A Mini-course on Multigrid Method: Introduction

Hardik Kothari

Euler Institute,
Università della Svizzera italiana,
Switzerland

May 16, 2023
Research school on Iterative Methods for Partial Differential Equations
Paris, France

Outline

Motivation

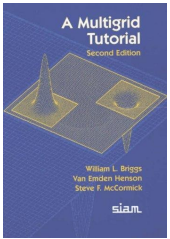
- Effective in solving large-scale problems from applied mathematics, engineering, physics, computational sciences
- More of a framework rather than a solution method, hence it requires a detailed knowledge of the algorithm for effective usage
- Properties of multigrid method
  - optimal time complexity
  - optimal memory complexity
  - level independent convergence
  - stable asymptotic convergence rates
- Included in libraries for solving numerical solutions of PDEs, e.g., PETSc, Trilinos, etc.

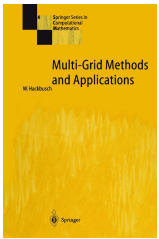
Historical developments

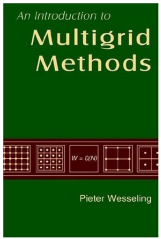
- Highly effective iterative numerical framework for solving linear system of equations arising from discretization of the partial differential equations (PDEs).
- Historical developments
  - Federenko (1964) - multigrid algorithm for standard 5-point stencil for solving Laplace problem
  - Bachvalov (1966) - generalization of the MG for central-difference scheme for linear elliptic PDE with variable smooth coefficients
  - Theoretical work estimates were pessimistic and method was not put in practice
  - First practical results were reported by Brandt, outlined the main principles and demonstrated practical utility of the method (1973, 1977)
  - The method was also discovered independently by Hackbusch (1978, 1980, 1981), who laid mathematical foundations
  - numerous contributions since then...

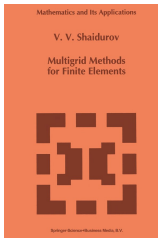
Main monographs

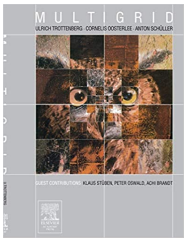
- A Multigrid Tutorial, - by *W. L. Briggs, Van E. Henson, S. F. McCormick* (1987, 2000)
- Multi-Grid Methods and Applications, - by *W. Hackbusch* (1985)
- An Introduction to Multigrid Methods, - by *P. Wesseling* (1992)
- Multigrid Methods for Finite Elements, - by *V. V. Shaidurov* (1995)
- Multigrid, - by *U. Trottenberg, C. W. Ooserlee, A. Schüler* (2001)





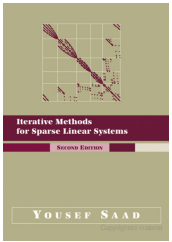


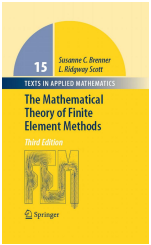


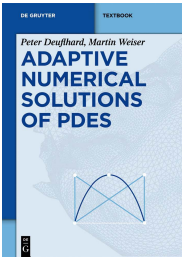


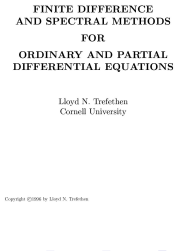
Chapters in other monographs

- Iterative methods for sparse linear systems - Chapter 13, - by *Y. Saad* (2003)
- Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics - Chapter 5, - by *D. Braess* (2007)
- The mathematical theory of Finite Element methods - Chapter 6, - by *S. C. Brenner, L. R. Scott* (2008)
- Adaptive Numerical Solutions of PDEs - Chapter 5, - by *P. Deuflhard, M. Weiser* (2012)
- Finite Difference and Spectral Methods for Ordinary and Partial differential equations - Chapter 9, - by *L. N. Trefethen* (1996)









Multigrid methods

- An effective iterative numerical method for solving linear system of equations arising from discretization of the partial differential equations.
- An optimal solution method
  - convergence rate of the method is independent of problem size
  - iteration till convergence do not grow with increasing problem size
  - only arithmetic operations grow with problem size
- It employs hierarchy of meshes with different resolutions
- Essential components:
  - Smoothers of each level
  - Transfer operators
  - A direct solver on the coarsest level

Table of Contents

A model problem

- Strong form:

$$\begin{aligned} -\nabla^2 u &= f, \qquad \text{in } \Omega, \\ u &= 0, \qquad \text{on } \partial\Omega, \end{aligned}$$

  where $f \in L^2(\Omega)$ regular, $\partial\Omega$ is Lipschitz

- Weak form: Find $u \in H_0^1(\Omega)$ such that

$$\begin{aligned} \int_\Omega \nabla u \cdot \nabla v \, dx &= \int_\Omega f v \, dx, \qquad \forall\ v \in H_0^1(\Omega), \\ a(u,v) &= F(v), \qquad \forall\ v \in H_0^1(\Omega). \end{aligned}$$

- Triangulation of Ω with elements
- Choosing basis $\{\phi_i\}_{i=1}^N$, where $N =$ number of nodes
- Discretized weak form can be given as a linear system

$$\boldsymbol{Ax} = \boldsymbol{b}, \qquad \text{where } \begin{cases} A_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, dx \\ b_i = \int_\Omega f \cdot \phi_i \, dx \end{cases}$$

Matrix stencil and Eigenvalues/Eigenvectors

- 1D problem

$$\boldsymbol{A} = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{bmatrix}$$

- $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ symmetric positive definite
- Diagonally dominant, i.e.,

$$|a_{ii}| \geqslant \sum_{j \neq i} |a_{ij}| \qquad \text{for } i = 1, 2, \ldots, n$$

- Eigenvalues in range $[0, 4]$

$$\lambda_k = 4 \sin^2\left(\frac{k\pi}{2n}\right)$$

- Eigenvectors are Fourier modes

$$(\boldsymbol{v}_k)_j = \sin\left(\frac{jk\pi}{n}\right)$$

Table of Contents

Iterative methods based on Matrix splitting

- Solve linear system of equation:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

- Split the matrix $\boldsymbol{A}$ to construct iterative method

$$\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$$

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{N}\boldsymbol{x} + \boldsymbol{b}$$

- Iterative method can be written as

$$\begin{aligned}
\boldsymbol{M}\boldsymbol{x}^{(k+1)} &= \boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{b} \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{M}^{-1}(\boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{b}) \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{M}^{-1}((\boldsymbol{M} - \boldsymbol{A})\boldsymbol{x}^{(k)} + \boldsymbol{b}) \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)})
\end{aligned}$$

- Alternatively, we can also write

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b} \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{G}\boldsymbol{x}^{(k)} + \boldsymbol{d}
\end{aligned}$$

Fixed point iteration

- Given an initial guess $\boldsymbol{x}^{(0)}$, an iterative method can be given as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{G}\boldsymbol{x}^{(k)} + \boldsymbol{d}, \qquad k = 0, 1, 2, \ldots$$

  where $\boldsymbol{G} = \boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}$ and $\boldsymbol{d} = \boldsymbol{M}^{-1}\boldsymbol{b}$

- The solution $\boldsymbol{x}^*$ of the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is a fixed point of above iteration, i.e.,

$$\boldsymbol{x}^* = \boldsymbol{G}\boldsymbol{x}^* + \boldsymbol{d}$$

- Error for a given iterate $\boldsymbol{x}^{(k)}$

$$\boldsymbol{e}^{(k)} = \boldsymbol{x}^* - \boldsymbol{x}^{(k)}$$

- Residual at a given iterate $\boldsymbol{x}^{(k)}$

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}$$

- Error equation

$$\begin{aligned}
\boldsymbol{r}^{(k)} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)} \\
\boldsymbol{r}^{(k)} &= \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{A}\boldsymbol{x}^{(k)} \\
\boldsymbol{r}^{(k)} &= \boldsymbol{A}\boldsymbol{e}^{(k)}
\end{aligned}$$

Error propagation

- Utilizing the fixed point and iterative method

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* &= \boldsymbol{G}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) \\
\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* &= \boldsymbol{G}^2(\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^*) \\
&\vdots \\
\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^* &= \boldsymbol{G}^{k+1}(\boldsymbol{x}^{(0)} - \boldsymbol{x}^*)
\end{aligned}$$

