Iterative Solvers

Steffen Börm

Christian-Albrechts-Universität zu Kiel

AKG Modellierung und Simulation, RWTH Aachen

Overview

Overview

Model problem

Poisson equation:

$$-\Delta u(x) = f(x) \qquad \text{for all } x \in \Omega,$$
$$u(x) = 0 \qquad \text{for all } x \in \partial\Omega.$$

Variational formulation:

$$\langle \nabla v, \nabla u \rangle_{L^2(\Omega)} = \langle v, f \rangle_{L^2(\Omega)} \qquad \text{for all } v \in H_0^1(\Omega).$$

Finite element discretization:

$$a_{ij} := \langle \nabla \varphi_i, \nabla \varphi_j \rangle_{L^2(\Omega)}, \qquad b_i := \langle \varphi_i, f \rangle_{L^2(\Omega)}, \qquad u_h = \sum_j \varphi_j x_j$$

$$\Longrightarrow \quad Ax = b$$

Direct solvers

Simple approach: Use LR or Cholesky factorization.
$\rightarrow \mathcal{O}(n^3)$ operations required for n degrees of freedom.

George (1973): Nested dissection ordering reduces factorization fill-in.

- $\mathcal{O}(n^{3/2})$ for two-dimensional problems.
- $\mathcal{O}(n^2)$ for three-dimensional problems.

Advantages:

- Highly optimized implementations (UMFPACK, PARDISO).
- Simple interface.
- Almost no a priori knowledge required.

Special cases: Circulant matrices $\mathcal{O}(n \log n)$, band matrices $\mathcal{O}(nk^2)$.

Properties of the model problem

Local basis: $\operatorname{supp}\varphi_i \cap \operatorname{supp}\varphi_j = \emptyset$ implies $a_{ij} = 0$.
$\rightarrow$ A is sparse.

Symmetry: $\langle \nabla v_h, \nabla u_h \rangle_{L^2(\Omega)} = \langle \nabla u_h, \nabla v_h \rangle_{L^2(\Omega)}$.
$\rightarrow$ A is symmetric.

Ellipticity: Friedrich's inequality implies $\|\nabla u_h\|^2_{L^2(\Omega)} \geq c\|u_h\|^2_{L^2(\Omega)}$.
$\rightarrow$ A is positive definite.

Unstructured grid: Non-zero entries in A are not arranged regularly.
$\rightarrow$ No (block) Toeplitz structure, no FFT solvers.

Inverse inequality: $\|\nabla u_h\|^2_{L^2(\Omega)} \sim h^{-2}\|u_h\|^2_{L^2(\Omega)}$ for certain u_h.
$\rightarrow$ Condition number of A grows like h^{-2}.

Overview

Richardson iteration

Goal: Solve $Ax^* = b$.

Approach: Try to improve an initial guess x.

Error: If we knew the error $e = x^* - x$, we could use $x' = x + e$.

Residual: We usually can compute the residual

$$r := Ae = A(x^* - x) = b - Ax.$$

If A is positive definite, $\langle e, r \rangle = \langle e, Ae \rangle > 0$ implies that r points (very) roughly in the same direction as e.

Richardson iteration: Choose damping parameter $\lambda > 0$ and use

$$x' := x + \lambda r = x + \lambda(b - Ax).$$

Preconditioned Richardson

Goal: Reduce the angle between e and Ae.

Idea: Introduce an invertible preconditioning matrix N that can be efficiently evaluated and apply Richardson to

$$NAx^* = Nb \qquad \Longleftrightarrow \qquad Ax^* = b.$$

Generalized iteration

$$x' := x + \lambda N(b - Ax).$$

Preconditioned Richardson

Goal: Reduce the angle between e and Ae.

Idea: Introduce an invertible preconditioning matrix N that can be efficiently evaluated and apply Richardson to

$$NAx^* = Nb \qquad \Longleftrightarrow \qquad Ax^* = b.$$

Generalized iteration with hidden damping parameter:

$$x' := x + \quad N(b - Ax).$$

Convergence

Question: Will the sequence $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ converge to x^*?

$$x^{(m+1)} := x^{(m)} + N(b - Ax^{(m)}) \qquad \text{for all } m \in \mathbb{N}_0.$$

Error propagation given by

$$\begin{aligned} e^{(m+1)} &= x^* - x^{(m+1)} = x^* - x^{(m)} - NA(x^* - x^{(m)}) \\ &= (I - NA)e^{(m)} = Me^{(m)} \qquad \text{for all } m \in \mathbb{N}_0 \end{aligned}$$

with the iteration matrix $M := I - NA$.

Goal: We have to ensure

$$e^{(m)} = M^m e^{(0)} \to 0.$$

Richardson: Convergence if A is positive definite and λ small enough.

Jacobi iteration

Idea: Use the diagonal D of $A \in \mathbb{R}^{n \times n}$ to construct preconditioner.

$$x' = x + \lambda D^{-1}(b - Ax).$$

Implementation:

$$x_i' = x_i + \frac{\lambda}{a_{ii}} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) \qquad \text{for all } i \in [1 : n]$$

Convergence

- guaranteed for diagonally dominant matrices,
- guaranteed for symmetric positive definite matrices if λ small.

Gauß-Seidel iteration

Idea: Use the lower triangular part L of $A \in \mathbb{R}^{n \times n}$.

$$x' = x + L^{-1}(b - Ax).$$

L^{-1} can be evaluated efficiently by forward substitution.

