

# Scientific Computing II

## Relaxation Methods and the Smoothing Property

Michael Bader  
Technical University of Munich

Summer 2018



*TUM Uhrenturm*

# Part I

## Relaxation Methods

### Residual-Based Correction

The Residual Equation

### Relaxation

Jacobi Relaxation

Gauss-Seidel Relaxation

Successive-Over-Relaxation (SOR)

# The Residual Equation

- we consider a system of linear equations:  $Ax = b$
- for which we compute a sequence of approximate solutions  $x^{(i)}$
- the **residual**  $r^{(i)}$  shall then be defined as:

$$r^{(i)} = b - Ax^{(i)}$$

- short computation:

$$r^{(i)} = b - Ax^{(i)} = Ax - Ax^{(i)} = A(x - x^{(i)}) = Ae^{(i)}.$$

- relates the residual  $r^{(i)}$  to the **error**  $e^{(i)} := x - x^{(i)}$  (note that  $x = x^{(i)} + e^{(i)}$ );
- we will call this equation the **residual equation**:

$$Ae^{(i)} = r^{(i)}$$

# Residual Based Correction

Solve  $Ax = b$  using the residual equation  $Ae^{(i)} = r^{(i)}$

- the residual  $r$  (which can be computed) is an indicator for the size of the error  $e$  (which is not known).
- therefore: use residual equation to compute a *correction* to  $x^{(i)}$
- one possible approach: solve a modified (easier) SLE

$$B \hat{e}^{(i)} = r^{(i)} \quad \text{where} \quad B \sim A$$

- use  $\hat{e}^{(i)}$  as an approximation for  $e^{(i)}$ , and set

$$x^{(i+1)} = x^{(i)} + \hat{e}^{(i)}$$

for the next iteration

# Relaxation

## How should we choose $B$ ?

- $B$  should be “similar” to  $A$  ( $B \sim A$ )
- more precisely  $B^{-1} \approx A^{-1}$
- or at least  $B^{-1}y \approx A^{-1}y$  for most vectors  $y$
- $Be = r$  should be easy/fast to solve

## Examples:

- $B = \text{diag}(A) = D_A$  (diagonal part of  $A$ )  
⇒ Jacobi method (“Jacobi relaxation”)
- $B = L_A$  (lower triangular part of  $A$ )  
⇒ Gauss-Seidel method (“Gauss-Seidel relaxation”)

# Jacobi Relaxation

Iteration formulas in matrix-vector notation:

1. residual notation:

$$x^{(i+1)} = x^{(i)} + D_A^{-1} r^{(i)} = x^{(i)} + D_A^{-1} (b - Ax^{(i)})$$

2. for implementation:

$$x^{(i+1)} = D_A^{-1} (b - (A - D_A)x^{(i)})$$

3. for analysis:

$$x^{(i+1)} = (I - D_A^{-1}A) x^{(i)} + D_A^{-1}b =: Mx^{(i)} + Nb$$

# Jacobi Relaxation – Algorithm

- based on:  $x^{(i+1)} = D_A^{-1} (b - (A - D_A)x^{(i)})$

```
for i from 1 to n do
    xnew[i] := ( b[i]
                - sum( A[i,j]*x[j], j=1..i-1)
                - sum( A[i,j]*x[j], j=i+1..n)
                ) / A[i,i];
end do;
for i from 1 to n do
    x[i] := xnew[i];
end do;
```

- **properties:**
  - additional storage required (xnew)
  - x, xnew can be computed **in any order**
  - x, xnew can be computed **in parallel**

# Gauss-Seidel Relaxation

Iteration formulas in matrix-vector notation:

1. residual notation:

$$x^{(i+1)} = x^{(i)} + L_A^{-1} r^{(i)} = x^{(i)} + L_A^{-1} (b - Ax^{(i)})$$

2. for implementation:

$$x^{(i+1)} = L_A^{-1} (b - (A - L_A)x^{(i)})$$

3. for analysis:

$$x^{(i+1)} = (I - L_A^{-1}A) x^{(i)} + L_A^{-1}b =: Mx^{(i)} + Nb$$



## Gauss-Seidel Relaxation – Algorithm

- based on:  $x^{(i+1)} = L_A^{-1} (b - (A - L_A)x^{(i)})$
- solve  $L_A x^{(i+1)} = b - (A - L_A)x^{(i)}$   
via backwards substitution:

```

for i from 1 to n do
  x[i] := ( b[i]
            - sum( A[i,j]*x[j], j=1..i-1)    !updated values of x
            - sum( A[i,j]*x[j], j=i+1..n)    !previous values of x
          ) / A[i,i];
end do;

