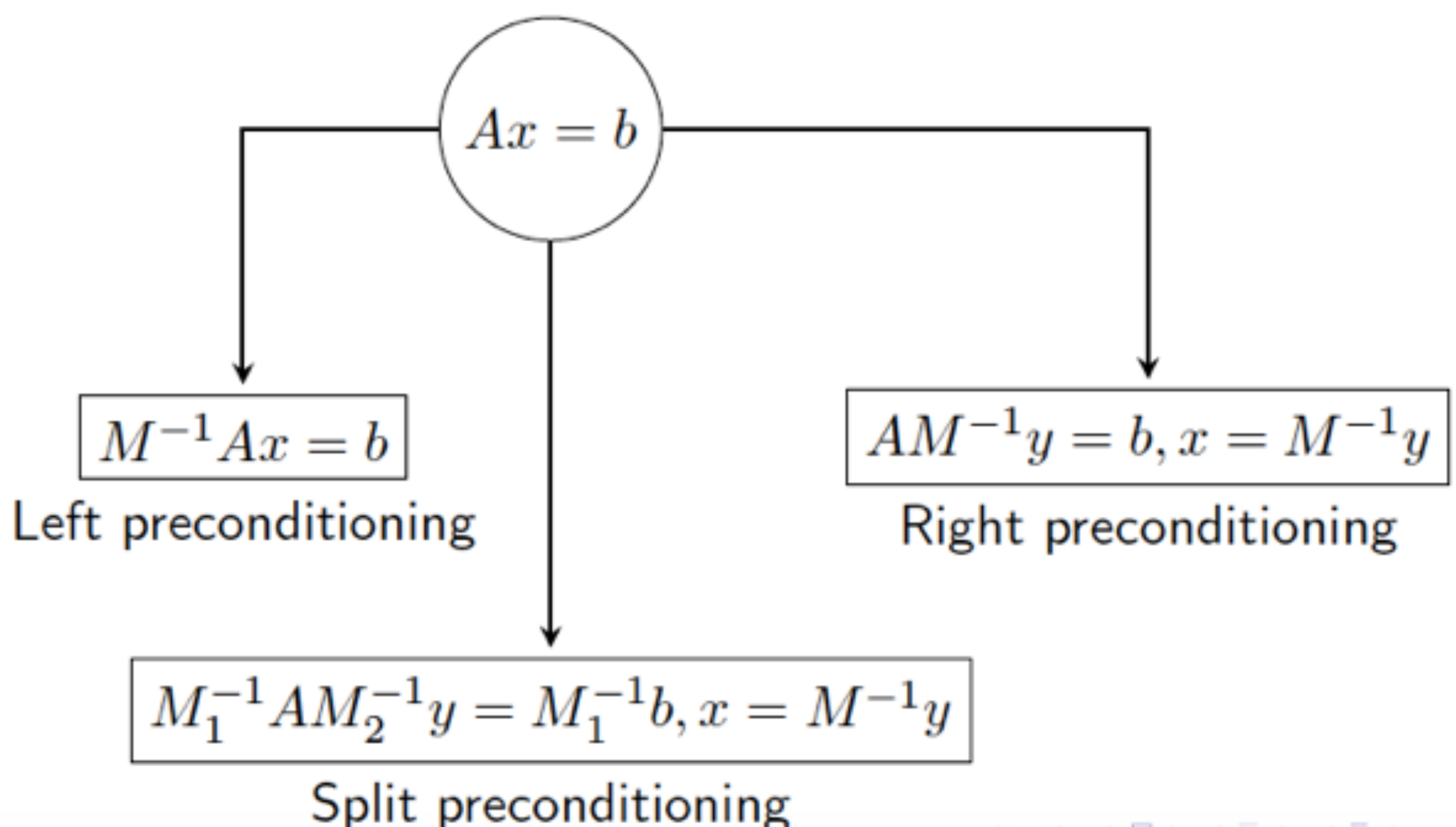
Does this Give a Good Preconditioner ?

The stationary iterative method

$$M\mathbf{u}^{n+1} = (M - A)\mathbf{u}^{n} - \mathbf{f},$$

converges fast, if $\rho(I - M^{-1}A) << 1$. This is equivalent to saying that the spectrum of the preconditioned operator $M^{-1}A$ is close to one. This implies, if the spectrum is real, that

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



Preconditioned Conjugate Gradient

- To keep symmetry, solve $(C^{-1}AC^{-*})C^*x = C^{-1}b$ with $CC^* = M$
- Can be written in terms of M^{-1} only, without reference to C:

Algorithm: Preconditioned Conjugate Gradients Method

$x_0 = 0, r_0 = b$

$p_0 = M^{-1}r_0, z_0 = p_0$

for $n = 1, 2, 3, \ldots$

$\alpha_n = (r_{n-1}^T z_{n-1})/(p_{n-1}^T A p_{n-1})$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual
$z_n = M^{-1} r_n$	preconditioning
$\beta_n = (r_n^T z_n)/(r_{n-1}^T z_{n-1})$	improvement this step
$p_n = z_n + \beta_n p_{n-1}$	search direction

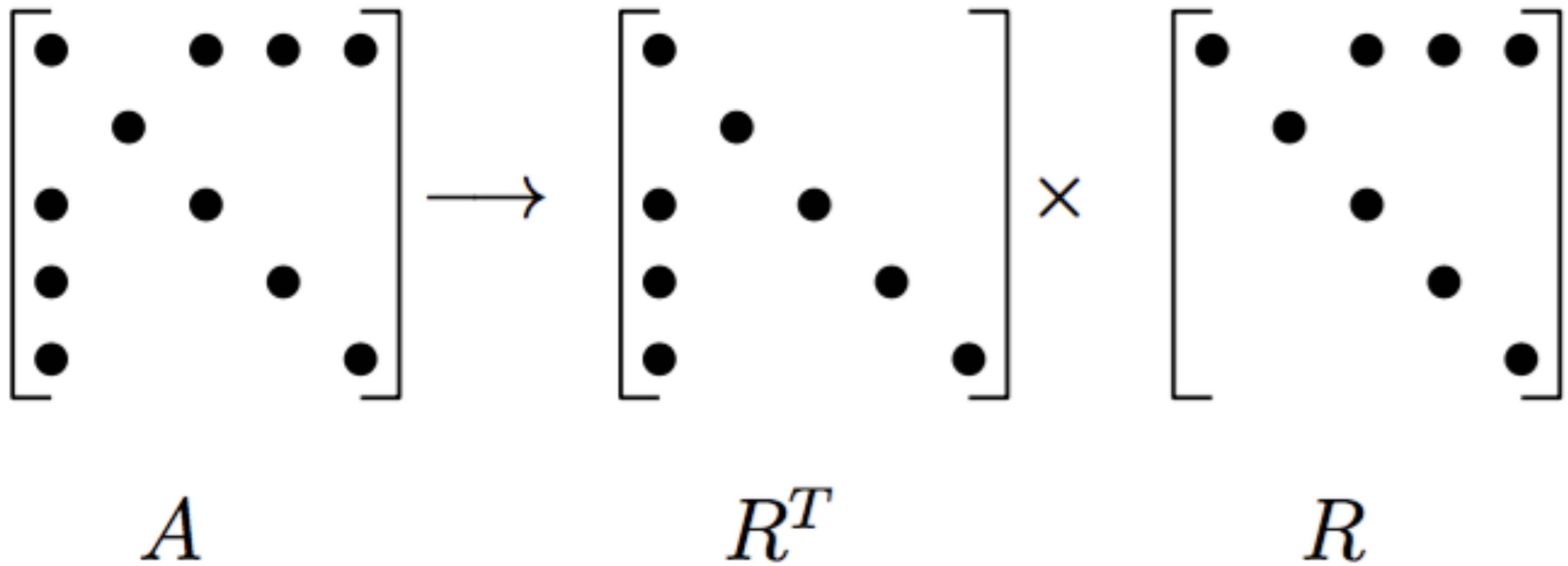
http://persson.berkeley.edu/228A/Fall10/doc/lec41-2x3.pdf

Various Preconditioners

- A preconditioner should "approximately solve" the problem $Ax = b$
- **Jacobi preconditioning** - $M = \text{diag}(A)$, very simple and cheap, might improve certain problems but usually insufficient
- **Block-Jacobi preconditioning** - Use block-diagonal instead of diagonal. Another variant is using several diagonals (e.g. tridiagonal)
- **Classical iterative methods** - Precondition by applying one step of Jacobi, Gauss-Seidel, SOR, or SSOR
- **Incomplete factorizations** - Perform Gaussian elimination but ignore fill, results in approximate factors $A \approx LU$ or $A \approx R^T R$ (more later)
- **Coarse-grid approximations** - For a PDE discretized on a grid, a preconditioner can be formed by transferring the solution to a coarser grid, solving a smaller problem, then transferring back (*multigrid*)

http://persson.berkeley.edu/228A/Fall10/doc/lec41-2x3.pdf

Incomplete LU (Cholesky) factorization

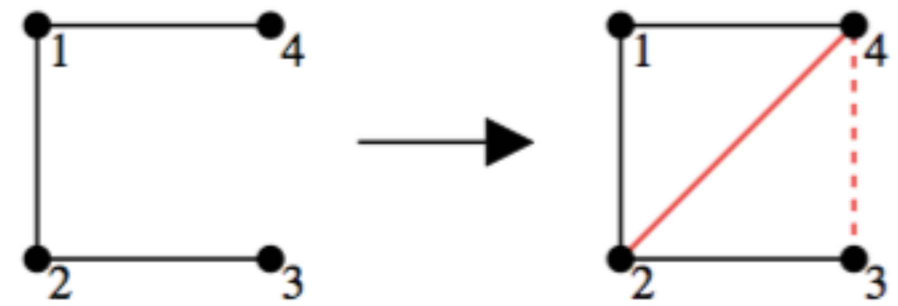


- Compute factors of A by Gaussian elimination, but ignore fill
- Preconditioner $B = R^T R \approx A$, not formed explicitly
- Compute $B^{-1}z$ by triangular solves in time $O(\texttt{nnz(A)})$
- Total storage is $O(\texttt{nnz(A)})$, static data structure
- Either symmetric (IC) or nonsymmetric (ILU)

http://persson.berkeley.edu/228A/Fall10/doc/lec41-2x3.pdf

Incomplete LU (Cholesky) factorization

- Allow one or more “levels of fill”
  - Unpredictable storage requirements



- Allow fill whose magnitude exceeds a “drop tolerance”
  - May get better approximate factors than levels of fill
  - Unpredictable storage requirements
  - Choice of tolerance is ad hoc
- Partial pivoting (for nonsymmetric A)
- “Modified ILU” (MIC): Add dropped fill to diagonal of U (R)
  - A and $R^T R$ have same row sums
  - Good in some PDE contexts

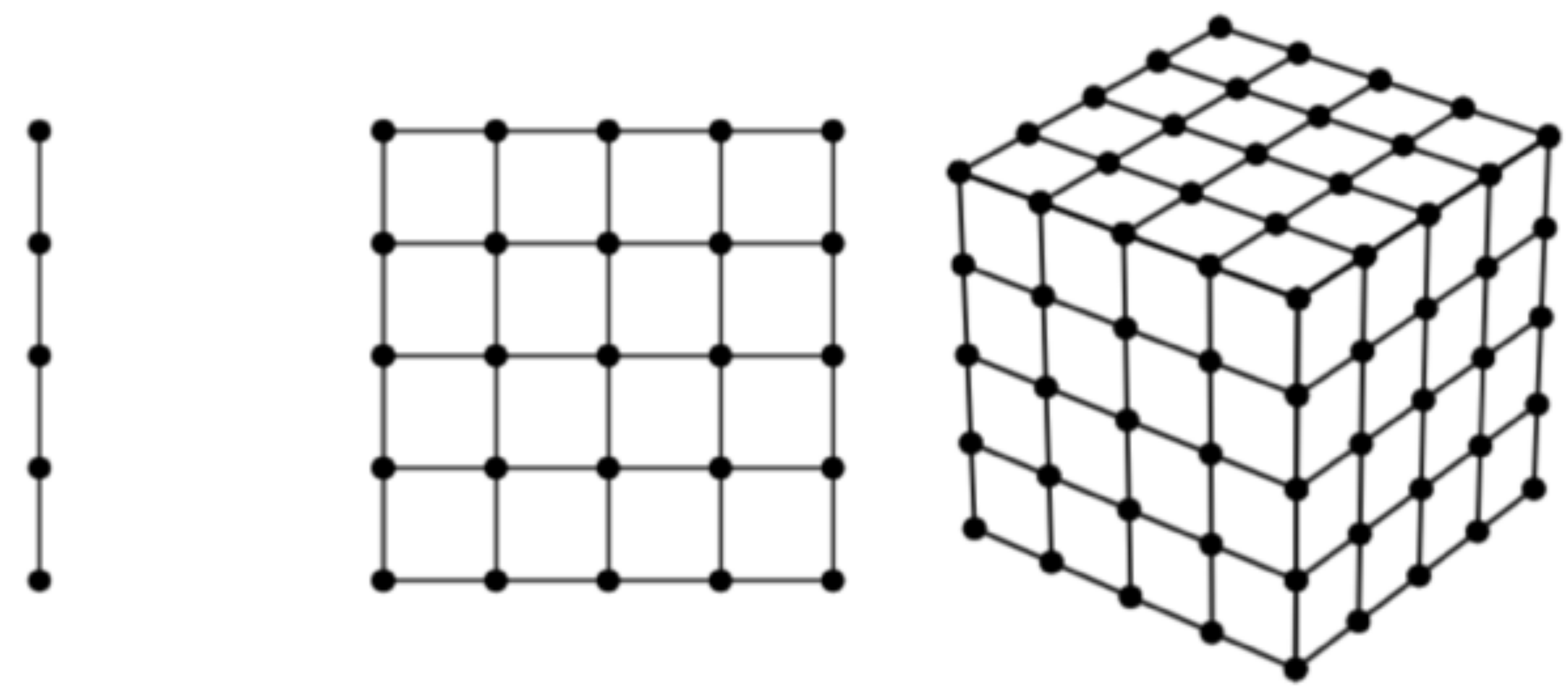
http://persson.berkeley.edu/228A/Fall10/doc/lec41-2x3.pdf