Implementation:

$$x'_i = x_i + \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x'_j - \sum_{j=i}^{n} a_{ij} x_j \right) \qquad \text{for all } i \in [1 : n]$$

Convergence

- guaranteed for diagonally dominant matrices,
- guaranteed for symmetric positive definite matrices

Gauß-Seidel iteration

Idea: Use the lower triangular part L of $A \in \mathbb{R}^{n \times n}$.

$$x' = x + L^{-1}(b - Ax).$$

L^{-1} can be evaluated efficiently by forward substitution.

Implementation:

$$x_i' = x_i + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j' - \sum_{j=i}^{n} a_{ij} x_j \right) \qquad \text{for all } i \in [1 : n]$$

Convergence

- guaranteed for diagonally dominant matrices,
- guaranteed for symmetric positive definite matrices,
- diagonal scaling parameter $\omega \in (0, 2)$ leads to SOR iteration.

ILU iteration

Idea: Compute incomplete LU factorization $A \approx LU$, where non-zero patterns of L and U are prescribed.

$$x' = x + U^{-1}L^{-1}(b - Ax).$$

L^{-1} and U^{-1} can be evaluated by forward and backward substitution.

Convergence

- guaranteed for M-matrices,
- strongly depends on prescribed sparsity pattern.

Summary

Linear iterative methods take the form

$$x^{(m+1)} = x^{(m)} + N(b - Ax^{(m)}.$$

Convergence determined by iteration matrix

$$e^{(m)} = M^m e^{(0)}, \qquad M = I - NA.$$

Classical methods based on $N \approx A^{-1}$

- Richardson iteration: $N = \lambda I$,
- Jacobi iteration: $N = \lambda D^{-1}$, D diagonal of A,
- Gauß-Seidel iteration: $N = L^{-1}$, L lower triangular part of A,
- ILU iteration: $N = U^{-1}L^{-1}$, incomplete factorization $LU \approx A$.

Overview

Minimization problem

Idea: Assume A symmetric positive definite. Use quadratic functional

$$J(x) = \frac{1}{2}\langle x, Ax\rangle - \langle x, b\rangle.$$

Minimization property: We have

$$\langle p, b - Ax\rangle = 0 \qquad \Longleftrightarrow \qquad J(x) \leq J(x + \lambda p) \text{ for all } \lambda \in \mathbb{R}.$$

In particular, x^* is the unique global minimum of J.

Descent algorithm: Choose direction p, let $x' = x + \lambda p$.

$$\langle p, b - Ax'\rangle = 0 \qquad \Longleftrightarrow \qquad \lambda = \frac{\langle p, b - Ax\rangle}{\langle p, Ap\rangle}.$$

Gradient method

Idea: Taylor expansion yields

$$J(x+\lambda p) = J(x) + \lambda\langle p, Ax-b\rangle + \frac{\lambda^2}{2}\langle p, Ap\rangle.$$

For $\lambda \ll 1$, we can ignore the last term and choose

$$p := -\nabla J(x) = b - Ax$$

to get direction of steepest descent.

Algorithm

$$p = b - Ax, \qquad \lambda = \frac{\langle p, p\rangle}{\langle p, Ap\rangle}, \qquad x' = x + \lambda p$$

Gradient method

Idea: Taylor expansion yields

$$J(x+\lambda p) = J(x) + \lambda\langle p, Ax-b\rangle + \frac{\lambda^2}{2}\langle p, Ap\rangle.$$

For $\lambda \ll 1$, we can ignore the last term and choose

$$p := -\nabla J(x) = b - Ax$$

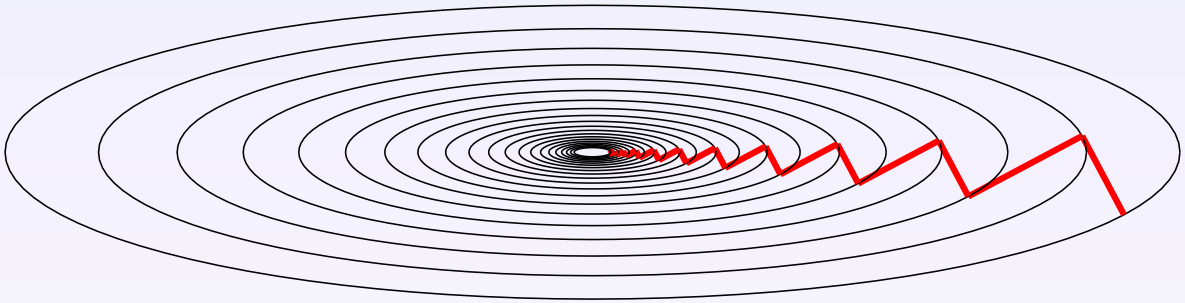
to get direction of steepest descent.

Algorithm keeps track of residual to save time:

$$p = b - Ax, \qquad \lambda = \frac{\langle p, p\rangle}{\langle p, Ap\rangle}, \qquad x' = x + \lambda p, \qquad p' = p - \lambda Ap.$$

Conjugate gradient method

Observation: Gradient method may choose the same direction repeatedly.



