Chapter 7: Elliptic equations and Sparse linear systems

Gustaf Söderlind
Numerical Analysis, Lund University

Multi-Grid Methods and Applications, by Wolfgang Hackbusch, 1985

© Gustaf Söderlind, Numerical Analysis, Mathematical Sciences, Lund University, 2009-2010
1. Overview

Consider simplest linear 2p-BVP (1D Poisson)

\[ y'' = f(x) \]

\[ y(0) = \alpha; \quad y(1) = \beta \]

Introduce equidistant grid with \( \Delta x = 1/(N + 1) \)

Discretization

\[
\frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2} = f(x_i)
\]

\[ y_0 = \alpha; \quad y_{N+1} = \beta \]
Linear system of equations

Tridiagonal $N \times N$ matrix formulation

\[
\frac{1}{\Delta x^2} \begin{pmatrix}
-2 & 1 & & \\
1 & -2 & 1 & \\
& & \ddots & \\
1 & -2 & & 1
\end{pmatrix} \begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{pmatrix} = \begin{pmatrix}
f(x_1) - \alpha / \Delta x^2 \\
f(x_2) \\
\vdots \\
 f(x_N) - \beta / \Delta x^2
\end{pmatrix}
\]

In matrix–vector form

\[ T_{\Delta x} y = f \]
Elliptic model problem: 2D Poisson equation

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \]

on \( \Omega = [0, 1] \times [0, 1] \) with Dirichlet conditions \( u = 0 \) on \( \partial \Omega \)

Uniform grid \( \{x_i, y_j\}_{i,j=1}^{N,N} \), mesh width \( \Delta x = \Delta y = 1/(N + 1) \)

**Discretization** Finite differences with \( u_{i,j} \approx u(x_i, y_j) \)

\[ \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = f(x_i, y_j) \]
Equidistant mesh $\Delta x = \Delta y$

$$
\frac{u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i,j+1} + u_{i+1,j}}{\Delta x^2} = f(x_i, y_j)
$$

Participating approximations and mesh points
Computational “stencil” for $\Delta x = \Delta y$

$$\frac{u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i,j+1} + u_{i+1,j}}{\Delta x^2} = f(x_i, y_j)$$

“Five-point operator”
The FDM linear system of equations

Lexicographic ordering of unknowns \( \Rightarrow \) *partitioned system*

\[
\begin{pmatrix}
T & I & 0 & \ldots \\
I & T & I & \\
I & T & I & \\
\vdots & \ddots & \ddots & I \\
\ldots & \ldots & 0 & I & T
\end{pmatrix}
\begin{pmatrix}
u_{i,1} \\
u_{i,2} \\
u_{i,3} \\
\vdots \\
u_{i,N}
\end{pmatrix}
=
\begin{pmatrix}
f(x_{i,1}, y_1) \\
f(x_{i,2}, y_2) \\
f(x_{i,3}, y_3) \\
\vdots \\
f(x_{i,N}, y_N)
\end{pmatrix}
\]

with Toeplitz matrix \( T = \text{tridiag}(1, -4, 1) \)

The system is \( N^2 \times N^2 \), hence *large and very sparse*
3D Poisson equation. The “curse” of dimension

Partitioned system

\[
\begin{pmatrix}
T & I & 0 & \cdots & I \\
I & T & I & & \\
& I & T & I & \\
& & I & T & I \\
& & & I & \cdots & I \\
& & & & 0 & I & T
\end{pmatrix}
\begin{pmatrix}
u_{1,1,1} \\
\vdots \\
\vdots \\
u_{N,N,N}
\end{pmatrix}
= \begin{pmatrix}
f_{1,1,1} \\
\vdots \\
\vdots \\
f_{N,N,N}
\end{pmatrix}
\]

with Toeplitz matrix \( T = \text{tridiag}(1, -6, 1) \)

The system is \( N^3 \times N^3 \), hence extremely large and sparse
Galerkin method (Finite Element Method)

1. **Basis functions** \{\varphi_i\}

2. **Approximate**
   
   \[ u = \sum c_j \varphi_j \]

3. **Determine** \(c_j\) from
   
   \[ \sum c_j \int \nabla \varphi_i \cdot \nabla \varphi_j = \int f \varphi_i \]

The \(c_j\) are determined by the **linear system**

\[ Kc = F \]

The **stiffness matrix** \(K\) has similar structure to FDM matrix

Stiffness matrix elements

\[ k_{ij} = \int \nabla \varphi_i \cdot \nabla \varphi_j = a(\varphi_i, \varphi_j) \]

Right-hand side

\[ F_i = \int \varphi_i f = \langle \varphi_i, f \rangle = \sum f_j \langle \varphi_i, \varphi_j \rangle \]
The FEM mesh. Domain triangulation

Piecewise linear basis \( \{ \varphi_j \} \) with *triangulation mesh*

Same number of nodes too
Iterative methods

All discretizations lead to very large linear systems

\[ Ty = f \]

typically having millions of equations

Matrix factorization methods are out of the question! Use iterative methods instead!