Incomplete LU (Cholesky) factorization

- Choice of parameters
  - Good: Smooth transition from iterative to direct methods
  - Bad: Very ad hoc, problem-dependent
  - Trade-off: Time per iteration vs # of iterations (more fill $\rightarrow$ more time
    but fewer iterations)
- Effectiveness
  - Condition number usually improves (only) by constant factor (except MIC for some problems from PDEs)
  - Still, often good when tuned for a particular class of problems
- Parallelism
  - Triangular solves are not very parallel
  - Reordering for parallel triangular solve by graph coloring

Incomplete LU (Cholesky) factorization

- Time to solve the Poisson model problem on regular mesh with N nodes:



Solver	1-D	2-D	3-D
Sparse Cholesky	$O(N)$	$O(N^{1.5})$	$O(N^2)$
CG, exact arith.	$O(N^2)$	$O(N^2)$	$O(N^2)$
CG, no precond.	$O(N^2)$	$O(N^{1.5})$	$O(N^{1.33})$
CG, modified IC	$O(N^{1.5})$	$O(N^{1.25})$	$O(N^{1.17})$
Multigrid	$O(N)$	$O(N)$	$O(N)$

http://persson.berkeley.edu/228A/Fall10/doc/lec41-2x3.pdf

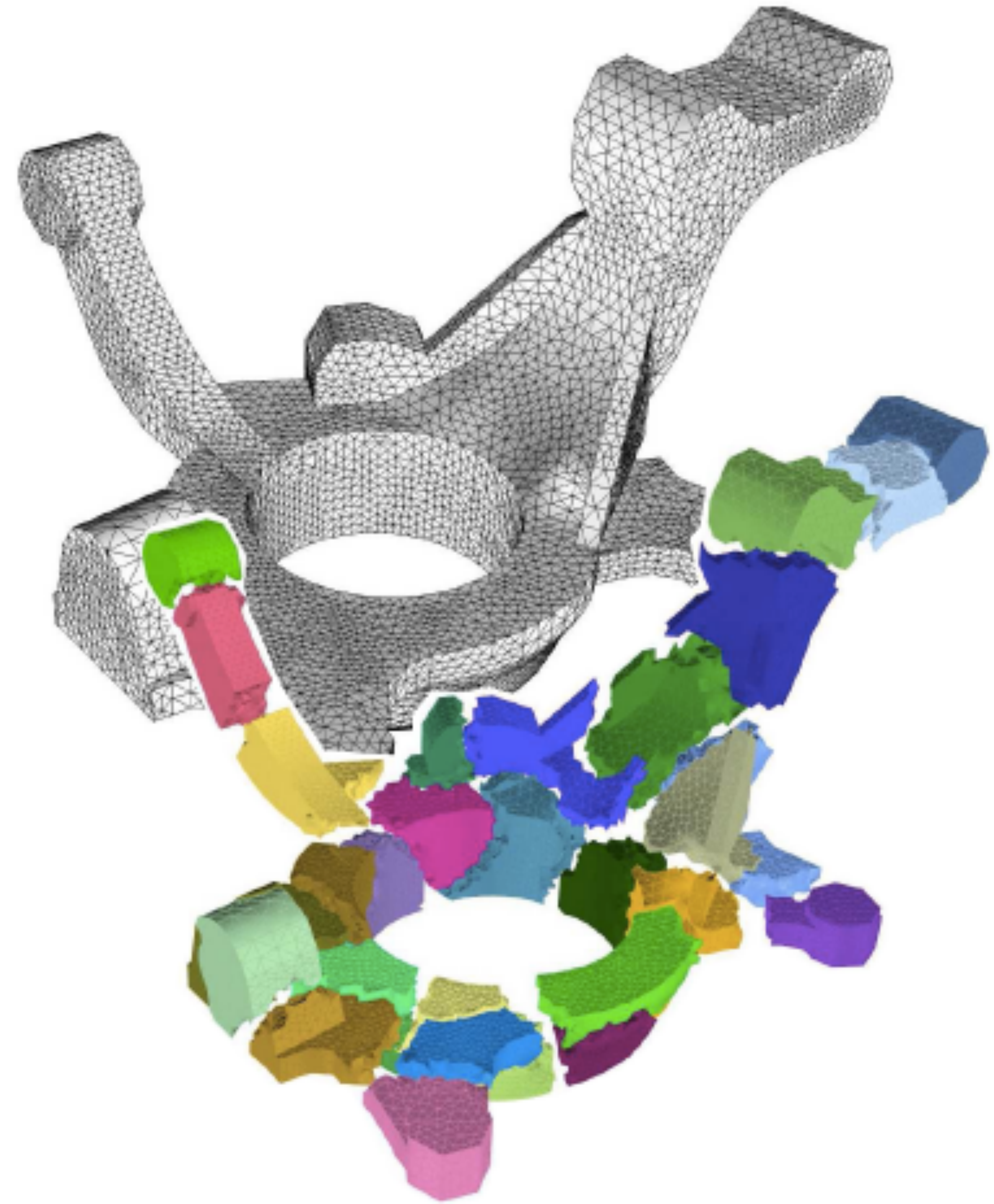
Domain Decomposition

Divide & conquer:

Solve **large** problem
by solving **sequence** of
local (smaller) problems

Applications:

- Iterative solver
- Parallelization
- Coupling different discretizations (BEM, FEM, IGA?)
- Multi physics



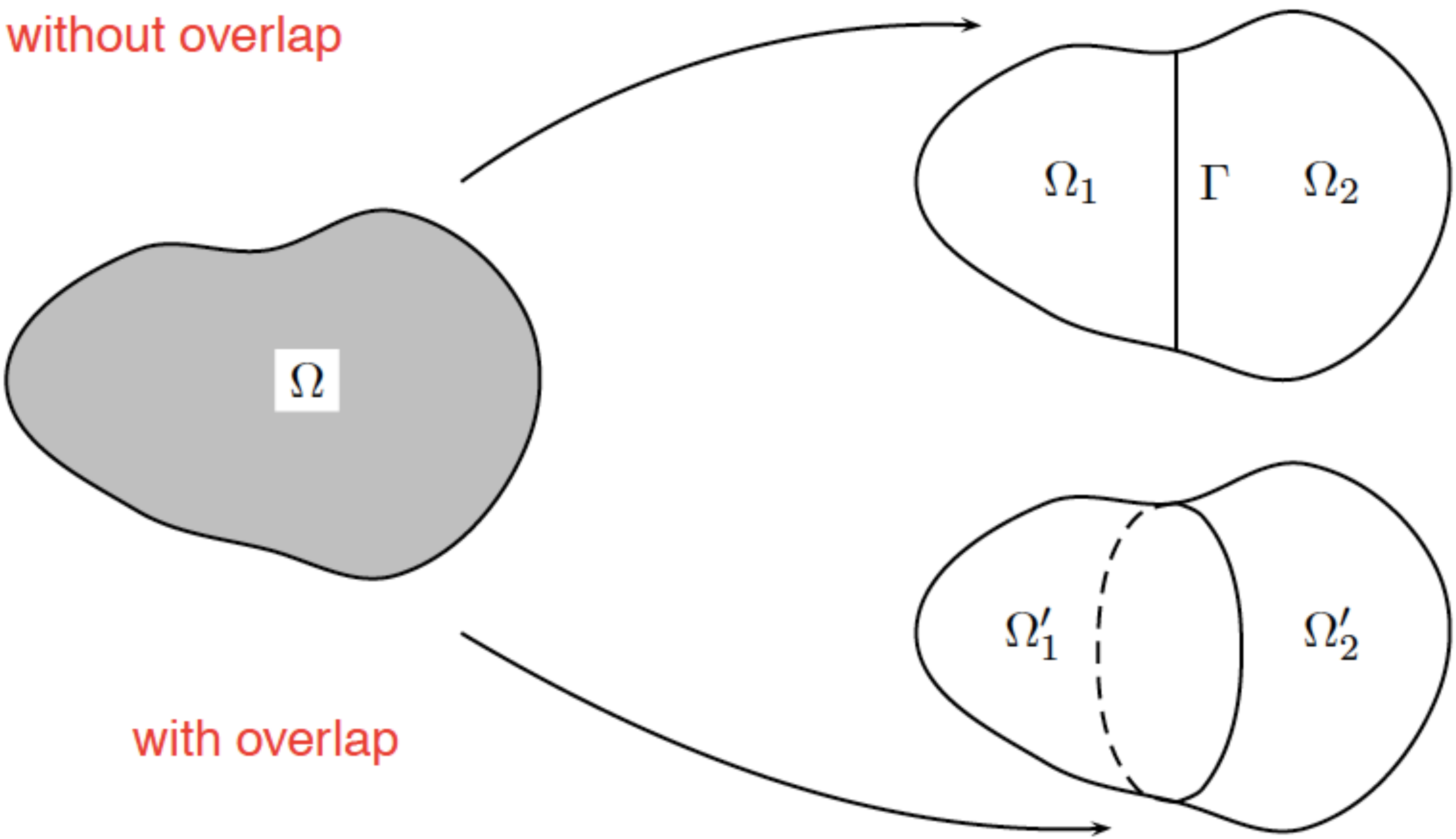
Courtesy of Charbel Farhat

http://people.ricam.oeaw.ac.at/c.pechstein/pechstein-IGAA.pdf

Domain Decomposition

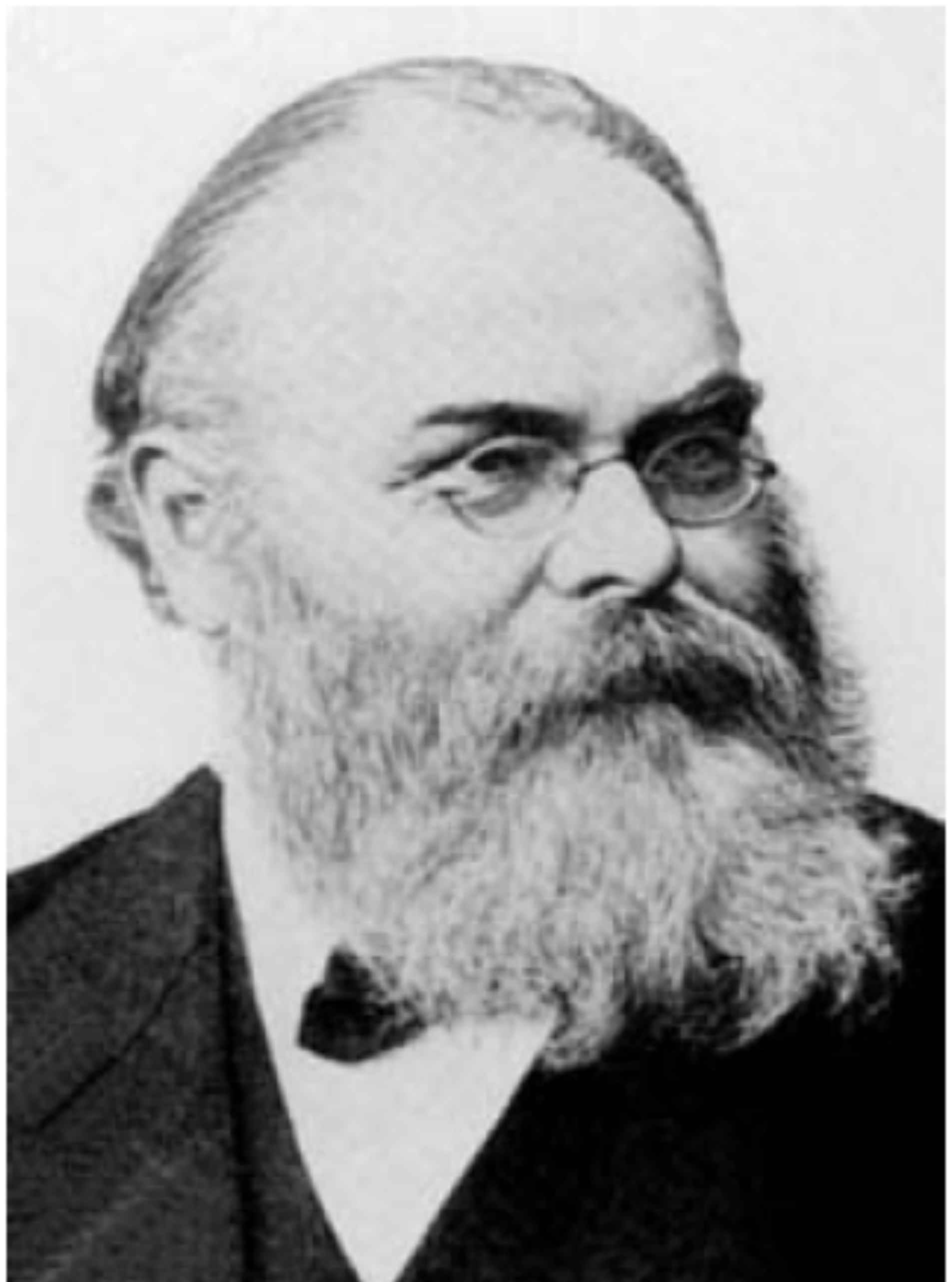
Decomposition of Ω

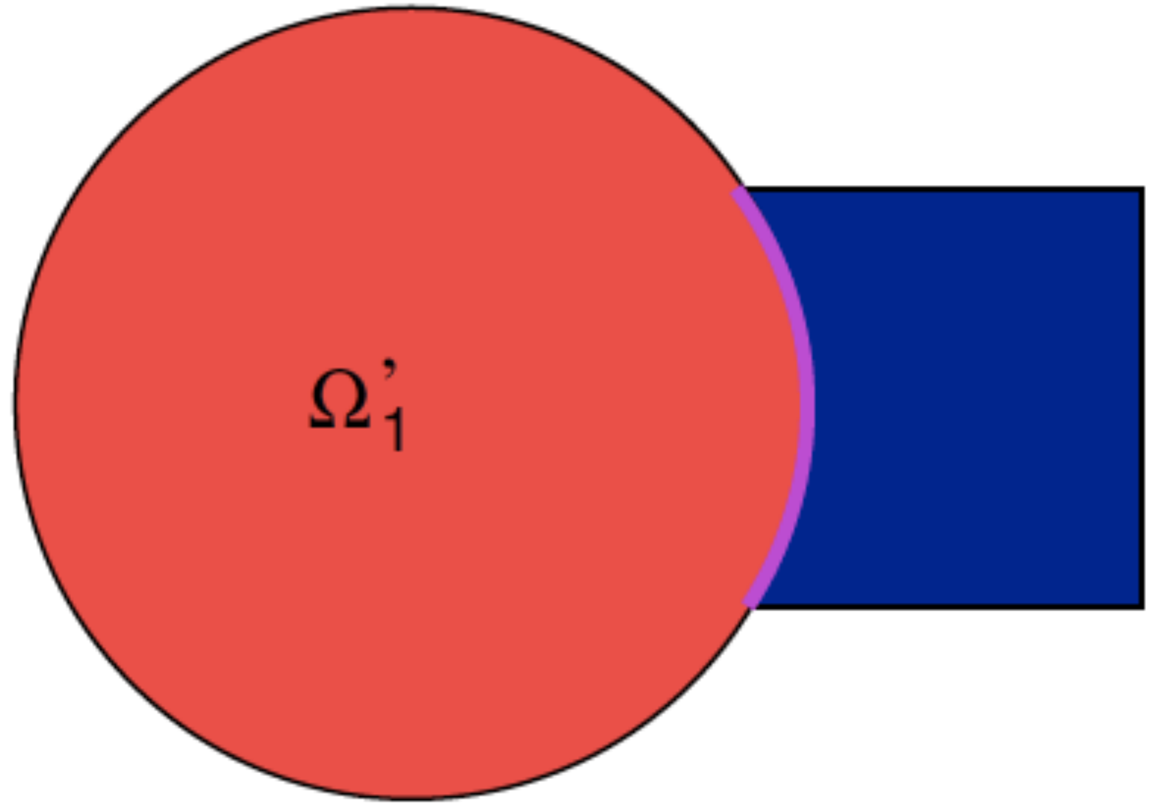
Let us decompose Ω in two subdomains:



Alternating Schwarz Method

Hermann Amandus
Schwarz
1843 – 1921





Schwarz's alternating method:

$u^{(0)} =$ given, satisfying B.C.