- Using the error

$$\boldsymbol{e}^{(k+1)} = \boldsymbol{G}^{k+1}\boldsymbol{e}^{(0)}$$

Theroem:
A fixed-point iterative method is called convergent, if for an arbitrary $\boldsymbol{x}^{(0)}$, we have $\lim_{k\to\infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^*$.
We can equivalently write,

$$\lim_{k\to\infty} \boldsymbol{G}^k = 0.$$

Iteration error can be bounded by

$$\|\boldsymbol{e}^{(k)}\| \leqslant \|\boldsymbol{G}\|^k \|\boldsymbol{e}^{(0)}\|$$

Fundamental theorem of iterative method

For an iteration matrix

$$G = I - M^{-1}A$$

- Iterative method is convergent if $\lim_{k\to\infty} G^k = 0$ if and only if $\rho(G) < 1$ or $\|G\| < 1$
- $\rho(G)$ denotes spectral radius, i.e., $\rho(G) = \max\{|\lambda_1(G)|, |\lambda_2(G)|, \ldots, |\lambda_n(G)|\}$

Speed of convergence:

- Using the bounds on iteration error

$$\frac{\|e^{(k)}\|}{\|e^{(0)}\|} \leqslant \|G^k\| \leqslant \|G\|^k \leqslant 10^{-d}$$

- Using spectral radius

$$\begin{aligned} \rho(G^k) &\leqslant 10^{-d} \\ \log(\rho(G^k)) &\leqslant \log(10^{-d}) \\ k\log(\rho(G)) &\leqslant -d \\ k &\geqslant \frac{d}{-\log(\rho(G))}, \qquad \text{where convergence rate} = -\log(\rho(G)) \end{aligned}$$

Classical iterative methods

- Richardson method:

$$\boldsymbol{M} = \omega \boldsymbol{I}, \quad \boldsymbol{N} = \omega \boldsymbol{I} - \boldsymbol{A} \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \omega(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}), \qquad k = 0, 1, 2, \ldots$$

- Jacobi method:

$$\boldsymbol{M} = \boldsymbol{D}, \quad \boldsymbol{N} = (\boldsymbol{D} - \boldsymbol{A}) \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{D}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}), \qquad k = 0, 1, 2, \ldots$$

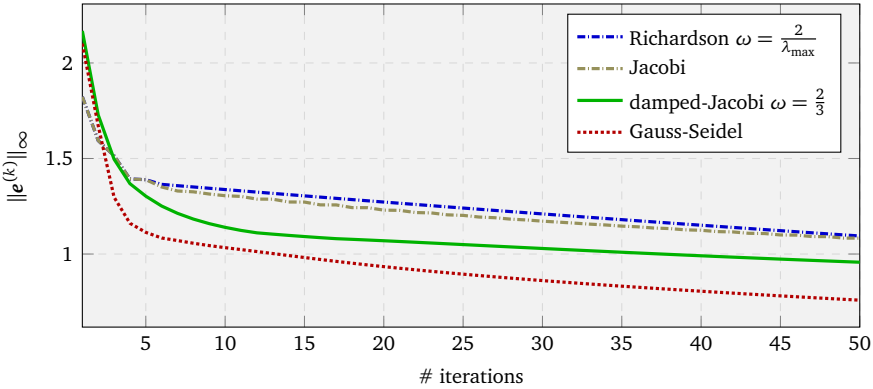
- Damped-Jacobi method:

$$\boldsymbol{M} = \omega \boldsymbol{D}, \quad \boldsymbol{N} = \omega \boldsymbol{D} - \boldsymbol{A} \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \omega \boldsymbol{D}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}), \qquad k = 0, 1, 2, \ldots$$

- Gauss-Seidel method:

$$\boldsymbol{M} = \boldsymbol{D} + \boldsymbol{L}, \quad \boldsymbol{N} = \boldsymbol{D} + \boldsymbol{L} - \boldsymbol{A} \qquad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + (\boldsymbol{D} + \boldsymbol{L})^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}), \qquad k = 0, 1, 2, \ldots$$

Error v/s iterations



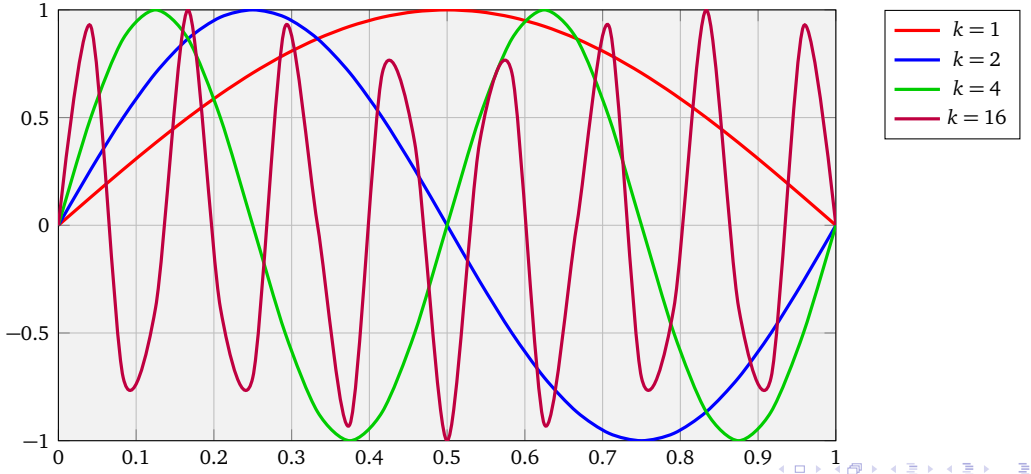
Observations:

- Iterative methods are very slow to converge
- Error reduced rapidly in few first iterations, but then it stagnates

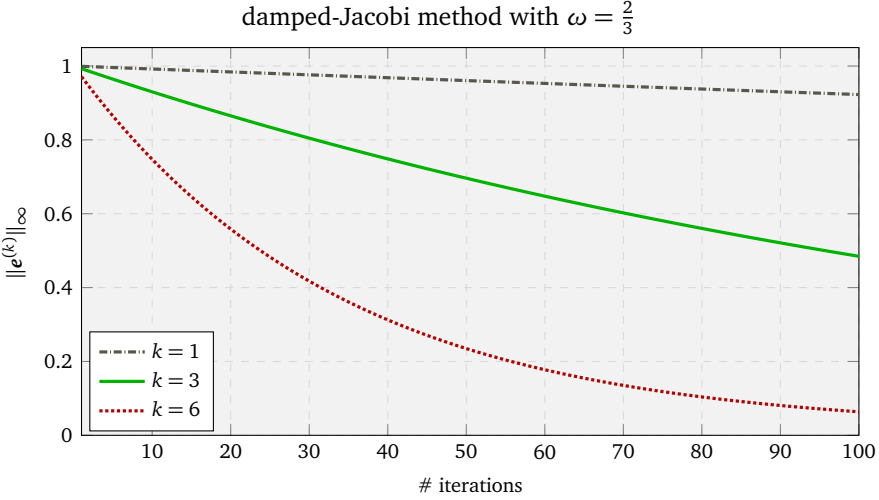
Table of Contents

Eigenvectors of $\boldsymbol{A}$

$$(\boldsymbol{v}_k)_j = \sin\left(\frac{jk\pi}{n}\right)$$



Convergence of different error components



Error reduces rapidly for high frequency component

Convergence analysis for damped Jacobi method

- Iteration matrix

$$\boldsymbol{G}_{\omega J} = \boldsymbol{I} - \omega \boldsymbol{D}^{-1}\boldsymbol{A}$$

- Eigenvalues of the iteration matrix

$$\lambda(\boldsymbol{G}_{\omega J}) = 1 - \frac{\omega}{2}\lambda(\boldsymbol{A})$$

- Recall: $\lambda_i(\boldsymbol{A}) = 4\sin^2\left(\frac{i\pi}{2n}\right)$

$$\lambda_i(\boldsymbol{G}_{\omega J}) = 1 - \frac{\omega}{2}4\sin^2\left(\frac{i\pi}{2n}\right)$$

- Convergence rate:

$$\rho(\boldsymbol{G}_{\omega J}) = \lambda_1 = 1 - 2\omega\sin^2\left(\frac{h\pi}{2}\right) = 1 - \mathcal{O}(h^2) < 1$$

- Eigenvectors of $\boldsymbol{G}$ and $\boldsymbol{A}$ are same for Jacobi/damped-Jacobi method, which is not true for Gauss-Seidel method, as Gauss-Seidel mixes the eigenvectors of $\boldsymbol{A}$