Explicit iterative methods \[ y^{m+1} = By^m + c \]

Implicit iterative methods \[ Dy^{m+1} = By^m + c \]
What are multigrid methods?

Multigrid methods are iterative methods that use the fact that the origin of the linear system is some discretization, and that the grid properties affect the convergence rate.

There is a relation to Fourier analysis as it turns out that mesh width (inverse spatial frequency) is a key factor governing convergence.

The methods are called multigrid, because the iteration will alternate between several different grids in order to speed up convergence.
2. Iterative methods for linear systems

Given a linear system $Au = f$ construct sequence $u^m \to u$

Then

$$Au^m = f + r^m$$

$$Au = f$$

Definitions

1. The *error* is defined by $e^m = u^m - u$

2. The *residual* is defined by $r^m = Au^m - f$

3. Relation via the *error–residual equation* $Ae^m = r^m$
The basic iterative methods

There are four different basic iterative methods

1. The *Jacobi* method
2. The *Gauss–Seidel* method
3. The *Successive Overrelaxation* (SOR) method
4. The *Symmetric SOR* method

Advanced methods include the *Conjugate Gradient* (CG) method; the *Generalized Minimum Residual* (GMRES) method; and various forms of *Multigrid* (MG) methods
The Jacobi method

**Splitting**  Write $Au = f$ as $(D - L - U)u = f$ with $D$ diagonal; $L$ lower triangular; $U$ upper triangular

$$Du = (L + U)u + f$$

$$u = D^{-1}(L + U)u + D^{-1}f$$

**Jacobi method**  Use fixed point iteration

$$u^{m+1} = D^{-1}(L + U)u^m + D^{-1}f$$
The Jacobi method. Implementation

Given $u^m$, calculate residual $r^m$, and update according to

\[ r^m \leftarrow Au^m - f \]
\[ u^{m+1} \leftarrow u^m - D^{-1}r^m \]

**Note** The scheme implies that *each single, scalar equation is solved independently of the other equations*

It can be directly used on massively parallel computers
Relation to fixed point iteration

From actual implementation

\[ u^{m+1} = u^m - D^{-1}((D - L - U)u^m - f) = D^{-1}(L + U)u^m + D^{-1}f \]

For analysis purposes this is written

\[ u^{m+1} = P_J u^m + D^{-1}f \]

where the *Jacobi iteration matrix* is

\[ P_J = D^{-1}(L + U) \]
If $A = T_{\Delta x}$ then

$$P_J = D^{-1}(L + U) = \text{tridiag}(1/2 \quad 0 \quad 1/2)$$

and

$$u^{m+1} = P_J u^m + D^{-1} f$$

For the exact solution

$$u = P_J u + D^{-1} f$$

**Error recursion**

$$e^{m+1} = P_J e^m$$
1D Poisson + Jacobi. Convergence

Error recursion \[ e^{m+1} = P_J e^m \]

Convergence \[ e^m \to 0 \]

1. Necessary condition \[ \rho[P_J] < 1 \]
2. Sufficient condition \[ \| P_J \| < 1 \]

Definition Spectral radius \[ \rho[A] = \max_k |\lambda_k[A]| \]

For 1D Poisson, we need to calculate the eigenvalues of the Jacobi iteration matrix \[ P_J = \text{tridiag}(1/2 \ 0 \ 1/2) \]
Eigenvalues of Toeplitz matrices

\[ P_J = \text{tridiag}(1/2 \ 0 \ 1/2) = S/2 \]

\[ Su = \begin{pmatrix}
0 & 1 & 0 & \ldots \\
1 & 0 & 1 \\
1 & 0 & 1 \\
\vdots & \ddots & 1 \\
\ldots & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_N
\end{pmatrix} = \lambda u \]

Find the eigenvalues of \( S \)!
Eigenvalues of symmetric Toeplitz matrix $S$

Consider the $n^{th}$ equation of $Su = \lambda u$

$$u_{n+1} + u_{n-1} = \lambda u_n$$

*Linear difference equation* with boundary values

$$u_0 = 0; \quad u_{N+1} = 0$$

*Characteristic equation*

$$z^2 - \lambda z + 1 = 0$$
Eigenvalues of $S$...