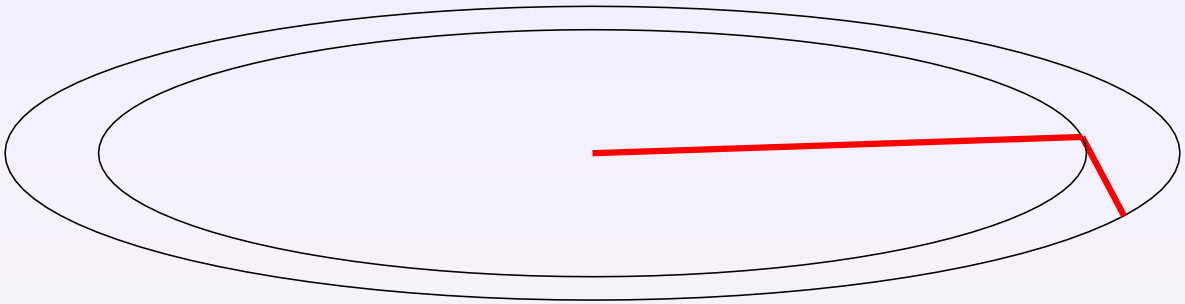
Idea: Avoid losing optimality with respect to old directions $\widehat{p}$.

$$\langle \widehat{p}, b - Ax \rangle = 0, \quad \langle \widehat{p}, b - A(x + \lambda p) \rangle = 0 \qquad \Longrightarrow \qquad \langle \widehat{p}, Ap \rangle = 0$$

Algorithm: Ensure p is conjugate with respect to all old directions.

Conjugate gradient method

Observation: Gradient method may choose the same direction repeatedly.



Idea: Avoid losing optimality with respect to old directions $\widehat{p}$.

$$\langle \widehat{p}, b - Ax \rangle = 0, \quad \langle \widehat{p}, b - A(x + \lambda p) \rangle = 0 \qquad \Longrightarrow \qquad \langle \widehat{p}, Ap \rangle = 0$$

Algorithm: Ensure p is conjugate with respect to all old directions.

Krylov spaces

Observation: "New" directions result from matrix-vector multiplications.

$$x^{(m)} - x^{(0)} \in \mathcal{K}_m := \operatorname{span}\{r, Ar, A^2 r, \ldots, A^{m-1} r\}, \quad r = b - Ax^{(0)}.$$

Optimality is guaranteed for all previous directions.

$$J(x^{(m)}) \leq J(x^{(m)} + p) \qquad \text{for all } p \in \mathcal{K}_m.$$

Energy norm satisfies $\|x^* - x^{(m)}\|_A^2 = 2\,J(x^{(m)}) + c$. This implies

$$\|x^* - x^{(m)}\|_A \leq \|(x^* - x^{(0)}) - y\|_A \qquad \text{for all } y \in \mathcal{K}_m.$$

Uzawa's method

Goal: Solve saddle point problem

$$\begin{pmatrix} A & B \\ B^* & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Problem: Indefinite system, although A is positive definite.

Idea: Block Gauss elimination.

$$\begin{pmatrix} A & B \\ & -B^* A^{-1} B \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 - B^* A^{-1} b_1 \end{pmatrix}$$

Uzawa's method

Goal: Solve saddle point problem

$$\begin{pmatrix} A & B \\ B^* & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Problem: Indefinite system, although A is positive definite.

Idea: Block Gauss elimination.

$$\begin{pmatrix} A & B \\ & B^* A^{-1} B \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ B^* A^{-1} b_1 - b_2 \end{pmatrix}$$

Matrix $S = B^* A^{-1} B$ is positive (semi-)definite.

Algorithm: Solve second row by conjugate gradient method to find x_2. $x_1 = A^{-1}(b_1 - Bx_2)$ can be constructed from intermediate results.

GMRES

Goal: Krylov solver for general matrices.

Approach: Minimize residual norm $\|b - Ax\|_2$ instead of J.

$$\|b - Ax^{(m)}\|_2 \leq \|b - A(x^{(0)} + p)\|_2 \qquad \text{for all } p \in \mathcal{K}_m.$$

Algorithm: Construct orthonormal basis of Krylov space $\mathcal{K}_m$, solve m-dimensional least squares problem to find $x^{(m)}$.
$\rightarrow$ Very time-consuming for $m \gg 1$.
$\rightarrow$ Restart method after fixed number of steps.

Related algorithms: BiCG, QMR, MINRES.

Summary

Krylov methods construct bases for Krylov spaces

$$\mathcal{K}_m = \operatorname{span}\{r, Ar, A^2 r, \ldots, A^{m-1} r\}$$

and look for updates $q^{(m)} \in \mathcal{K}_m$ such that $x^{(m)} = x^{(0)} + q^{(m)} \approx x^*$.

Conjugate gradient method constructs conjugate vectors to minimize

$$\|x^{(m)} - x^*\|_A.$$

GMRES method constructs orthogonal vectors to minimize

$$\|b - Ax^{(m)}\|_2.$$

Overview

Iteration steps

Model problem: $-u'' = f$ in $[0,1]$, n nodal basis functions.

Experiment: Count iteration steps required to reduce residual by 10^{-4}.

$n =$	7	15	31	63	127	255
Rich	105	373	1284	4278	13667	40863
Jac	105	373	1284	4278	13667	40863
GS	51	179	608	1997	6260	18132
CG	4	8	16	32	64	128

Observation: Number of steps grows with n

Iteration steps

Model problem: $-u'' = f$ in $[0,1]$, n nodal basis functions.

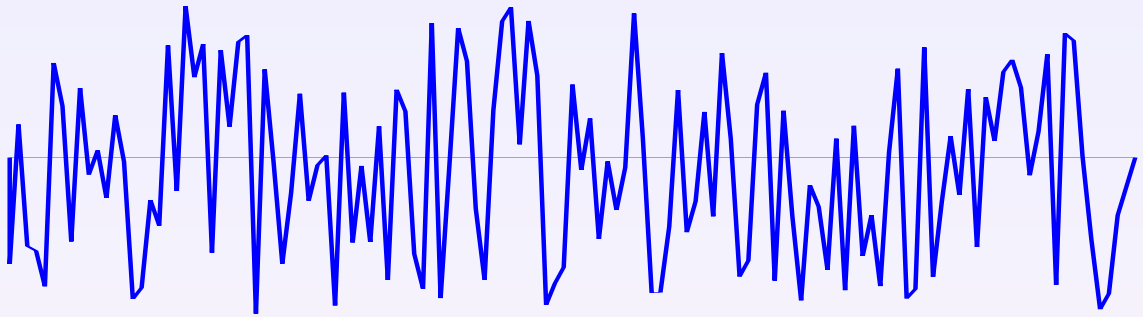
Experiment: Count iteration steps required to reduce residual by 10^{-4}.

