Contents

- Elliptic problems with FDM and FEM
- Classical iterative methods for sparse linear systems
- Power iteration
- Eigenvectors of $P_J$
- The Laplacian and the 5-point operator
- Speeding up convergence
- Successive over-relaxation (SOR)
- Why classical iterative methods are hopeless
- Conclusions
Consider simplest linear 2p-BVP (1D Poisson)

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

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

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

**Discretization**

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

\[ u_0 = \alpha; \quad u_{N+1} = \beta \]
Linear system of equations

Tridiagonal $N \times N$ matrix formulation

\[
\frac{1}{\Delta x^2} \begin{pmatrix}
-2 & 1 & & & \\
1 & -2 & 1 & & \\
& 1 & -2 & \ddots & \\
& & \ddots & \ddots & 1 \\
& & & 1 & -2
\end{pmatrix} \begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_N
\end{pmatrix} = \begin{pmatrix}
f(x_1) - \alpha/\Delta x^2 \\
f(x_2) \\
\vdots \\
f(x_N) - \beta/\Delta x^2
\end{pmatrix}
\]

In matrix–vector form

\[ T_{\Delta x} u = f \]

Minor modifications with Neumann or Robin boundary condition
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

\[ \Delta x = \Delta y \]

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

Computational stencil

"Five-point operator"
The FDM linear system of equations

Lexicographic ordering of unknowns ⇒ partitioned system

\[
\begin{pmatrix}
\frac{1}{\Delta x^2} & T & 0 & \cdots & 0 \\
0 & 1 & -4 & 1 & 0 \\
\vdots & 0 & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & 1 & -4 & 1
\end{pmatrix}
\begin{pmatrix}
u.,1 \\
u.,2 \\
u.,3 \\
\vdots \\
u.,N
\end{pmatrix}
=
\begin{pmatrix}
f(x., y_1) \\
f(x., y_2) \\
f(x., y_3) \\
\vdots \\
f(x., y_N)
\end{pmatrix}
\]

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

Partitioned system

\[ \frac{1}{\Delta x^2} \begin{pmatrix} T & I & 0 & \cdots & I \\
I & T & I & \cdots & I \\
I & I & T & I & \ddots \\
I & \cdots & I & \ddots & I \\
\cdots & \cdots & \cdots & \cdots & 0 & I & T \end{pmatrix} \begin{pmatrix} u_{1,1,1} \\
\vdots \\
\vdots \\
\vdots \\
u_{N,N,N} \end{pmatrix} = \begin{pmatrix} f_{1,1,1} \\
\vdots \\
\vdots \\
\vdots \\
f_{N,N,N} \end{pmatrix} \]

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

The system is \( N^3 \times N^3 \), hence \textit{extremely large and sparse}.
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** \( Kc = F \), where 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 \( F_i = \int \varphi_i f = \langle \varphi_i, f \rangle = \sum f_j \langle \varphi_i, \varphi_j \rangle \)
Finite element mesh

Domain triangulation

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

Same number of nodes as in FDM
2. Classical iterative methods for linear systems

All discretizations lead to very large linear systems

\[ Tu = f \]

typically having *many million equations*, even billions... 

Matrix *factorization methods are out of the question* – use iterative methods instead!

Iterative methods are often derived from splitting \( T = M - N \) and \( Mu^{m+1} = Nu^m + f \), where “inverting” \( M \) is inexpensive, producing explicit iterative method \( u^{m+1} = Bu^m + c \)
What are multigrid methods?

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

There is a relation to Fourier analysis as 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.
Given a linear system $Au = f$ construct sequence $u^m \rightarrow u$. 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 equation* $Ae^m = r^m$
Classical iterative methods . . .

There are four classical iterative methods

1. The \textit{Jacobi} method
2. The \textit{Gauss–Seidel} method
3. The \textit{Successive Overrelaxation} (SOR) method
4. The \textit{Symmetric SOR} (SSOR) method

Advanced methods include the \textit{Conjugate Gradient} (CG) method; the \textit{Generalized Minimum Residual} (GMRES) method; and various forms of \textit{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

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

\[ P_J = D^{-1}(L + U) \]
1D Poisson + Jacobi method

If $A = T_{\Delta x}$ then

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

and

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

For the exact solution

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

This gives the error recursion $e^{m+1} = P_Je^m$
1D Poisson + Jacobi method

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** \( \|P_J\|_2 < 1 \)

