~

Numerical Linear Algebra

Scientific Computing Winter 2016/2017

Part II

With material from Y. Saad "Iterative Methods for Sparse Linear Systems", R. S. Varga "Matrix Iterative Analysis", J. Shewchuk: "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain"

Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

made wit pandoc

Floating point representation

- Scientific notation of floating point numbers: e.g. $x = 6.022 \cdot 10^{23}$
- Representation formula:

$$x = \pm \sum_{i=0}^{\infty} d_i \beta^{-i} \beta^e$$

  - $\beta \in \mathbb{N}, \beta \geq 2$: base
  - $d_i \in \mathbb{N}, 0 \leq d_i \leq \beta$: mantissa digits
  - $e \in \mathbb{Z}$: exponent
- Representation on computer:

$$x = \pm \sum_{i=0}^{t-1} d_i \beta^{-i} \beta^e$$

  - $\beta = 2$
  - t: mantissa length, e.g. $t = 53$ for IEEE double
  - $L \leq e \leq U$, e.g. $-1022 \leq e \leq 1023$ (10 bits) for IEEE double
  - $d_0 \neq 0 \Rightarrow$ normalized numbers, unique representation

Floating point limits

- symmetry wrt. 0 because of sign bit
- smallest positive normalized number: $d_0 = 1, d_i = 0, i = 1 \dots t-1$
  $x_{min} = \beta^L$
- smallest positive denormalized number: $d_i = 0, i = 0 \dots t-2, d_{t-1} = 1$
  $x_{min} = \beta^{1-t}\beta^L$
- largest positive normalized number: $d_i = \beta - 1, 0 \dots t-1$
  $x_{max} = \beta(1 - \beta^{1-t})\beta^U$

Machine precision

- Exact value x
- Approximation $\tilde{x}$
- Then: $|\frac{\tilde{x}-x}{x}| < \epsilon$ is the best accuracy estimate we can get, where
  - $\epsilon = \beta^{1-t}$ (truncation)
  - $\epsilon = \frac{1}{2}\beta^{1-t}$ (rounding)
- Also: ϵ is the smallest representable number such that $1 + \epsilon > 1$.
- Relative errors show up in partiular when
  - subtracting two close numbers
  - adding smaller numbers to larger ones

Matrix + Vector norms

- Vector norms: let $x = (x_i) \in \mathbb{R}^n$
  - $||x||_1 = \sum_i =^n |x_i|$: sum norm, l_1-norm
  - $||x||_2 = \sqrt{\sum_{i=1}^n x_i^2}$: Euclidean norm, l_2-norm
  - $||x||_\infty = \max_{i=1\dots n} |x_i|$: maximum norm, l_∞-norm
- Matrix $A = (a_{ij}) \in \mathbb{R}^n \times \mathbb{R}^n$
  - Representation of linear operator $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^n$ defined by $\mathcal{A} : x \mapsto y = Ax$ with

$$y_i = \sum_{j=1}^n a_{ij} x_j$$

  - Induced matrix norm:

$$\begin{aligned} ||A||_\nu &= \max_{x \in \mathbb{R}^n, x \neq 0} \frac{||Ax||_\nu}{||x||_\nu} \\ &= \max_{x \in \mathbb{R}^n, ||x||_\nu = 1} \frac{||Ax||_\nu}{||x||_\nu} \end{aligned}$$

Matrix norms

- $||A||_1 = \max_{j=1\dots n} \sum_{i=1}^n |a_{ij}|$ maximum of column sums
- $||A||_\infty = \max_{i=1\dots n} \sum_{j=1}^n |a_{ij}|$ maximum of row sums
- $||A||_2 = \sqrt{\lambda_{max}}$ with λ_{max}: largest eigenvalue of $A^T A$.

Matrix condition number and error propagation

Problem: solve $Ax = b$, where b is inexact.

$$A(x + \Delta x) = b + \Delta b.$$

Since $Ax = b$, we get $A\Delta x = \Delta b$. From this,

$$\left\{ \begin{array}{ll} \Delta x &= A^{-1}\Delta b \\ Ax &= b \end{array} \right\} \Rightarrow \left\{ \begin{array}{ll} ||A|| \cdot ||x|| & \geq ||b|| \\ ||\Delta x|| & \leq ||A^{-1}|| \cdot ||\Delta b|| \end{array} \right.$$

$$\Rightarrow \frac{||\Delta x||}{||x||} \leq \kappa(A) \frac{||\Delta b||}{||b||}$$

where $\kappa(A) = ||A|| \cdot ||A^{-1}||$ is the *condition number* of A.

Approaches to linear system solution

Solve $Ax = b$

Direct methods:

- Deterministic
- Exact up to machine precision
- Expensive (in time and space)

Iterative methods:

- Only approximate
- Cheaper in space and (possibly) time
- Convergence not guaranteed

Really bad example of direct method

Cramer's rule
write $|A|$ for determinant, then

$$x_i = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1i-1} & b_1 & a_{1i+1} & \dots & a_{1n} \\ a_{21} & & \dots & & b_2 & & \dots & a_{2n} \\ \vdots & & & & \vdots & & & \vdots \\ a_{n1} & & \dots & & b_n & & \dots & a_{nn} \end{vmatrix} / |A| \quad (i = 1 \dots n)$$

$O(n!)$ operations...