Roots of $z^2 - \lambda z + 1 = 0$ are $z$ and $1/z$ (product 1)

General solution $u_n = \alpha z^n + \beta z^{-n}$

Boundary condition $u_0 = 0 = \alpha + \beta \quad \Rightarrow$

Solution $u_n = \alpha(z^n - z^{-n})$

Boundary condition $u_{N+1} = 0 = \alpha(z^{N+1} - z^{-(N+1)}) \quad \Rightarrow \quad z^{2(N+1)} = 1$

Roots $z_k = \exp\left(\frac{k\pi i}{N+1}\right) \quad k = 1 : N$
Eigenvalues...

Sum of the roots of \( z^2 - \lambda z + 1 = 0 \) are

\[
\lambda_k = z_k + 1/z_k \quad \Rightarrow
\]

\[
\lambda_k[S] = \exp\left(\frac{k\pi i}{N + 1}\right) + \exp\left(-\frac{k\pi i}{N + 1}\right) = 2 \cos \frac{k\pi}{N + 1}
\]

Hence the eigenvalues of \( P_J = \text{tridiag}(1/2 \quad 0 \quad 1/2) \) are

\[
\lambda_k[P_J] = \cos \frac{k\pi}{N + 1} \in (-1, 1)
\]
Eigenvalue locations

Eigenvalues get very close to $\pm 1$ for large $N$ ($N = 31$)
Peacock plot. Eigenvalues and unit circle, $N = 31$

Eigenvalues $\lambda_k = \cos k\pi \Delta x$ are projections on real axis

In terms of Chebyshev zeros, $T'_{N+1}(\lambda_k) = 0$ and $U_N(\lambda_k) = 0$
Slow, slower, slowest

With \( \lambda_k[P_J] = \cos k\pi/(N + 1) \) we have

\[
\rho[P_J] = \lambda_1[P_J] = -\lambda_N[P_J] \approx 1 - \frac{\pi^2 \Delta x^2}{2} + O(\Delta x^4)
\]

So Jacobi’s method will converge as \( \rho[P_J] < 1 \), but convergence will be painfully slow!

Example  \( N = 99 \Rightarrow \rho[P_J] \approx 0.9995 \)
Plot of $L^2$ error as a function of iteration number ($N = 99$)
Unit error reduction requires $O(N^2)$ iterations

Plot of $L^2$ error after $N^2$ iterations ($N = 15, 31, 63, 127$)
3. Power iteration

The fixed point iteration $y^{m+1} = Ay^m$ is also called power iteration.

It implies $y^m = A^m y^0$ \textit{Note powers of $A$!}

Assume distinct eigenvalues $Ax^k = \lambda_k x^k$ with $|\lambda_1| > |\lambda_k|$, for $k = 2, \ldots, N$. Let $y^0 = \sum \alpha_k x^k$. Then

$$\frac{y^m}{\lambda_1^m} = \alpha_1 x^1 + \sum_{k=2}^N \alpha_k x^k \left( \frac{\lambda_k}{\lambda_1} \right)^m \rightarrow \alpha_1 x^1$$

So the vector $y^m$ gets aligned with eigenvector $x^1$. 

The power method

Once we have a good approximation to $x^1$ we obtain the corresponding eigenvalue from the Rayleigh quotient

$$\lambda_1 \approx \frac{\langle y^m, Ay^m \rangle}{\langle y^m, y^m \rangle}$$

The power method  Take $y^0$ and iterate until convergence:

$$\hat{y}^m := y^m / \|y^m\|_2$$

$$y^{m+1} := A\hat{y}^m$$

$$\sigma_m := \langle \hat{y}^m, y^{m+1} \rangle \rightarrow \lambda_1$$
Why study the power method?

The error recursion $e^{m+1} = P_J e^m$ is a power iteration

Let $P_J v = \lambda v$, with $|\lambda| = \rho[P_J]$. Then any convergent fixed point type iteration will have the following properties:

1. The error will initially decay relatively fast
2. Convergence then slows to be governed by $\rho[P_J]$
3. The error $e^m$ will become aligned with the eigenvector $v$
What is the convergence to $x^1$ like?

