# FEM and Sparse Linear System Solving

## Lecture 7, Nov 3, 2015: Introduction to Iterative Solvers: Stationary Methods

http://people.inf.ethz.ch/arbenz/FEM16

Peter Arbenz
Computer Science Department, ETH Zürich
E-mail: arbenz@inf.ethz.ch

# Survey on lecture

- The finite element method
- Direct solvers for sparse systems
- Iterative solvers for sparse systems
  - Stationary iterative methods, preconditioning
  - Preconditioned conjugate gradient method (PCG)
  - Krylov space methods
  - Incomplete factorization preconditioning
  - Multigrid preconditioning
  - Nonsymmetric problems (GMRES, BiCGstab, IDR($s$))
  - Indefinite problems (SYMMLQ, MINRES)

# Today's topic

1. Comparison of direct and iterative linear solvers
2. Stationary iterative methods: theory
3. Stationary iterative methods: practical schemes
4. Smoothing properties of Jacobi and Gauss–Seidel

- Today: introduce some 'classical' iterative methods.
- Reason: still important as *preconditioners*, *smoothers*.
- Next time(s): conjugate gradient method, general Krylov subspace methods.

# References


- G. Golub & C. van Loan: *Matrix Computations*, 3rd ed., pp. 508–519, Johns Hopkins, 1996.
- Y. Saad: *Iterative Methods for Sparse Linear Systems*, 2nd ed., pp. 103–128, SIAM, 2003 (online).
- O. Axelsson: *Iterative solution methods*. Chapter 5, Cambridge University Press, 1996.
- M. A. Olshanskii and E. E. Tyrtyshnikov *Iterative Methods for Linear Systems: Theory and Applications*. SIAM, 2014.
- V. Dolean, P. Jolivet, F. Nataf. *An Introduction to Domain Decomposition Methods.* SIAM, 2015.

# Comparison of direct and iterative linear solvers

*Direct solvers*

- Computation is numerically stable in many relevant cases.
- Can solve economically for several right-hand sides.
- Accuracy can be improved via 'iterative refinement.'
- 'Essentially' a black box.
- But: fill-in limits usefulness (memory, flops).

*Iterative solvers*

- Matrix often only *implicitly* needed via MatVec product.
- One might not care about exact solution of linear system.
- Good preconditioner often necessary for convergence.
- Quality often dependent on 'right' choice of parameters, e.g. start vector, basis size, restart (see later).

# Typical scenarios

*Direct solvers*

- Inverse Iteration
- Determinants
- Many linear systems with the same matrix $A$
- 'Difficult' applications (e.g. circuit simulation)

*Iterative solvers*

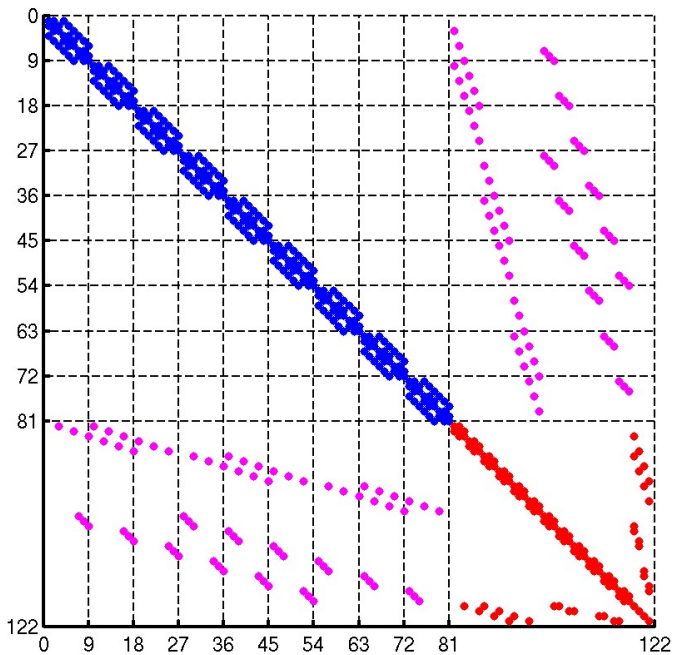
- Inexact Newton-Methods
- Many linear systems with 'slightly changing' matrices
- Matrix-free applications (e.g. MatVec product via FFT)
- Very large problems

# Cross-over & synergy

Direct solvers as preconditioners for iterative ones:

- Incomplete Cholesky
- Incomplete $LU$

Combination as *hybrid* direct-iterative methods, example:




Compute $[A_{kk}]^{-1} = U_k^{-1} L_k^{-1}$
but evaluate Schur complement

$$S = A_{10,10} - \sum A_{10,k} [A_{kk}]^{-1} A_{k,10}$$

iteratively instead of storing it
as dense matrix!