Gaussian elimination

- Essentially the only feasible direct solution method
- Solve $Ax = b$ with square matrix A.

Gauss 1

$$\begin{pmatrix} 6 & -2 & 2 \\ 12 & -8 & 6 \\ 3 & -13 & 3 \end{pmatrix} x = \begin{pmatrix} 16 \\ 26 \\ -19 \end{pmatrix}$$

Step 1

$$\begin{pmatrix} 6 & -2 & 2 \\ 0 & 4 & -2 \\ 0 & -12 & 2 \end{pmatrix} x = \begin{pmatrix} 16 \\ -6 \\ -27 \end{pmatrix}$$

Step 2

$$\begin{pmatrix} 6 & -2 & 2 \\ 0 & 4 & -2 \\ 0 & -0 & -4 \end{pmatrix} x = \begin{pmatrix} 16 \\ -6 \\ -9 \end{pmatrix}$$

Gauss 2

Solve upper triangular system

$$\begin{pmatrix} 6 & -2 & 2 \\ 0 & 4 & -2 \\ 0 & 0 & -4 \end{pmatrix} x = \begin{pmatrix} 16 \\ -6 \\ -9 \end{pmatrix}$$

$$-4x_3 = -9 \qquad \Rightarrow x_3 = \frac{9}{4}$$

$$-4x_2 - 2x_3 = -6 \quad \Rightarrow -4x_2 = \frac{21}{2} \qquad \Rightarrow x_2 = -\frac{21}{8}$$

$$6x_1 - 2x_2 + 2x_3 = 2 \quad \Rightarrow 6x_1 = 2 - \frac{21}{4} - \frac{18}{4} = -\frac{31}{4} \quad \Rightarrow x_1 = -\frac{-31}{24}$$

Gaussian elimination expressed in matrix operations: LU factorization

$$L_1Ax = \begin{pmatrix} 6 & -2 & 2 \\ 0 & 4 & -2 \\ 0 & -12 & 2 \end{pmatrix} x = \begin{pmatrix} 16 \\ -6 \\ -27 \end{pmatrix} = L_1b, \qquad L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{pmatrix}$$

$$L_2L_1Ax = \begin{pmatrix} 6 & -2 & 2 \\ 0 & 4 & -2 \\ 0 & -0 & -4 \end{pmatrix} x = \begin{pmatrix} 16 \\ -6 \\ -9 \end{pmatrix} = L_2L_1b, \qquad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix}$$

- Let $L = L_1^{-1}L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ \frac{1}{2} & 3 & 1 \end{pmatrix}$, $U = L_2L_1A$. Then $A = LU$
- Inplace operation. Diagonal elements of L are always 1, so no need to store them $\Rightarrow$ work on storage space for A and overwrite it.

Problem example

Consider

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 1+\epsilon \\ 2 \end{pmatrix}$$

with solution $x = (1,1)^t$

Ordinary elimination:

$$\begin{pmatrix} \epsilon & 1 \\ 0 & (1 - \frac{1}{\epsilon}) \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 - \frac{1}{\epsilon} \end{pmatrix}$$

$$\Rightarrow x_2 = \frac{2 - \frac{1}{\epsilon}}{1 - \frac{1}{\epsilon}} \Rightarrow x_1 = \frac{1 - x_2}{\epsilon}$$

If $\epsilon < \epsilon_{\mathrm{mach}}$, then $2 - 1/\epsilon = -1/\epsilon$ and $1 - 1/\epsilon = -1/\epsilon$, so

$$x_2 = \frac{2 - \frac{1}{\epsilon}}{1 - \frac{1}{\epsilon}} = 1, \Rightarrow x_1 = \frac{1 - x_2}{\epsilon} = 0$$

Partial Pivoting

- Before elimination step, look at the element with largest absolute value in current column and put the corresponding row "on top" as the "pivot"
- This prevents near zero divisions and increases stability

$$\begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 \\ 0 & 1-\epsilon \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 - 2\epsilon \end{pmatrix}$$

If ϵ very small:

$$x_2 = \frac{1 - 2\epsilon}{1 - \epsilon} = 1, \qquad x_1 = 2 - x_2 = 1$$

- Factorization: $PA = LU$, where P is a permutation matrix which can be encoded usin an integer vector

Gaussian elimination and LU factorization

- Full pivoting: in addition to row exchanges, perform column exchanges to ensure even larger pivots. Seldomly used in practice.
- Gaussian elimination with partial pivoting is the "working horse" for direct solution methods
- Standard routines from LAPACK: `dgetrf`, (factorization) `dgetrs` (solve) used in overwhelming number of codes (e.g. matlab, scipy etc.). Also, C++ matrix libraries use them. Unless there is special need, they should be used.
- Complexity of LU-Factorization: $O(n^3)$, some theoretically better algorithms are known with e.g. $O(n^{2.736})$