**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 symmetric tridiagonal Toeplitz matrices

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

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

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

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

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

with boundary conditions $u_0 = 0; u_{N+1} = 0$

Linear difference equation with characteristic equation

$$z^2 - \lambda z + 1 = 0$$

Roots are $z$ and $1/z$ (product 1), so the general solution is

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

Boundary condition $u_0 = 0$ implies $u_n = z^n - z^{-n}$
Eigenvalues of $P_J$

Boundary conditions imply $z^{2(N+1)} = 1$ so the roots are

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

As the sum of the roots are $\lambda_k = z_k + 1/z_k$ we have

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

**Theorem**  The eigenvalues of $P_J = S/2$ are

$$\lambda_k[P_J] = \cos \frac{k\pi}{N + 1} \in (-1, 1)$$
Eigenvalues get very close to $\pm 1$ for large $N$ for $N = 31$. The following graph illustrates the behavior of the eigenvalues of the Jacobi iteration matrix as a function of the frequency $k\pi/(N+1)$. The points on the graph show the eigenvalues, which approach 0 as $k\pi/(N+1)$ increases.
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, . . .

With \( \lambda_k[P_J] = \cos \frac{k\pi}{(N+1)} \) 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 \( N = 99 \) \( \Rightarrow \rho[P_J] \approx 0.9995 \)

Note As \( P_J \) is symmetric, \( \|P_J\|_2 = \rho[P_J] \)
... slowest!

And with a Neumann boundary condition it gets even worse!

For \( u'' = f \) with \( u'(0) = u(1) = 0 \) and lowest eigenvalue \(-\pi^2/4\), the largest eigenvalue of the Jacobi matrix becomes

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

So Jacobi’s method will converge, but...

convergence is 4 times slower than with Dirichlet condition!
Convergence history

Plot of $L^2$ error as a function of iteration number

- Iteration number: $0$, $1000$, $2000$, $3000$, $4000$, $5000$, $6000$, $7000$, $8000$, $9000$, $10000$
- $L^2$ Error norm: $10^{-3}$, $10^{-2}$, $10^{-1}$, $10^0$

$N = 99$

© G Söderlind 2015–2016  FMNN15 Multigrid V3.15  Classical Iterative Methods
Unit error reduction requires $O(N^2)$ iterations

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

Convergence history: N=15, 31, 63, 127

Iteration number

$L^2$ Error norm
3. Power iteration

The fixed point iteration \( e^{m+1} = A e^m \) is also known as power iteration as it implies \( e^m = A^m e^0 \)

Assume distinct eigenvalues \( A v_k = \lambda_k v_k \) with \( |\lambda_k| < |\lambda_1| \), for \( k = 2, \ldots, N \). Let \( e^0 = \sum \alpha_k v_k \). Then

\[
\frac{e^m}{\lambda_1^m} = \alpha_1 v_1 + \sum_{k=2}^{N} \alpha_k v_k \left( \frac{\lambda_k}{\lambda_1} \right)^m \to \alpha_1 v_1
\]

So the vector \( e^m \) gets aligned with dominant eigenvector \( v_1 \)
Why study power iteration?

The error recursion $e^{m+1} = P_J e^m$ is a power iteration with 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(s) corresponding to $\rho[P_J]$
What is the convergence to \( \max |\lambda| \) like?

Non-normal matrices  \( \text{linear convergence} \)
Normal matrices  \( \text{quadratic convergence} \)

Example

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

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

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

Normal differential operator $\Rightarrow$ *normal matrix*

\[
\begin{align*}
  u_t &= u_{xx} \\
  u_t &= \nabla \cdot (p \nabla u) \\
  u_t + u_x &= 0 \\
  \rho_t + \nabla \cdot (\rho \mathbf{v}) &= 0
\end{align*}
\]

Non-normal differential operators

\[
  u_t = u_x + \frac{1}{\text{Pe}} u_{xx}
\]

*Reduced performance on non-selfadjoint problems*
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

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

Same eigenvectors as for $T_{\Delta x}$

With grid points $x_n = n/(N + 1)$ the $k^{\text{th}}$ mode is $u^n_k = \sin k\pi x_n$

Same eigenfunctions as for $d^2/dx^2$
Eigenvectors of $P_J$
Eigenfunctions of $d^2/dx^2$

$N = 31$

- **Lowest mode**: A smooth, decreasing curve from left to right, peaking at the center.
- **Highest mode**: A series of peaks and troughs, oscillating violently, indicating a higher frequency pattern.