# Stationary iterative methods: motivation

Let $A$ be so large that it is impossible to compute its LU factorization (fill-in). Then, we try to solve $A\boldsymbol{x} = \boldsymbol{b}$ iteratively. Let $\boldsymbol{x}_k$ be an *approximation* to the solution $\boldsymbol{x}^*$ of $A\boldsymbol{x} = \boldsymbol{b}$. Then

$$\boldsymbol{x}^* = \boldsymbol{x}_k + \underbrace{\boldsymbol{e}_k}_{\text{error}}$$

$$A\boldsymbol{e}_k = A\boldsymbol{x}^* - A\boldsymbol{x}_k = \boldsymbol{b} - A\boldsymbol{x}_k =: \underbrace{\boldsymbol{r}_k}_{\text{residual}}$$

$$\boldsymbol{x}^* = \boldsymbol{x}_k + A^{-1}\boldsymbol{r}_k \tag{1}$$

Of course, the above assumption prevents us from solving directly $A\boldsymbol{e}_k = \boldsymbol{r}_k$ for the error $\boldsymbol{e}_k$.

# Stationary iterative methods: motivation (cont.)

Let $M$ be an invertible matrix with the properties

1. $M$ is a good approximation to $A$.
2. $M\boldsymbol{z} = \boldsymbol{r}$ can be solved (relatively) cheaply.
3. $M$ can be formed (relatively) cheaply.

Then, instead of solving (1) we *iterate* according to

$$\boxed{\begin{aligned} \boldsymbol{r}_k &= \boldsymbol{b} - A\boldsymbol{x}_k \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + M^{-1}\boldsymbol{r}_k \end{aligned}} \tag{2}$$

$M$ is called a *preconditioner*.

# Stationary iterative methods: motivation (cont.)

## Practical procedure:

$$
\begin{aligned}
\boldsymbol{r}_k &= \boldsymbol{b} - A\boldsymbol{x}_k && \text{(Compute residual)} \\
M\boldsymbol{z}_k &= \boldsymbol{r}_k && \text{(Solve with preconditioner)} \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{z}_k && \text{(Update approximate)}
\end{aligned}
$$

## Remarks:

- In step 1, we have to multiply a matrix with a vector. In practice: we need to have a procedure that computes $\boldsymbol{y} \leftarrow A\,\boldsymbol{x}$.
- In step 2, we have to solve a system of equations with the preconditioner which is "by definition" cheap to do.
  *Still, this step is usually the most expensive one.*
- $\boldsymbol{z}_k$ is called the preconditioned residual.

# Stationary iterative methods: Theory

## Definition (Matrix splitting)

Let $A$ be nonsingular. Then $A = M - N$, with $M$ *nonsingular*, is called a *matrix splitting*.

We consider the iteration

$$M\boldsymbol{x}_{k+1} = N\boldsymbol{x}_k + \boldsymbol{b} \;\Leftrightarrow\; \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + M^{-1}\boldsymbol{r}_k.$$

Note:

- If $M^{-1} = A^{-1}$, then one-step convergence: $\boldsymbol{x}_{k+1} = A^{-1}\boldsymbol{b}$.
- In practice, $M^{-1}$ should be a good (but cheap) approximation to $A^{-1}$ (*'preconditioner'*).

# Formulation as fixed-point iteration

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + M^{-1}\boldsymbol{r}_k = M^{-1}\left(M\boldsymbol{x}_k + \boldsymbol{r}_k\right) \\ &= M^{-1}\left((M-A)\boldsymbol{x}_k + \boldsymbol{b}\right) \\ &= \underbrace{M^{-1}N}_{=:\,G}\boldsymbol{x}_k + \underbrace{M^{-1}\boldsymbol{b}}_{=:\,\boldsymbol{c}}, \qquad N = M - A. \end{aligned}$$

$$\Rightarrow \boxed{\boldsymbol{x}_{k+1} = G\boldsymbol{x}_k + \boldsymbol{c}} \qquad \text{fixed point iteration}$$

$G = M^{-1}N = I - M^{-1}A$ is called the *iteration matrix*.

# Formulation as fixed-point iteration (cont.)

## Theorem

*The error satisfies* $\boldsymbol{e}_{k+1} = G\boldsymbol{e}_k$ *with* $G = I - M^{-1}A$.

*Proof:* With the fixed point $\boldsymbol{x}^* = A^{-1}\boldsymbol{b}$, one has $\boldsymbol{x}^* = G\boldsymbol{x}^* + \boldsymbol{c}$.
Subtract this from the fixed point iteration. □

A similar equation holds for the residuals.

$$\boldsymbol{r}_{k+1} = A\boldsymbol{e}_{k+1} = AG\boldsymbol{e}_k = AGA^{-1}A\boldsymbol{e}_k = AGA^{-1}\boldsymbol{r}_k.$$

This iteration matrix can be written as

$$A(I - M^{-1}A)A^{-1} = I - AM^{-1}AA^{-1} = I - AM^{-1}.$$

Note that the two iteration matrices are *similar*!
This means that they have the same eigenvalues.

# Some tools

Let $\|\cdot\|$ be *any vector norm* and

$$\|G\| := \max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|G\boldsymbol{x}\|}{\|\boldsymbol{x}\|}, \qquad \text{induced matrix norm,} \tag{3}$$

$$\rho(G) := \max_{\lambda\in\sigma(G)} |\lambda|, \qquad \text{spectral radius of } G, \tag{4}$$

where $\sigma(G)$ is the spectrum (set of eigenvalues) of $G$.