As

$$\frac{y^m}{\lambda_1^m} = \alpha_1 x^1 + \sum_{k=2}^{N} \alpha_k x^k \left(\frac{\lambda_k}{\lambda_1}\right)^m \rightarrow \alpha_1 x^1$$

we see that the error decay is *exponential* and governed by the *ratio of the largest to the next largest eigenvalue*

The decay may be very *slow if eigenvalue gap is small*.

For $P_J$ the gap is very small, with ratio

$$\frac{\lambda_2}{\lambda_1} = \frac{\cos 2\pi \Delta x}{\cos \pi \Delta x} \approx \frac{2 - 4\pi^2 \Delta x^2}{2 - \pi^2 \Delta x^2} \approx 1 - \frac{3\pi^2 \Delta x^2}{2}$$
What is the convergence to \( \max |\lambda| \) like?

Non-normal matrices  \textit{linear convergence}  
Normal matrices  \textit{quadratic convergence}  

Example  

\[
A = \begin{pmatrix}
12 & 2 & 2.05 \\
2 & 9 & 1 \\
1.95 & 1 & 7 \\
\end{pmatrix}
\quad B = \begin{pmatrix}
12 & 2 & 2 \\
2 & 9 & 1 \\
2 & 1 & 7 \\
\end{pmatrix}
\]

Approximate eigenvalues \( \lambda \in \{13.74, 8.00, 6.26\} \), but the \textit{normal matrix has orthogonal eigenvectors}
Computation. Non-normal vs. normal matrix

Convergence history: Error in $\lambda_{\text{max}}$ vs. iteration number
Good news, bad news

**Normal differential operator ⇒ normal matrix**

\[ u_t = u_{xx} \]

\[ u_t = \nabla \cdot (p \nabla u) \]

\[ u_t + u_x = 0 \]

\[ \rho_t + \nabla \cdot (\rho \mathbf{v}) = 0 \]

**Non-normal differential operators**

\[ u_t = u_x + \frac{1}{Pe} u_{xx} \]
4. Eigenvectors of $P_J$

From eigenvalue problem of $P_J = \text{tridiag}(1/2 \ 0 \ 1/2)$

Recall $u_n = z^n_k - z^{-n}_k$ and $z_k = \exp\left(\frac{k\pi i}{N+1}\right)$, then

$$u_n^k = \exp\left(\frac{kn\pi i}{N+1}\right) - \exp\left(-\frac{kn\pi i}{N+1}\right) \sim \sin\left(\frac{kn\pi}{N+1}\right)$$

Same eigenvectors as for $T_{\Delta x}$
Eigenvectors of $P_J$ at $N = 31$

Lowest mode, $\sin(n\pi dx)$, on fine grid

Highest mode, $\sin(nN\pi dx)$, on fine grid
Eigenvectors of $P_J$ at $N = 31$

Below are two graphs showing the lowest and highest modes for sin($n\pi dx$) on a fine grid. The lowest mode exhibits a smooth curve, while the highest mode shows a highly oscillatory pattern. These modes are essential in understanding the behavior of the $P_J$ operator at the specified grid size.
From fine grid to coarse – now you see it

Lowest mode, \( \sin(n\pi dx) \), on fine grid

Highest mode, \( \sin(nN\pi dx) \), on fine grid
From fine grid to coarse – now you don’t

Lowest mode, $\sin(n\pi dx)$, on coarse grid

Highest mode, $\sin(nN\pi dx)$, on coarse grid
And yet... Nyquist Sampling Theorem

**Lowest mode, \( \sin(n\pi dx) \), on coarse grid**

**Highest mode, \( \sin(nN\pi dx) \), on coarse grid**
High frequency modes and grid density

What is a high frequency is a grid property

**Nyquist Sampling Theorem**  
*On a grid with $N$ interior points the highest frequency that can be represented is*  

$$u_n = \sin \frac{N \pi n}{N + 1}$$

**Note**

$$\sin \frac{N \pi n}{N + 1} = \sin (\pi n - \frac{\pi n}{N + 1}) = (-1)^{n+1} \sin \left( \frac{\pi n}{N + 1} \right)$$

*Highest frequency is* $(-1)^n$ *modulation of lowest frequency!*
Downsampling and aliasing

**Aliasing** If we take only every second sample, the highest frequency \( u_n = \sin \left( \frac{N \pi n}{N + 1} \right) \) “maps to” the function

\[
v_l = \sin \left( \frac{(2l + 1)\pi}{N + 1} \right) \approx \sin \pi x
\]

or to the function

\[
\omega_l = - \sin \left( \frac{2l\pi}{N + 1} \right) \approx - \sin \pi x
\]

The former highest frequency maps to the lowest!