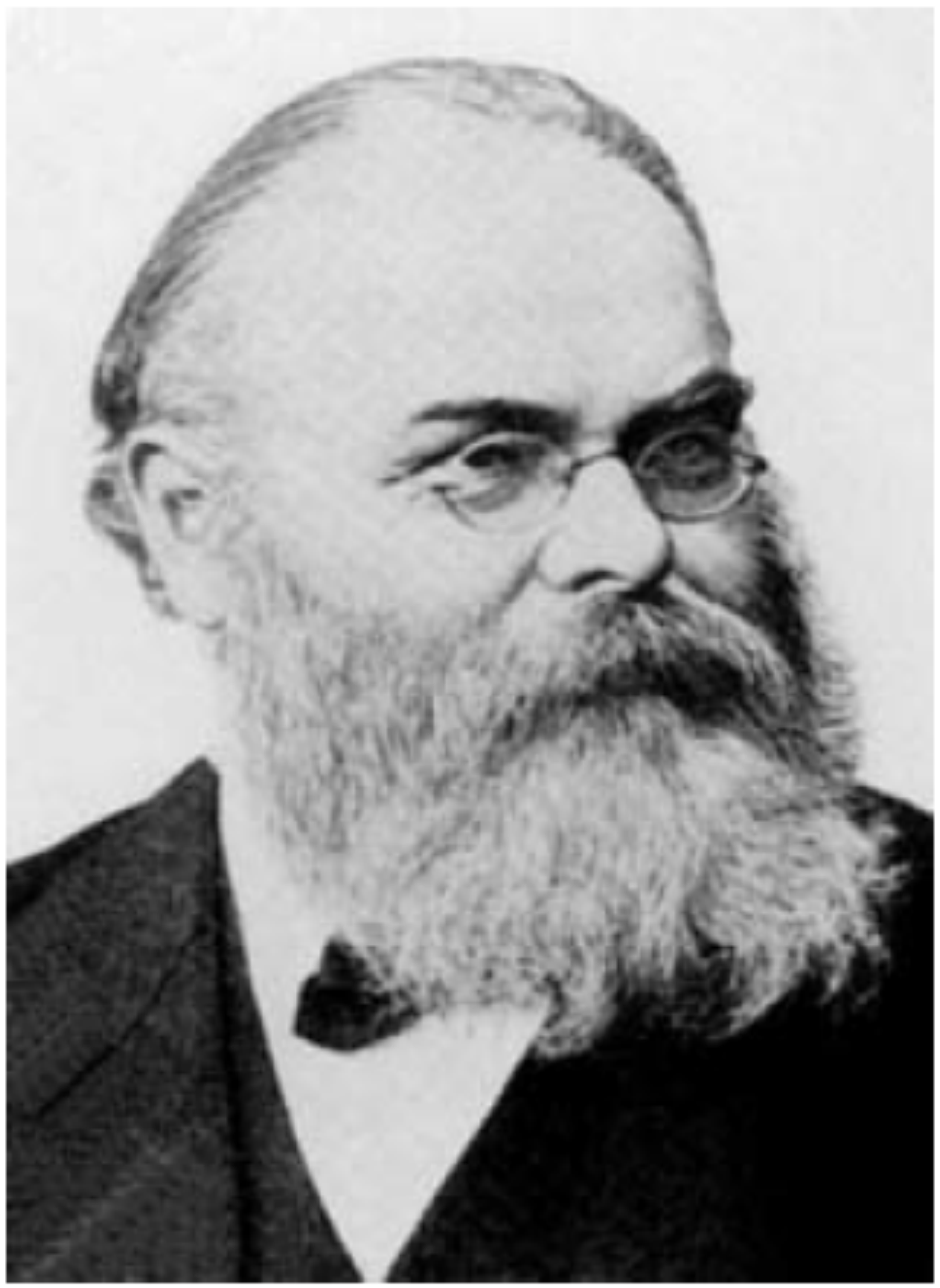
$$u^{(n+1/2)} : \begin{cases} -\Delta u^{(n+1/2)} = f & \text{in } \Omega'_1 \\ u^{(n+1/2)} = u^{(n)} & \text{on } \partial\Omega'_1 \\ u^{(n+1/2)} = u^{(n)} & \text{on } \Omega \setminus \Omega'_1 \end{cases}$$

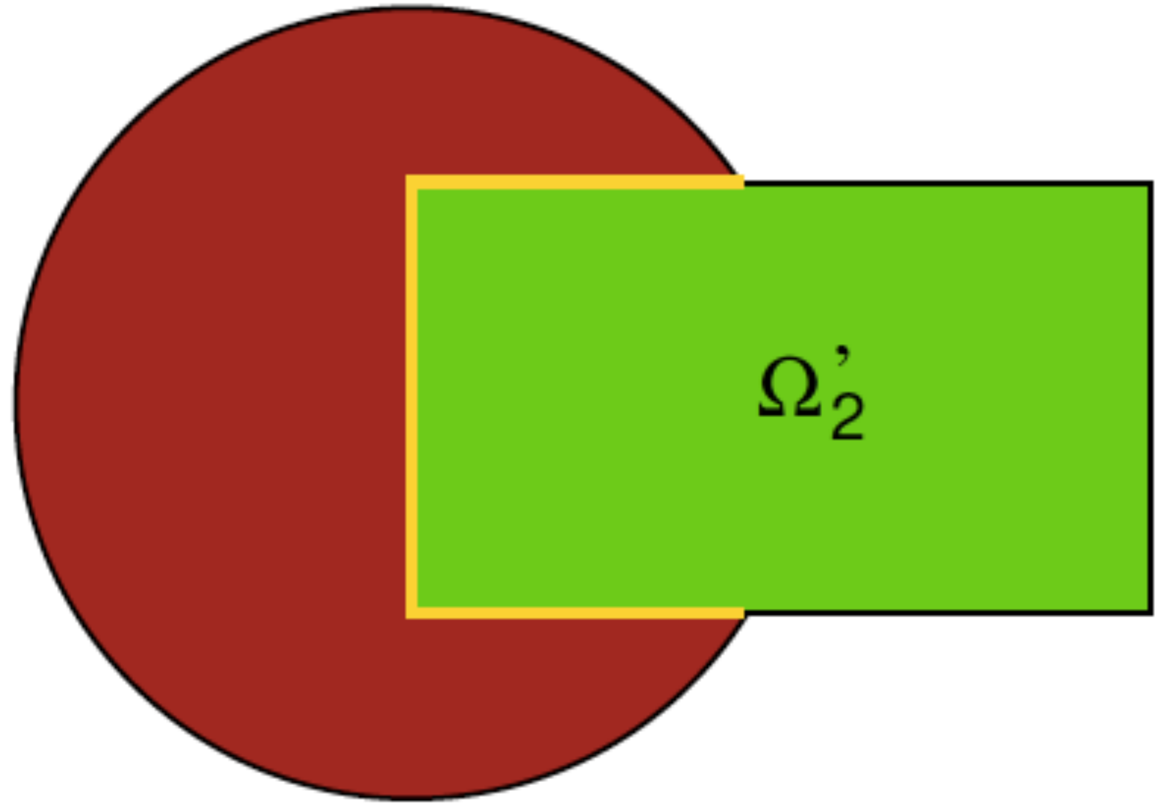
$$u^{(n+1)} : \begin{cases} -\Delta u^{(n+1)} = f & \text{in } \Omega'_2 \\ u^{(n+1)} = u^{(n+1/2)} & \text{on } \partial\Omega'_2 \\ u^{(n+1)} = u^{(n+1/2)} & \text{on } \Omega \setminus \Omega'_2 \end{cases}$$

http://people.ricam.oeaw.ac.at/c.pechstein/pechstein-IGAA.pdf

Alternating Schwarz Method

Hermann Amandus
Schwarz
1843 – 1921





Schwarz's alternating method:

$u^{(0)} =$ given, satisfying B.C.

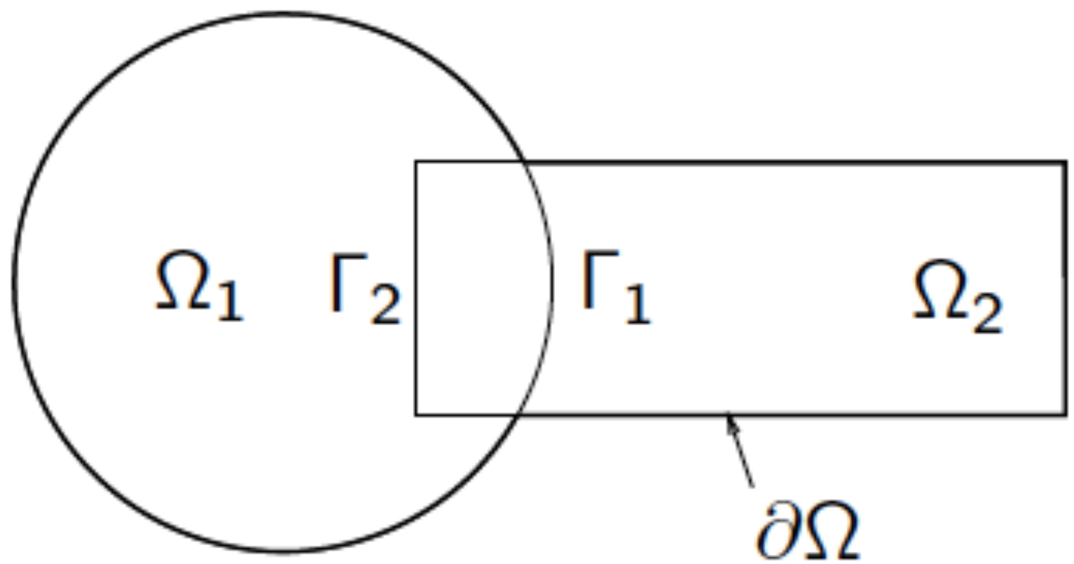
$$u^{(n+1/2)} : \begin{cases} -\Delta u^{(n+1/2)} = f & \text{in } \Omega_1' \\ u^{(n+1/2)} = u^{(n)} & \text{on } \partial\Omega_1' \\ u^{(n+1/2)} = u^{(n)} & \text{on } \Omega \setminus \Omega_1' \end{cases}$$

$$u^{(n+1)} : \begin{cases} -\Delta u^{(n+1)} = f & \text{in } \Omega_2' \\ u^{(n+1)} = u^{(n+1/2)} & \text{on } \partial\Omega_2' \\ u^{(n+1)} = u^{(n+1/2)} & \text{on } \Omega \setminus \Omega_2' \end{cases}$$

http://people.ricam.oeaw.ac.at/c.pechstein/pechstein-IGAA.pdf

Alternating Schwarz Method

Schwarz invents a method to proof that the infimum is attained: for a general domain $\Omega := \Omega_1 \cup \Omega_2$:



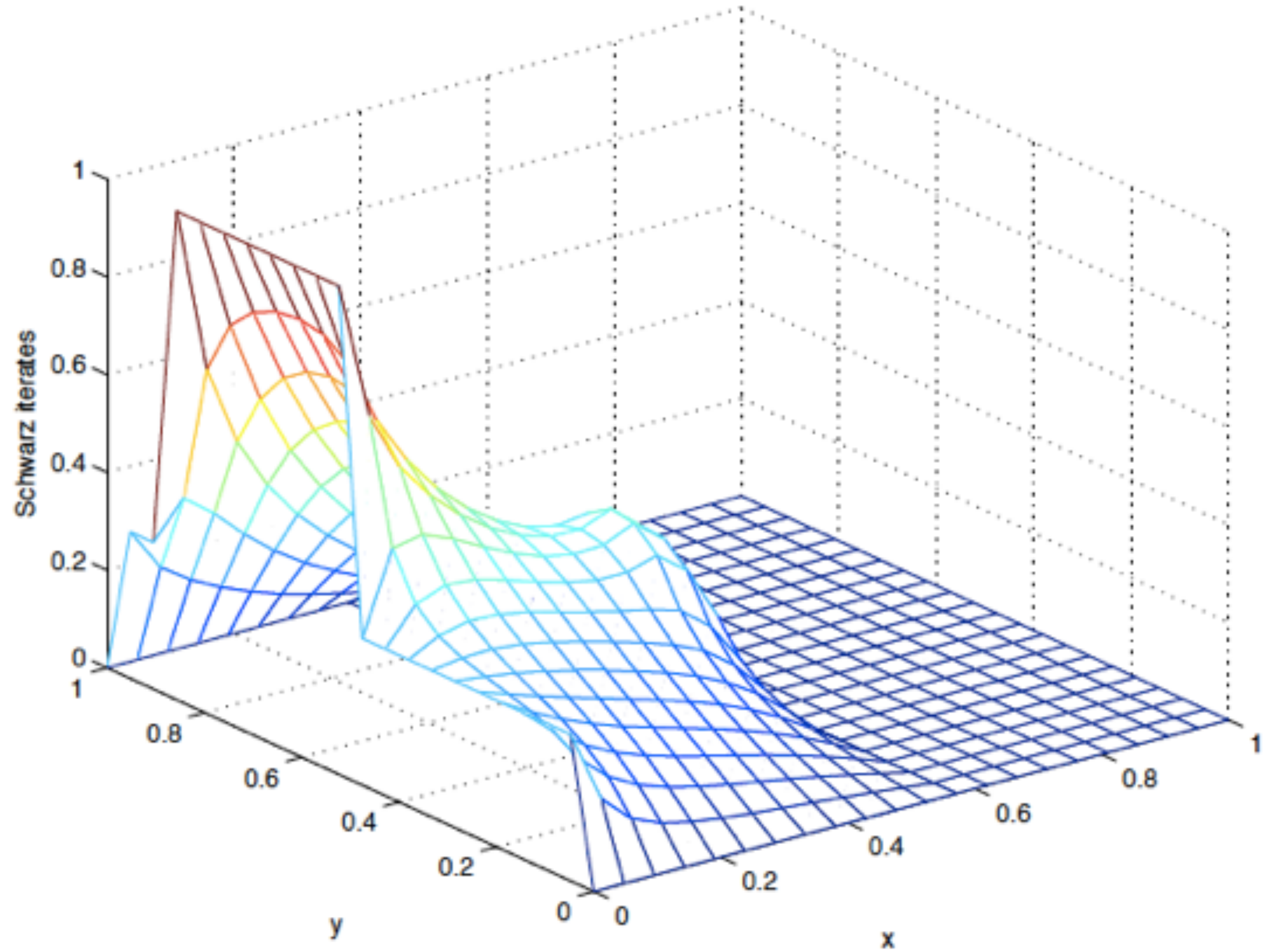
$$\begin{aligned} \Delta u_1^n &= 0 && \text{in } \Omega_1 \\ u_1^n &= g && \text{on } \partial\Omega \cap \overline{\Omega}_1 \\ u_1^n &= u_2^{n-1} && \text{on } \Gamma_1 \end{aligned} \qquad \begin{aligned} \Delta u_2^n &= 0 && \text{in } \Omega_2 \\ u_2^n &= g && \text{on } \partial\Omega \cap \overline{\Omega}_2 \\ u_2^n &= u_1^n && \text{on } \Gamma_2 \end{aligned}$$

solve on the disk   solve on the rectangle

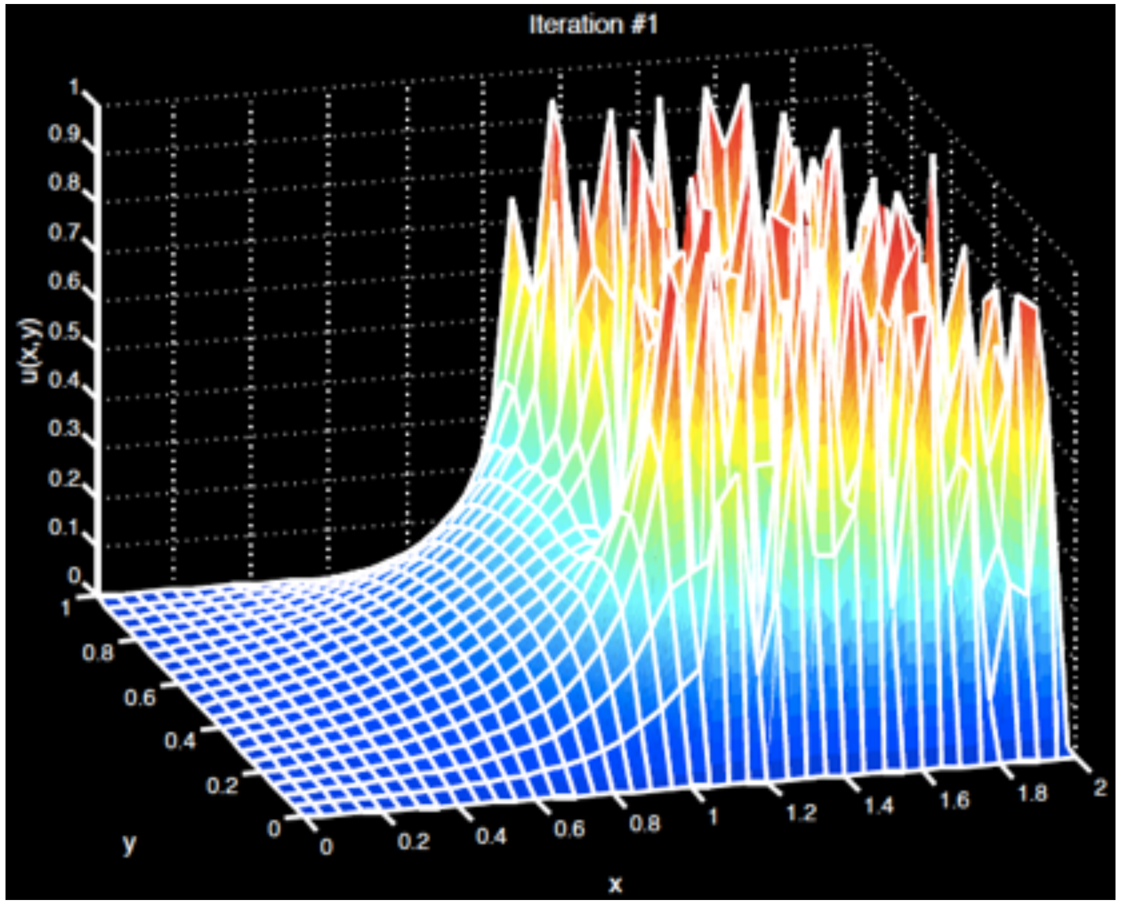
- Schwarz proved convergence in 1869 using the maximum principle.