Cholesky factorization

- $A = LL^T$ for symmetric, positive definite matrices

Matrices from PDE: a first example

- "Drosophila": Poisson boundary value problem in rectangular domain

Given:

- Domain $\Omega = (0, X) \times (0, Y) \subset \mathbb{R}^2$ with boundary $\Gamma = \partial\Omega$, outer normal $\mathbf{n}$
- Right hand side $f : \Omega \to \mathbb{R}$
- "Conductivity" λ
- Boundary value $v : \Gamma \to \mathbb{R}$
- Transfer coefficient α

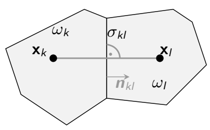
Search function $u : \Omega \to \mathbb{R}$ such that

$$
\begin{aligned}
-\nabla \cdot \lambda \nabla u = f \quad & \text{in}\Omega \\
-\lambda \nabla u \cdot \mathbf{n} + \alpha(u - v) = 0 \quad & \text{on}\Gamma
\end{aligned}
$$

- Example: heat conduction:
  - u: temperature
  - f: volume heat source
  - λ: heat conduction coefficient
  - v: Ambient temperature
  - α: Heat transfer coefficient

The finite volume idea

- Assume Ω is a polygon
- Subdivide the domain Ω into a finite number of **control volumes** : $\bar{\Omega} = \bigcup_{k \in \mathcal{N}} \bar{\omega}_k$ such that
  - ω_k are open (not containing their boundary) convex domains
  - $\omega_k \cap \omega_l = \emptyset$ if $\omega_k \neq \omega_l$
  - $\sigma_{kl} = \bar{\omega}_k \cap \bar{\omega}_l$ are either empty, points or straight lines
    - we will write $|\sigma_{kl}|$ for the length
    - if $|\sigma_{kl}| > 0$ we say that ω_k, ω_l are neigbours
    - neigbours of ω_k: $\mathcal{N}_k = \{l \in \mathcal{N} : |\sigma_{kl}| > 0\}$
- To each control volume ω_k assign a **collocation point**: $\mathbf{x}_k \in \bar{\omega}_k$ such that
  - **admissibility condition**: if $l \in \mathcal{N}_k$ then the line $\mathbf{x}_k\mathbf{x}_l$ is orthogonal to σ_{kl}
  - if ω_k is situated at the boundary, i.e. $\gamma_k = \partial\omega_k \cap \partial\Omega \neq \emptyset$, then $\mathbf{x}_k \in \partial\Omega$



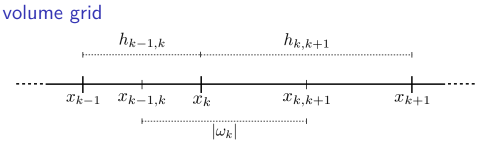
Discretization ansatz

- Given control volume ω_k, integrate equation over control volume

$$
\begin{aligned}
0 &= \int_{\omega_k} (-\nabla \cdot \lambda \nabla u - f)\, d\omega \\
&= -\int_{\partial\omega_k} \lambda \nabla u \cdot \mathbf{n}_k d\gamma - \int_{\omega_k} f d\omega \qquad \text{(Gauss)} \\
&= -\sum_{L \in \mathcal{N}_k} \int_{\sigma_{kl}} \lambda \nabla u \cdot \mathbf{n}_{kl} d\gamma - \int_{\gamma_k} \lambda \nabla u \cdot \mathbf{n} d\gamma - \int_{\omega_k} f d\omega \\
&\approx \sum_{L \in \mathcal{N}_k} \frac{\sigma_{kl}}{h_{kl}} (u_k - u_l) + |\gamma_k| \alpha (u_k - v_k) - |\omega_k| f_k
\end{aligned}
$$

- Here,
  - $u_k = u(\mathbf{x}_k)$
  - $v_k = v(\mathbf{x}_k)$
  - $f_k = f(\mathbf{x}_k)$
- $N = |\mathcal{N}|$ equations (one for each control volume)
- $N = |\mathcal{N}|$ unknowns (one in each collocation point $\equiv$ control volume)

1D finite volume grid



- $\Omega = [0, X]$
- Collocation points: $0 = x_1 < x_2 < \dots < x_{n-1} < x_n = X$
- Control volumes:

$$
\begin{aligned}
\omega_1 &= (x_1, (x_1 + x_2)/2) \\
\omega_2 &= ((x_1 + x_2)/2, (x_2 + x_3)/2) \\
&\vdots \\
\omega_{N-1} &= ((x_{N-2} + x_{N-1})/2, (x_{N-1} + x_N)/2) \\
\omega_N &= ((x_{N-1} + x_N)/2, x_N)
\end{aligned}
$$

- Maximum number of neighbours: 2

Discretization matrix (1D)

Assume $\lambda = 1$, $h_{kl} = h$ and we count collocation points from $1 \dots N$. For $k = 2 \dots N-1$, $\omega_K = h$, and

$$
\sum_{L \in \mathcal{N}_k} \frac{\sigma_{kl}}{h_{kl}} (u_k - u_l) = \frac{1}{h}(-u_{k-1} + 2u_k - u_{k+1})
$$

