SF2520 — Applied numerical methods

Lecture 14

Numerical linear algebra

Anna Nissen
Numerical analysis
Department of Mathematics, KTH

2023-11-13

Today's lecture

- Direct methods for linear systems of equations (Gaussian elimination)
  - Full matrices, repetition
  - Sparse matrices
  - Banded matrices
- Stationary iterative methods

Linear systems of equations

We consider

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{R}^{N\times N}, \quad \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^N.$$

- Finding $\boldsymbol{x}$ given A and $\boldsymbol{b}$ done using Gaussian elimination (GE).
- GE is called a *"direct method"*, contains no approximations.

1. Reduce the extended matrix $A|\boldsymbol{b}$ to triangular form

$$\underbrace{\left(\begin{array}{cccccccc|c}
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
\times&\times&\times&\times&\times&\times&\times&\times&\times
\end{array}\right)}_{A\qquad b}
\Rightarrow
\left(\begin{array}{cccccccc|c}
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
&\times&\times&\times&\times&\times&\times&\times&\times\\
&&\times&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times
\end{array}\right)
\Rightarrow
\left(\begin{array}{cccccccc|c}
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
&\times&\times&\times&\times&\times&\times&\times&\times\\
&&\times&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times\\
&&&&\times&\times&\times&\times&\times\\
&&&&&\times&\times&\times&\times\\
&&&&&\times&\times&\times&\times\\
&&&&&\times&\times&\times&\times
\end{array}\right)
\Rightarrow
\underbrace{\left(\begin{array}{cccccccc|c}
\times&\times&\times&\times&\times&\times&\times&\times&\times\\
&\times&\times&\times&\times&\times&\times&\times&\times\\
&&\times&\times&\times&\times&\times&\times&\times\\
&&&\times&\times&\times&\times&\times&\times\\
&&&&\times&\times&\times&\times&\times\\
&&&&&\times&\times&\times&\times\\
&&&&&&\times&\times&\times\\
&&&&&&&\times&\times
\end{array}\right)}_{U\qquad \tilde{b}}$$

Subtract multiple of current row from subsequent rows.

2. Solve lower triangular system $U\boldsymbol{x} = \tilde{\boldsymbol{b}}$ by backward substitution.

Computational cost: 1 is $O(N^3)$ and 2 is $O(N^2)$.
Hard to improve if A is a full matrix.

Linear systems of equations

Connection to LU factorization

The reduction step 1 can be seen as simultaneously

(a) Factorizing $A = LU$ where
- L is lower triangular with ones on diagonal
- U is upper triangular

$$L = \begin{pmatrix} 1 & & & \\ \times & 1 & & \\ \vdots & \vdots & \ddots & \\ \times & \times & \times & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} \times & \times & \times & \times \\ & \ddots & \vdots & \vdots \\ & & \times & \times \\ & & & \times \end{pmatrix},$$

(b) Solving $L\tilde{\boldsymbol{b}} = \boldsymbol{b}$ by forward substitution.

- Here U, $\tilde{\boldsymbol{b}}$ are as in 1 above and L is given by the GE multipliers.
- GE can then be seen as: LU factorization + forward substitution for $L\tilde{\boldsymbol{b}} = \boldsymbol{b}$ + backward substitution for $U\boldsymbol{x} = \tilde{\boldsymbol{b}}$.

1	LU factorize A	$\Rightarrow$ $LU\boldsymbol{x} = \boldsymbol{b}$	Cost = $O(N^3)$
2	Solve $L\tilde{\boldsymbol{b}} = \boldsymbol{b}$	$\Rightarrow$ $U\boldsymbol{x} = \tilde{\boldsymbol{b}}$	Cost = $O(N^2)$
3	Solve $U\boldsymbol{x} = \tilde{\boldsymbol{b}}$	$\Rightarrow$ $\boldsymbol{x}$	Cost = $O(N^2)$

- Note: LU factorization very useful when same linear system should be solved many times with different right hand sides, e.g. when time stepping PDEs with an implicit method.