In general, if the Nyquist frequency of the coarse grid is \( \hat{\omega} \) then a frequency \( \hat{\omega} + \delta \omega < 2\hat{\omega} \) “folds back” down to \( \hat{\omega} - \delta \omega \)
Jacobi iteration demonstration

Error vector sequence (50 iterations) at $N = 19$

Error reduction in 50 iterations
5. The Laplacian and the 5-point FD operator

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \lambda u \]

Computational domain \( \Omega = [0, 1] \times [0, 1] \) (unit square)

Dirichlet conditions \( u(x, y) = 0 \) on boundary

Variable separation \( u(x, y) := v(x)w(y) \) implies

\[ v_{xx}w + vw_{yy} = \lambda vw \]

or

\[ \frac{v_{xx}}{v} + \frac{w_{yy}}{w} = \lambda \]
Eigenvalues and eigenfunctions of the Laplacian

Let $v'' = \kappa_k v$ and $w'' = \mu_m w$, then

$$\lambda_{k,m} = \kappa_k + \mu_m$$

$$u^{k,m}(x, y) = v^k(x) \cdot w^m(y)$$

Modulation Theorem  

The Laplacian on the unit square has eigenvalues and eigenvectors, given by

$$\lambda_{k,m} = (k^2 + m^2)\pi^2$$  \hspace{1cm} k, m \in \mathbb{Z}^+$$

$$u^{k,m}(x, y) = \sin k\pi x \sin m\pi y$$

Analogous modulation result in 3D

Introduction to Multigrid Methods – p. 46/61
The 5-point finite difference Laplacian

Equidistant mesh width $\Delta x = \Delta y = 1/(N + 1)$

**Discretization** Finite differences with $u_{i,j} \approx u(x_i, y_j)$

$$
\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta x^2} = \lambda_{k,m} \Delta x \cdot u_{i,j}
$$

**Discrete variable separation** $u_{i,j}^{k,m} := v_i^k \cdot w_j^m$ implies

$$
\frac{v_{i-1}^k - 2v_i^k + v_{i+1}^k}{v_i^k \Delta x^2} + \frac{w_{j-1}^m - 2w_j^m + w_{j+1}^m}{w_j^m \Delta x^2} = \kappa_k^\Delta x + \mu_m^\Delta x
$$
Discrete eigenvalues and eigenfunctions

Discrete Modulation Theorem  The five-point finite
difference Laplacian on the unit square has $N^2$ discrete
eigenvalues and eigenvectors, given by

$$\lambda_{k,m} = -4(N + 1)^2 \cdot \left( \sin^2 \frac{k\pi}{2(N + 1)} + \sin^2 \frac{m\pi}{2(N + 1)} \right)$$

$$u_{i,j}^{k,m} = \sin \frac{k\pi i}{N + 1} \cdot \sin \frac{m\pi j}{N + 1}; \quad i, j, k, m = 1 : N$$

Again analogous results hold in 3D
Thus 1D theory gives most of the necessary insight
6. Advanced iteration methods

As Jacobi iteration is so slow, what can be done to speed up convergence?

**Damped Jacobi (under-relaxation)** Use a damping factor $\omega$ and iterate according to

\[
    r^m \leftarrow Au^m - f \\
    u^{m+1} \leftarrow u^m - \omega D^{-1}r^m
\]

The iteration stops only when $r^m = 0$, so it still solves the problem $Au = f$
Damped Jacobi method

Damped Jacobi is equivalent to

\[ u^{m+1} = ((1 - \omega)I + \omega D^{-1}(L + U))u^m + \omega D^{-1}f \]

so the iteration has \textit{damped Jacobi iteration matrix}

\[ P_\omega = (1 - \omega)I + \omega P_J \]

**Theorem**  \textit{The eigenvalues of} \( P_\omega \) \textit{are}

\[ \lambda_k[P_\omega] = 1 - \omega + \omega \cos \frac{k\pi}{N + 1} \]
Eigenvalue locations of damped Jacobi matrix

Eigenvalues for \( N = 31 \) and \( \omega = 1, 0.5, 0.3, 0.2 \)

Introduction to Multigrid Methods – p. 51/61
Eigenvalues of damped Jacobi matrix

By choosing $\omega$ we can (at least) eliminate large negative eigenvalues; what happens to the largest eigenvalue?