Expanding on eigenvalues of iteration matrix

- Expanding initial error using eigenvectors

$$\boldsymbol{e}^{(0)} = \sum_{i=1}^{n} c_i \boldsymbol{v}_i$$

- After ν iterations

$$\boldsymbol{e}^{(\nu)} = \boldsymbol{G}\boldsymbol{e}^{(\nu-1)}$$

- Error equation for damped Jacobi iteration

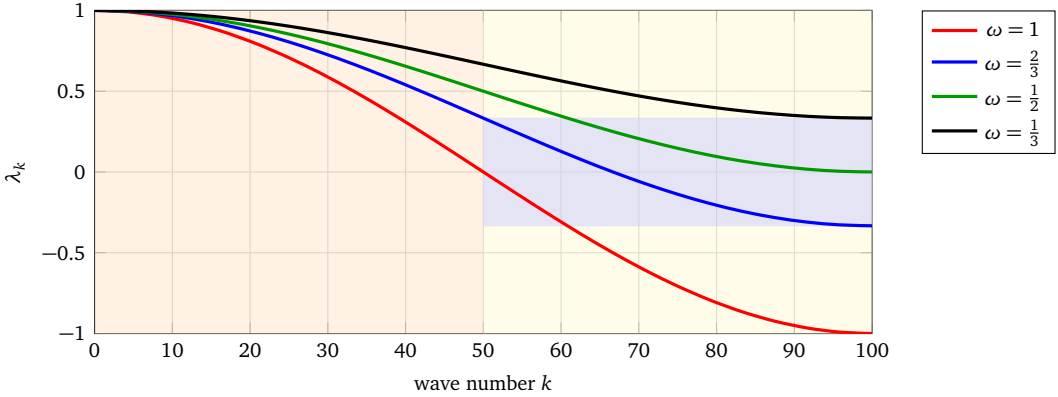
$$\boldsymbol{e}^{(\nu)} = (\boldsymbol{I} - \omega \boldsymbol{D}^{-1}\boldsymbol{A})\boldsymbol{e}^{(\nu-1)} = \boldsymbol{G}_{\omega J}^{\nu}\boldsymbol{e}^{(0)}$$

- For ν iterations, we get

$$\begin{aligned} \boldsymbol{G}_{\omega J}^{\nu} e^{(0)} &= \sum_{i=1}^{n} c_i \boldsymbol{G}_{\omega J}^{\nu} \boldsymbol{v}_i \\ &= \sum_{i=1}^{n} c_i \lambda_i^{\nu} \boldsymbol{v}_i \end{aligned}$$

- The i^{th} mode of the error is reduced by magnitude of λ_i at each iteration

Relaxation smooths different eigenmodes



- Eigenvalues of iteration matrix: $\lambda_i(\boldsymbol{G}_{\omega J}) = 1 - 2\omega \sin^2\left(\frac{i\pi}{2n}\right)$
- Which value of ω provides best smoothing of high frequency error?

Smoothing factor

The **smoothing factor** of a relaxation method is the largest absolute value of the eigenvalues in the upper half of the spectrum of the iteration matrix:

$$\text{smoothing factor} = \max\{|\lambda_{n/2}(\boldsymbol{G})|, |\lambda_{n/2+1}(\boldsymbol{G})|, |\lambda_{n/2+2}(\boldsymbol{G})|, \ldots, |\lambda_n(\boldsymbol{G})|\}$$

Smoothing factors for damped-Jacobi method:

- for $\omega = 2/3$

$$|\lambda_{n/2}(\boldsymbol{G}_{\omega J})| = |\lambda_n(\boldsymbol{G}_{\omega J})| = \frac{1}{3}$$

- for $\omega = 1/2$

$$|\lambda_{n/2}(\boldsymbol{G}_{\omega J})| = \frac{1}{2}, \quad |\lambda_n(\boldsymbol{G}_{\omega J})| = 0$$

- for $\omega = 1/3$

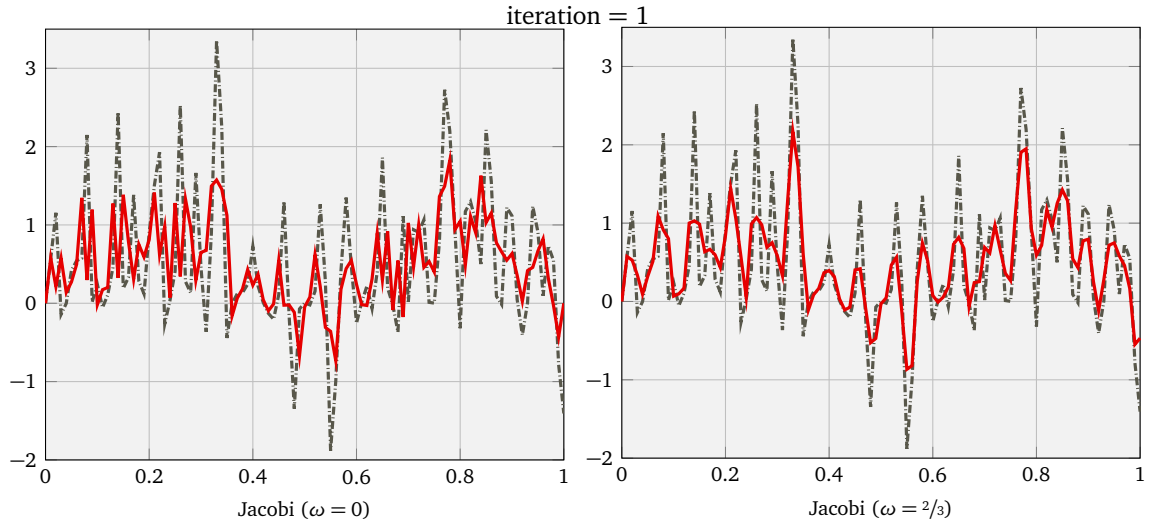
$$|\lambda_{n/2}(\boldsymbol{G}_{\omega J})| = \frac{2}{3}, \quad |\lambda_n(\boldsymbol{G}_{\omega J})| = \frac{1}{3}$$

- for $\omega = 1$

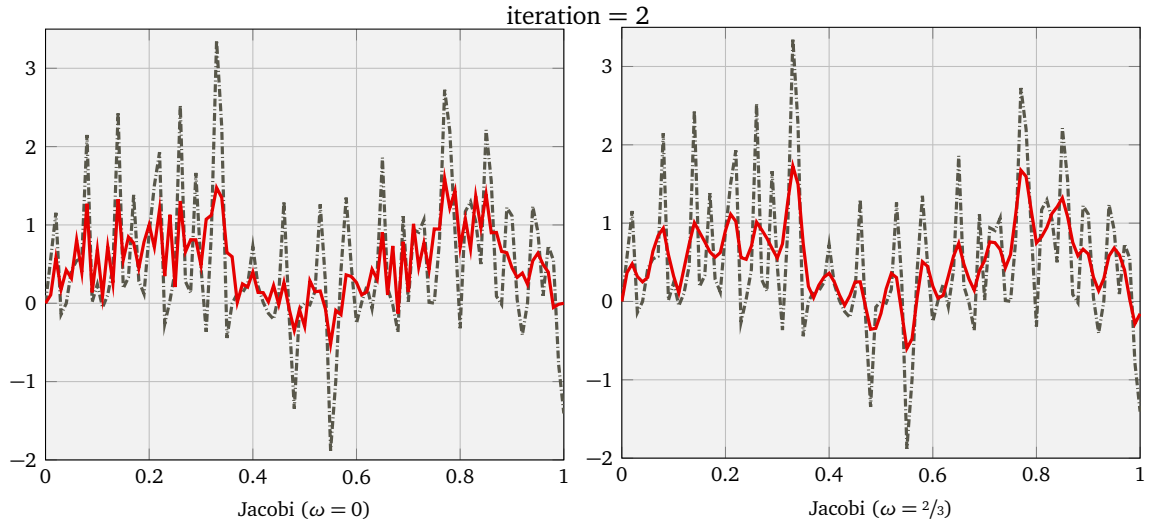
$$|\lambda_{n/2}(\boldsymbol{G}_{\omega J})| = 0 \quad |\lambda_n(\boldsymbol{G}_{\omega J})| = 1$$