The linear system then is (only nonzero entries marked):

$$
\begin{pmatrix}
\alpha + \frac{1}{h} & -\frac{1}{h} & & & & & \\
-\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & & \\
& -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & \\
& \ddots & \ddots & \ddots & \ddots & & \\
& & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & \\
& & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & \\
& & & & -\frac{1}{h} & \frac{1}{h} + \alpha
\end{pmatrix}
\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{pmatrix}
=
\begin{pmatrix} \frac{h}{2} f_1 + \alpha v_1 \\ hf_2 \\ hf_3 \\ \vdots \\ hf_{N-2} \\ hf_{N-1} \\ \frac{h}{2} f_N + \alpha v_n \end{pmatrix}
$$

General tridiagonal matrix

$$
\begin{pmatrix}
b_1 & c_1 & & & \\
a_2 & b_2 & c_2 & & \\
& a_3 & b_3 & \ddots & \\
& & \ddots & \ddots & c_{n-1} \\
& & & a_n & b_n
\end{pmatrix}
\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{pmatrix}
=
\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{pmatrix}
$$

Gaussian elimination for tridiagonal systems

- TDMA (tridiagonal matrix algorithm)
- "Thomas algorithm" (Llewellyn H. Thomas, 1949 (?))
- "Progonka method" (Gelfand, Lokutsievski, 1952, published 1960)

$a_i u_{i-1} + b_i u_i + c_i u_{i+1} = f_i$, $a_1 = 0$, $c_N = 0$

For $i = 1 \dots n-1$, assume there are coefficients α_i, β_i such that $u_i = \alpha_{i+1} u_{i+1} + \beta_{i+1}$.

Then, we can express u_{i-1} and u_i via u_{i+1}:
$(a_i \alpha_i \alpha_{i+1} + c_i \alpha_{i+1} + b_i) u_{i+1} + a_i \alpha_i \beta_{i+1} + a_i \beta_i + c_i \beta_{i+1} - f_i = 0$

This is true independently of u if

$$
\begin{cases}
a_i \alpha_i \alpha_{i+1} + c_i \alpha_{i+1} + b_i & = 0 \\
a_i \alpha_i \beta_{i+1} + a_i \beta_i + c_i \beta_{i+1} - f_i & = 0
\end{cases}
$$

or for $i = 1 \dots n-1$:

$$
\begin{cases}
\alpha_{i+1} & = -\frac{b_i}{a_i \alpha_i + c_i} \\
\beta_{i+1} & = \frac{f_i - a_i \beta_i}{a_i \alpha_i + c_i}
\end{cases}
$$

Progonka algorithm

Forward sweep:

$$\begin{cases} \alpha_2 &= -\frac{b_1}{c_1} \\ \beta_2 &= \frac{f_i}{c_1} \end{cases}$$

for $i = 2 \dots n-1$

$$\begin{cases} \alpha_{i+1} &= -\frac{b_i}{a_i\alpha_i + c_i} \\ \beta_{i+1} &= \frac{f_i - a_i\beta_i}{a_i\alpha_i + c_i} \end{cases}$$

Backward sweep:

$$u_n = \frac{f_n - a_n\beta_n}{a_n\alpha_n + c_n}$$

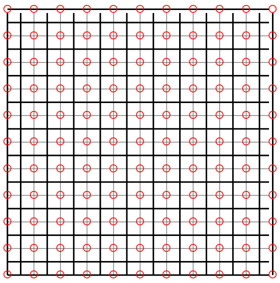
for $n-1 \dots 1$:

$$u_i = \alpha_{i+1}u_{i+1} + \beta_{i+1}$$

Progonka algorithm - properties

- n unknowns, one forward sweep, one backward sweep $\Rightarrow O(n)$ operations vs. $O(n^3)$ for algorithm using full matrix
- No pivoting $\Rightarrow$ stability issues
  - Stability for diagonally dominant matrices ($|b_i| > |a_i| + |c_i|$)
  - Stability for symmetric positive definite matrices

2D finite volume grid



- Red circles: discretization nodes
- Thin lines: original "grid"
- Thick lines: boundaries of control volumes
- Each discretization point has not more then 4 neighbours

Sparse matrices

- Regardless of number of unknowns n, the number of non-zero entries per row remains limited by n_r
- If we find a scheme which allows to store only the non-zero matrix entries, we would need $nn_r = O(n)$ storage locations instead of n^2
- The same would be true for the matrix-vector multiplication if we program it in such a way that we use every nonzero element just once: martrix-vector multiplication uses $O(n)$ instead of $O(n^2)$ operartions
- In the special case of tridiagonal matrices, progonka gives an algorithm which allows to solve the nonlinear system with $O(n)$ operations

Sparse matrix questions

- What is a good format for sparse matrices?
- Is there a way to implement Gaussian elimination for general sparse matrices which allows for linear system solution with $O(n)$ operation
- Is there a way to implement Gaussian elimination *with pivoting* for general sparse matrices which allows for linear system solution with $O(n)$ operations?
- Is there *any algorithm* for sparse linear system solution with $O(n)$ operations?