$n =$	7	15	31	63	127	255
Rich	105	373	1284	4278	13667	40863
Jac	105	373	1284	4278	13667	40863
GS	51	179	608	1997	6260	18132
CG	4	8	16	32	64	128
MG	6	6	6	6	6	6

Observation: Number of steps grows with n for classical iterations, but remains constant for multigrid iteration.

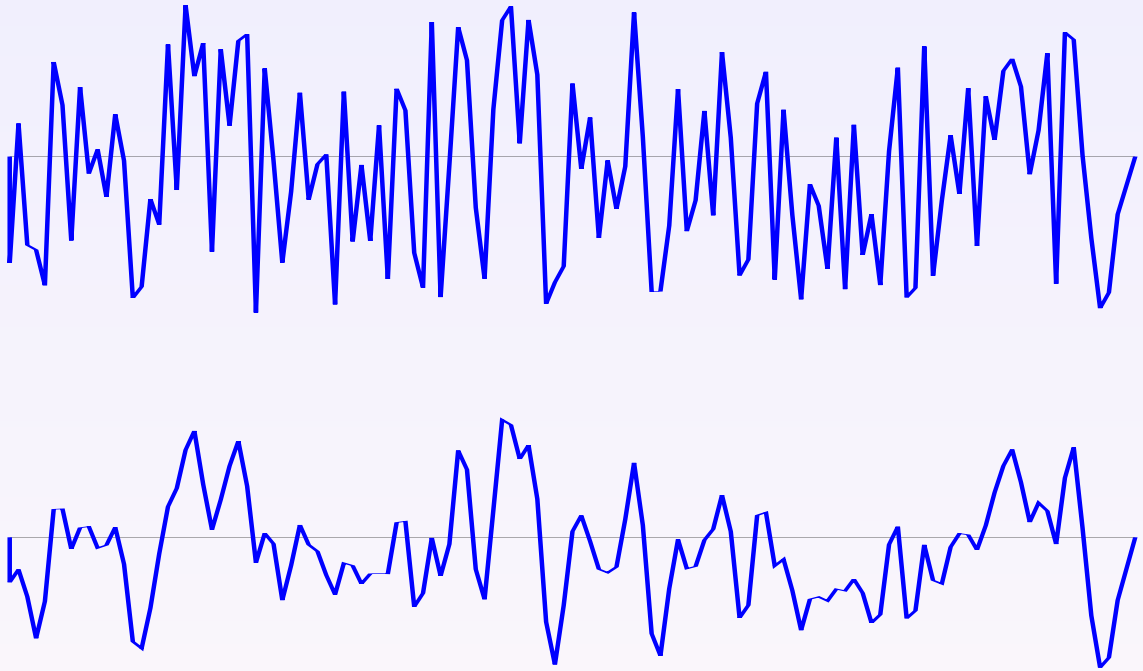
Smoothing iteration

Approach: Apply classical iteration, e.g., Jacobi.
How does the error change?



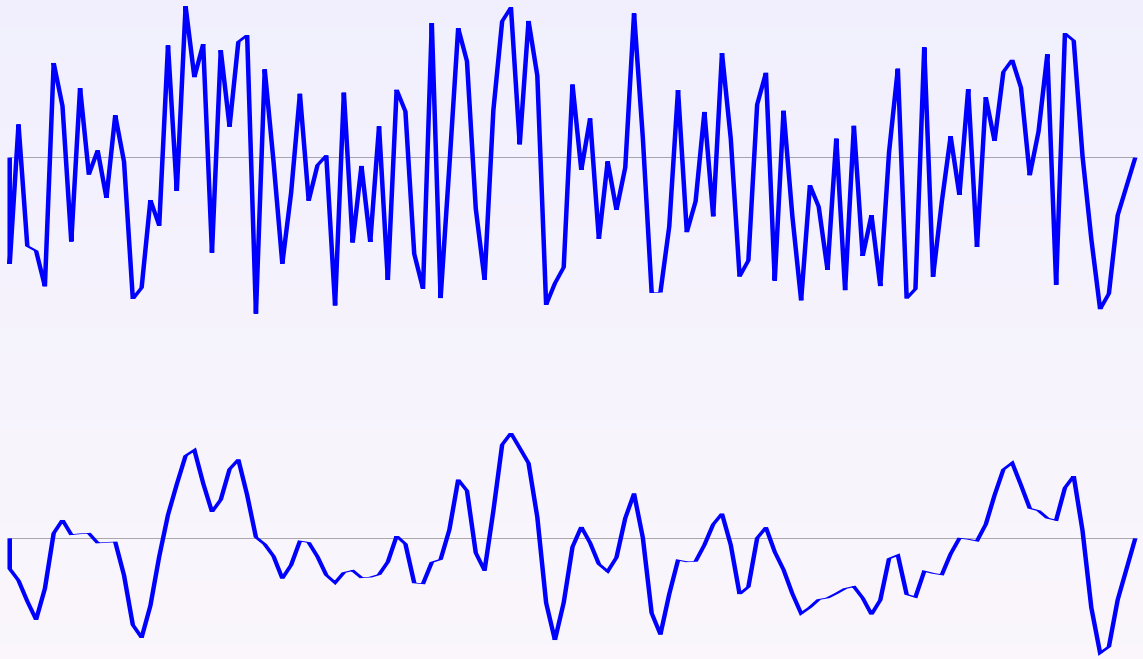
Smoothing iteration

Approach: Apply classical iteration, e.g., Jacobi.
How does the error change?