Lemma 1. $\rho(G) \leq \|G\|$

*Proof.* For all $\lambda \in \sigma(G)$ and associated eigenvector $\boldsymbol{x}$ we have

$$G\boldsymbol{x} = \lambda\boldsymbol{x} \Rightarrow \|\lambda\boldsymbol{x}\| = |\lambda| \cdot \|\boldsymbol{x}\| = \|G\boldsymbol{x}\| \leq \|G\| \cdot \|\boldsymbol{x}\|. \qquad \square$$

(In fact: $\rho(G) \leq |||G|||$ for any matrix norm $|||\cdot|||$.)

# Some tools (cont.)

Lemma 2. For any matrix $G$ and any $\varepsilon > 0$ there is a matrix norm such that

$$|||G||| \leq \rho(G) + \varepsilon.$$

*Proof.* (See e.g. Horn-Johnson, Matrix analysis, p. 297.)
Let $G = UDU^*$ be the Schur decomposition of $G$ with $UU^* = I$ and $D$ upper-triangular. The eigenvalues of $G$ appear on the diagonal of $D$

$$U^*GU = D = \begin{pmatrix} \lambda_1 & d_{12} & d_{13} & \cdots & d_{1n} \\ & \lambda_2 & d_{23} & \cdots & d_{2n} \\ & & \lambda_3 & \cdots & d_{3n} \\ & & & \ddots & \vdots \\ & & & & \lambda_n \end{pmatrix}$$

# Some tools (cont.)

Let $D_t = \mathrm{diag}(t, t^2, t^3, \cdots, t^n)$. Then,

$$D_t U^* G U D_t^{-1} = \begin{pmatrix} \lambda_1 & t^{-1}d_{12} & t^{-2}d_{13} & \cdots & t^{-n+1}d_{1n} \\ & \lambda_2 & t^{-1}d_{23} & \cdots & t^{-n+2}d_{2n} \\ & & \lambda_3 & \cdots & t^{-n+3}d_{3n} \\ & & & \ddots & \vdots \\ & & & & \lambda_n \end{pmatrix}.$$

For $t$ large enough, the off-diagonal elements are so small, that the sum of all off-diagonal elements of one column (or row) is smaller than $\varepsilon$.

# Some tools (cont.)

Define the matrix norm (which in fact is induced by the vector 1-norm)

$$|||A|||_1 := \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|.$$

Then, for large enough $t$, the norm

$$|||G||| := |||D_t U^* G U D_t^{-1}|||_1$$

is smaller than $\rho(G) + \varepsilon$. □

# Theorem on convergence

## Theorem (Convergence theorem for stationary iterations)

*Let $A = M - N$ be a matrix splitting with $M$ invertible. Then,*

**(1)** *The iteration*

$$M\boldsymbol{x}_{k+1} = N\boldsymbol{x}_k + \boldsymbol{b}, \qquad (+)$$

*converges for any $\boldsymbol{x}_0$ if and only if*

$$\rho(G) < 1, \qquad G = M^{-1}N = I - M^{-1}A.$$

**(2)** *If, for any vector norm, $\|G\| < 1$, then iteration $(+)$ converges.*

# Theorem on convergence (cont.)

*Proof.* Recall that if $(+)$ converges then $\lim \boldsymbol{e}_k = \mathbf{0}$ for any starting vector $\boldsymbol{x}_0$.

**(1)** "$\Longrightarrow$" Let $\boldsymbol{e}_0$ be an eigenvector of $G$ corresponding to the eigenvalue $\lambda$. Then,
$G\boldsymbol{e}_0 = \lambda \boldsymbol{e}_0 \Rightarrow \boldsymbol{e}_k = G^k \boldsymbol{e}_0 = \lambda^k \boldsymbol{e}_0 \Longrightarrow |\lambda| < 1$, since $\lim \boldsymbol{e}_k = \mathbf{0}$. As this holds for all $\lambda$ we have $\rho(G) < 1$.

"$\Longleftarrow$" If $\rho(G) < 1$ then by Lemma 2 there is a vector norm $\|\cdot\|$ such that $\|G\| < 1$.
$\Rightarrow \|\boldsymbol{e}_k\| = \|G^k \boldsymbol{e}_0\| \leq \|G^k\| \cdot \|\boldsymbol{e}_0\| \leq \|G\|^k \cdot \|\boldsymbol{e}_0\| \to 0$
$\Rightarrow \|\boldsymbol{e}_k\| \to 0$ for *any* $\boldsymbol{e}_0$. Thus we have convergence.

**(2)** trivial because of Lemma 1.

# Remarks

- The convergence theorem is based entirely on the eigenvalues of the iteration matrix $G$.
- There is however a big difference of the convergence behavior of normal (diagonalizable by a unitary matrix) and nonnormal matrices.
- Compare in MATLAB the behavior of the error norm of $\|\boldsymbol{e}_k\|$ ($\boldsymbol{e}_0 = [\sqrt{1/2}, \sqrt{1/2}]^T$) with the two matrices

$$\begin{pmatrix} 0.9 & 0 \\ 0 & 0.9 \end{pmatrix}, \qquad \begin{pmatrix} 0.9 & 10 \\ 0 & 0.9 \end{pmatrix}.$$

# Practicalities: Initial vector

## Starting vector, iterates & convergence

Improve *initial vector* $\boldsymbol{x}_0$ so that $\boldsymbol{x}_k \to \boldsymbol{x} = A^{-1}\boldsymbol{b}$ in fewer steps.