```

- **properties:**
  - no additional storage required
  - inherently **sequential** computation of  $x$
  - usually faster convergence than Jacobi

# Successive-Over-Relaxation (SOR)

- observation: Gauss-Seidel corrections are “too small”
- add an over-relaxation-factor  $\alpha$ :

```
for i from 1 to n do
  x[i] := x[i] + alpha * ( b[i]
    - sum( A[i,j]*x[j], j=1..n)
  ) / A[i,i];
end do;
```

- for 2D Poisson model problem:  
optimal  $\alpha$  ( $\approx 1.7$ ) improves convergence:  $\mathcal{O}(n^2) \rightarrow \mathcal{O}(n^{3/2})$

# Does It Always Work?

- simple answer: no (life is not that easy ...)
- Jacobi: matrix  $A$  needs to be *diagonally dominant*
- Gauss-Seidel: matrix  $A$  needs to be *positive definite*
- How about performance?  
→ usually quite slow

## Our next topics:

1. How slow are the methods exactly?
2. What is the underlying reason?
3. Is there a fix?

## Part II

# Smoothing Property of Relaxation Methods

The Model Problem – 1D Poisson  
Convergence of Relaxation Methods  
The Smoothing Property

# The Model Problem – 1D Poisson

## 1D Poisson equation:

- $-u''(x) = 0$  on  $\Omega = (0, 1)$ ,  $u(0) = u(1) = 0$
- thus:  $u(x) = 0 \rightsquigarrow$  boring, but easy to examine the error
- discretised on a uniform grid of mesh size  $h = \frac{1}{n}$
- compute approximate values  $u_j \approx u(x_j)$   
at grid points  $x_j := jh$ , with  $j = 1, \dots, (n-1)$
- system matrix  $A_h$  built from 3-point stencil:

$$\frac{1}{h^2} [-1 \quad 2 \quad -1]$$

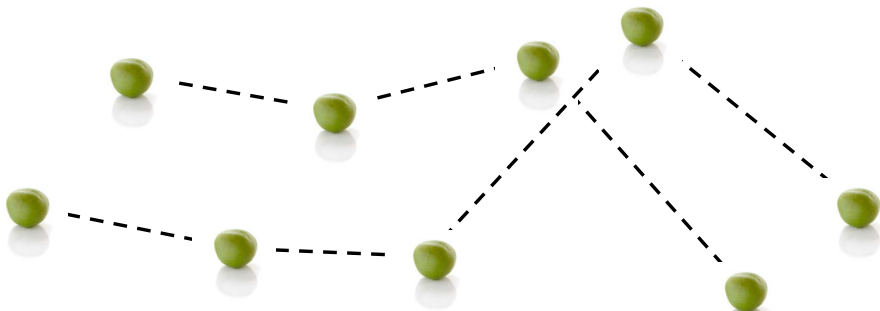
- $A_h$  a tridiagonal  $(n-1) \times (n-1)$ -matrix