Coordinate (triplet) format

- store all nonzero elements along with their row and column indices
- one real, two integer arrays, length = nnz= number of nonzero elements

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

AA	12.	9.	7.	5.	1.	2.	11.	3.	6.	4.	8.	10.
JR	5	3	3	2	1	1	4	2	3	2	3	4
JC	5	5	3	4	1	4	4	1	1	2	4	3

Y.Saad, Iterative Methods, p.92

Compressed Row Storage (CRS) format

(aka Compressed Sparse Row (CSR) or IA-JA etc.)

- real array `AA`, length nnz, containing all nonzero elements row by row
- integer array `JA`, length nnz, containing the column indices of the elements of `AA`
- integer array `IA`, length n+1, containing the start indizes of each row in the arrays `IA` and `JA` and `IA(n+1)=nnz+1`

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

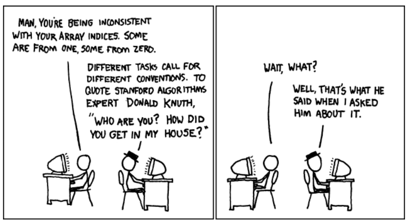
AA	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
JA	1	4	1	2	4	1	3	4	5	3	4	5
IA	1	3	6	10	12	13						

Y.Saad, Iterative Methods, p.93

- Used in most sparse matrix packages

The big schism

- Worse than catholics vs. protestants or shia vs. sunni...
- Should array indices count from zero or from one ?
- Fortran, Matlab, Julia count from one
- C/C++, python count from zero
- I am siding with the one fraction
- but I am tolerant, so for this course ...
  - It matters when passing index arrays to sparse matrix packages



http://xkcd.com/1739/

CRS again

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

```
AA: 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12.
JA: 0 3 0 1 3 0 2 3 4 2 3 4
IA: 0 2 4 0 11 12
```

- some package APIs provide the possibility to specify array offset
- index shift is not very expensive compared to the rest of the work

Sparse direct solvers

- Sparse direct solvers implement Gaussian elimination with different pivoting strategies
  - UMFPACK
  - Pardiso (omp + MPI parallel)
  - SuperLU
  - MUMPS (MPI parallel)
  - Pastix
- Quite efficient for 1D/2D problems
- They suffer from *fill-in*: $\Rightarrow$ huge memory usage for 3D

Sparse direct solvers: solution steps (Saad Ch. 3.6)

1. Pre-ordering
   - The amount of non-zero elements generated by fill-in can be decreases by re-ordering of the matrix
   - Several, graph theory based heuristic algorithms exist
2. Symbolic factorization
   - If pivoting is ignored, the indices of the non-zero elements are calculated and stored
   - Most expensive step wrt. computation time
3. Numerical factorization
   - Calculation of the numerical values of the nonzero entries
   - Not very expensive, once the symbolic factors are available
4. Upper/lower triangular system solution
   - Fairly quick in comparison to the other steps

- Separation of steps 2 and 3 allows to save computational costs for problems where the sparsity structure remains unchanged, e.g. time dependent problems on fixed computational grids
- With pivoting, steps 2 and 3 have to be performed together
- Instead of pivoting, *iterative refinement* may be used in order to maintain accuracy of the solution

Interfacing UMFPACK from C++ (numcxx)

(shortened version of the code)

```
#include <suitesparse/umfpack.h>

// Calculate LU factorization
template<> inline void TSolverUMFPACK<double>::update()
{
    pMatrix->flush(); // Update matrix, adding newly created elements
    int n=pMatrix->shape(0);
    double *control=nullptr;

    //Calculate symbolic factorization only if matrix patter
    //has changed
    if (pMatrix->pattern_changed())
    {
       umfpack_di_symbolic (n, n, pMatrix->pIA->data(), pMatrix->pJA->data(), pMatrix->pA->data(),
       &Symbolic, 0, 0);
    }

    umfpack_di_numeric (pMatrix->pIA->data(), pMatrix->pJA->data(), pMatrix->pA->data(),
    Symbolic, &Numeric, control, 0) ;

   pMatrix->pattern_changed(false);
}

// Solve LU factorized system
template<> inline void TSolverUMFPACK<double>::solve( TArray<T> & Sol,  const TArray<T> & Rhs)
{
    umfpack_di_solve (UMFPACK_At,pMatrix->pIA->data(), pMatrix->pJA->data(), pMatrix->pA->data(),
                      Sol.data(), Rhs.data(),
                      Numeric, control, 0 ) ;
}
```


How to use ?

```
#include <numcxx/numcxx.h>
auto pM=numcxx::DSparseMatrix::create(n,n);
auto pF=numcxx::DArray1::create(n);
auto pU=numcxx::DArray1::create(n);

auto &M=*pM;
auto &F=*pF;
auto &U=*pU;

F=1.0;
for (int i=0;i<n;i++)
{
    M(i,i)=3.0;
    if (i>0) M(i,i-1)=-1;
    if (i<n-1) M(i,i+1)=-1;
}

auto pUmfpack=numcxx::DSolverUMFPACK::create(pM);
pUmfpack->solve(U,F);
```