## Where to get $\boldsymbol{x}_0$ from?

There are various possibilities (some better than others):

- Zero vector.
- Random vector.
- Insights into underlying problem.
- Solution from a 'similar' previously solved problem.

## Note

- For nonsingular $A$, iteration should converge for *any* starting vector!
- But: better to make productive use of all information at hand!

# Practicalities: Stopping criterion

A few practical possibilities

- $\|\boldsymbol{r}_k\| \leq \tau \|\boldsymbol{b}\|$.
- $\|\boldsymbol{r}_k\| \leq \tau \left(\|A\| \|\boldsymbol{x}_k\| + \|\boldsymbol{b}\|\right)$.
- $\|\boldsymbol{r}_k\| \leq \tau \|\boldsymbol{r}_0\|$.

Note:

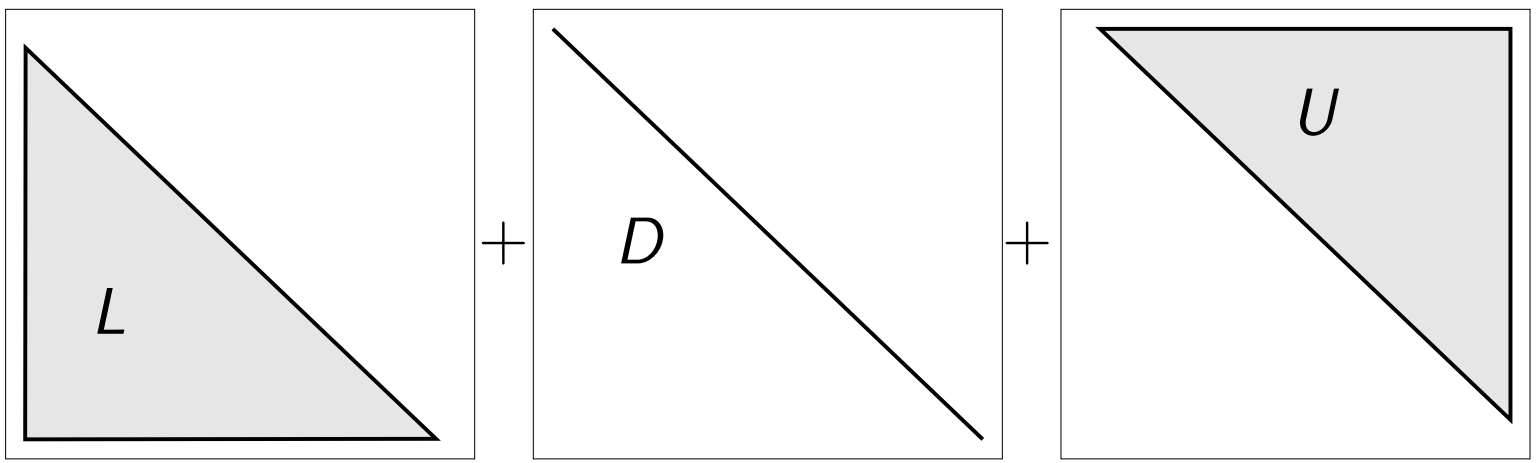
- $\|e_k\| \leq \|A^{-1}\| \, \|\boldsymbol{r}_k\|$, commonly estimate $\|A^{-1}\|$.
- All aforementioned criteria are scaling-invariant.
- Usually also set a maximum number of iterations.
- The criterion $\|\boldsymbol{r}_k\| \leq \tau \|\boldsymbol{r}_0\|$ can be misleading if $\boldsymbol{x}_0$ is very far from the true solution.

For more information, see: Chapter 4.2. of
Barrett et al.: *Templates for the Solution of Linear Systems*.

# Practical schemes: Jacobi iteration

Let $A = L + D + U$ where

- $D$ is diagonal,
- $L$ is strictly lower triangular, and
- $U$ is strictly upper triangular.



# Practical schemes: Jacobi iteration (cont.)

We set

$$M = D = \mathrm{diag}(a_{11}, a_{22}, \ldots, a_{nn}), \qquad N = -(L + U).$$

Thus,

$$\boldsymbol{x}_{k+1} = -D^{-1}(L + U)\boldsymbol{x}_k + D^{-1}\boldsymbol{b}.$$

Component-wise notation:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right) = x_i^{(k)} + \frac{1}{a_{ii}} \underbrace{\left( b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)} \right)}_{r_i^{(k)}}$$

# Practical schemes: Jacobi iteration (cont.)

## Theorem

*The Jacobi iteration converges if $A$ is row-wise strictly diagonally dominant.*

*Proof:* We show that $\|D^{-1}(L+U)\|_\infty < 1$. In fact,

$$\rho(M^{-1}N) \le \|D^{-1}(L+U)\|_\infty = \max_{1\le i\le n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1.$$

□

# Practical schemes: Gauss–Seidel iteration

Again: $A = L + D + U$.