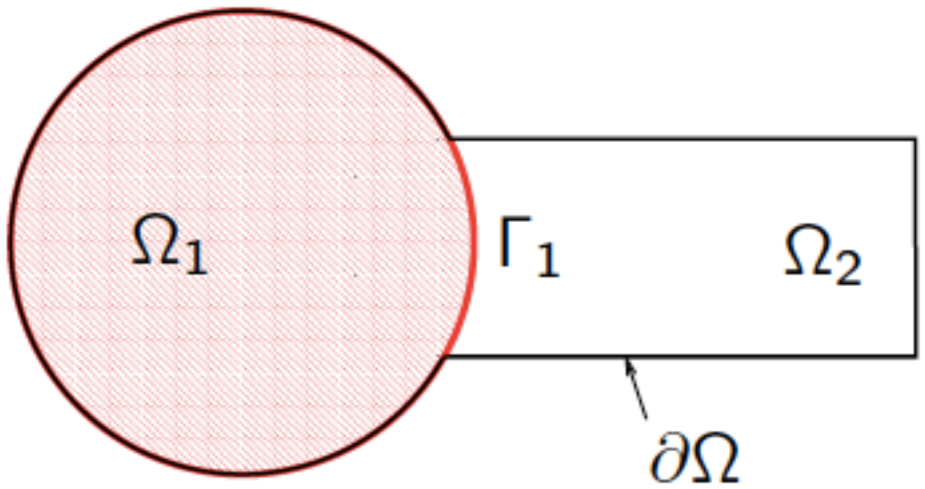
Alternating Schwarz Method

Iteration 1



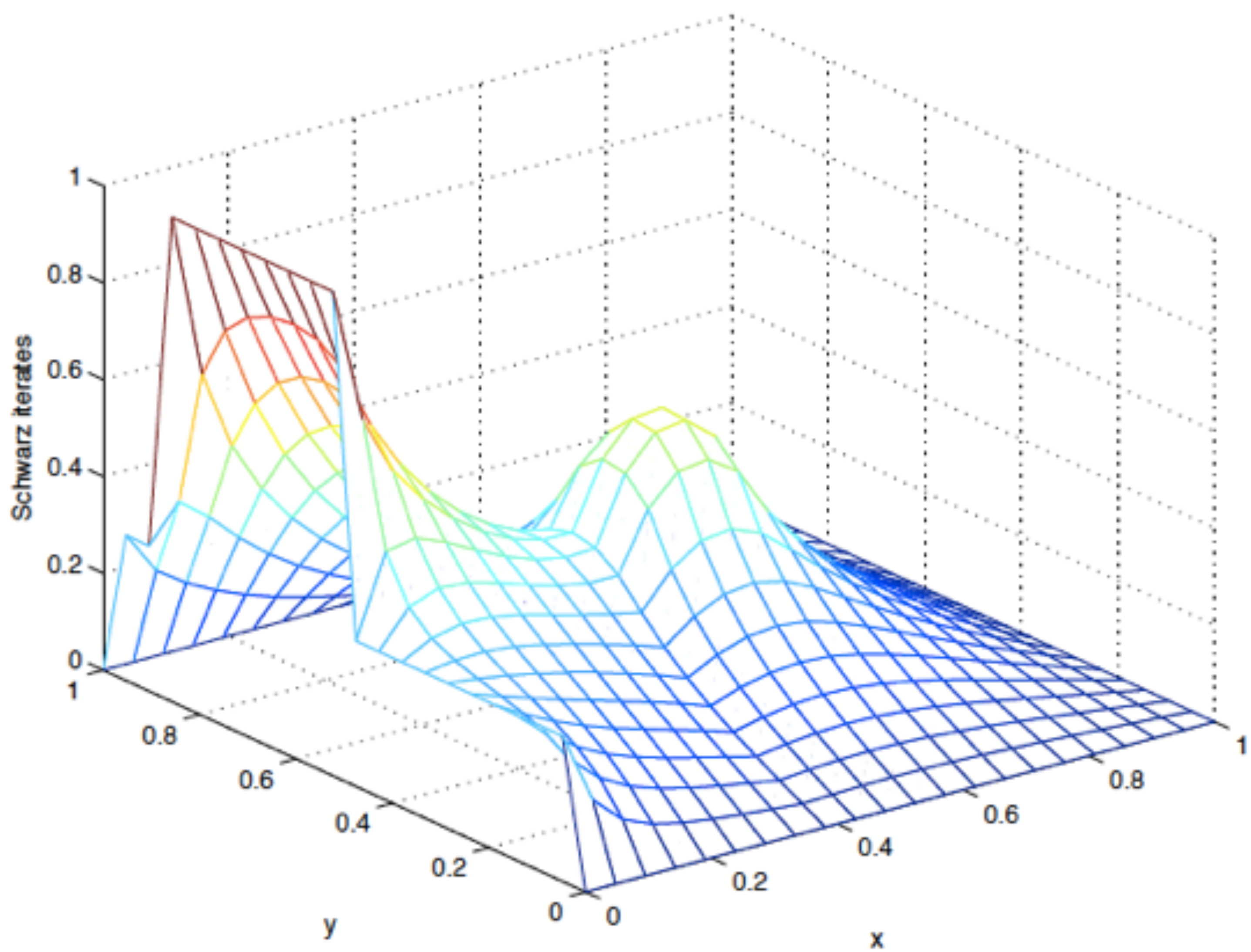
Error



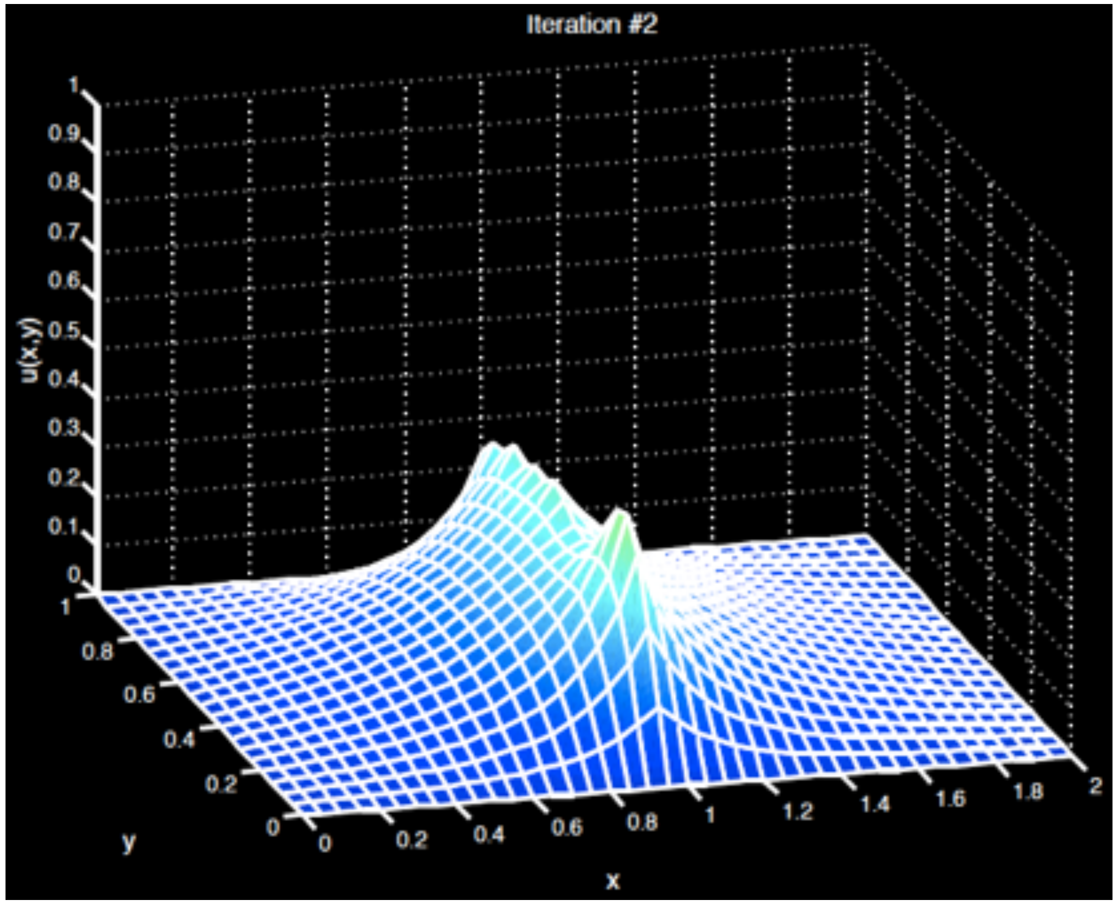


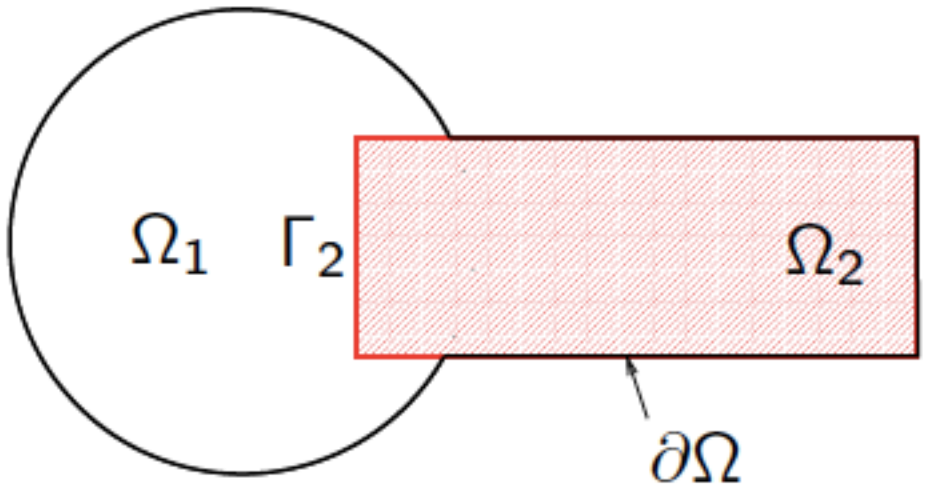
Alternating Schwarz Method

Iteration 2



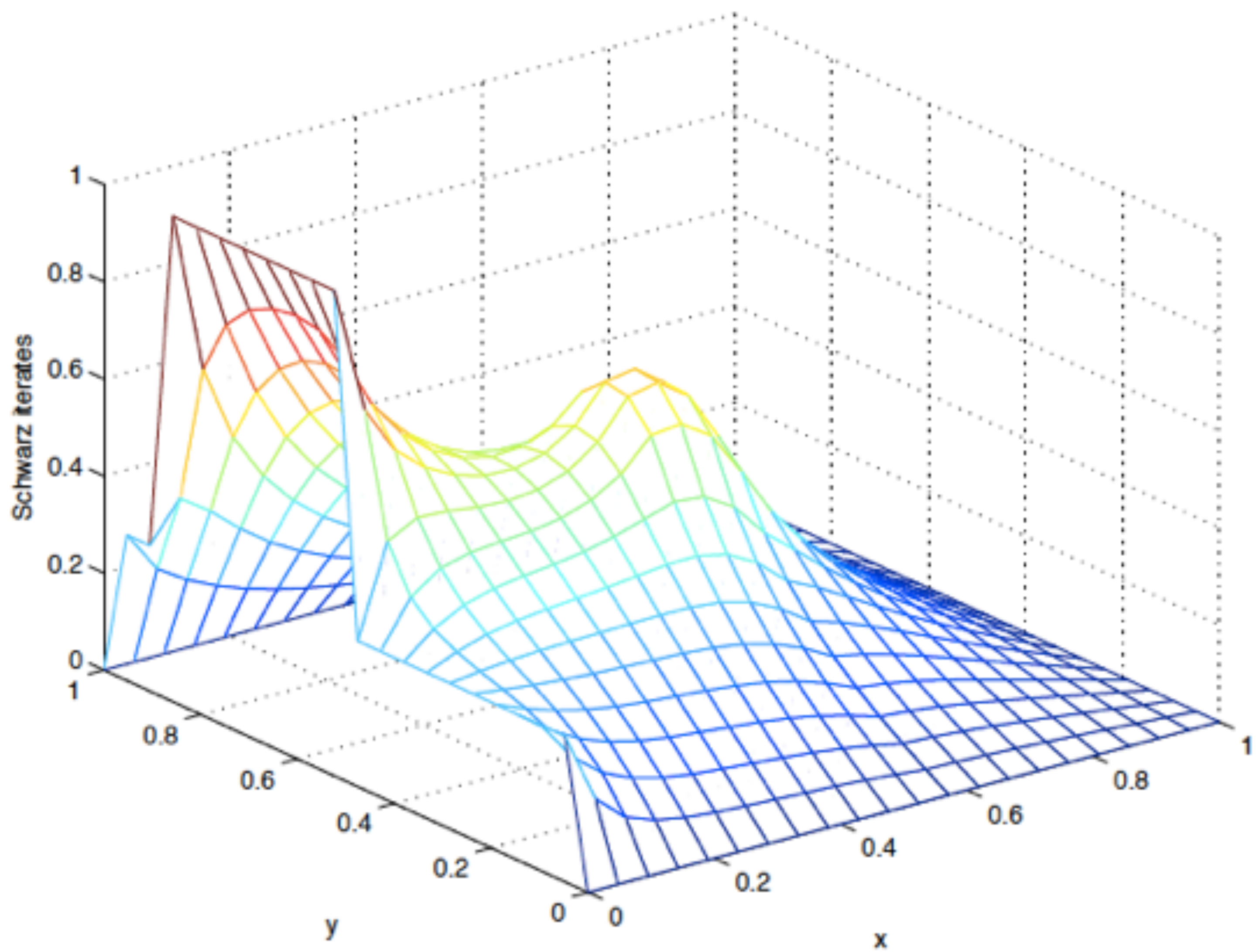
Error



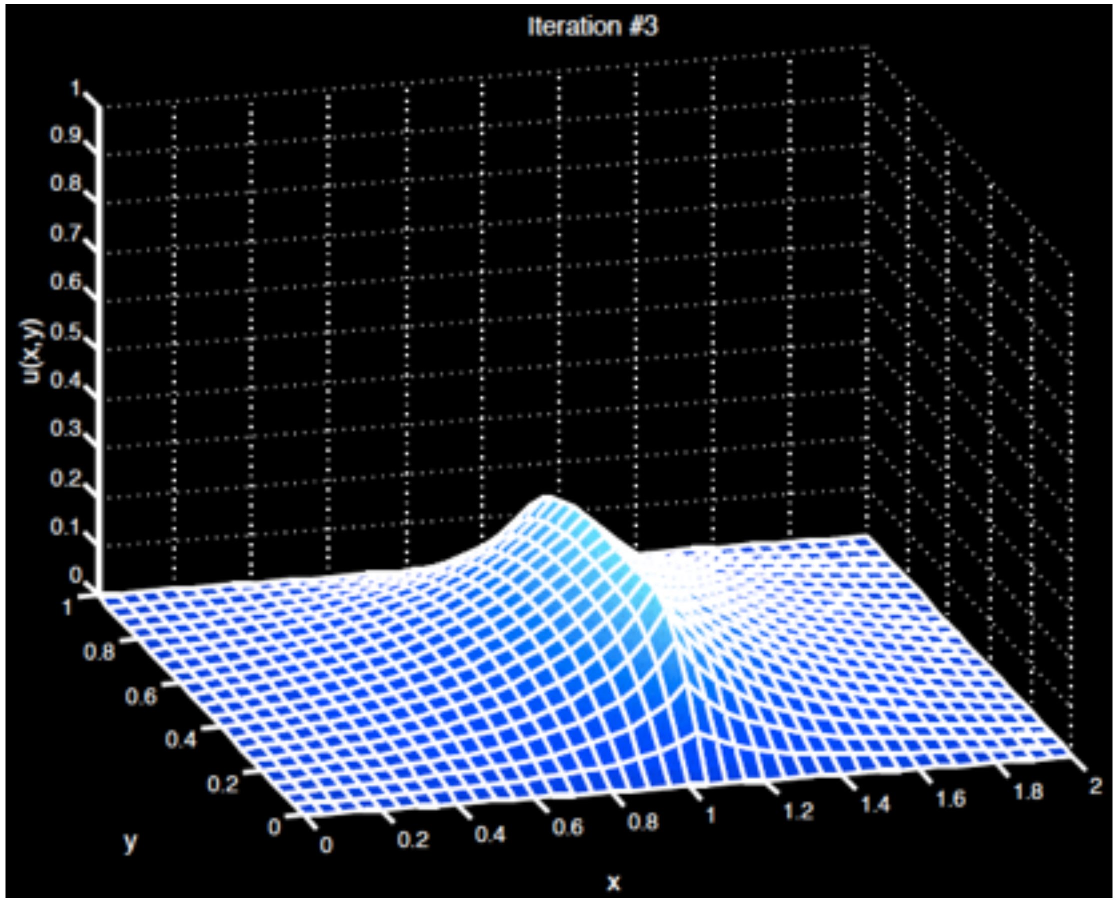


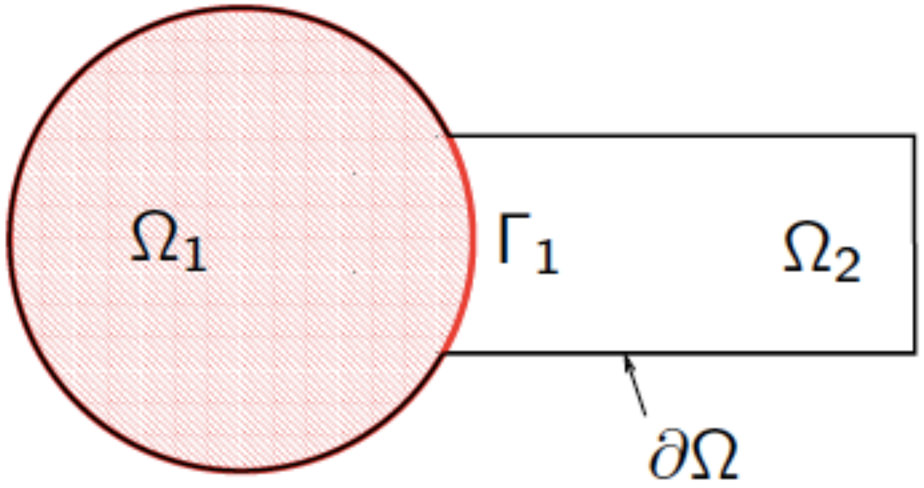
Alternating Schwarz Method

Iteration 3



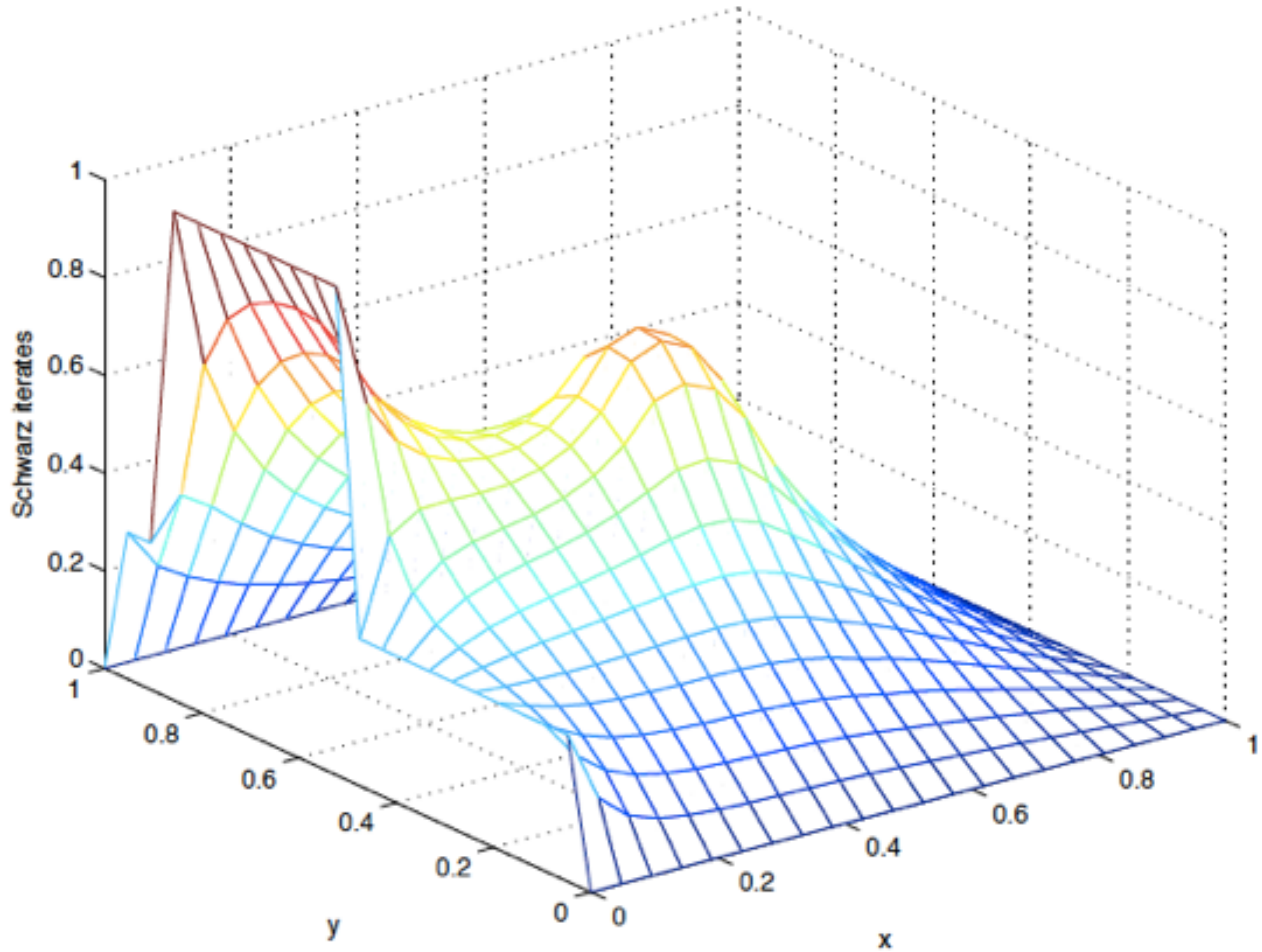