- Jacobi is not a smoother, but damped-Jacobi is a smoother

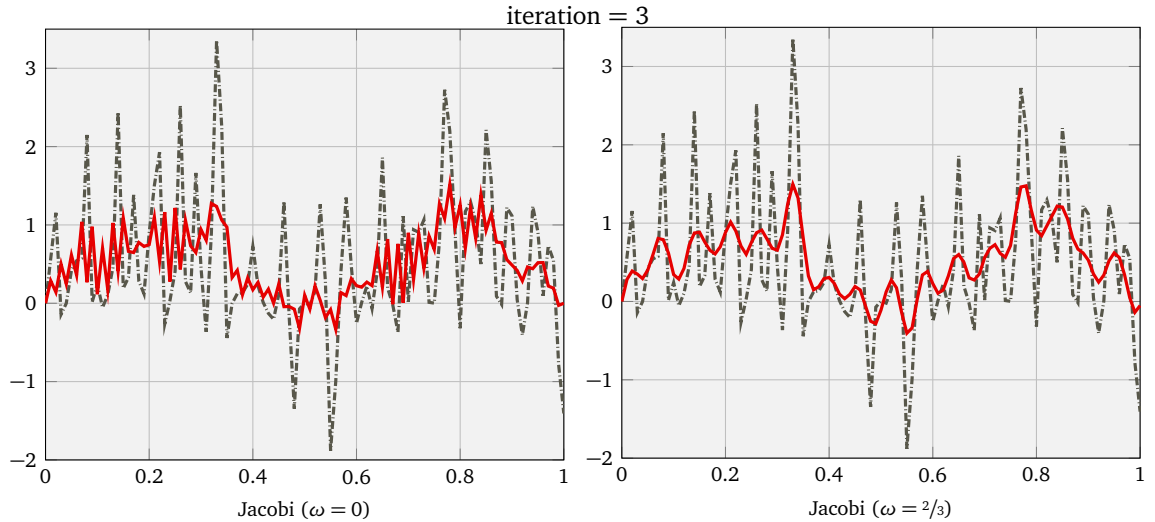
Smoothing error for Jacobi v/s damped-Jacobi methods



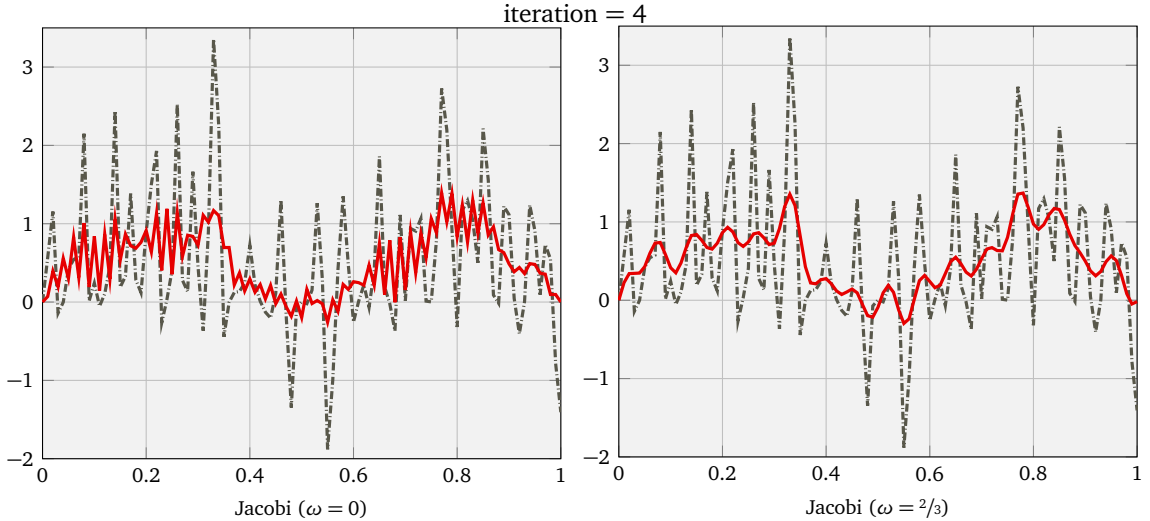
Smoothing error for Jacobi v/s damped-Jacobi methods



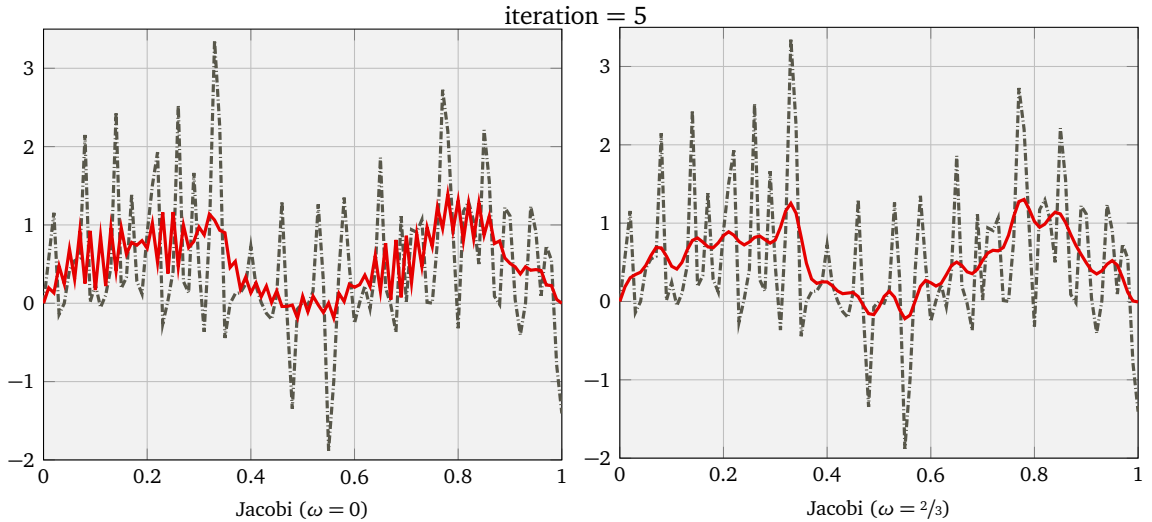
Smoothing error for Jacobi v/s damped-Jacobi methods



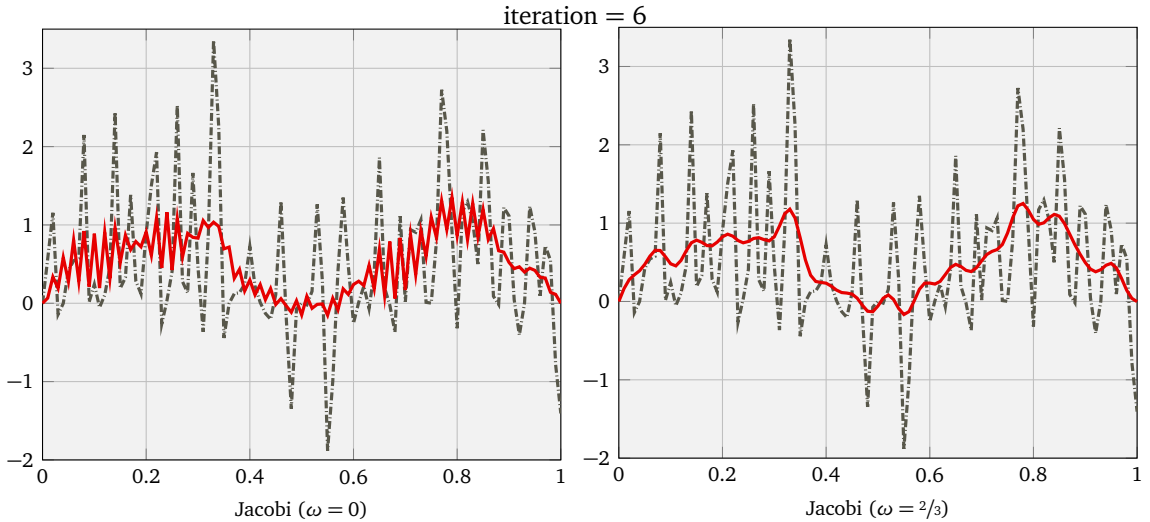
Smoothing error for Jacobi v/s damped-Jacobi methods