Let

$$M = D + L, \qquad N = -U.$$

Thus,

$$(D + L)\boldsymbol{x}_{k+1} = -U\boldsymbol{x}_k + \boldsymbol{b}.$$

Component-wise notation:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right)$$

# Practical schemes: Gauss–Seidel iteration (cont.)

## Theorem (Convergence of Gauss–Seidel)

*The Gauss–Seidel iteration converges if $A$ is SPD.*

*Proof:* We show that the eigenvalues of the iteration matrix $G = -(D + L)^{-1} L^T$ are inside the unit disc.

Since $D$ is SPD we can define $D_1 = D^{1/2}$ and set $G_1 := D_1 G D_1^{-1} = -(I + L_1)^{-1} L_1^T$ with $L_1 = D_1^{-1} L D_1^{-1}$.

$G$ and $G_1$ are similar and thus have the same eigenvalues.

If $G_1 \boldsymbol{x} = \lambda \boldsymbol{x}$ with $\|\boldsymbol{x}\|_2 = 1$ then

$$-L_1^T \boldsymbol{x} = \lambda (I + L_1) \boldsymbol{x}.$$

and

$$-\boldsymbol{x}^* L_1^T \boldsymbol{x} = \lambda \boldsymbol{x}^* (I + L_1) \boldsymbol{x} = \lambda (1 + \boldsymbol{x}^* L_1 \boldsymbol{x}).$$

# Practical schemes: Gauss–Seidel iteration (cont.)

Note that the eigenvalue $\lambda$ and thus the eigenvector $\boldsymbol{x}$ can be complex.

Set $\boldsymbol{x}^* L_1 \boldsymbol{x} = a + bi$. Then

$$|\lambda|^2 = \left| \frac{-a + bi}{1 + a + bi} \right|^2 = \frac{a^2 + b^2}{1 + 2a + a^2 + b^2}.$$

We now show that $1 + 2a > 0$ from which we get that the numerator is smaller than the denominator.
Since $D^{-1/2} A D^{-1/2} = I + L_1 + L_1^T$ is spd we have

$$0 < 1 + \underbrace{\boldsymbol{x}^* L_1 \boldsymbol{x}}_{a+bi} + \underbrace{\boldsymbol{x}^* L_1^T \boldsymbol{x}}_{a-bi} = 1 + 2a.$$

# Practical schemes: Block iterations

Let

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mm} \end{bmatrix}$$

Block Jacobi iteration

$$M = \operatorname{diag}(A_{11}, A_{22}, \ldots, A_{mm})$$

Block Gauss–Seidel iteration

$$M = \begin{bmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & \vdots & \ddots & \\ A_{m1} & A_{m2} & \cdots & A_{mm} \end{bmatrix}$$

# Practical schemes: Successive Over-Relaxation (SOR)

Jacobi iteration: $$D\boldsymbol{x}_{k+1}^{J} = -L\boldsymbol{x}_{k}^{J} - U\boldsymbol{x}_{k}^{J} + \boldsymbol{b}.$$

Gauss–Seidel iteration: $$D\boldsymbol{x}_{k+1}^{GS} = -L\boldsymbol{x}_{k+1}^{GS} - U\boldsymbol{x}_{k}^{GS} + \boldsymbol{b}.$$

Successive Over-Relaxation (SOR) iteration:

$$D\boldsymbol{x}_{k+1}^{SOR} = \omega\left(-L\boldsymbol{x}_{k+1}^{SOR} - U\boldsymbol{x}_{k}^{SOR} + \boldsymbol{b}\right) + (1-\omega)D\boldsymbol{x}_{k}^{SOR}.$$

Thus,

$$\underbrace{\left(\frac{1}{\omega}D + L\right)}_{M_\omega}\boldsymbol{x}_{k+1}^{SOR} = \underbrace{\left(\frac{1-\omega}{\omega}D - U\right)}_{N_\omega}\boldsymbol{x}_{k}^{SOR} + \boldsymbol{b}.$$

# Practical schemes: Successive Over-Relaxation (SOR) (cont.)

## Theorem

*The SOR iteration converges if $A$ is SPD and $0 < \omega < 2$.*

- SOR($\omega = 1$) = Gauss–Seidel iteration.
- Idea: $\omega \neq 1$ may yield *faster* convergence than GS.
- Usually $\omega \geq 1$, therefore *over*relaxation.