Pivoting

Pivoting often needed to stabilize GE.

$$\left(\begin{array}{cccccccc|c} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times & \times & \times \\ & & & \mathbf{x_0} & x_1 & x_2 & x_3 & x_4 & b_4 \\ & & & y_0 & y_1 & y_2 & y_3 & y_4 & b_5 \\ & & & \times & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times & \times & \times \\ & & & z_0 & z_1 & z_2 & z_3 & z_4 & b_8 \end{array}\right) \Rightarrow \left(\begin{array}{cccccccc|c} \times & \times & \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times & \times & \times \\ & & & \mathbf{z_0} & z_1 & z_2 & z_3 & z_4 & b_8 \\ & & & y_0 & y_1 & y_2 & y_3 & y_4 & b_5 \\ & & & \times & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times & \times & \times \\ & & & x_0 & x_1 & x_2 & x_3 & x_4 & b_4 \end{array}\right)$$

- Element $\boldsymbol{x}_0$ called the *"pivot element"*. In the reduction step, elements in the current row are divided by it.
- Small pivot elements introduces instabilities. Risk of large amplification of rounding errors.
- **Pivoting:** Changing around rows to get a large pivot element. Above: $|z_0| > |x_0|$ is assumed. (Sometimes also columns are changed.)
- Gives modified LU factorization,

$$PA = LU,$$

where P is a permutation matrix. (An identity matrix with shuffled rows.)

Theory

If A is a non-singular matrix, then

- LU and GE may not always work. e.g. $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- LU and GE with pivoting always works.
- If A is also symmetric positive definite[1] LU and GE always works without pivoting...
- ...and GE gives an even simpler "Cholesky" factorization $A = LL^T$ (with L lower triangular).

Remark: Do not compute A^{-1} explicitly and multiply $\boldsymbol{x} = A^{-1}\boldsymbol{b}$. This is more expensive and less stable than solving $A\boldsymbol{x} = \boldsymbol{b}$ with GE, in particular for sparse matrices.

[1] Or diagonally dominant, $|a_{kk}| > \sum_{j \neq k} |a_{kj}|$ for all j.

Sparse linear systems

- The system

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{R}^{N\times N}, \quad \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^N.$$

  is sparse if the number of non-zero elements in A is $\ll N^2$.
- A typical situation is that the number of non-zero elements are $O(N)$, e.g. if each row or column has a fixed number of them.
- Cost of matrix-vector multiply is then $O(N)$.
- A special type is banded matrices...

$$A = \left(\begin{array}{ccc} & & 0 \\ & \neq 0 & \\ 0 & & \end{array} \right) \begin{array}{l} \left.\vphantom{\begin{array}{c}0\\0\\0\end{array}}\right\} N, \\ \left.\vphantom{\begin{array}{c}0\\0\\0\end{array}}\right\} p \end{array}$$

Here $p \ll N$.

p called the *bandwidth*.

Ex: $p = 2$ for tridiagonal matrices.

PDE discretizations

- Discretizations of PDEs lead to sparse systems, e.g. in elliptic equations or implicit time stepping methods for parabolic PDEs.
- Finite difference methods on simple geometries (with standard ordering) give banded systems:

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N\times N}$$

$-u_{xx} = f$ in 1D
$h \sim 1/N$,
bandwidth = 2

$$\underbrace{A = \frac{1}{h^2}\begin{pmatrix} T & -I & & \\ -I & T & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & T \end{pmatrix}}_{\in\mathbb{R}^{N\times N}}, \quad \underbrace{T = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 \end{pmatrix}}_{\in\mathbb{R}^{n\times n}}$$

$-\Delta u = f$ in 2D
$h \sim 1/n$,
bandwidth $= n \sim \sqrt{N}$

- Finite element methods also give banded systems with $\approx$ same bandwidth if a good ordering of the elements is used. (Good ordering needed for FD too in general.)

PDE discretizations

Remarks:

- Discretization of second derivatives ($-\partial_{xx}$, $-\Delta$) typically leads to symmetric positive definite matrices (with the right BC). These have real positive eigenvalues.
- Discretization of first derivatives (∂_x, ∇) typically leads to skew symmetric matrices $A = -A^T$, (with the right BC). These have purely imaginary eigenvalues.