# 1D Poisson: Jacobi Relaxation

## Iterative scheme for Jacobi relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

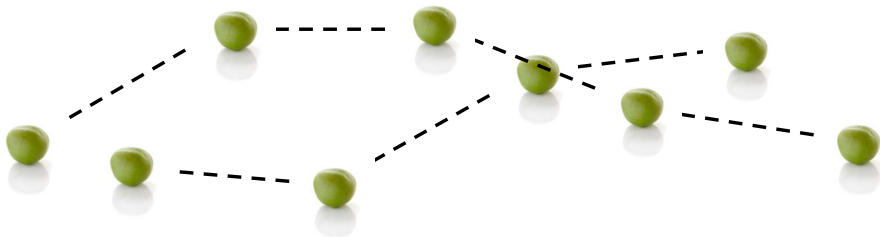


# 1D Poisson: Jacobi Relaxation

## Iterative scheme for Jacobi relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

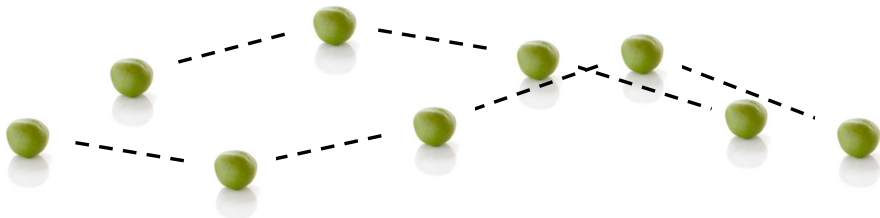


# 1D Poisson: Jacobi Relaxation

## Iterative scheme for Jacobi relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:



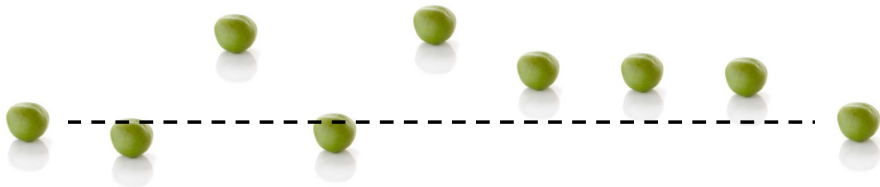


# 1D Poisson: Jacobi Relaxation

## Iterative scheme for Jacobi relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

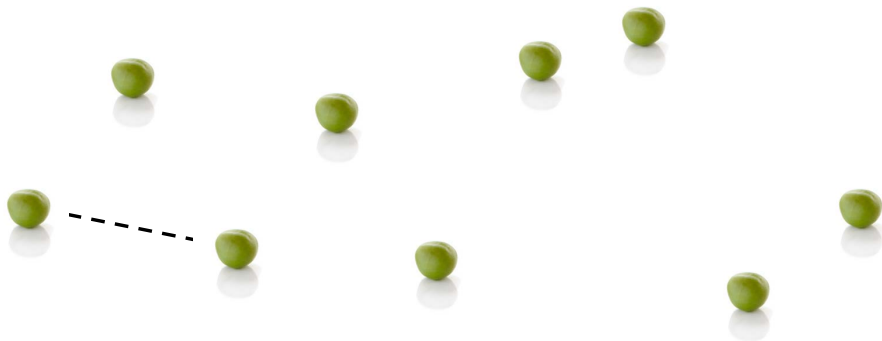


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} \left( u_{j+1}^{(i)} + u_{j-1}^{(i+1)} \right)$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

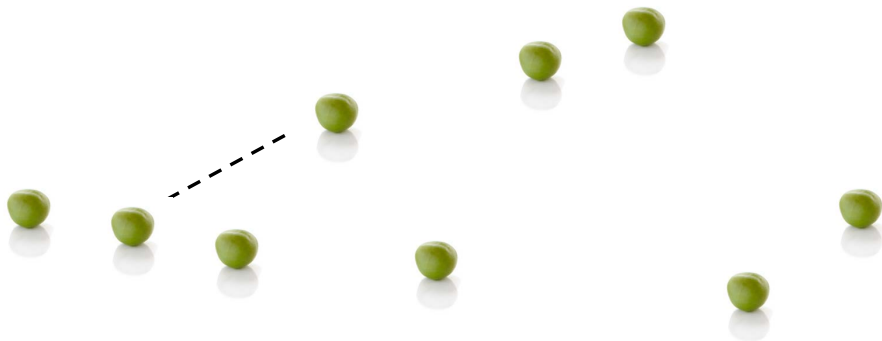


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} \left( u_{j+1}^{(i)} + u_{j-1}^{(i+1)} \right)$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