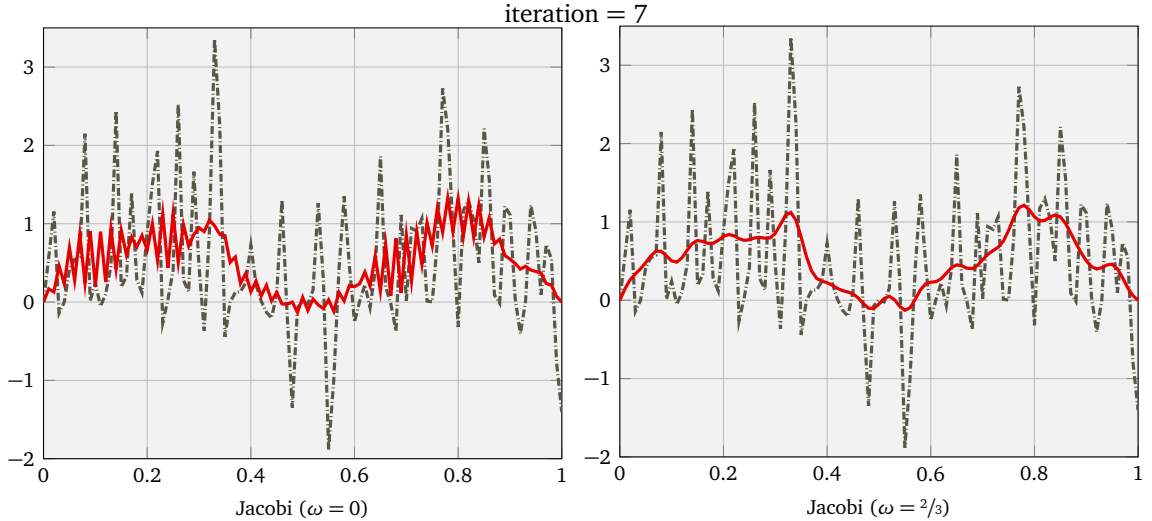
Smoothing error for Jacobi v/s damped-Jacobi methods



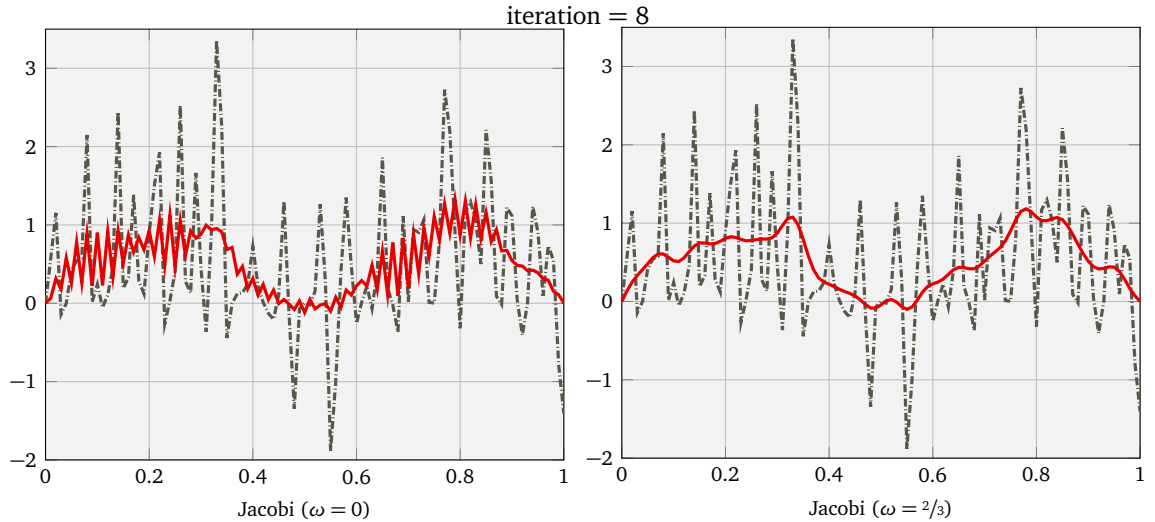
Smoothing error for Jacobi v/s damped-Jacobi methods



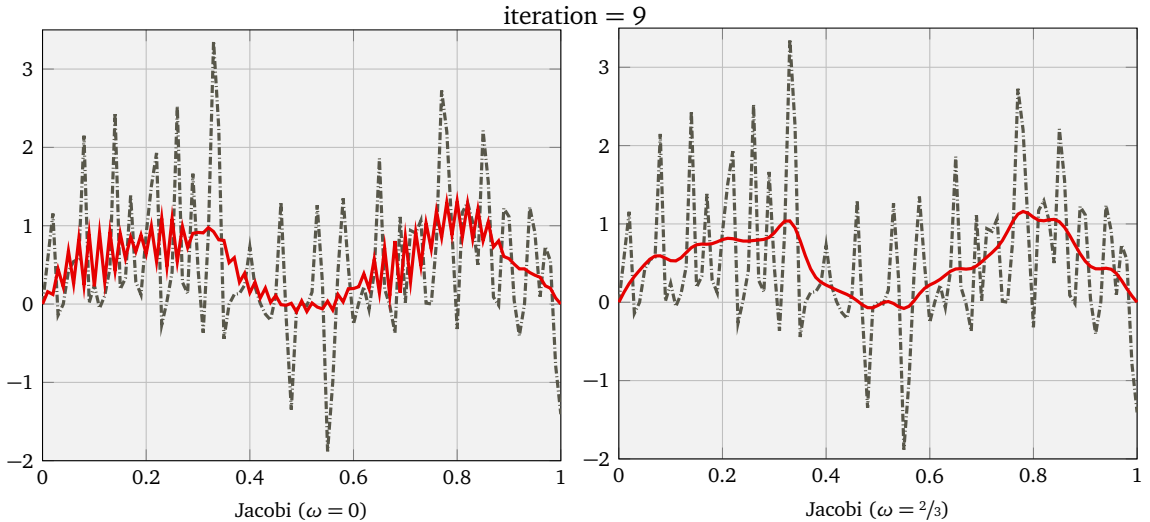
Smoothing error for Jacobi v/s damped-Jacobi methods



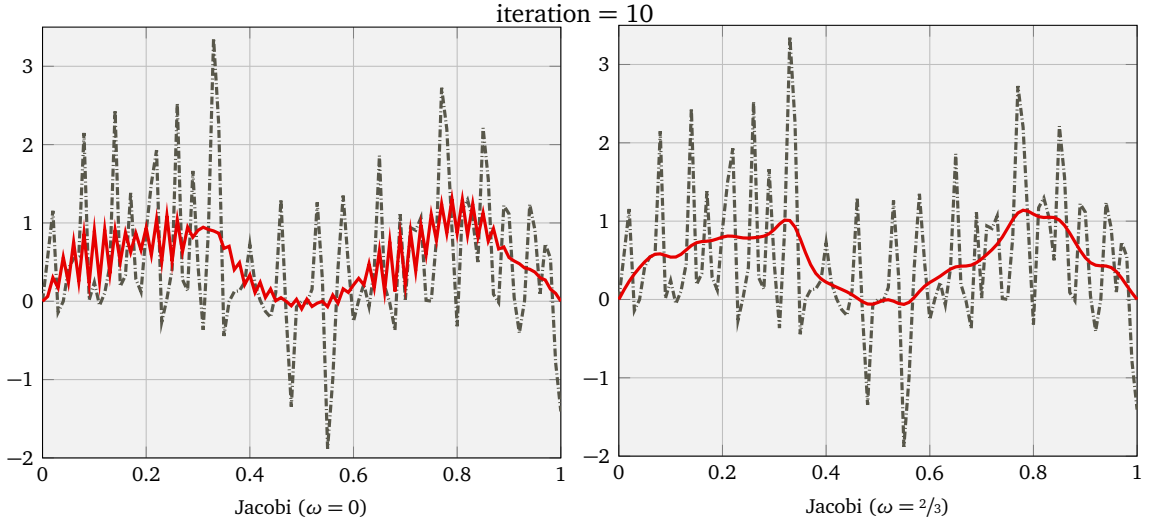
Smoothing error for Jacobi v/s damped-Jacobi methods