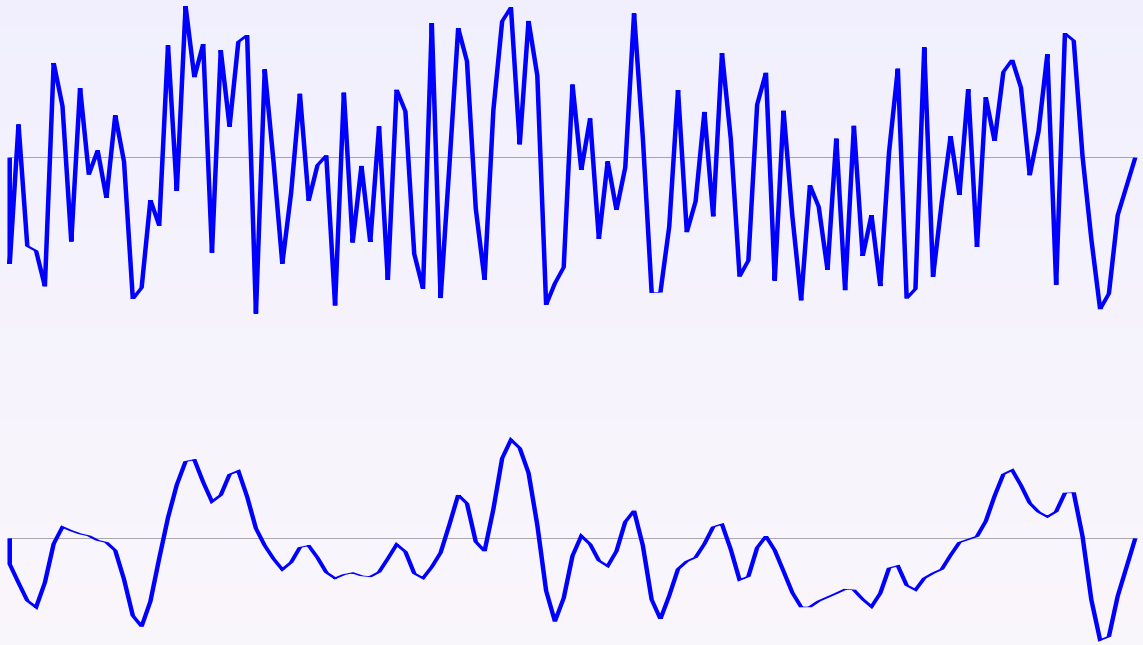
Smoothing iteration

Approach: Apply classical iteration, e.g., Jacobi.
How does the error change?



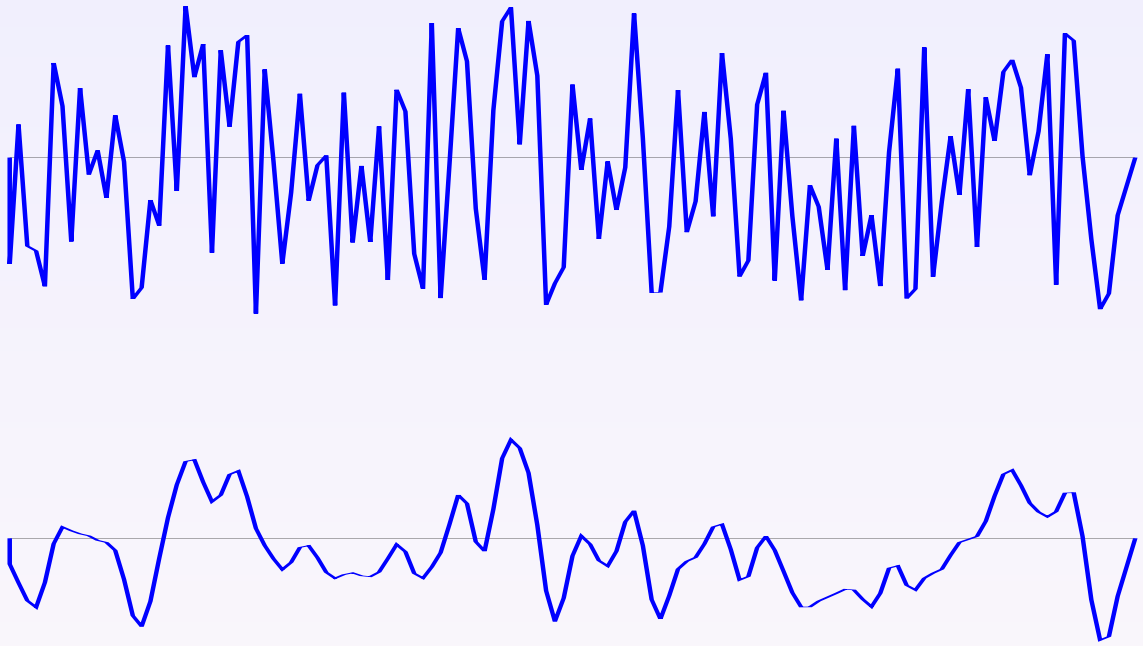
Smoothing iteration

Approach: Apply classical iteration, e.g., Jacobi.
How does the error change?