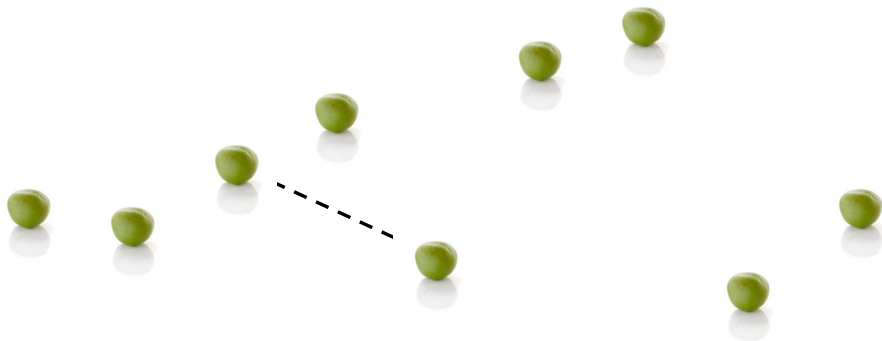


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

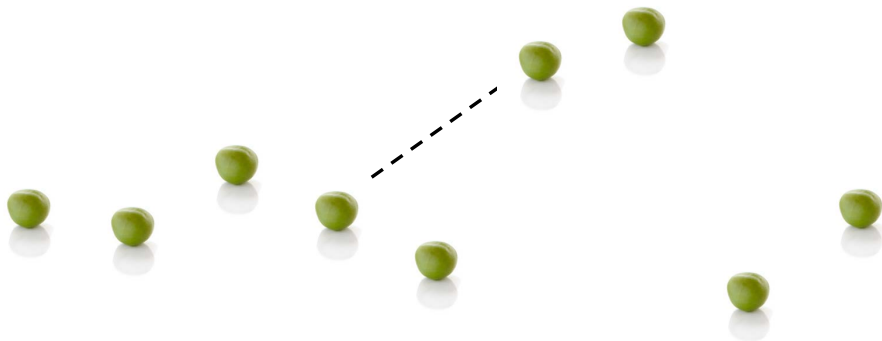


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

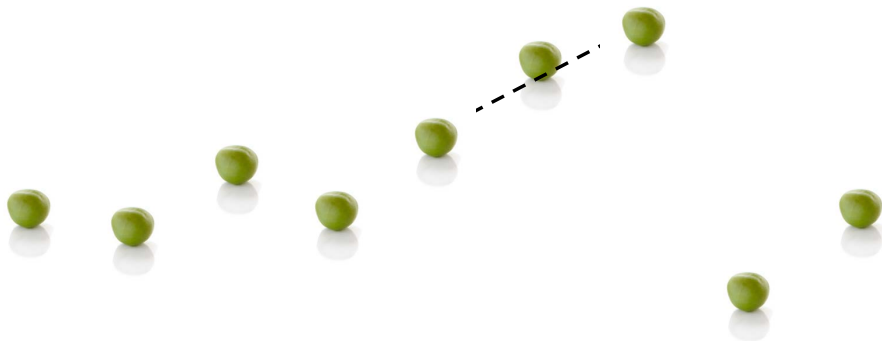


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

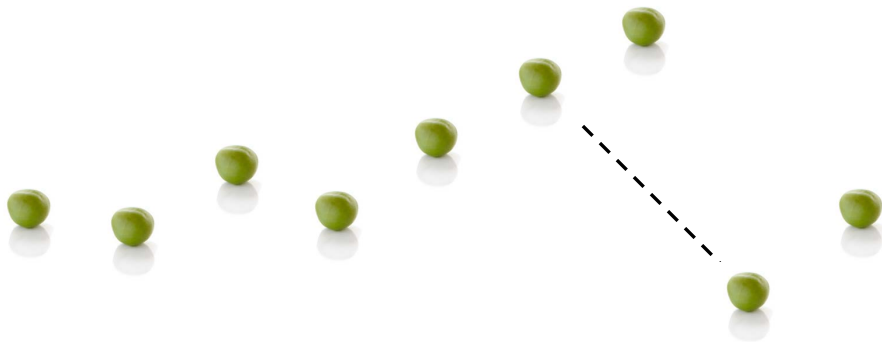


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:



# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:





# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:



# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:

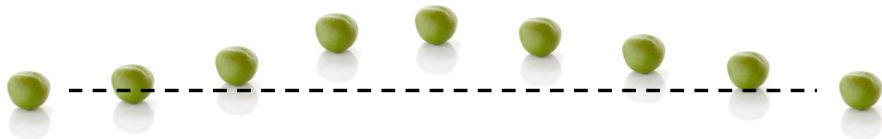


# 1D Poisson: Gauss-Seidel Relaxation

## Iterative scheme for Gauss-Seidel relaxation:

- leads to relaxation scheme  $u_j^{(i+1)} = \frac{1}{2} (u_{j+1}^{(i)} + u_{j-1}^{(i+1)})$
- start with initial guess  $u_j^{(0)} \neq 0$
- in this case:  $e_j^{(i)} = u_j - u_j^{(i)} = -u_j^{(i)}$

## Visualisation of relaxation process:



# Convergence of Relaxation Methods

**Observation** (see also tutorials)

- slow convergence
- smooth error components are reduced very slowly
- high frequency error components are damped more efficiently (esp. for Gauss-Seidel relaxation)

# Convergence Analysis

- remember iteration scheme:  $x^{(i+1)} = Mx^{(i)} + Nb$
- derive iterative scheme for the error  $e^{(i)} := x - x^{(i)}$ :

$$e^{(i+1)} = x - x^{(i+1)} = x - Mx^{(i)} - Nb$$

- for a **consistent** scheme,  $x$  is a fixpoint of the iteration equation:  
 $x = Mx + Nb$
- hence:

$$\begin{aligned} e^{(i+1)} &= Mx + Nb - Mx^{(i)} - Nb \\ &= Mx - Mx^{(i)} = Me^{(i)} \\ \Rightarrow e^{(i)} &= M^i e^{(0)}. \end{aligned}$$