Solving sparse linear systems

Suppose A is sparse, when can we solve $A\boldsymbol{x} = \boldsymbol{b}$ faster than $O(N^3)$?

- If A is triangular and sparse, backward/forward substitution cost is O(# non zero elements in A). (Most operations inlvolve only zeros and can be skipped.)
- In general, if $A = LU$, both L and U must also be sparse, in order to be able to solve $A\boldsymbol{x} = \boldsymbol{b}$ fast.
  (Then there are few entries in L and U to compute, and both $L\tilde{\boldsymbol{b}} = \boldsymbol{b}$ and $U\boldsymbol{x} = \tilde{\boldsymbol{b}}$ can be solved fast by the previous point.)
- Unfortunately, A sparse does not imply that L and U are sparse.
  Example:

$$\underbrace{\begin{pmatrix} \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & & & & & & \\ \times & & \times & & & & & \\ \times & & & \times & & & & \\ \times & & & & \times & & & \\ \times & & & & & \times & & \\ \times & & & & & & \times & \\ \times & & & & & & & \times \\ \times & & & & & & & \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} \times & & & & & & & \\ \times & \times & & & & & & \\ \times & \times & \times & & & & & \\ \times & \times & \times & \times & & & & \\ \times & \times & \times & \times & \times & & & \\ \times & \times & \times & \times & \times & \times & & \\ \times & \times & \times & \times & \times & \times & \times & \\ \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \times & \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times & \times \\ & & & & \times & \times & \times & \times \\ & & & & & \times & \times & \times \\ & & & & & & \times & \times \\ & & & & & & & \times \end{pmatrix}}_{U}$$

L and U full. New non-zero entries in L and U called "fill-in".

Solving sparse linear systems

- Appearance of fill-in depends a lot on column and row ordering. If we reverse columns and rows in the previous example we get:

$$\underbrace{\begin{pmatrix} \times & & & & & & & \times \\ & \times & & & & & & \times \\ & & \times & & & & & \times \\ & & & \times & & & & \times \\ & & & & \times & & & \times \\ & & & & & \times & & \times \\ & & & & & & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}}_{PAQ} = \begin{pmatrix} \times & & & & & & & \\ & \times & & & & & & \\ & & \times & & & & & \\ & & & \times & & & & \\ & & & & \times & & & \\ & & & & & \times & & \\ & & & & & & \times & \\ \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix} \begin{pmatrix} \times & & & & & & & \times \\ & \times & & & & & & \times \\ & & \times & & & & & \times \\ & & & \times & & & & \times \\ & & & & \times & & & \times \\ & & & & & \times & & \times \\ & & & & & & \times & \times \\ & & & & & & & \times \end{pmatrix}$$

where P, Q are permutation matrices. L and U sparse!

- When solving a linear system we are free to use any row and column ordering. Finding the optimal reordering (or permutations P, Q) to minimize fill-in is an NP-complete problem, however. Too expensive!
- Note also: Pivoting may restrict possible reorderings, or cause fill-in by itself.

Solving sparse linear systems

- Banded systems can always be solved fast by GE.
- In this case, GE involves mostly zeros and L, U remains sparse (banded).