# Symmetric Successive Over-Relaxation (SSOR)

$M_\omega$ of SOR is nonsymmetric. Sometimes we need a symmetric preconditioner $\longrightarrow$ Symmetric Successive Over-Relaxation (SSOR)

Combination of standard forward and 'backward' SOR (same $\omega$):

$$M_\omega \boldsymbol{x}_{k+\frac{1}{2}} = N_\omega \boldsymbol{x}_k + \boldsymbol{b},$$
$$\widetilde{M}_\omega \boldsymbol{x}_{k+1} = \widetilde{N}_\omega \boldsymbol{x}_{k+\frac{1}{2}} + \boldsymbol{b},$$

with 'backward' SOR

$$\widetilde{M}_\omega = \frac{1}{\omega} D + U, \qquad \widetilde{N}_\omega = \frac{1-\omega}{\omega} D - L$$

Note: $A$ symmetric $\iff U = L^T \implies \widetilde{M}_\omega = M_\omega^T, \widetilde{N}_\omega = N_\omega^T.$

# Symmetric Successive Over-Relaxation (SSOR) (cont.)

The iteration matrix of SSOR is

$$G = \widetilde{M}_\omega^{-1} \widetilde{N}_\omega M_\omega^{-1} N_\omega.$$

The associated preconditioner is

$$M_\omega^{SSOR} = \frac{\omega}{2-\omega} \left( \frac{1}{\omega} D + L \right) D^{-1} \left( \frac{1}{\omega} D + U \right)$$

## Theorem (Convergence of SSOR)

*SSOR converges if $A$ is symmetric positive definite and $0 < \omega < 2$.*

- SSOR used when $A$ symmetric, i.e. $U = L^t$.
- Symmetric Gauss–Seidel: SSOR($\omega = 1$).
- Finding the optimal $\omega$ is mostly by experiment.

# MATLAB - Experiment

```
function [x,k] = statit(A,M,M2,b,x,tol)
%STATIT Stationary Iteration
%
%       x^{k+1} = x^{k} + M \ r^{k},   r^{k} = b - A x^{k}
%       for solving A x = b
%
%       [x,k] = statit(A,M1,M2,b,x,tol)
%       Input:  A  system matrix
%               M1,M2  M = M1*M2 'preconditioner'
%                      (M2 = [] indicates M2=identity)
%               b  right hand side
%               x  initial vector x^{0} (default x = 0)
%               tol (default tol = eps)
%       Output: x approximate solution
%               k number of iteration until convergence
%       convergence criterion:
%       norm(b - A*x) <= tol*norm(b - A*x0)
```

```
if (nargin < 6), tol = eps; end
if (nargin < 5), x = zeros(size(A,1),1); end

r = b - A*x;
rnrm0 = norm(r);  rnrm = rnrm0;
for k=1:5000
   if isempty(M2),
      x = x + M\r;
   else
      x = x + M2\(M\r);
   end
   r = b - A*x;
   rnrm = norm(r);
   if rnrm < tol*rnrm0, return, end
end
```

# Poisson equation on square $11 \times 11$ - grid

```
n=11; e = ones(n,1);
T=spdiags([-e,2*e,-e],[-1:1],n,n); I = speye(n);
A=kron(T,I) + kron(I,T);
b = A*[1:n^2]'; tol = 1e-6, x = zeros(11^2,1)
```

| Solver | MATLAB | nit |
|---|---|---|
| Jacobi | M = D = diag(diag(A)) | 341 |
| Block Jacobi | M= $D_B$ = triu(tril(A,1),-1) | 176 |
| Gauss–Seidel | M=tril(A) | 174 |
| Block Gauss–Seidel | M=tril(A,1) | 90 |
| SGS ($A$ SPD) | $M_1$=tril(A)/sqrt(D); $M_2 = M_1^T$ | 90 |
| Block SGS | $M_1$=tril(A,1)/chol($D_B$); $M_2 = M_1^T$ | 48 |
| SOR($\omega = 1.6$) | D/omega + tril(A,-1) | 32 |
| Block SOR($\omega = 1.5$) | triu(tril(A,1),-1)/omega + tril(A,-n) | 24 |

# Poisson equation on slightly larger grids

| solver | $n = 31^2$ | $n = 63^2$ |
|---|---|---|
| Jacobi | 2157 | 7787 |
| Block Jacobi | 1093 | 3943 |
| Gauss–Seidel | 1085 | 3905 |
| Block Gauss–Seidel | 547 | 1959 |
| SSOR ($\omega = 1.8$) | 85 | 238 |
| Block SSOR ($\omega = 1.8$) | 61 | 132 |

Tabelle 1: Iteration steps for solving the Poisson equation on a 31-by-31 and on a 63-by-63 grid with an relative residual accuracy of $10^{-6}$.

# Illustration of smoothing

- Let $A$ be the matrix obtained from a FD discretization of the Poisson equation on a rectangular grid. (5-point stencil with diagonal elements 4.)
- It is easy to see, that the (real) eigenvalues $\lambda_i = \lambda_i(A)$ of $A$ satisfy
$$0 < \lambda_i = \lambda_i(A) < 8.$$
- Then, with $D = \mathrm{diag}(A)$, the eigenvalues of $D^{-1}A$ satisfy
$$0 < \lambda_i(D^{-1}A) < 2,$$
and
$$-1 < \lambda_i(G) = \lambda_i(I - D^{-1}A) = 1 - \frac{\lambda_i}{4} < 1.$$
- Thus, Jacobi iteration converges. (What we know already.)