Smoothing iteration

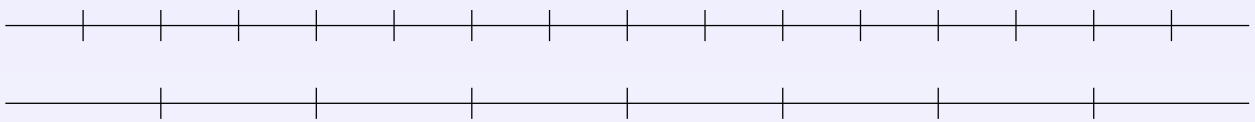
Approach: Apply classical iteration, e.g., Jacobi.
How does the error change?



Observation: Many classical iterations smoothe the error.

Coarse grid correction

Idea: If the error is smooth, approximate it on coarser grid.

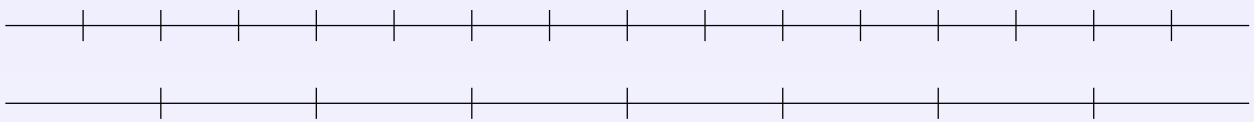


Grid transfer described by prolongation matrix p.

$$p\widetilde{x} \approx x^* - x$$

Coarse grid correction

Idea: If the error is smooth, approximate it on coarser grid.

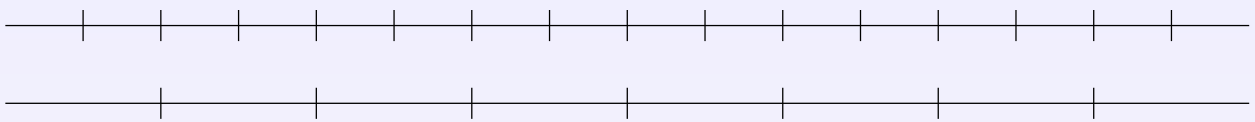


Grid transfer described by prolongation matrix p.

$$p\widetilde{x} \approx x^* - x$$
$$Ap\widetilde{x} \approx A(x^* - x) = b - Ax,$$

Coarse grid correction

Idea: If the error is smooth, approximate it on coarser grid.

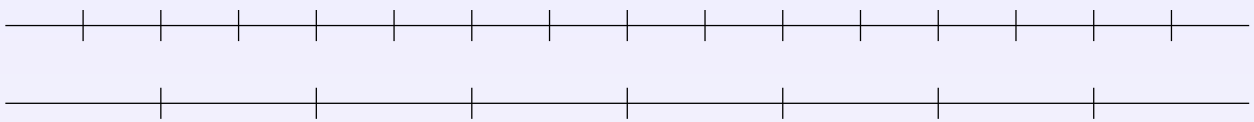


Grid transfer described by prolongation matrix p.

$$\begin{aligned} p\widetilde{x} &\approx x^* - x \\ Ap\widetilde{x} &\approx A(x^* - x) = b - Ax, \\ p^* Ap\widetilde{x} &= p^*(b - Ax), \end{aligned}$$

Coarse grid correction

Idea: If the error is smooth, approximate it on coarser grid.

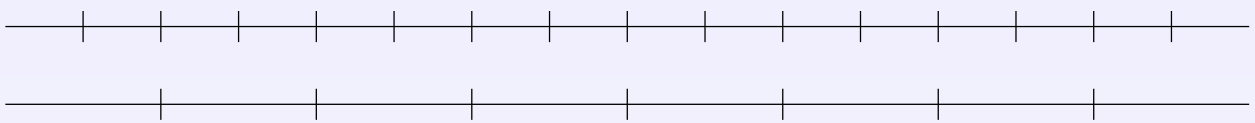


Grid transfer described by prolongation matrix p.

$$\begin{aligned} p\widetilde{x} &\approx x^* - x \\ Ap\widetilde{x} &\approx A(x^* - x) = b - Ax, \\ p^* Ap\widetilde{x} &= p^*(b - Ax), \\ \widetilde{A}\widetilde{x} &= p^*(b - Ax). \end{aligned}$$

Coarse grid correction

Idea: If the error is smooth, approximate it on coarser grid.

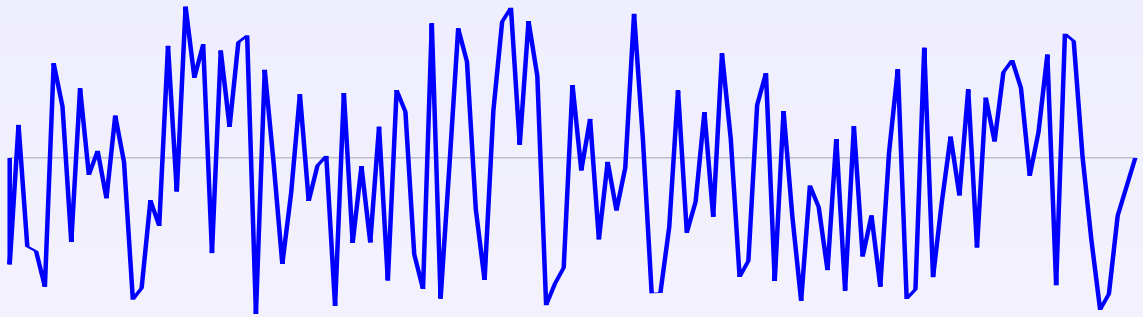


Grid transfer described by prolongation matrix p.

$$\begin{aligned} p\widetilde{x} &\approx x^* - x \\ Ap\widetilde{x} &\approx A(x^* - x) = b - Ax, \\ p^* Ap\widetilde{x} &= p^*(b - Ax), \\ \widetilde{A}\widetilde{x} &= p^*(b - Ax). \end{aligned}$$

Finite elements: $\widetilde{A} = p^* Ap$ results from discretization on coarser grid.