$$
\underbrace{\begin{pmatrix}
\times & \times & \times & \times & & & & & & & & \\
\times & \times & \times & \times & \times & & & & & & & \\
\times & \times & \times & \times & \times & \times & & & & & & \\
\times & \times & \times & \times & \times & \times & \times & & & & & \\
 & \times & \times & \times & \times & \times & \times & \times & & & & \\
 & & \times & \times & \times & \times & \times & \times & \times & & & \\
 & & & \times & \times & \times & \times & \times & \times & \times & & \\
 & & & & \times & \times & \times & \times & \times & \times & \times & \\
 & & & & & \times & \times & \times & \times & \times & \times & \times \\
 & & & & & & \times & \times & \times & \times & \times & \times \\
 & & & & & & & \times & \times & \times & \times & \times \\
 & & & & & & & & \times & \times & \times & \times
\end{pmatrix}}_{A}
=
\underbrace{\begin{pmatrix}
\times & & & & & & & & & & & \\
\times & \times & & & & & & & & & & \\
\times & \times & \times & & & & & & & & & \\
\times & \times & \times & \times & & & & & & & & \\
 & \times & \times & \times & \times & & & & & & & \\
 & & \times & \times & \times & \times & & & & & & \\
 & & & \times & \times & \times & \times & & & & & \\
 & & & & \times & \times & \times & \times & & & & \\
 & & & & & \times & \times & \times & \times & & & \\
 & & & & & & \times & \times & \times & \times & & \\
 & & & & & & & \times & \times & \times & \times & \\
 & & & & & & & & \times & \times & \times & \times
\end{pmatrix}}_{L}
\quad
\underbrace{\begin{pmatrix}
\times & \times & \times & \times & & & & & & & & \\
 & \times & \times & \times & \times & & & & & & & \\
 & & \times & \times & \times & \times & & & & & & \\
 & & & \times & \times & \times & \times & & & & & \\
 & & & & \times & \times & \times & \times & & & & \\
 & & & & & \times & \times & \times & \times & & & \\
 & & & & & & \times & \times & \times & \times & & \\
 & & & & & & & \times & \times & \times & \times & \\
 & & & & & & & & \times & \times & \times & \times \\
 & & & & & & & & & \times & \times & \times \\
 & & & & & & & & & & \times & \times \\
 & & & & & & & & & & & \times
\end{pmatrix}}_{U}
$$

Solving sparse linear systems

Why?

$$
\begin{array}{cc}
 & p \\
p & \begin{pmatrix}
\times & \times & \times & \times & & & & & & & \\
\times & \times & \times & \times & \times & & & & & & \\
\times & \times & \times & \times & \times & \times & & & & & \\
\times & \times & \times & \times & \times & \times & \times & & & & \\
 & \times & \times & \times & \times & \times & \times & \times & & & \\
 & & \times & \times & \times & \times & \times & \times & \times & & \\
 & & & \times & \times & \times & \times & \times & \times & \times & \\
 & & & & \times & \times & \times & \times & \times & \times & \times \\
 & & & & & \times & \times & \times & \times & \times & \times \\
 & & & & & & \times & \times & \times & \times & \times \\
 & & & & & & & \times & \times & \times & \times \\
 & & & & & & & & \times & \times & \times \\
 & & & & & & & & & \times & \times \\
 & & & & & & & & & & \times
\end{pmatrix}
\end{array}
$$

First step

Row 1 is subtracted only from rows $2, \ldots, p$. (Rows $p, \ldots, N$ already have a leading zero.)
$\Rightarrow$ Only the squared part of matrix is affected. Cost = $O(p^2)$.

$$
\begin{array}{cc}
 & \\
 & \begin{pmatrix}
\times & \times & \times & \times & & & & & & & \\
 & \times & \times & \times & \times & & & & & & \\
 & & \times & \times & \times & \times & & & & & \\
 & & & \times & \times & \times & \times & & & & \\
 & & & & \times & \times & \times & \times & & & \\
 & & & & \times & \times & \times & \times & \times & & \\
 & & & & \times & \times & \times & \times & \times & \times & \\
 & & & & & \times & \times & \times & \times & \times & \times \\
 & & & & & & \times & \times & \times & \times & \times \\
 & & & & & & & \times & \times & \times & \times \\
 & & & & & & & & \times & \times & \times \\
 & & & & & & & & & \times & \times \\
 & & & & & & & & & & \times
\end{pmatrix}
\end{array}
$$

row j

p

Step j

Row j is subtracted only from rows $j+1, \ldots, p+j-1$.
$\Rightarrow$ Only the squared part of matrix is affected. Cost = $O(p^2)$.

Same cost $O(p^2)$ in every step.