Error



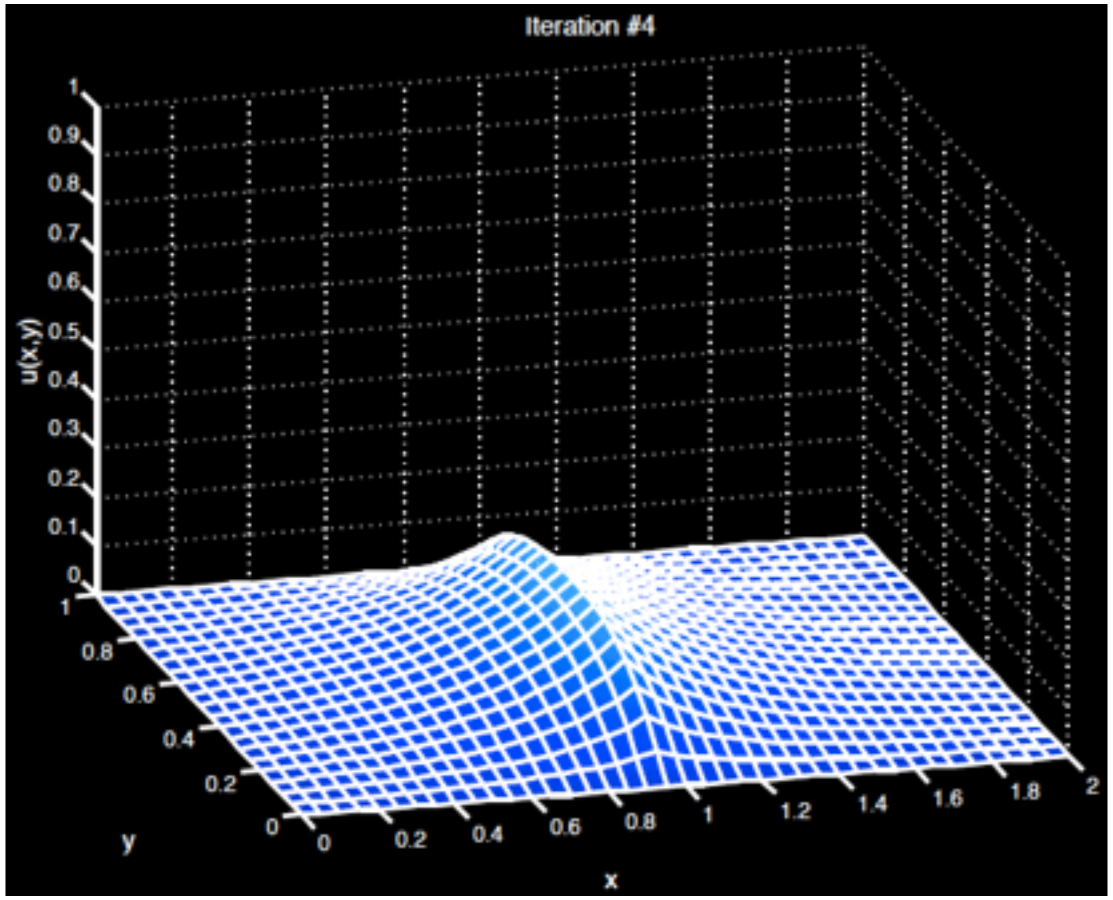


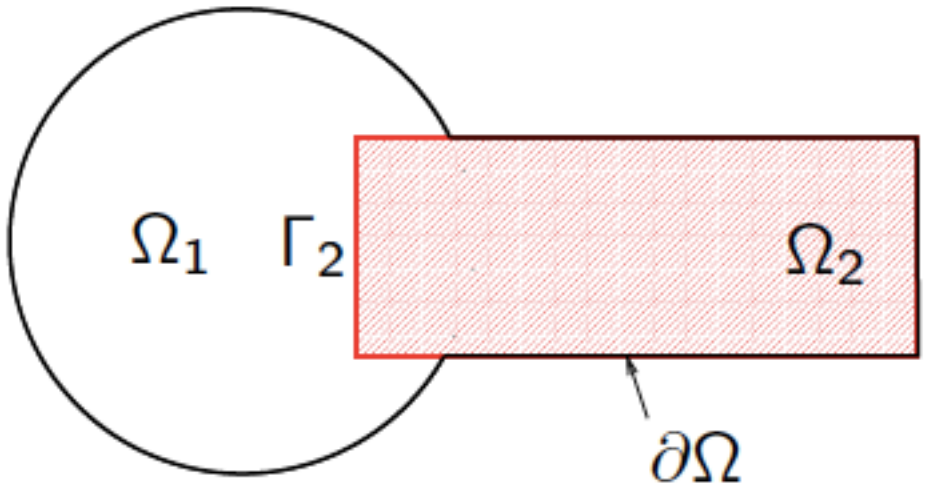
Alternating Schwarz Method

Iteration 5



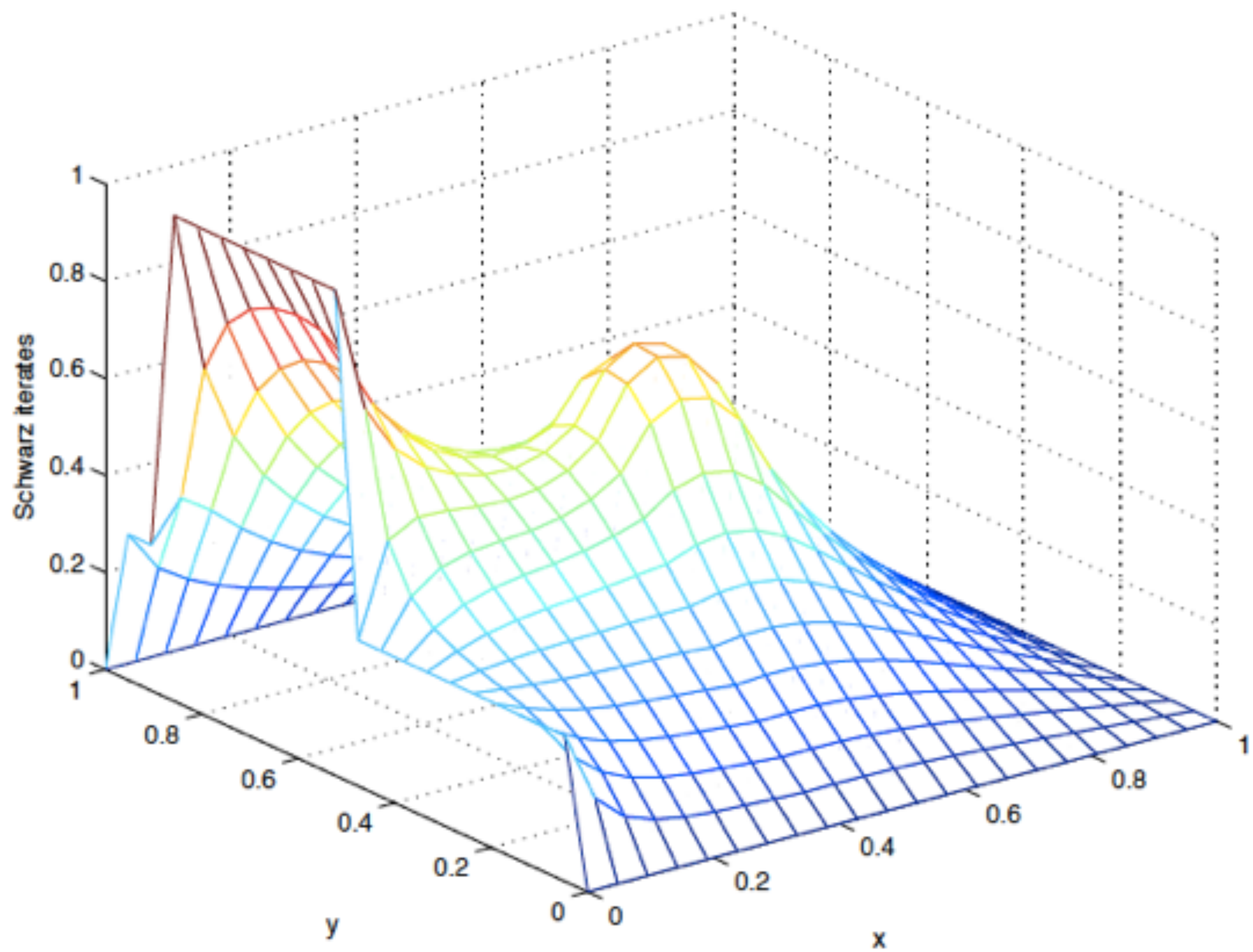
Error



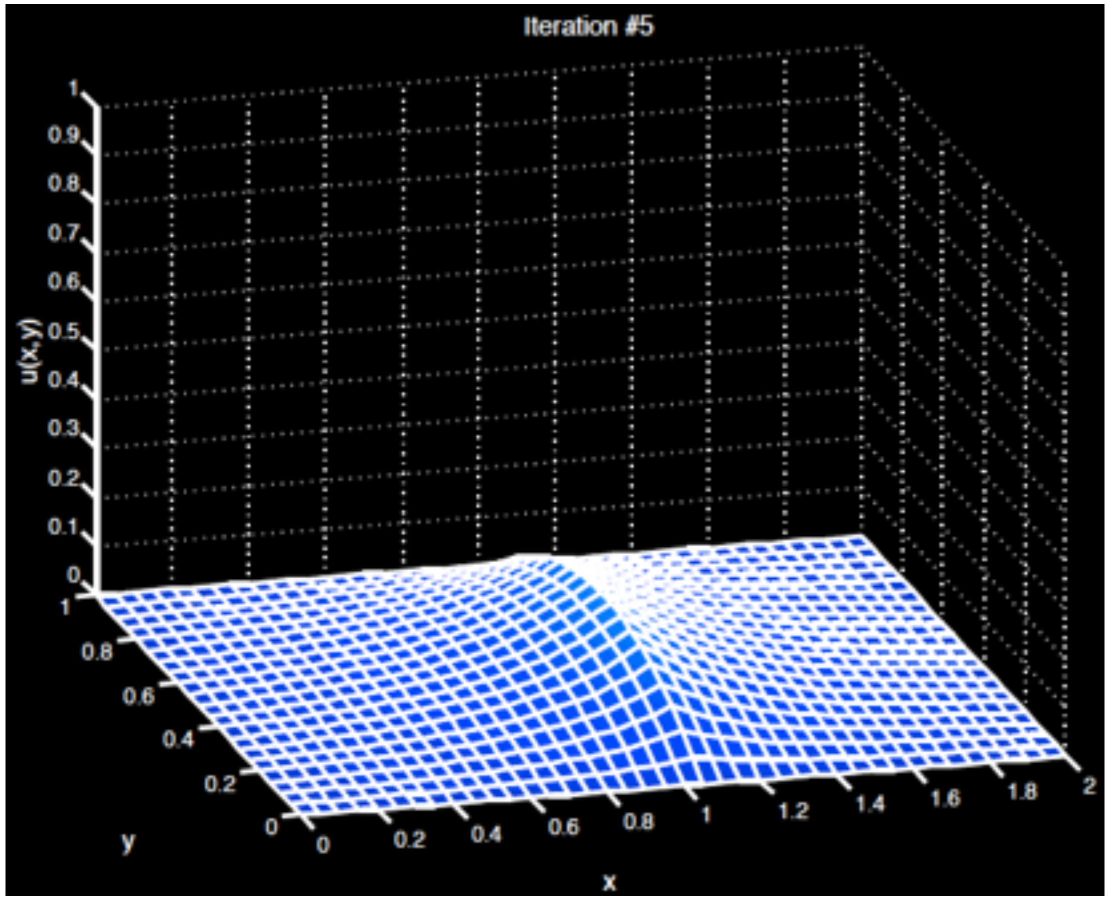


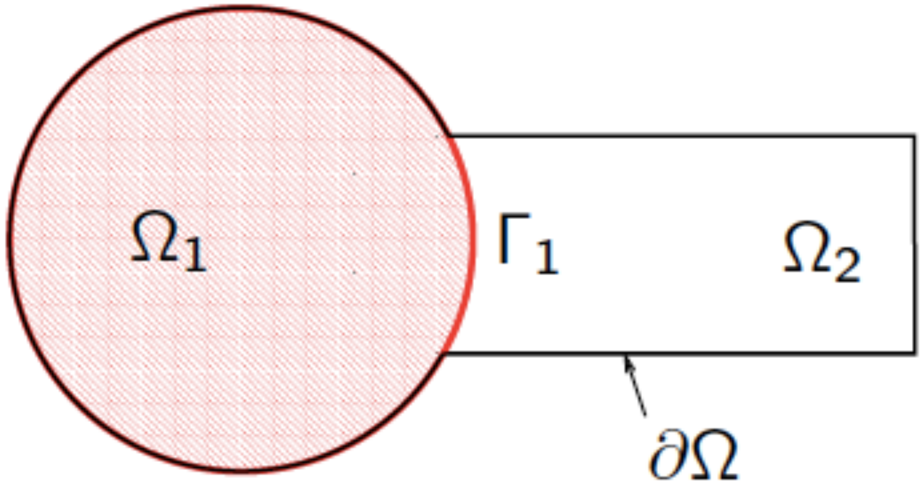
Alternating Schwarz Method

Iteration 4



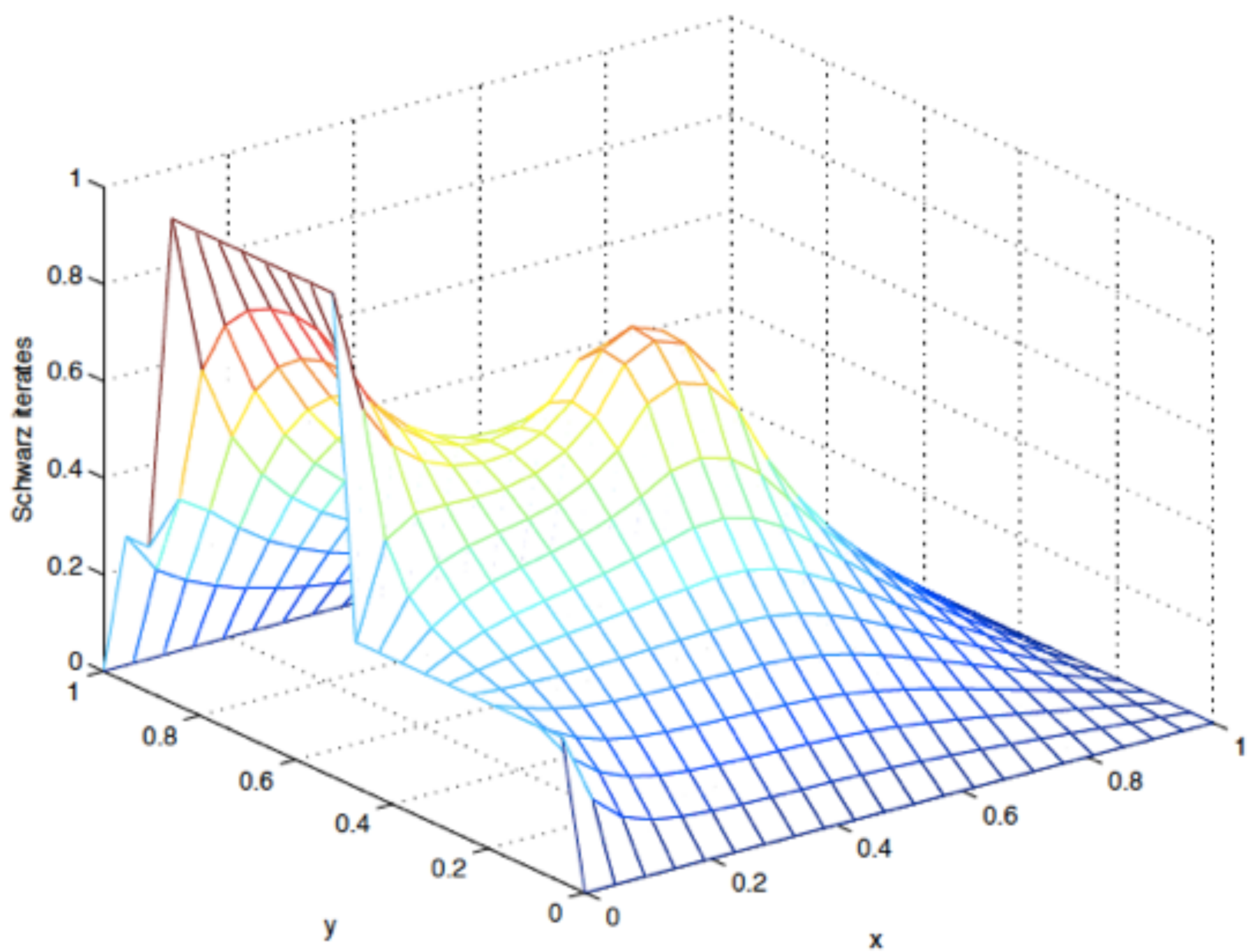
Error



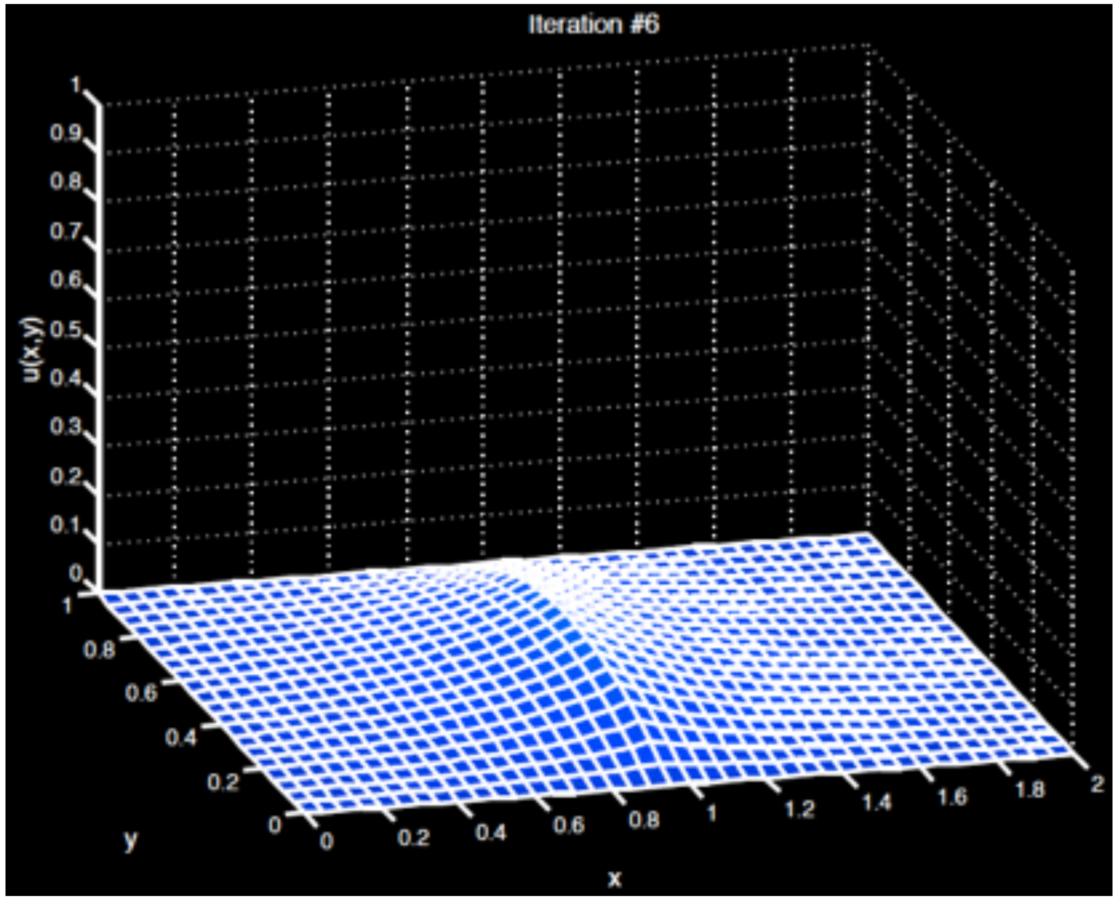


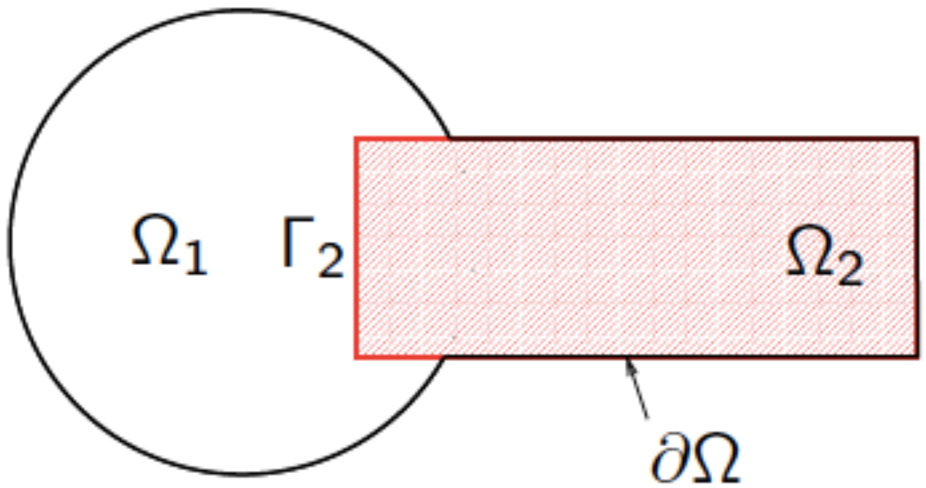
Alternating Schwarz Method