# Illustration of smoothing (cont.)

- The eigenvectors of $A$ corresponding to small eigenvalues are 'smooth', like

$$\sin(x\pi)\sin(y\pi)$$

  on the square $(0,1)^2$.
- In contrast, the eigenvectors of $A$ corresponding to eigenvalues close to 8 are 'oscillatory', like

$$\sin(nx\pi)\sin(ny\pi).$$

- Let $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$, $1 \leq i \leq n$. Then

$$G\boldsymbol{x}_i = \left(1 - \frac{\lambda_i}{4}\right)\boldsymbol{x}_i.$$

# Illustration of smoothing (cont.)

- Let $\boldsymbol{e}_0$ be the initial error, which we decompose in the directions of the eigenvectors,

$$\boldsymbol{e}_0 = \sum_{i=1}^{n} \eta_i \boldsymbol{x}_i.$$

- Then,

$$\boldsymbol{e}_k = G^k \boldsymbol{e}_0 = \sum_{i=1}^{n} \eta_i \left(1 - \frac{\lambda_i}{4}\right)^k \boldsymbol{x}_i.$$

- The error component corresponding to eigenvalues $\lambda_i \approx 0$ and $\lambda_i \approx 8$ decrease very slowly as then

$$\left|1 - \frac{\lambda_i}{4}\right| \approx 1.$$

The other error components converge to 0 (quite) quickly.

# Illustration of smoothing (cont.)

- We now introduce a damping coefficient to Jacobi,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \omega D^{-1} \boldsymbol{r}_k.$$

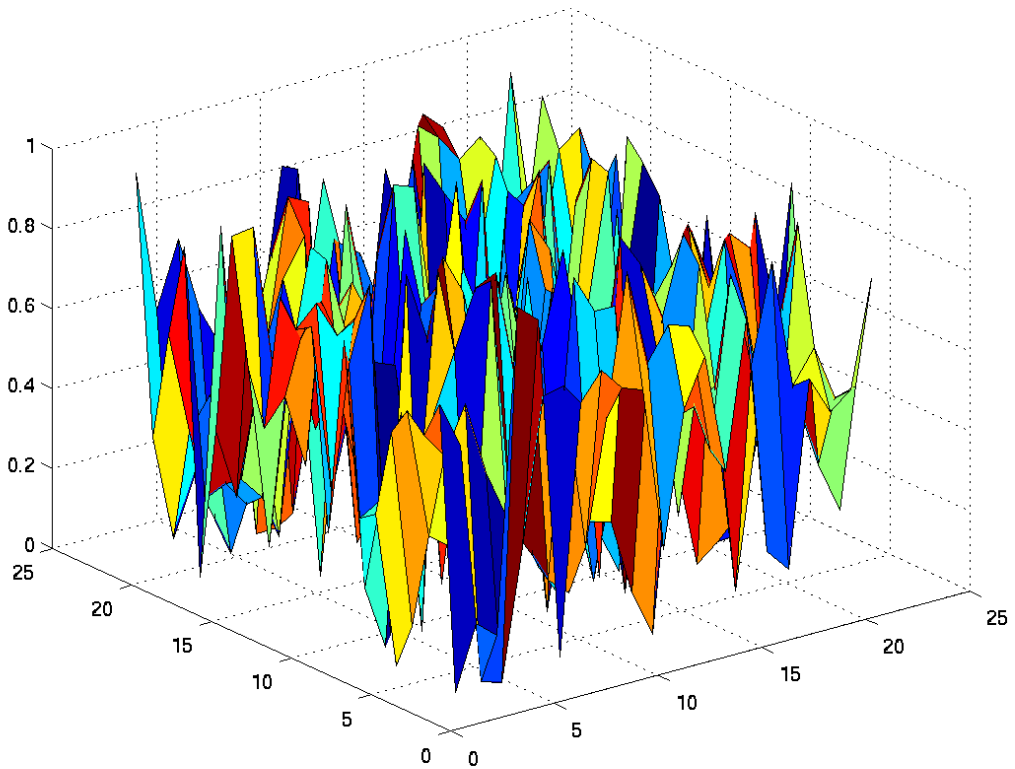
  With this modification

$$1 - 2\omega < 1 - \omega \frac{\lambda_i}{4} < 1.$$

  Typical factors are $\omega = 2/3$ for 1D problems, $\omega = 4/5$ in 2D problems.