~

Towards iterative methodsx

Elements of iterative methods (Saad Ch.4)

Solve $Au = b$ iteratively

- Preconditioner: a matrix $M \approx A$ "approximating" the matrix A but with the property that the system $Mv = f$ is easy to solve
- Iteration scheme: algorithmic sequence using M and A which updates the solution step by step

Simple iteration with preconditioning

Idea: $A\hat{u} = b \Rightarrow$

$$\hat{u} = \hat{u} - M^{-1}(A\hat{u} - b)$$

$\Rightarrow$ iterative scheme

$$u_{k+1} = u_k - M^{-1}(Au_k - b) \quad (k = 0, 1 \ldots)$$

1. Choose initial value u_0, tolerance ε, set $k = 0$
2. Calculate *residuum* $r_k = Au_k - b$
3. Test convergence: if $||r_k|| < \varepsilon$ set $u = u_k$, finish
4. Calculate *update*: solve $Mv_k = r_k$
5. Update solution: $u_{k+1} = u_k - v_k$, set $k = i + 1$, repeat with step 2.

The Jacobi method

- Let $A = D - E - F$, where D: main diagonal, E: negative lower triangular part F: negative upper triangular part
- Jacobi: $M = D$, where D is the main diagonal of A.

$$u_{k+1,i} = u_{k,i} - \frac{1}{a_{ii}}\left(\sum_{j=1\dots n} a_{ij}u_{k,j} - b_i\right) \quad (i = 1\dots n)$$

$$a_{ii}u_{k+1,i} + \sum_{j=1\dots n, j\neq i} a_{ij}u_{k,j} = b_i \quad (i = 1\dots n)$$

- Alternative formulation:

$$u_{k+1} = D^{-1}(E + F)u_k + D^{-1}b$$

- Essentially, solve for main diagonal element row by row
- Already calculated results not taken into account
- Variable ordering does not matter

The Gauss-Seidel method

- Solve for main diagonal element row by row
- Take already calculated results into account

$$a_{ii}u_{k+1,i} + \sum_{j<i} a_{ij}u_{k+1,j} + \sum_{j>i} a_{ij}u_{k,j} = b_i \quad (i = 1\dots n)$$

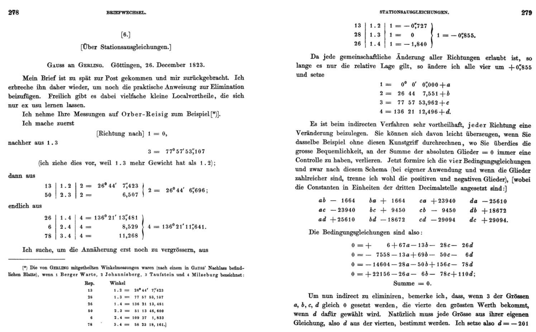
$$(D - E)u_{k+1} - Fu_k = b$$

$$u_{k+1} = (D - E)^{-1}Fu_k + (D - E)^{-1}b$$

- May be it is faster
- Variable order probably matters
- The preconditioner is $M = D - E$
- Backward Gauss-Seidel: $M = D - F$
- Splitting formulation: $A = M - N$, then

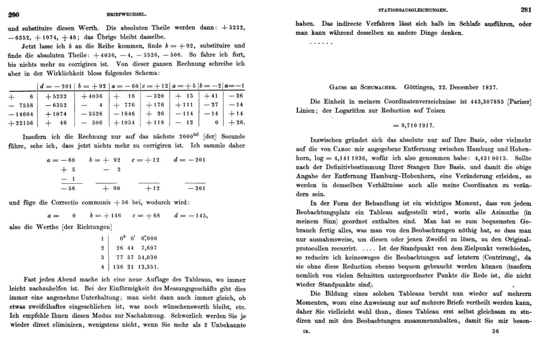
$$u_{k+1} = M^{-1}Nu_k + M^{-1}b$$

Gauss an Gerling I



http://gdz.sub.uni-goettingen.de/

Gauss an Gerling II



http://gdz.sub.uni-goettingen.de/

SOR and SSOR

- SOR: Successive overrelaxation: solve $\omega A = \omega B$ and use splitting

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega D))$$

$$M = \frac{1}{\omega}(D - \omega E)$$

leading to

$$(D - \omega E)u_{k+1} = (\omega F + (1 - \omega D)u_k + \omega b$$

- SSOR: Symmetric successive overrelaxation

$$(D - \omega E)u_{k+\frac{1}{2}} = (\omega F + (1 - \omega D)u_k + \omega b$$

$$(D - \omega F)u_{k+1} = (\omega E + (1 - \omega D)u_{k+\frac{1}{2}} + \omega b$$

$$M = \frac{1}{\omega(2 - \omega)}(D - \omega E)D^{-1}(D - \omega F)$$

- Gauss-Seidel and symmetric Gauss-Seidel are special cases for $\omega = 1$.