Iteration 6



Error





Restriction Operator

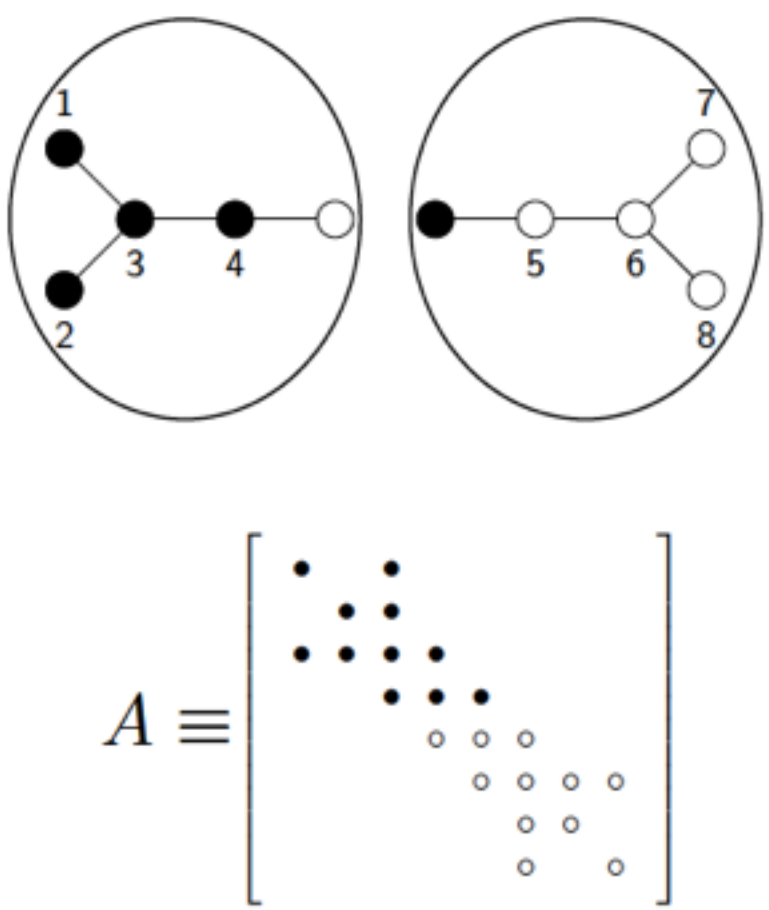
- Subdomain operator

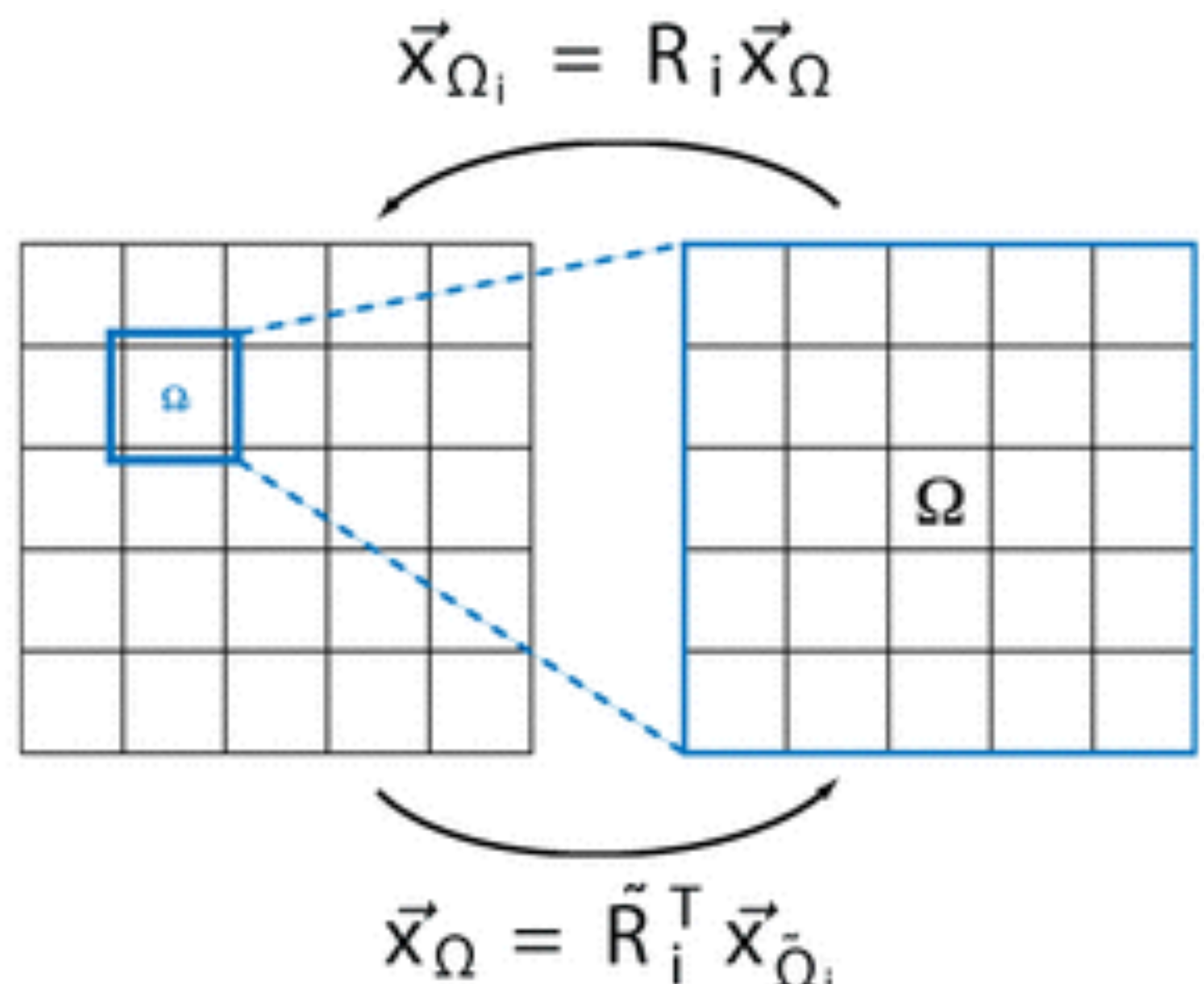
$$A_i = R_i^\delta A R_i^\delta$$

- R_i^δ is the "restriction" operator (mapping to subdomain)
- Overlapping additive Schwarz preconditioner

$$\boxed{M_{AS}^{-1} = \sum R_i^\delta A_i^{-1} R_i^\delta}$$

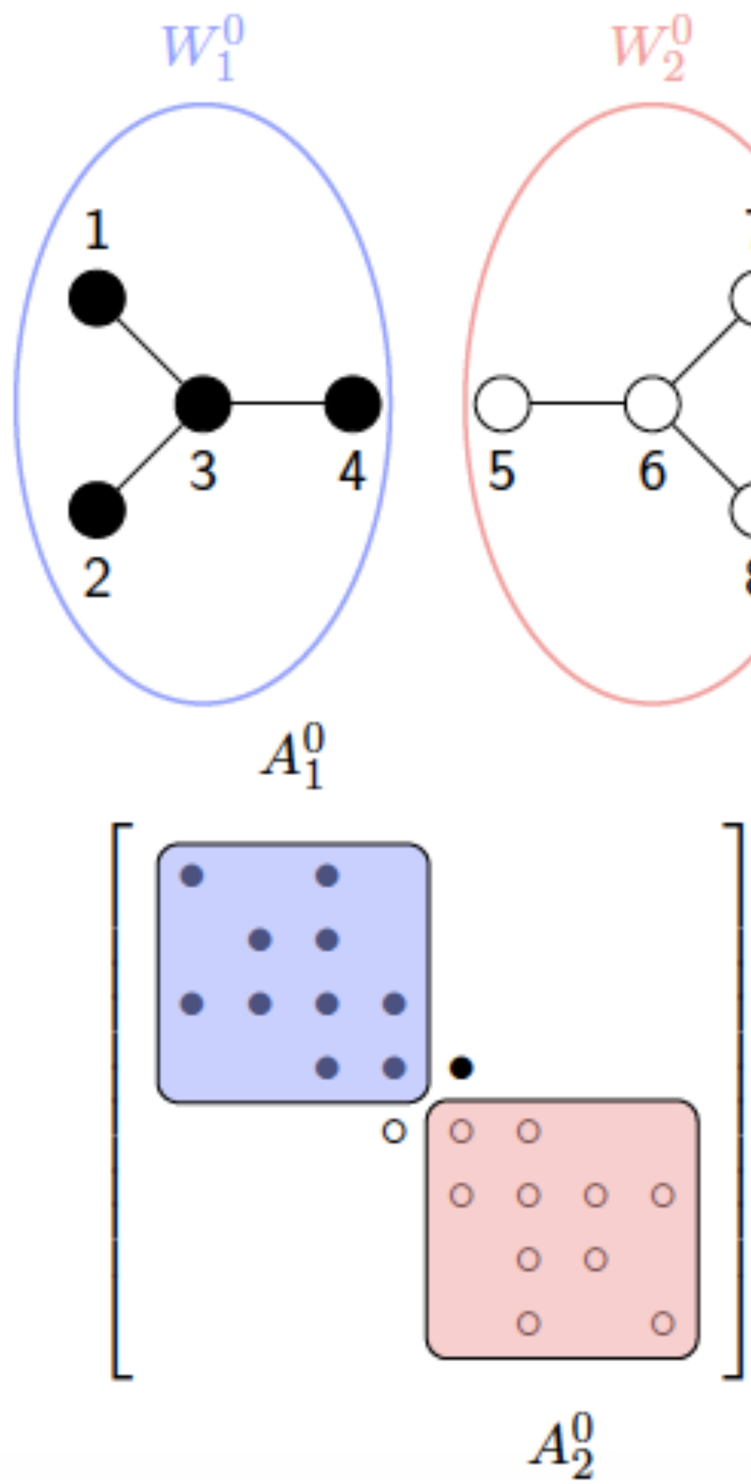
- A_i is not invertible but its restriction to the subspace is invertible