Classical Iterative Methods
Jacobi iteration demonstration

$N = 19$

Error vector sequence (50 iterations)

Error reduction in 50 iterations

© G Söderlind 2015–2016

Classical Iterative Methods
5. The Laplacian and the 5-point 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 boundary conditions \( u(x, y) = 0 \)

**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

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

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

Analogous modulation result in 3D
The 5-point finite difference Laplacian

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

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

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

**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 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. Speeding up convergence

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

**Damped Jacobi (under-relaxation)** Use a damping factor (or “step size”) \( \gamma \) and iterate according to

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

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

Damped Jacobi is equivalent to

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

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

\[ P_J(\gamma) = (1 - \gamma)I + \gamma P_J \]

\textbf{Theorem} \hspace{1em} \textit{The eigenvalues of} \hspace{0.5em} P_J(\gamma) \hspace{0.5em} \textit{are}

\[ \lambda_k[P_J(\gamma)] = 1 - \gamma + \gamma \cos \frac{k\pi}{N + 1} \]
Eigenvalue locations of damped Jacobi matrix

Damping parameter $\gamma = 1, \frac{2}{3}, \frac{1}{2}, \frac{1}{3}$

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

Largest eigenvalue becomes

$$\lambda_1[P_J(\gamma)] = 1 - \gamma + \gamma \cos \frac{\pi}{N + 1} = 1 - 2\gamma \sin^2 \frac{\pi \Delta x}{2}$$

So

$$\lambda_1[P_J(\gamma)] \approx 1 - \frac{\gamma \pi^2 \Delta x^2}{2}$$

(Ouch, this is even closer to 1 for $\gamma < 1$)
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

Practical implementation

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

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

The scheme implies that each scalar equation is solved in sequence. It cannot easily be parallelized.

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 $Uu = \lambda(D - L)u$, and according to Briggs:

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

but this result is incorrect
Gauss–Seidel eigenvalues and eigenvectors

\[ \frac{N}{2} \] eigenvalues of \( P_{GS} \) 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

Gauss–Seidel is twice as fast as Jacobi – or half as slow

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

Therefore we need other ways of studying convergence
Study error recursion

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

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

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

Plot frequency response vs frequency \( \theta_k = k\pi \Delta x \in (0, \pi) \) or as a function of wave number \( k = 1 : N \) (Bode diagram)
Bode diagram

Linear attenuation vs frequency: **Gauss–Seidel** and **Jacobi**

© G Söderlind 2015–2016

FMNN15 Multigrid V3.15
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. With “damped Jacobi” this can be changed into a smoothing iteration.
Gauss–Seidel iteration demonstration

$N = 19$

Error vector sequence (50 iterations)
An attempt to speed up convergence in Gauss–Seidel

\[ Du = Du - \gamma (Au - f) \]

\[ Du = Du - \gamma (D - L - U)u + \gamma f \]

\[ (D - \gamma L)u = (\gamma U + (1 - \gamma)D)u + \gamma f \]

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

**Iteration matrix** \[ P_{SOR} = (D - \gamma L)^{-1}((1 - \gamma)D + \gamma U) \]

**Acceleration parameter** \[ \gamma = 1 \Rightarrow \text{Gauss–Seidel method} \]
Spectral radius of SOR matrix for $T_{\Delta x}$ $N = 31, 63, 127$
Faster convergence is possible!

For 1D Poisson, the spectral radius is minimized at

\[ \hat{\gamma} = \frac{2}{1 + \sqrt{1 - \rho^2[P_J]}} \]

As \( \rho[P_J] = \cos \pi \Delta x \approx 1 - \frac{\pi^2 \Delta x^2}{2} \) we have

\[ \hat{\gamma} \approx 2 - 2\pi \Delta x \]

At this value the spectral radius of the SOR matrix is

\[ \rho[P_{SOR}(\hat{\gamma})] = \hat{\gamma} - 1 \approx 1 - 2\pi \Delta x \]

Much improved over Jacobi and Gauss–Seidel
Run error “transportation” demo live!
**Note**  Unit error reduction requires *only* $O(N)$ iterations compared to $O(N^2)$ for Gauss–Seidel (5$N$ iterations used)
Symmetric SOR (SSOR)