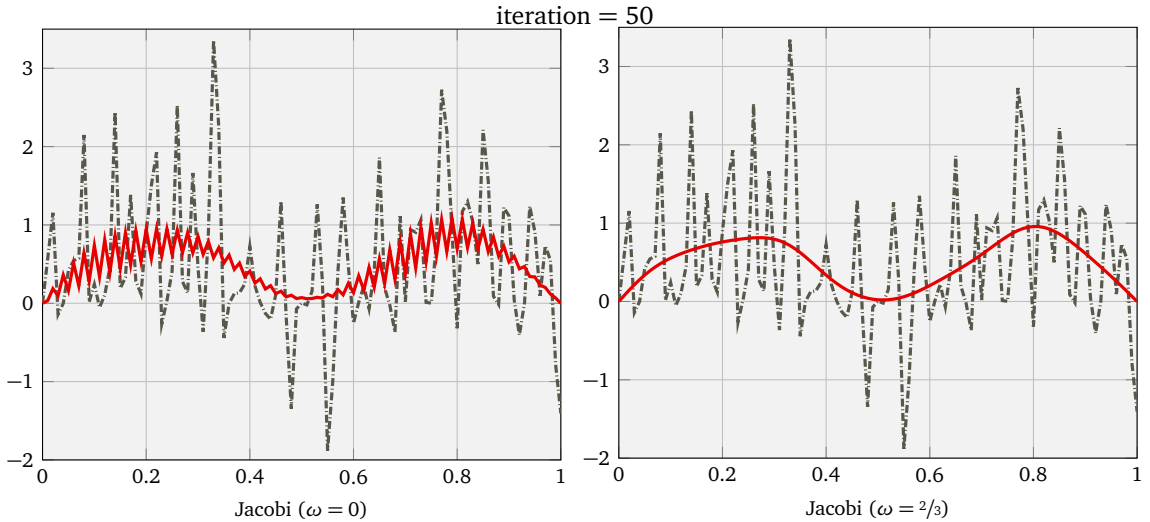
Smoothing error for Jacobi v/s damped-Jacobi methods



Smoothing error for Jacobi v/s damped-Jacobi methods



Smoothing error for Jacobi v/s damped-Jacobi methods



Towards projection method

Observation:

- Classical iterative schemes have **smoothing property**,
- The oscillatory modes of the error are damped very quickly, the smooth modes are damped very slowly

Idea:

- Do ν smoothing iterations to improve on the initial guess

$$\boldsymbol{x}^{(k)} = \boldsymbol{S}^{\nu}(\boldsymbol{x}^{(k-1)})$$

- Look for an update $\delta\boldsymbol{x}$ that provides the "best" improvement for iterate

$$\boldsymbol{x}^{*} = \boldsymbol{x}^{(k)} + \delta\boldsymbol{x}$$

- Construct $\delta\boldsymbol{x} \approx \boldsymbol{e}^{(k)}$ by least square minimization on a smaller space $\boldsymbol{V}$

$$\min_{\delta\boldsymbol{x} \in \text{span}\{\boldsymbol{V}\}} \|\delta\boldsymbol{x} - \boldsymbol{e}^{(k)}\|$$

- Then $\delta\boldsymbol{x} = \boldsymbol{V}\boldsymbol{w}$

$$\boldsymbol{V}^{\top}\boldsymbol{V}\boldsymbol{w} = \boldsymbol{V}^{\top}\boldsymbol{e}^{(k)}$$

- Update from the projection method

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{V}(\boldsymbol{V}^{\top}\boldsymbol{V})^{-1}\boldsymbol{V}^{\top}\boldsymbol{e}^{(k)}$$

A-orthogonal projection

- Instead of standard Euclidean norm, we can employ A-norm
- Thus employing A-orthogonal projection onto the range of $\boldsymbol{V}$
- Construct $\delta \boldsymbol{x} \approx \boldsymbol{e}^{(k)}$ by minimization in A-norm

$$\min_{\delta \boldsymbol{x} \in \text{span}\{\boldsymbol{V}\}} \|\delta \boldsymbol{x} - \boldsymbol{e}^{(k)}\|_A$$

- Then $\delta \boldsymbol{x} = \boldsymbol{V}\boldsymbol{w}$

$$\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}\boldsymbol{w} = \boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{e}^{(k)}$$

- Update from the projection method

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{e}^{(k)}$$
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{r}^{(k)}$$

Table of Contents

Coarse grids

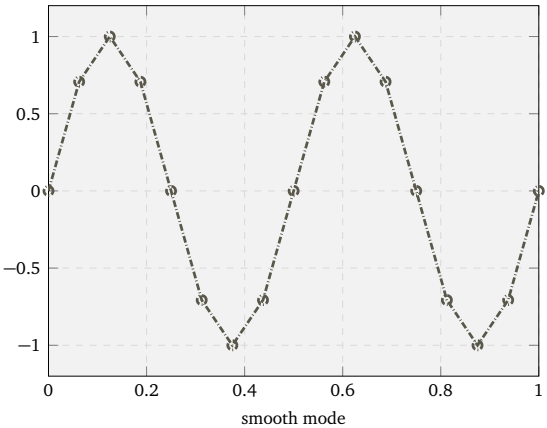
- Smooth error modes on a fine grid appear oscillatory on a coarse grid
- Thus, smoothing iteration on coarse grid can damp the oscillatory errors
- Relaxation on coarse grid is much cheaper, as the mesh size grows with the coarsening factor of $\frac{1}{2}$
- Relaxation on the coarse grid has better convergence rate

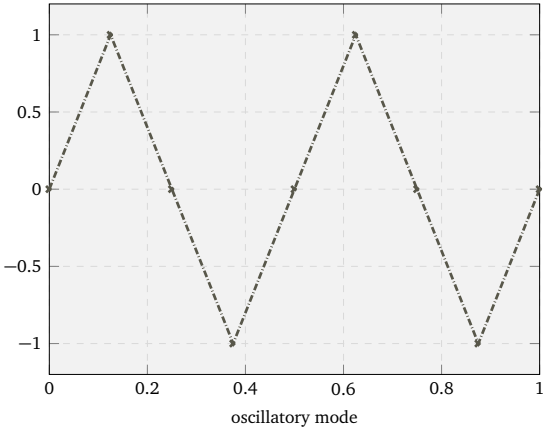
$$\rho(\boldsymbol{G}) = 1 - \mathcal{O}(4h^2)$$

- The A-orthogonal projection property ensure that for a given subspace $\boldsymbol{V}$, we get optimal coarse-grid correction

Smooth error on coarse grid

- Smooth modes appear oscillatory on coarse grid





Coarse modes

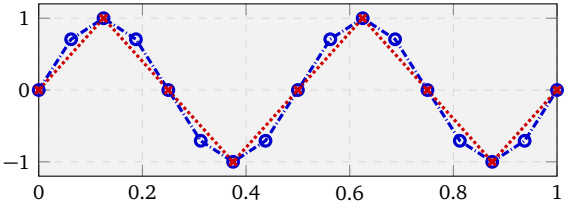
- For $k \in [1, n/2]$ the k^{th} mode is preserved on the coarse grid

$$(\boldsymbol{v}_k^h)_{2j} = \sin\frac{jk\pi}{n/2} = \sin\frac{2jk\pi}{n} = (\boldsymbol{v}_k^{2h})_j$$

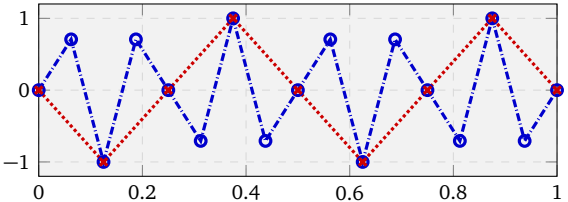
- For $k \in (n/2, n]$ the k^{th} mode is invisible on the coarse grid (Aliasing)

$$\begin{aligned}(\boldsymbol{v}_k^h)_{2j} = \sin\frac{2jk\pi}{n} &= -\sin\frac{2j(n-k)\pi}{n} \\ = -\sin\frac{j(n-k)\pi}{n/2} &= -(\boldsymbol{v}_{n-k}^{2h})_j\end{aligned}$$

- It is necessary to damp the oscillating error modes on fine grid before a problem on coarse grid is considered. Otherwise, one would get additional smooth error modes on the coarser grid.



smooth mode



oscillatory mode

Incorporating coarse grid

Recall: Projection iteration

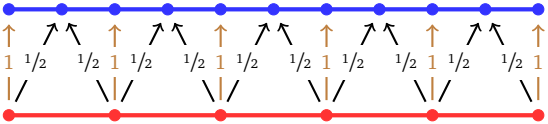
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{V}(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{r}^{(k)}$$

Questions:

- How to approximate $\boldsymbol{V}$ to transfer information between a fine level and a coarse level?
- How to solve the problem on coarse space?
- Which problem do we solve on coarse grid?

Prolongation operator

- Construct a prolongation operator $\boldsymbol{P} : \Omega_H \to \Omega_h$
- We employ linear interpolation as a prolongation operator: the simplest approach!