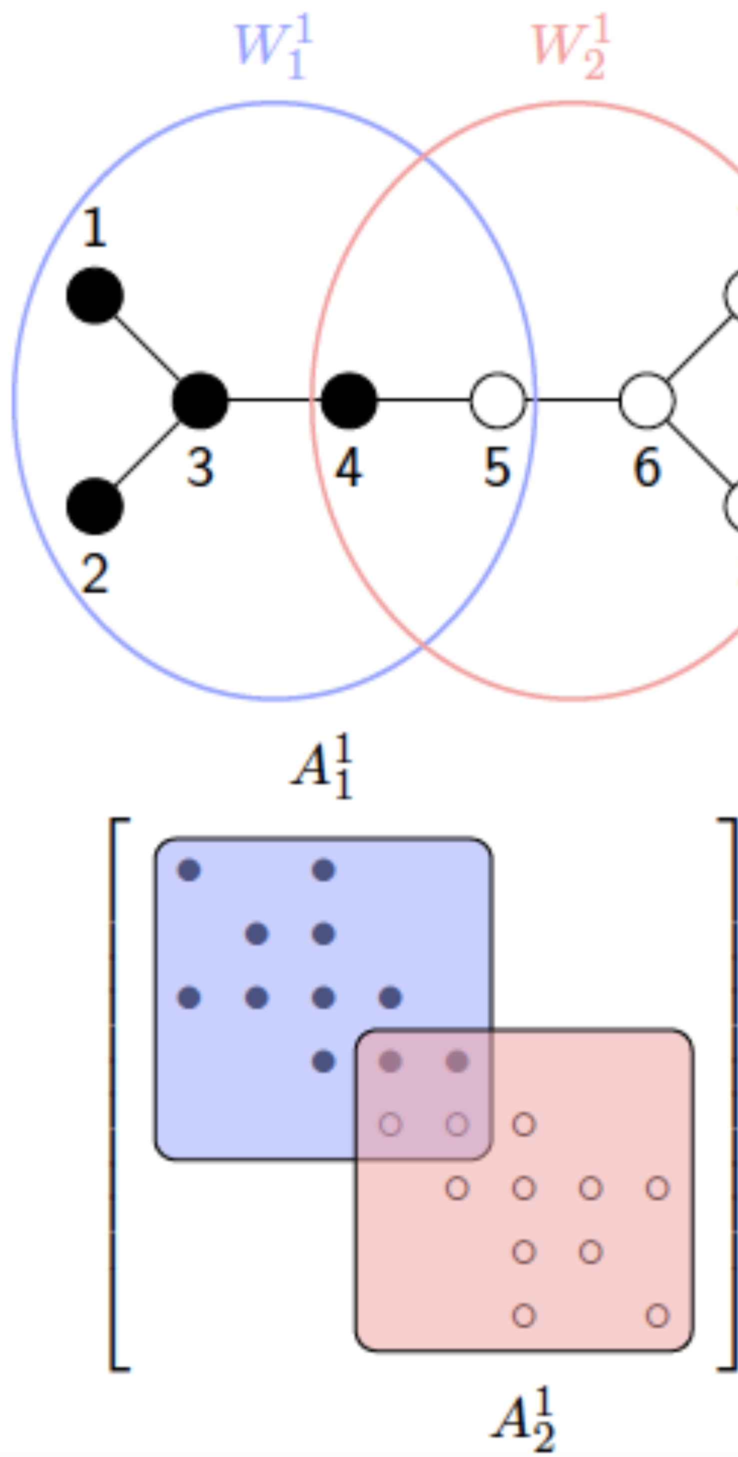
Non-overlapping Schwarz Methods

Overlap δ=0 (Block-Jacobi preconditioner)

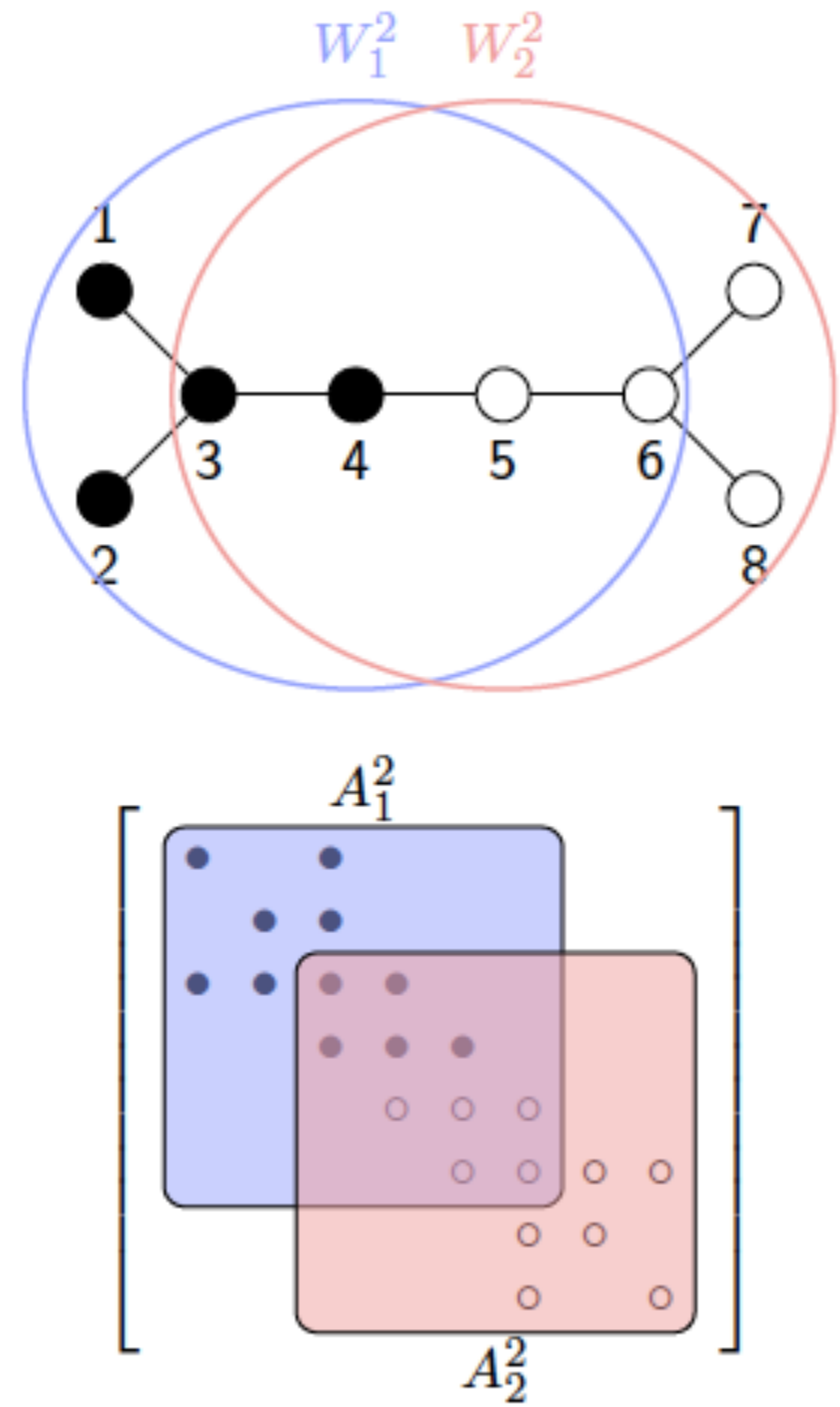


Overlapping Schwarz Methods

Overlap δ=1



Overlap δ=2



Additive & Multiplicative Schwarz Methods

- Subdomain operator

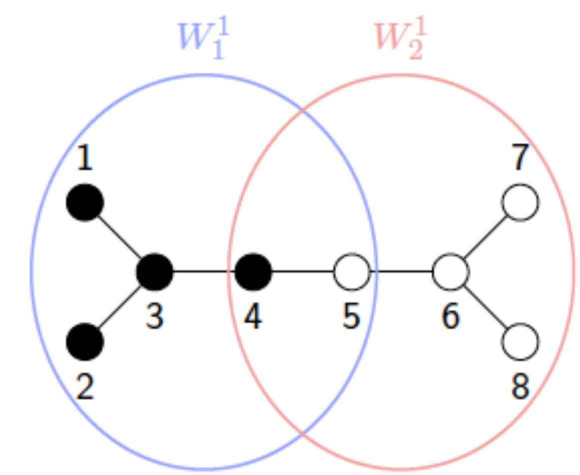
$$A_i = R_i^\delta A R_i^\delta$$

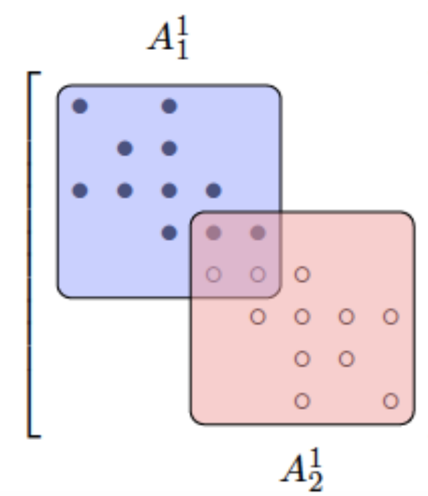
- R_i^δ is the "restriction" operator (mapping to subdomain)
- Overlapping additive Schwarz preconditioner

$$\boxed{M_{AS}^{-1} = \sum R_i^\delta A_i^{-1} R_i^\delta}$$

- A_i is not invertible but its restriction to the subspace is invertible

Overlap δ=1

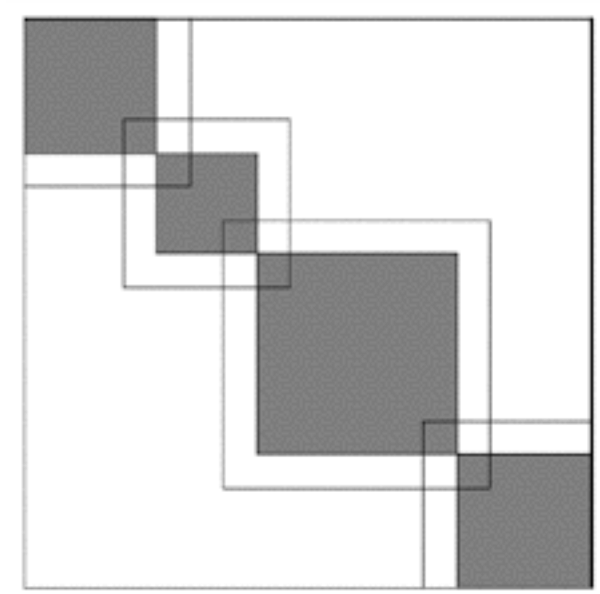




Multiplicative Schwarz $$M^{-1} = \prod R_i^\delta A_i^{-1} R_i^\delta$$

Additive Schwarz $$M^{-1} = \sum R_i^\delta A_i^{-1} R_i^\delta$$

Restricted Additive Schwarz $$M^{-1} = \sum R_i^0 A_i^{-1} R_i^\delta$$



Additive & Multiplicative Schwarz Methods

	Overlap = 0				Overlap = 1			
$H\backslash h$	1/32	1/64	1/128	1/256	1/32	1/64	1/128	1/256
1/4	15	22	30	43	13	17	25	34
1/8	20	28	40	60	15	22	31	46
1/16	26	39	55	80	20	29	42	61
1/32	31	54	77	111	23	40	57	83
	Overlap = 1%				Overlap = 2%			
1/4	13	17	20	25	13	15	18	20
1/8	15	22	26	33	15	18	23	26
1/16	20	29	34	43	20	25	30	34
1/32	23	40	47	58	23	30	41	44

05/09	Class 9	Dense direct solvers	Understand the principle of LU decomposition and the optimization and parallelization techniques that lead to the LINPACK benchmark.
05/12	Class 10	Dense eigensolvers	Determine eigenvalues and eigenvectors and understand the fast algorithms for diagonalization and orthonormalization.
05/16	Class 11	Sparse direct solvers	Understand reordering in AMD and nested dissection, and fast algorithms such as skyline and multifrontal methods.
05/19	Class 12	Sparse iterative solvers	Understand the notion of positive definiteness, condition number, and the difference between Jacobi, CG, and GMRES.
05/23	Class 13	Preconditioners	Understand how preconditioning affects the condition number and spectral radius, and how that affects the CG method.
05/26	Class 14	Multigrid methods	Understand the role of smoothers, restriction, and prolongation in the V-cycle.
05/30	Class 15	Fast multipole methods, H-matrices	Understand the concept of multipole expansion and low-rank approximation, and the role of the tree structure.