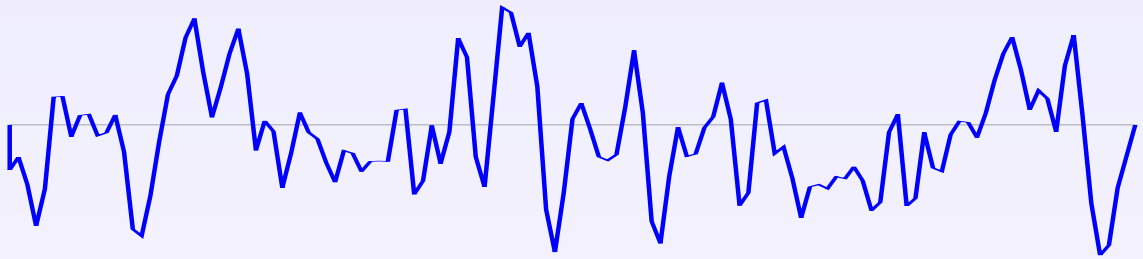
Multigrid step



Idea:

- Classical iterations like Jacobi smoothe the error.

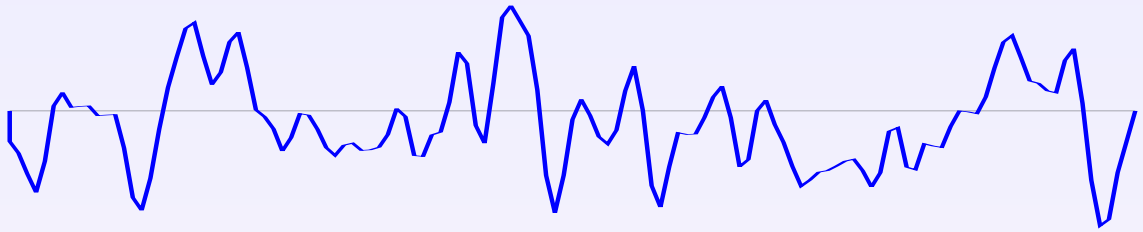
Multigrid step



Idea:

- Classical iterations like Jacobi smoothe the error.

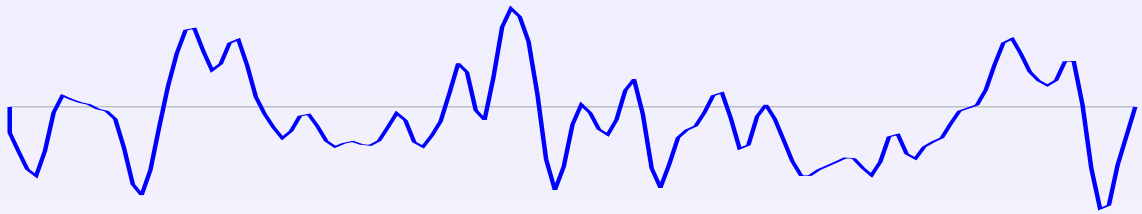
Multigrid step



Idea:

- Classical iterations like Jacobi smoothe the error.

Multigrid step



Idea:

- Classical iterations like Jacobi smoothe the error.

Multigrid step

Idea:

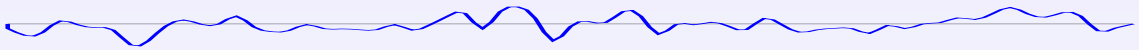
- Classical iterations like Jacobi smoothe the error.
- Smooth error can be approximated on coarser grid.

Multigrid step

Idea:

- Classical iterations like Jacobi smoothe the error.
- Smooth error can be approximated on coarser grid.
- Resulting oscillations can again be smoothed.

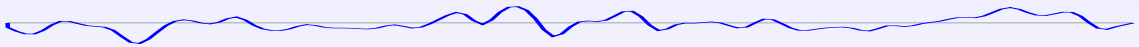
Multigrid step



Idea:

- Classical iterations like Jacobi smoothe the error.
- Smooth error can be approximated on coarser grid.
- Resulting oscillations can again be smoothed.

Multigrid step

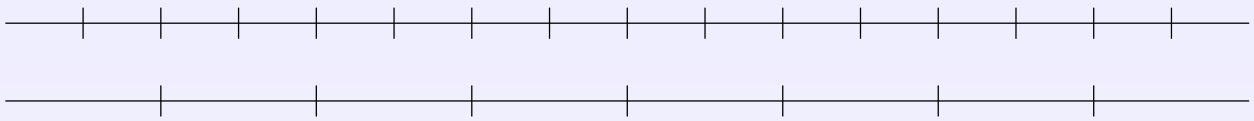


Idea:

- Classical iterations like Jacobi smoothe the error.
- Smooth error can be approximated on coarser grid.
- Resulting oscillations can again be smoothed.

Result: Fast convergence, optimal complexity $\mathcal{O}(n)$.

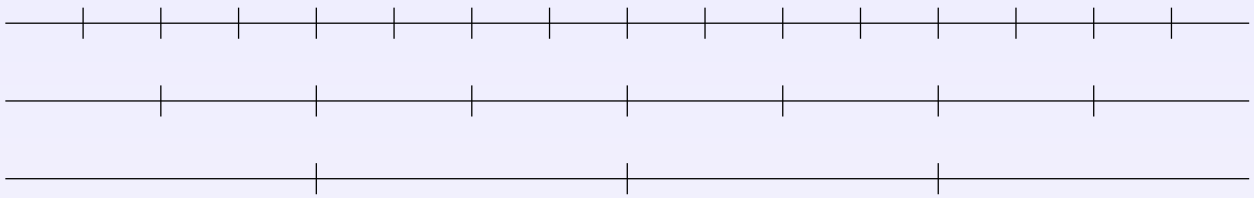
Grid hierarchy



Goal: Approximate smooth error.