Solving sparse linear systems

- Same cost $O(p^2)$ in every step, $A \in \mathbb{R}^{N \times N}$
- $\Rightarrow$ total cost, GE for banded systems = $O(Np^2)$.
- U becomes upper triangular with bandwidth p
- L becomes lower triangular with bandwidth p
- Backward/forward substitution has smaller cost than the reduction step: $O(Np)$.
- For Cholesky factorization $A = LL^T$ cost will be same $O(Np^2)$ and L as bandwidth p.
- When pivoting is needed, $PA = LU$, the L and U factors are *still sparse* but less so:
  - U has bandwidth $2p$
  - L not banded, but lower triangular with $p + 1$ non-zero entries per column. (I.e. forward substitution still fast, at $O(Np)$.)

Costs for solving Poisson equation

Consider a finite difference discretization of

$$-\Delta u = f,$$

in d dimensions. This leads to a matrix equation

$$A\boldsymbol{u} = \boldsymbol{f}, \qquad A \in \mathbb{R}^{N\times N}.$$

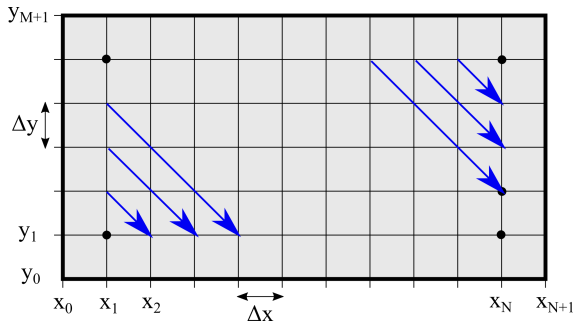
Suppose discretization has n grid points in each coordinate direction. Then

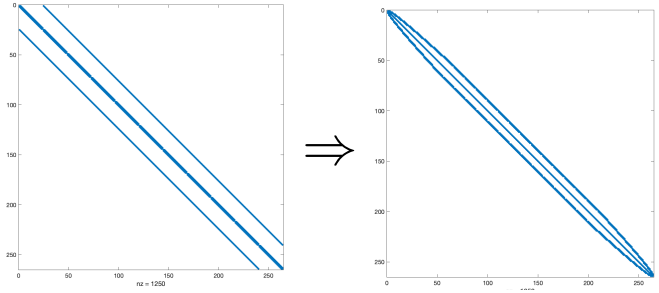
- Total number of points is $N = n^d$.
- Bandwidth is $p = O(n^{d-1})$.
- Since cost is $O(Np^2)$ we get
  - Poisson in 1D. Then $N = n$, $p = 1$, gives cost = $O(N)$.
  - Poisson in 2D. Then $N = n^2$, $p = n$, gives cost = $O(n^4) = O(N^2)$.
  - Poisson in 3D. Then $N = n^3$, $p = n^2$, gives cost = $O(n^7) = O(N^{2\frac{1}{3}})$.
- Higher dimension means not only more unknowns N but also higher cost to solve matrix equation for the same N.

Sparse but not banded systems

Remarks on sparse but not banded systems

- There are many heuristic methods (greedy algorithms) to reorder rows and columns to a better form that reduces fill-in. Works well in practice (e.g. for FEM) but no guarantee of success.
- Cost of these metods are O(# non-zero elements in A), i.e. smaller than cost of LU-factorization and GE.
- Matlab-commands, examples:
  - "Reverse Cuthill–McKee" (RCM)
    `>> p = symrcm(A);` $\Rightarrow$ `A(p,p)` reordered to low bandwidth.
  - Minimum degree, for A symmetric positive definite
    `>> p = symamd(A);` $\Rightarrow$ `A(p,p)` has sparse Cholesky factors.
- **Ex.:** 2D Poisson with RCM gives ordering along *diagonals*. Can be solved in $O(n^3)$ instead of $O(n^4)$!