Block methods

- Jacobi, Gauss-Seidel, (S)SOR methods can as well be used block-wise, based on a partition of the system matrix into larger blocks,
- The blocks on the diagonal should be square matrices, and invertible
- Interesting variant for systems of partial differential equations, where multiple species interact with each other

Convergence

Let $\hat{u}$ be the solution of $Au = b$.

$$\begin{aligned} u_{k+1} &= u_k - M^{-1}(Au_k - b) \\ &= (I - M^{-1}A)u_k + M^{-1}b \\ u_{k+1} - \hat{u} &= u_k - \hat{u} - M^{-1}(Au_k - A\hat{u}) \\ &= (I - M^{-1}A)(u_k - \hat{u}) \\ &= (I - M^{-1}A)^k(u_0 - \hat{u}) \end{aligned}$$

So when does $(I - M^{-1}A)^k$ converge to zero for $k \to \infty$?

Jordan canonical form of a matrix A

- λ_i $(i = 1\dots p)$: eigenvalues of A
- $\sigma(A) = \{\lambda_1 \dots \lambda_p\}$: spectrum of A
- μ_i: algebraic multiplicity of λ_i:
  multiplicity as zero of the characteristic polynomial $\det(A - \lambda I)$
- γ_i geometric multiplicity of λ_i: dimension of $\mathrm{Ker}(A - \lambda I)$
- l_i: index of the eigenvalue: the smallest integer for which $\mathrm{Ker}(A - \lambda I)^{l_i+1} = \mathrm{Ker}(A - \lambda I)^{l_i})$
- $l_i \leq \mu_i$

Theorem (Saad, Th. 1.8) Matrix A can be transformed to a block diagonal matrix consisting of p diagonal blocks, each associated with a distinct eigenvalue λ_i.

- Each of these diagonal blocks has itself a block diagonal structure consisting of γ_i *Jordan blocks*
- Each of the Jordan blocks is an upper bidiagonal matrix of size not exceeding l_i with λ_i on the diagonal and 1 on the first upper diagonal.

Jordan canonical form of a matrix II

$$X^{-1}AX = J = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_p \end{pmatrix}$$

$$J_i = \begin{pmatrix} J_{i,1} & & & \\ & J_{i,2} & & \\ & & \ddots & \\ & & & J_{i,\gamma_i} \end{pmatrix}$$

$$J_{i,k} = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & 1 & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}$$

Each $J_{i,k}$ is of size l_i and corresponds to a different eigenvector of A.

Spectral radius and convergence

- $\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|$: spectral radius

Theorem (Saad, Th. 1.10) $\lim_{k\to\infty} A^k = 0 \Leftrightarrow \rho(A) < 1$.

Proof, $\Rightarrow$: Let u_i be a unit eigenvector associated with an eigenvalue λ_i. Then

$$\begin{aligned} Au_i &= \lambda_i u_i \\ A^2 u_i &= \lambda_i A_i u_i = \lambda^2 u_i \\ &\vdots \\ A^k u_i &= \lambda^k u_i \\ \text{therefore} \quad ||A^k u_i||_2 &= |\lambda^k| \\ \text{and} \quad \lim_{k\to\infty} |\lambda^k| &= 0 \end{aligned}$$

so we must have $\rho(A) < 1$

Spectral radius and convergence II

Proof, $\Leftarrow$: Jordan form $X^{-1}AX = J$. Then $X^{-1}A^kX = J^k$.
Sufficient to regard Jordan block $J_i = \lambda_i I + E_i$ where $|\lambda_i| < 1$ and $E_i^{l_i} = 0$.
Let $k \geq l_i$. Then

$$J_i^k = \sum_{j=0}^{l_i-1} \binom{k}{j} \lambda^{k-j} E_i^j$$

$$||J_i||^k \leq \sum_{j=0}^{l_i-1} \binom{k}{j} |\lambda|^{k-j} ||E_i||^j$$

One has $\binom{k}{j} = \frac{k!}{j!(k-j)!} = \sum_{i=0}^{j} \begin{bmatrix} j \\ i \end{bmatrix} \frac{k^i}{j!}$ is a polynomial
where for $k > 0$, the Stirling numbers of the first kind are given by
$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1, \quad \begin{bmatrix} j \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ j \end{bmatrix} = 0, \quad \begin{bmatrix} j+1 \\ i \end{bmatrix} = j \begin{bmatrix} j \\ i \end{bmatrix} + \begin{bmatrix} j \\ i-1 \end{bmatrix}$.

Thus, $\binom{k}{j} |\lambda|^{k-j} \to 0 \ (k \to \infty)$.

Corollary from proof

Theorem (Saad, Th. 1.12)

$$\lim_{k\to\infty} ||A^k||^{\frac{1}{k}} = \rho(A)$$

Back to iterative methods

Sufficient condition for convergence: $\rho(I - M^{-1}A) < 1$.