Idea: Since its smooth, use a coarser grid.

Grid hierarchy

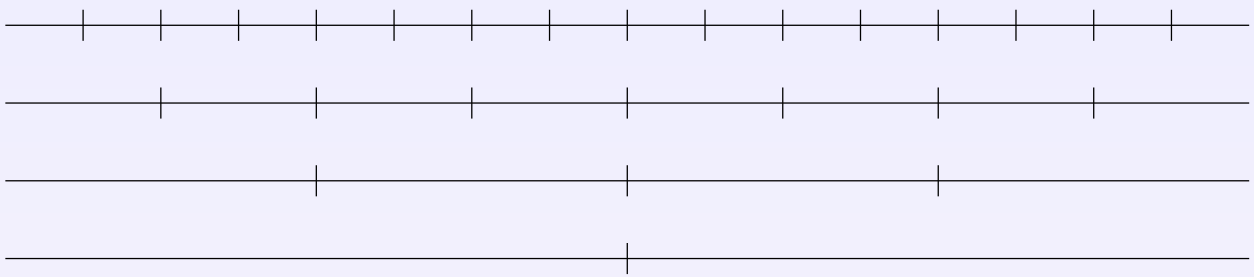


Goal: Approximate smooth error.

Idea: Since its smooth, use a coarser grid.

Since the coarser grid may still be too fine, use even coarser grids

Grid hierarchy



Goal: Approximate smooth error.

Idea: Since its smooth, use a coarser grid.

Since the coarser grid may still be too fine, use even coarser grids

Result: Hierarchy of grids, each with its own discretization.

- Matrices $A_\ell \in \mathbb{R}^{n_\ell \times n_\ell}$ for each grid $\ell \in \{0, \ldots, L\}$.
- Prolongation matrices $p_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ mapping coarse to fine.

Multigrid algorithm

Exact correction on coarse grid given by

$$x'_\ell = x_\ell + p_\ell A_{\ell-1}^{-1} p_\ell^* (b_\ell - A_\ell x_\ell).$$

Multigrid algorithm

Exact correction on coarse grid given by

$$x'_\ell = x_\ell + p_\ell A_{\ell-1}^{-1} p_\ell^* (b_\ell - A_\ell x_\ell).$$

Problem: Evaluating $A_{\ell-1}^{-1}$ may still be too expensive.
$\rightarrow$ Use recursion to even coarser grids.

procedure *multigrid(ℓ);*
  presmoothing(b_ℓ, x_ℓ);
  if $\ell > 0$ ***then begin***
    $b_{\ell-1} \leftarrow p_\ell^*(b_\ell - A_\ell x_\ell)$; $x_{\ell-1} \leftarrow 0$;
    multigrid($\ell - 1$);
    $x_\ell \leftarrow x_\ell + p_\ell x_{\ell-1}$
  end;
  postsmoothing(b_ℓ, x_ℓ)

Full multigrid

Given: Multigrid iteration, $\|M_\ell\| \leq \varrho < 1$.

Challenge: Solve discretized system with accuracy $\epsilon_\ell \sim q^\ell$ ($q < 1$).
Naive approach needs $\log(\epsilon_\ell)/\log(\varrho) \sim \ell \log(q)/\log(\varrho)$ steps.

Full multigrid

Given: Multigrid iteration, $\|M_\ell\| \leq \varrho < 1$.

Challenge: Solve discretized system with accuracy $\epsilon_\ell \sim q^\ell$ ($q < 1$).
Naive approach needs $\log(\epsilon_\ell)/\log(\varrho) \sim \ell \log(q)/\log(\varrho)$ steps.

Idea: All levels provide approximations of the same solution x.
If $\|x_{\ell-1} - x\| \leq \epsilon_{\ell-1}$, we can use $p_\ell x_{\ell-1}$ as initial guess $x_\ell^{(0)}$ for level ℓ.

$$\|x_\ell^{(0)} - x\| = \quad \|x_{\ell-1} - x\| \lesssim \quad q^{\ell-1},$$
$$\|x_\ell^{(m)} - x\| \leq \varrho^m \|x_{\ell-1} - x\| \lesssim \varrho^m q^{\ell-1} \overset{!}{\leq} q^\ell.$$

Result: We only have to ensure $\varrho^m \leq q$, i.e., $m \sim \log(q)/\log(\varrho)$.
$\rightarrow$ Constant number of steps.
$\rightarrow$ Solve n-dimensional system in $\mathcal{O}(n)$ operations.

Summary

Smoothing: Classical iterations quickly reduce oscillations of the error.

Coarse grid correction: Smooth error approximated on coarser grid.

Convergence rate bounded independently of the mesh parameter.

Full multigrid solves n-dimensional system in linear complexity.

Challenges:

- Suitable grid hierarchy has to be provided.
- Algorithm has to be adapted to handle special properties, e.g., Maxwell's equation, singular perturbations.
- Proof of robust convergence requires sophisticated techniques.

Literature

Wolfgang Hackbusch:
Iterative Solution of Large Sparse Systems of Equations,
Springer (1994)

Yousef Saad:
Iterative Methods for Sparse Linear Systems,
SIAM (2003)

Maxim E. Olshanskii, Eugene E. Tyrtyshnikov:
Iterative Methods for Linear Systems: Theory and Applications,
SIAM (2014)