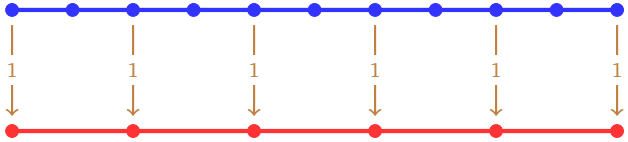


- Prolongation gives the best results, if the error on the fine grid is smooth
- Hence, it is an appropriate complement to the smoother which works most efficiently if the error is oscillating
- Prolongation operator

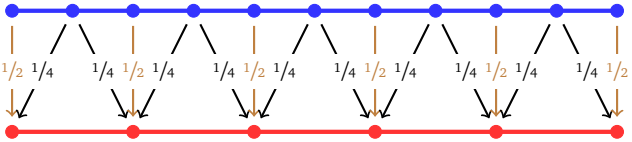
$$\boldsymbol{P} = \begin{bmatrix} 1 & & & & \\ 1/2 & 1/2 & & & \\ & 1 & & & \\ & 1/2 & 1/2 & & \\ & & \ddots & & \\ & & \ddots & 1/2 & \\ & & & 1 & \end{bmatrix}$$

Restriction operators

- Construct a restriction operator $\boldsymbol{R} : \Omega_h \rightarrow \Omega_H$
- Injection is the simplest restriction



- Not an efficient approach, as the errors on the fine node is not corrected on the coarse level
- Weighted restriction: $\boldsymbol{R} = c\boldsymbol{P}^\top$ (used restrict primal quantities)

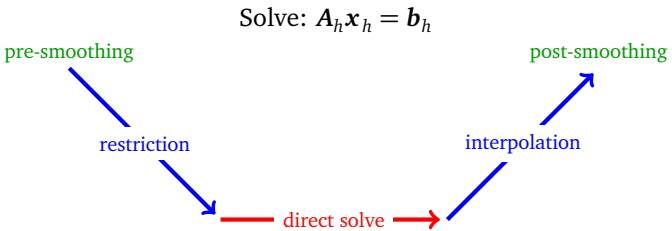


- In FE framework $\boldsymbol{R} = \boldsymbol{P}^\top$ (used to restrict residuals/dual quantities)

Further observations on transfer operators

- Columns of interpolation matrix represent basis functions
- Thus the basis functions on coarse level are constructed as linear combination of the basis functions from the finest level
- Hierarchy of nested spaces
- Full rank for of the transfer operator
- The constant functions are preserved when going to the fine level

Two-level method



Coarse grid correction scheme:

1. Perform ν_{pre} pre-smoothing steps: $x_h \leftarrow x_h + S^{\nu_{pre}}(b_h - A_h x_h)$
2. Restrict residual to the coarse grid: $r_H \leftarrow R(b - A_h x_h)$
3. Solve coarse grid equation: $A_H e_H = r_H$
4. Interpolate correction to the fine grid: $x_h \leftarrow x_h + P e_H$
5. Perform ν_{post} post-smoothing steps: $x_h \leftarrow x_h + S^{\nu_{post}}(b_h - A_h x_h)$

Coarse grid matrix

Question: How to construct $\boldsymbol{A}_H$?

- A straightforward approach consists of constructing $\boldsymbol{A}_H$ by discretizing the PDE on Ω_H
- Galerkin projection approach can be given as:

$$\begin{aligned} \boldsymbol{A}_h \boldsymbol{e}_h &= \boldsymbol{r}_h \\ \boldsymbol{A}_h \boldsymbol{P} \boldsymbol{e}_H &= \boldsymbol{r}_h \\ \boldsymbol{R} \boldsymbol{A}_h \boldsymbol{P} \boldsymbol{e}_H &= \boldsymbol{R} \boldsymbol{r}_h \\ \boldsymbol{A}_H \boldsymbol{e}_H &= \boldsymbol{r}_H \end{aligned}$$

- Galerkin projection based on the assumption that the error $\boldsymbol{e}_h$ is in the range of the prolongation. This property is in general not given.
- For simple cases, $\boldsymbol{A}_H$ constructed using rediscretization and Galerkin projection coincides
- Although, it does not hold in general

Analysis of two-grid method

- Recall iterative method

$$\boldsymbol{x}_h \leftarrow (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{x}_h + \boldsymbol{M}^{-1}\boldsymbol{b}_h$$

- Two grid iteration process

$$\boldsymbol{I} - \boldsymbol{M}_{TG}^{-1}\boldsymbol{A}_h = (\boldsymbol{I} - \boldsymbol{M}^{-\top}\boldsymbol{A}_h)^{\nu_{post}}(\boldsymbol{I} - \boldsymbol{P}(\boldsymbol{P}^\top\boldsymbol{A}_h\boldsymbol{P})^{-1}\boldsymbol{P}^\top\boldsymbol{A}_h)(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}_h)^{\nu_{pre}}$$

- Error propagation

$$\boldsymbol{e} \leftarrow (\boldsymbol{I} - \boldsymbol{M}_{TG}^{-1}\boldsymbol{A})^k\boldsymbol{e}$$

- The two-grid method converges, if and only if

$$\rho(\boldsymbol{I} - \boldsymbol{M}_{TG}^{-1}\boldsymbol{A}) < 1$$

Analysis of two-grid method

- Two-grid iteration matrix without post-smoothing steps:

$$G_{TG} = I - M_{TG}^{-1} A_h = (I - P(P^\top A_h P)^{-1} P^\top A_h)(I - M^{-1} A_h)^\nu$$

$$\begin{aligned}
\|G_{TG}\|_2 &= \|(I - P(P^\top A_h P)^{-1} P^\top A_h) G_s^\nu\|_2 \\
&= \|(I - P(P^\top A P)^{-1} P^\top A) A_h^{-1} A_h G_s^\nu\|_2 \\
&\leqslant \|A_h^{-1} - P(P^\top A_h P)^{-1} P^\top)\|_2 \; \|A_h G_s^\nu\|_2 \\
&= \underbrace{\|A_h^{-1} - P A_c^{-1} P^\top)\|_2}_{\text{approximation property}} \quad \underbrace{\|A_h G_s^\nu\|_2}_{\text{smoothing property}}
\end{aligned}$$

- Smoothing property and approximation property

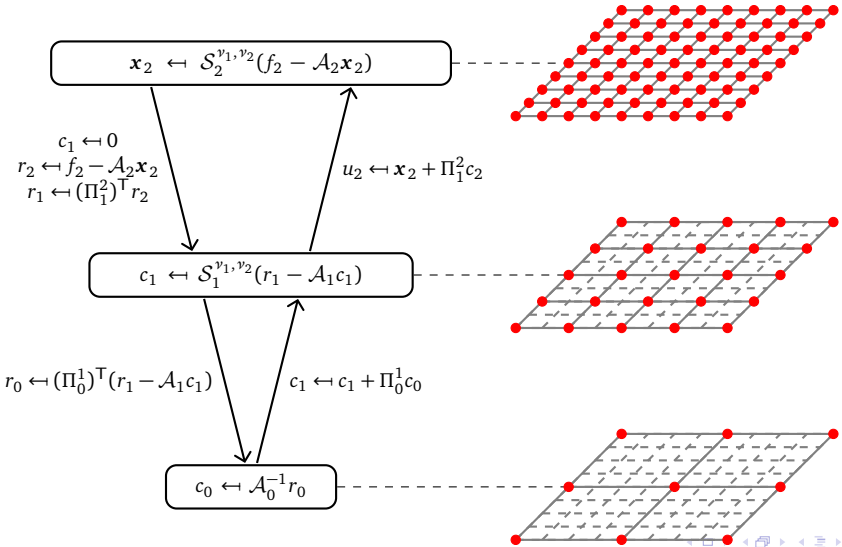
$$\|A_h G_s^\nu\|_2 \leqslant \frac{c}{\nu} h^{-2} \qquad \|A_h^{-1} - P A_c^{-1} P^\top\|_2 \leqslant c h^2$$