## Convergence Analysis (2)

- iteration equation for error:  $e^{(i)} = M^i e^{(0)}$
- consider eigenvalues  $\mu_k$  and eigenvectors  $v_k$  of iteration matrix  $M$ :

$$Mv_k = \mu_k v_k \quad \Rightarrow \quad M\left(\sum_k \alpha_k v_k\right) = \sum_k \alpha_k Mv_k = \sum_k \mu_k \alpha_k v_k$$

- write error as combination of eigenvectors:  $e^{(0)} = \sum_k \alpha_k v_k$ , then:

$$M^i e^{(0)} = M^i \left(\sum_k \alpha_k v_k\right) = \sum_k (\mu_k)^i \alpha_k v_k$$

- convergence, if all  $|\mu_k| < 1$
- speed of convergence dominated by largest  $\mu_k$

# The Smoothing Property

## Eigenvalues and -vectors of $A_h$ : (compare with tutorials!)

- eigenvalues:  $\lambda_k = \frac{4}{h^2} \sin^2\left(\frac{k\pi}{2n}\right) = \frac{4}{h^2} \sin^2\left(\frac{k\pi h}{2}\right)$
- eigenvectors:  $v^{(k)} = (\sin(k\pi j/n))_{j=1, \dots, n-1}$
- both for  $k = 1, \dots, (n-1)$

## For Jacobi relaxation:

- iteration matrix  $M = I - D_A^{-1}A = I - \frac{h^2}{2}A$
- eigenvalues of  $M$ :  $\mu_k := 1 - 2 \sin^2\left(\frac{k\pi h}{2}\right)$
- $|\mu_k| < 1$  for all  $k$ , but  $|\mu_k| \approx 1$  if  $k = 1$  or  $k = n-1$
- $\mu_1 \in \mathcal{O}(1 - h^2)$ : slow convergence of smooth errors
- $\mu_{n-1} \approx -1$ : “sign-flip” (but slow reduction) of “zig-zag” error components
- convergence factor determined by  $\mathcal{O}(1 - h^2)$

# The Smoothing Property

**Eigenvalues and -vectors of  $A_h$ :** (compare with tutorials!)

- eigenvalues:  $\lambda_k = \frac{4}{h^2} \sin^2\left(\frac{k\pi}{2n}\right) = \frac{4}{h^2} \sin^2\left(\frac{k\pi h}{2}\right)$
- eigenvectors:  $v^{(k)} = (\sin(k\pi j/n))_{j=1, \dots, n-1}$
- both for  $k = 1, \dots, (n-1)$

**For weighted Jacobi relaxation:**

- iteration matrix  $M = I - \omega D_A^{-1} A = I - \frac{h^2}{2} \omega A$
- eigenvalues of  $M$ :  $1 - 2\omega \sin^2\left(\frac{k\pi h}{2}\right)$
- $\mu_1 \in \mathcal{O}(1 - h^2)$ : slow convergence of smooth errors
- $\mu_{n-1} \approx 0$  for  $\omega = \frac{1}{2}$ ;  $\mu_{n-1} \approx -\frac{1}{3}$  for  $\omega = \frac{2}{3}$   
thus quick reduction of high-frequency errors
- convergence determined by  $\mathcal{O}(1 - n^{-2})$   
(slower than normal Jacobi due to  $\omega$ )



## The Smoothing Property (2)

### “Fourier mode analysis”

- decompose the error  $e^{(i)}$  into eigenvectors  $\rightarrow$  for 1D Poisson:  $\sin(k\pi x_j)$
- determine convergence factors for “eigenmodes”

### Observation for weighted Jacobi and Gauss-Seidel:

- The *high* frequency part (with respect to the underlying grid) is reduced quite quickly.
- The *low* frequency part (w.r.t. the grid) decreases only very slowly; actually the slower, the finer the grid is.

$\Rightarrow$  “smoothing property”

## The Smoothing Property (2)

### “Fourier mode analysis”

- decompose the error  $e^{(i)}$  into eigenvectors  $\rightarrow$  for 1D Poisson:  $\sin(k\pi x_j)$
- determine convergence factors for “eigenmodes”

### Another Observation:

- the smoothest (slowest converging) component corresponds to the smallest eigenvalue of  $A$  ( $k = 1$ )
- remember residual equation:  $Ae = r$ :  
if  $e = v^{(1)}$ , then  $r = \lambda_1 v^{(1)}$

$\Rightarrow$  “small residual, but large error”

$\Rightarrow$  in such a situation, any residual-based correction will normally fail