Largest eigenvalue becomes

$$
\lambda_1[P_\omega] = 1 - \omega + \omega \cos \frac{\pi}{N + 1} = 1 - 2\omega \sin^2 \frac{\pi \Delta x}{2}
$$

So

$$
\lambda_1[P_\omega] \approx 1 - \frac{\omega \pi^2 \Delta x^2}{2}
$$

(Ouch, this is even closer to 1 for $\omega < 1$... )
The Gauss–Seidel method

Splitting Write $Au = f$ as $(D - L - U)u = f$ with

$D$ diagonal; $L$ lower triangular; $U$ upper triangular

$$(D - L)u = Uu + f$$

$u = (D - L)^{-1}Uu + (D - L)^{-1}f$

Gauss–Seidel method Use fixed point iteration

$$u^{m+1} = (D - L)^{-1}Uu^m + (D - L)^{-1}f$$
The Gauss–Seidel method. Implementation

Given $u^m$, calculate residual $r^m$, and update according to

$$r^m \leftarrow Au^m - f$$

$$u^{m+1} \leftarrow u^m - (D - L)^{-1}r^m$$

**Note** The scheme implies that *each equation is solved in sequence of scalar equations*. It cannot be parallelized in a straightforward way

The **Gauss–Seidel iteration matrix** is

$$P_{GS} = (D - L)^{-1}U$$
1D Poisson + Gauss–Seidel method

Major features compared to Jacobi iterations are

- Regularizing iteration – no high-frequency error
- Faster convergence, although still slow

Eigenvalues of $P_{GS}$ satisfy generalized eigenvalue problem $Uu = \lambda (D - L)u$, according to Briggs:

$$\lambda_k[P_{GS}] = \cos^2 \frac{k\pi}{N + 1} \approx 1 - k^2 \pi^2 \Delta x^2$$

but this result is \textit{incorrect}!
Gauss–Seidel eigenvalues and eigenvectors

\[ \frac{N}{2} \text{ eigenvalues of } P_{GS} \text{ are} \]

\[
\lambda_k[P_{GS}] = \cos^2 \frac{k\pi}{N + 1} \approx 1 - k^2\pi^2 \Delta x^2
\]

and the remaining eigenvalues are zero!

**Note** Eigenvectors of \( P_{GS} \) and \( T_{\Delta x} \) do not coincide!

Therefore we need other ways of studying convergence
Digital filters and Bode diagrams

Study error recursion

\[ e^{m+1} = P_{GS}e^m \]

**one frequency at a time!** Study “output” \( y = P_{GS}u \) when \( u = \sin k\pi \Delta x \), an eigenvector of \( T_{\Delta x} \), and compute the \( L^2 \) (root-mean-square) “attenuation”

\[
\frac{\|y\|_{\Delta x}}{\|u\|_{\Delta x}} = \frac{\|P_{GS}u\|_{\Delta x}}{\|u\|_{\Delta x}} = \frac{\|P_{GS}u\|_2}{\|u\|_2}
\]

**Frequency response** Plot vs frequency \( \omega = k\pi \Delta x \in (0, \pi) \) or as a function of wave number \( k = 1 : N \) (Bode diagram)
Linear attenuation vs frequency: Gauss–Seidel and Jacobi
GS is a low-pass filter and J a band-stop filter

Logarithmic attenuation (dB): Gauss–Seidel and Jacobi

Frequency response: Gauss–Seidel (b) and Jacobi (r)
Gauss–Seidel as LP filter

Gauss–Seidel has good damping of all frequencies except low frequencies which are only weakly suppressed.

High frequency damping is better than 0.5, meaning that high frequency modes decay faster than $0.5^m$.

Thus Gauss–Seidel is smoothing (LP filter) – the error becomes a smoother function as the iterations pass.

As Jacobi does not damp high frequencies it is not a smoother; it blocks a narrow band of mid-frequencies.
Gauss–Seidel iteration demonstration

Error vector sequence (50 iterations) at $N = 19$