- Contraction property for sufficiently large ν

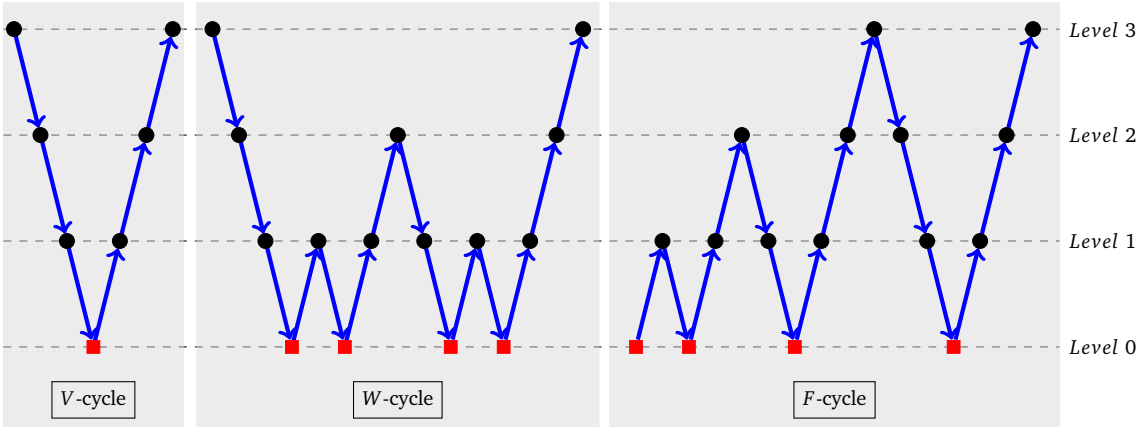
$$\|G_{TG}\|_2 \leqslant \frac{c}{\nu} < 1$$

Table of Contents

Sketch of multigrid method



Type of multigrid cycles



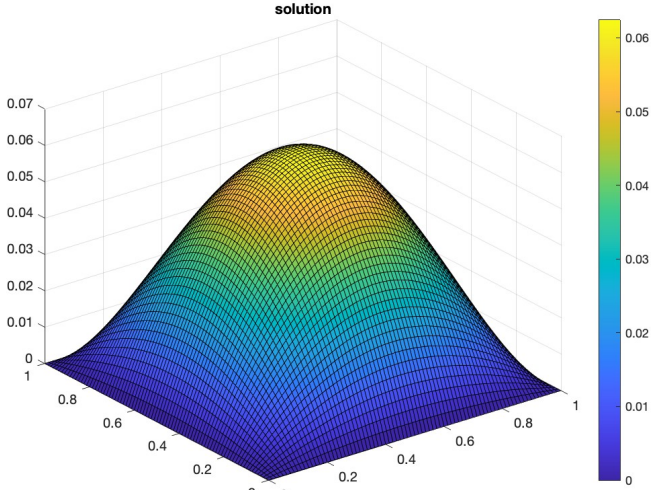
Multigrid extension

- V-cycle
  - Recursive formulation of the two-grid method
- W-cycle
  - Useful when coarse grid is not optimal
  - Realized by calling V-cycle twice on each level
- F-cycle
  - Requires discretization on grid
  - Motivated by nested iteration
  - Eliminates errors from coarse to fine level

Table of Contents

Numerical experiments

Poisson problem



Comparing various MG-cycles

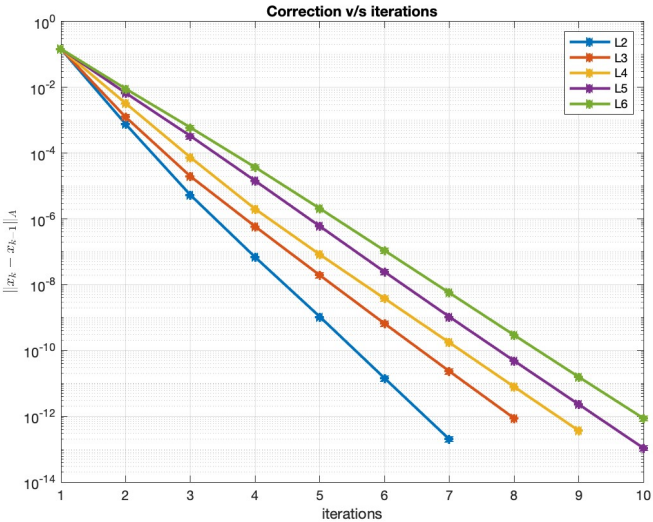
mesh size	# levels	V(3,3)	V(5,5)	W(3,3)	W(5,5)
20×20	2	7	6	6	6
40×40	3	8	8	7	6
80×80	4	9	8	7	6
160×160	5	10	9	7	6
320×320	6	10	9	7	6

\# Iterations

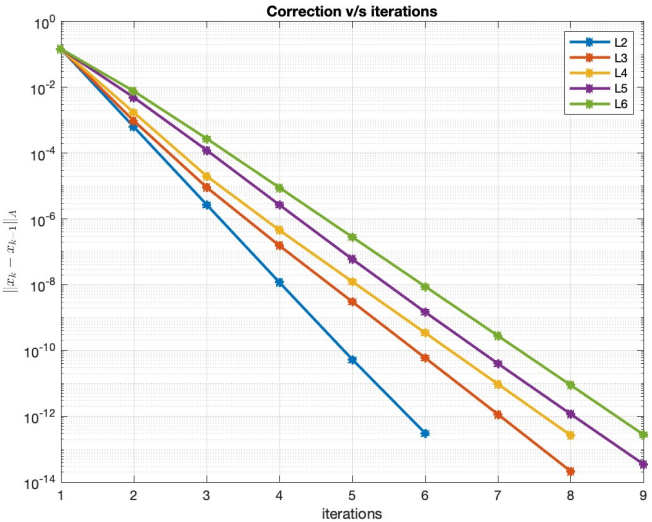
mesh size	# levels	V(3,3)	V(5,5)	W(3,3)	W(5,5)
20×20	2	0.0139	0.0057	0.0139	0.0057
40×40	3	0.0372	0.0185	0.0182	0.0093
80×80	4	0.0460	0.0276	0.0184	0.0106
160×160	5	0.0467	0.0299	0.0186	0.0108
320×320	6	0.0551	0.0312	0.0179	0.0108

Asymptotic convergence rate ($\rho^* = \frac{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_A}{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_A}$)

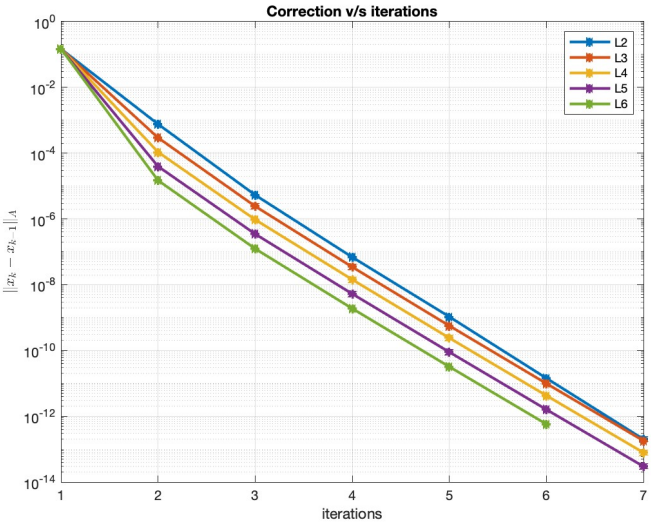
Convergence of V(3,3) cycle



Convergence of V(5,5) cycle



Convergence of W(3,3) cycle



Convergence of W(5,5) cycle

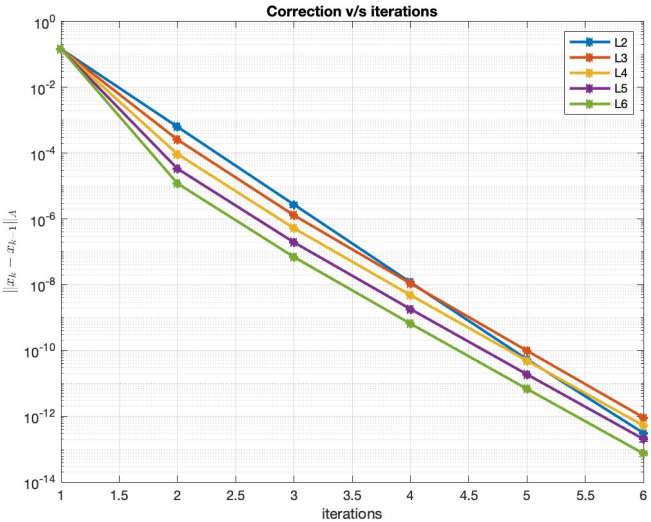


Table of Contents

Still many topics remain

Algebraic multigrid methods

- Aims to construct better transfer operators
- When mesh is not available
- Utilizes only graph information from matrix
- Compute strength between the edges of graph

AMG libraries:

- Hypre - BoomerAMG: http://acts.nersc.gov/hypre/
- Trilinos - muLU: https://trilinos.org/
- PyAMG: http://pyamg.org/
- PETSc: https://petsc.org

Questions?

Contact: hardik.kothari@usi.ch