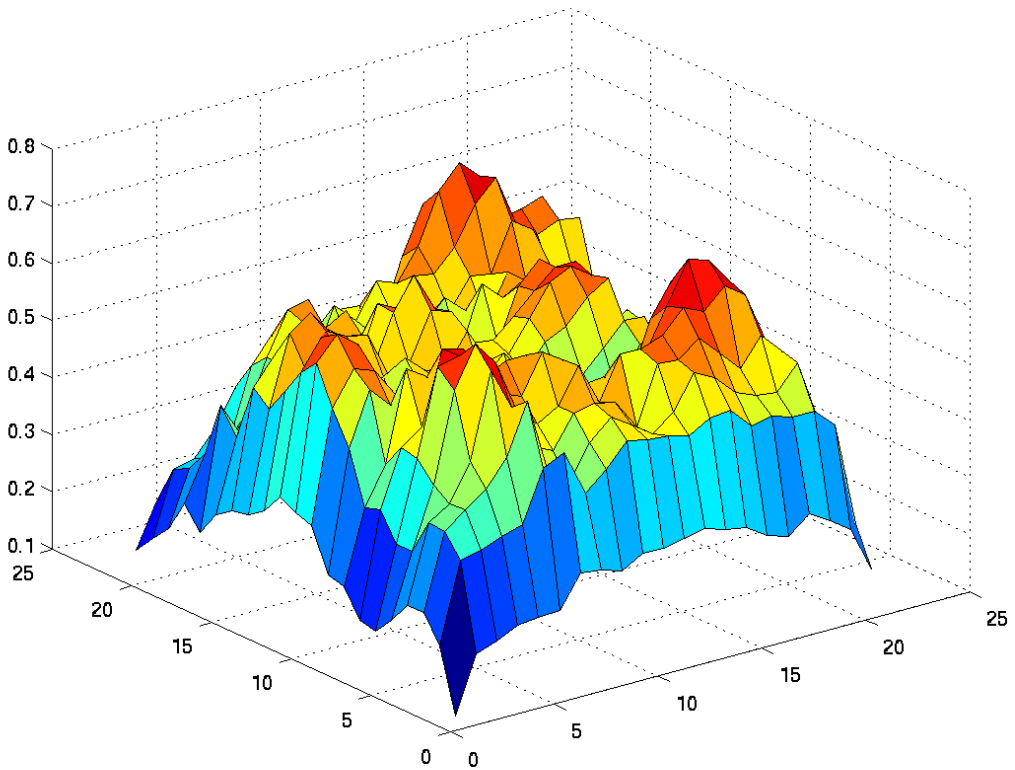
- Smooting with symmetric Gauss–Seidel is easier (automatic), since the iteration matrix has positive eigenvalues only, with the eigenvalues close to unity corresponding to smooth eigenmodes.

# Numerical example: Smoothing with sym. Gauss–Seidel

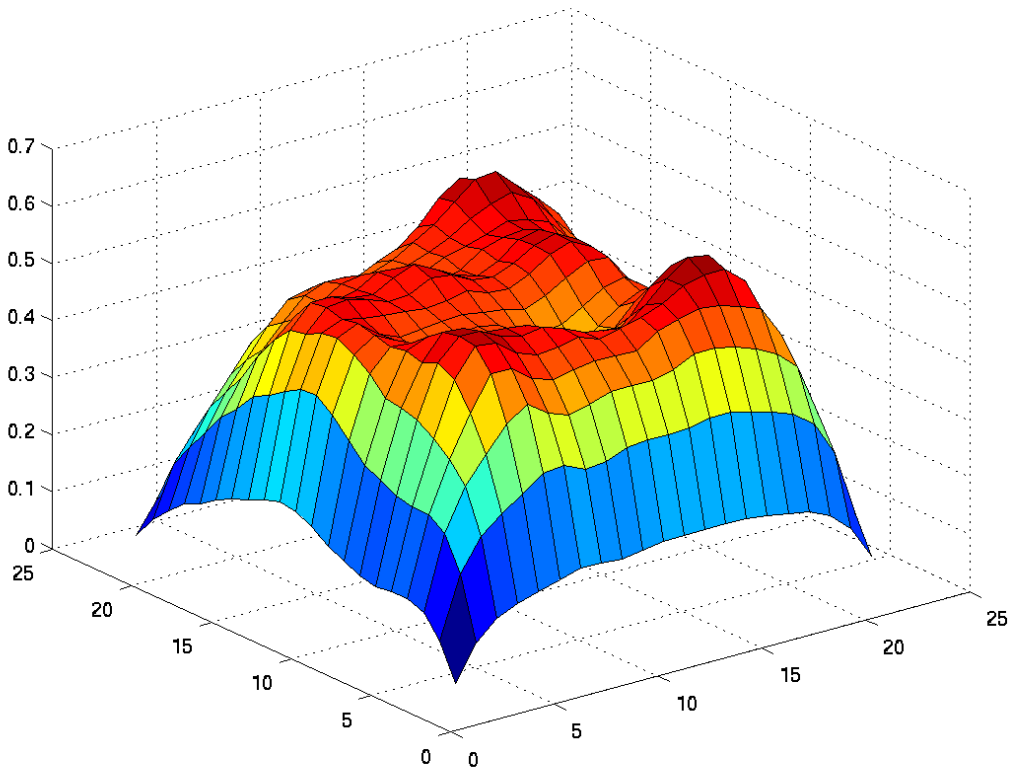



- 2D Poisson equation, $21 \times 21$ mesh. Random initial condition.

# Sym. Gauss–Seidel: error after 1 step



# Sym. Gauss–Seidel: error after 2 steps



# Summary

1. Comparison of direct and iterative linear solvers
2. Stationary iterative methods: theory
3. Stationary iterative methods: practical schemes
4. Illustration of smoothing

Next time: Steepest descent and conjugate gradient algorithms.

# Exercise 7:

http://people.inf.ethz.ch/arbenz/FEM17/pdfs/ex7.pdf