Iterative methods

We consider

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{R}^{N\times N}, \quad \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^N.$$

- Solve this via iteration $\Rightarrow$

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$$

- Stop e.g. when $||A\boldsymbol{x}_k - \boldsymbol{b}|| \leq \varepsilon$ or when $||\boldsymbol{x}_k - \boldsymbol{x}_{k-1}|| \leq \varepsilon$, for some given tolerance ε.
- Primarily used for large sparse linear systems.
- Main cost of an iteration is typically a multiplication of $\boldsymbol{x}_k$ by (a part of) the matrix A. Hence, cost of one iteration =O(# non-zero elements of A).
- Methods are mostly insensitive to the precise sparsity structure. No need to worry about "fill-in" as in direct methods.
- Easier to exploit sparseness with iterative methods than with direct methods.

Stationary methods

We consider

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{R}^{N\times N}, \quad \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^N.$$

Construct iteration as follows:

- Split $A = M - T$, so that $M\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{b}$.
- Iterate

$$M\boldsymbol{x}_{k+1} = T\boldsymbol{x}_k + \boldsymbol{b}.$$

- Choose M invertible, and so that $M\boldsymbol{x} = \boldsymbol{f}$ is easy to solve.
- Example 1: $M =$ diagonal of A gives "Jacobi method"
- Example 2: $M =$ lower triangular part of A gives "Gauss–Seidel method"

Stationary methods

Jacobi and Gauss–Seidel methods

Let $A = \{a_{j,\ell}\}$ and $\boldsymbol{x}_k = \{x_j^k\}$. Consider the methods elementwise, for a 3×3 matrix.

Jacobi

$$\begin{aligned} a_{11}x_1^{k+1} + a_{12}x_2^k + a_{13}x_3^k &= b_1, \\ a_{21}x_1^k + a_{22}x_2^{k+1} + a_{23}x_3^k &= b_2, \\ a_{31}x_1^k + a_{32}x_2^k + a_{33}x_3^{k+1} &= b_3 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x_1^{k+1} &= \tfrac{1}{a_{11}}(b_1 - a_{12}x_2^k - a_{13}x_3^k), \\ x_2^{k+1} &= \tfrac{1}{a_{22}}(b_2 - a_{21}x_1^k - a_{23}x_3^k), \\ x_3^{k+1} &= \tfrac{1}{a_{33}}(b_3 - a_{31}x_1^k - a_{32}x_2^k). \end{aligned}$$

Gauss–Seidel

$$\begin{aligned} a_{11}x_1^{k+1} + a_{12}x_2^k + a_{13}x_3^k &= b_1, \\ a_{21}x_1^{k+1} + a_{22}x_2^{k+1} + a_{23}x_3^k &= b_2, \\ a_{31}x_1^{k+1} + a_{32}x_2^{k+1} + a_{33}x_3^{k+1} &= b_3 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x_1^{k+1} &= \tfrac{1}{a_{11}}(b_1 - a_{12}x_2^k - a_{13}x_3^k), \\ x_2^{k+1} &= \tfrac{1}{a_{22}}(b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k), \\ x_3^{k+1} &= \tfrac{1}{a_{33}}(b_3 - a_{31}x_1^{k+1} - a_{32}x_2^{k+1}). \end{aligned}$$

Hence, Gauss–Seidel not much harder to implement than Jacobi. Just use the previously computed x_j^{k+1} values in the update, instead of the old ones x_j^k.

Convergence

Consider the stationary method

$$M\boldsymbol{x}_{k+1} = T\boldsymbol{x}_k + \boldsymbol{b}.$$

- Can be written as fixed point iteration

$$\boldsymbol{x}_{k+1} = \phi(\boldsymbol{x}_k), \qquad \phi(\boldsymbol{x}) = M^{-1}(T\boldsymbol{x} + \boldsymbol{b}).$$

- This iteration converges if spectral radius of jacobian < 1. Here,

$$\rho(M^{-1}T) < 1, \qquad (\rho(\,\cdot\,) \text{ is the spectral radius.})$$

- For the Jacobi method this is fulfilled when $A \in \mathbb{R}^{N\times N}$ is *diagonally dominant*,

$$|a_{kk}| > \sum_{\ell\neq k} |a_{k,\ell}|, \qquad k = 1, \ldots, N.$$

- For Gauss–Seidel it is fulfilled if A is diagonally dominant or symmetric positive definite.
- Gauss–Seidel usually converges a bit faster than Jacobi.