Neutralize “error transport” to the left by alternating between forward and backward sweeps

\[(D - \gamma L)u^{m+1} = (\gamma U + (1 - \gamma)D)u^m + \gamma f\]
\[(D - \gamma U)u^{m+2} = (\gamma L + (1 - \gamma)D)u^{m+1} + \gamma f\]

**Iteration matrix**

\[P_{SORfwd} = (D - \gamma L)^{-1}(\gamma U + (1 - \gamma)D)\]
\[P_{SORbwd} = (D - \gamma U)^{-1}(\gamma L + (1 - \gamma)D)\]

\[P_{SSOR} = P_{SORbwd}P_{SORfwd}\]
SOR and SSOR spectral radii vs $\gamma$

Optimal acceleration parameter $\hat{\gamma}$ does not move
8. Why classical methods are hopeless

For the damped Jacobi method the error recursion is

\[ e^{m+1} = P_J(\gamma) e^m \]
\[ P_J(\gamma) = (1 - \gamma) I + \gamma P_J \]

Consider the diffusion equation \( u_t = u_{xx} \) with homogeneous boundary conditions and 2nd order MOL semidiscretization

\[ \dot{u} = T_{\Delta x} u \]

Use explicit Euler time-stepping to get

\[ u^{m+1} = u^m + \Delta t \cdot T_{\Delta x} u^m \]
The diffusion equation

Introduce the *Courant number* $\mu = \Delta t / \Delta x^2$ and the Toeplitz matrix

$$T = \begin{pmatrix}
-2 & 1 & 0 \\
1 & -2 & 1 \\
& & \ddots \\
0 & 1 & -2
\end{pmatrix}$$

The Explicit Euler recursion becomes

$$u^{m+1} = u^m + \mu \cdot Tu^m$$
Diffusion equation . . .

Split the matrix \( T = D - L - U = -2I + S \) and note that

\[
S = \begin{pmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
& \ddots & \ddots \\
0 & 1 & 0
\end{pmatrix} = 2P_J
\]

So the recursion can be written

\[
u^{m+1} = u^m + \mu \cdot Tu^m = [(1 - 2\mu)I + 2\mu P_J] u^m = P_J(2\mu)u^m
\]
Error diffusion

Solving diffusion equation $u_t = u_{xx}$ using Explicit Euler time stepping with finite difference MOL produces

$$u^{m+1} = P_J(2\mu)u^m$$

Solving 2pBVP $T_{\Delta x}u = f$ using damped Jacobi iteration produces the error recursion

$$e^{m+1} = P_J(\gamma)e^m$$

The recursions are \textit{identical}, with the interpretation $2\mu = \gamma$

\textit{Errors in damped Jacobi “diffuse” according to} $e_t = e_{xx}/2$
Discrete diffusion

Explicit Euler for diffusion is extremely slow (stiffness)

CFL condition \(0 < \mu \leq 1/2\)

The same thing will happen to Jacobi iterations

Convergence if \(0 < \gamma \leq 1\)

After many steps/iterations \(not on the CLF limit\), the solution will approach the low frequency mode \(\sin \pi x\), while \(high frequencies\) are efficiently damped

What is a high frequency is a grid property
The dynamic/static connection

If we want to solve the (dynamic) diffusion problem

\[ u_t = u_{xx} - f(x) \]

its (static) stationary solution solves the 1D Poisson problem

\[ u'' = f \]

When we try to solve the latter, every classical iterative method introduces (discrete) pseudo–dynamics

\[ u^{m+1} = u^m + \gamma M^{-1} r^m \]

Because \( r \) depends on \( u \), this is the Explicit Euler method for \( \dot{u} = M^{-1} r(u) \) with step size \( \gamma \) and residual \( r(u) = u'' - f \)
Iterative methods for $Au = f$ produce

$$u^{m+1} = u^m + M^{-1}(Au^m - f)$$

Because in boundary value problems $A$ represents an unbounded operator, the scheme is equivalent to explicit (pseudo) time stepping in a stiff problem.

**Remedy** Counteract stiffness using inexpensive preconditioner $M^{-1} \approx A^{-1}$

**Note** Taking $M = \text{diag}(A)$ gives Jacobi, while $M = A$ produces Newton’s method – direct method, but unaffordable.
Classical iterative methods for elliptic problems have a convergence rate that depends on the mesh width $\Delta x$.

Can fast iterative methods be constructed to achieve a convergence rate independent of $\Delta x$?

This is the objective of multigrid methods.