Convergence rate

Assume λ with $|\lambda| = \rho(I - M^{-1}A)$ is the largest eigenvalue and has a single Jordan block. Then the convergence rate is dominated by this Jordan block, and therein by the term

$$\lambda^{k-p+1} \binom{k}{p-1} E^{p-1}$$

$$||(I - M^{-1}A)^k (u_0 - \hat{u})|| = O\left(|\lambda^{k-p+1}| \binom{k}{p-1}\right)$$

and the "worst case" convergence factor ρ equals the spectral radius:

$$\begin{aligned} \rho &= \lim_{k\to\infty} \left(\max_{u_0} \frac{||(I - M^{-1}A)^k (u_0 - \hat{u})||}{||u_0 - \hat{u}||} \right)^{\frac{1}{k}} \\ &= \lim_{k\to\infty} ||(I - M^{-1}A)^k||^{\frac{1}{k}} \\ &= \rho(I - M^{-1}A) \end{aligned}$$

Depending on u_0, the rate may be faster, though

Richardson iteration

$M = \frac{1}{\alpha}$, $I - M^{-1}A = I - \alpha A$. Assume for the eigenvalues of A:
$\lambda_{min} \leq \lambda_i \leq \lambda_{max}$.

Then for the eigenvalues μ_i of $I - \alpha A$ one has $1 - \alpha\lambda_{max} \leq \lambda_i \leq 1 - \alpha\lambda_{min}$.

If $\lambda_{min} < 0$ and $\lambda_{max} < 0$, at least one $\mu_i > 1$.

So, assume $\lambda_{min} > 0$. Then we must have

$1 - \alpha\lambda_{max} > -1, 1 - \alpha\lambda_{min} < 1 \Rightarrow$
$0 < \alpha < \frac{2}{\lambda_{max}}$.

$\rho = \max(|1 - \alpha\lambda_{max}|, |1 - \alpha\lambda_{min}|)$

$\alpha_{opt} = \frac{2}{\lambda_{min} + \lambda_{max}}$

$\rho_{opt} = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}$

Regular splittings

$A = M - N$ is a regular splitting if - M is nonsingular - M^{-1}, N are nonnegative, i.e. have nonnegative entries

- Regard the iteration $u_{k+1} = M^{-1}Nu_k + M^{-1}b$.

When does it converge ?

Theory of nonnegative matrices

1.10 Nonnegative Matrices, M-Matrices

Nonnegative matrices play a crucial role in the theory of matrices. They are important in the study of convergence of iterative methods and arise in many applications including economics, queuing theory, and chemical engineering.

A *nonnegative matrix* is simply a matrix whose entries are nonnegative. More generally, a partial order relation can be defined on the set of matrices.

Definition 1.23 *Let A and B be two $n \times m$ matrices. Then*

$$A \leq B$$

if by definition, $a_{ij} \leq b_{ij}$ for $1 \leq i \leq n$, $1 \leq j \leq m$. If O denotes the $n \times m$ zero matrix, then A is nonnegative if $A \geq O$, and positive if $A > O$. Similar definitions hold in which "positive" is replaced by "negative".

The binary relation "$\leq$" imposes only a *partial* order on $\mathbb{R}^{n\times m}$ since two arbitrary matrices in $\mathbb{R}^{n\times m}$ are not necessarily comparable by this relation. For the remainder of this section, we now assume that only square matrices are involved. The next proposition lists a number of rather trivial properties regarding the partial order relation just defined.

Properties of $\leq$ for matrices

Proposition 1.24 *The following properties hold.*

1. *The relation $\leq$ for matrices is reflexive ($A \leq A$), antisymmetric (if $A \leq B$ and $B \leq A$, then $A = B$), and transitive (if $A \leq B$ and $B \leq C$, then $A \leq C$).*
2. *If A and B are nonnegative, then so is their product AB and their sum $A + B$.*
3. *If A is nonnegative, then so is A^k.*
4. *If $A \leq B$, then $A^T \leq B^T$.*
5. *If $O \leq A \leq B$, then $\|A\|_1 \leq \|B\|_1$ and similarly $\|A\|_\infty \leq \|B\|_\infty$.*

Irreducible matrices

A is *irreducible* if there is no permutation matrix P such that PAP^T is upper block triangular.

Perron-Frobenius Theorem

Theorem (Saad Th.1.25) Let A be a real $n \times n$ nonnegative irreducible martrix. Then:

- The spectral radius $\rho(A)$ is a simple eigenvalue of A.
- There exists an eigenvector u associated wit $\rho(A)$ which has positive elements

Comparison of products of nonnegative matrices

Proposition 1.26 *Let A, B, C be nonnegative matrices, with $A \leq B$. Then*

$$AC \leq BC \quad \textit{and} \quad CA \leq CB.$$

Proof. Consider the first inequality only, since the proof for the second is identical. The result that is claimed translates into

$$\sum_{k=1}^{n} a_{ik}c_{kj} \leq \sum_{k=1}^{n} b_{ik}c_{kj}, \quad 1 \leq i, j \leq n,$$

which is clearly true by the assumptions. □

Comparison of powers of nonnegative matrices

Corollary 1.27 *Let A and B be two nonnegative matrices, with $A \leq B$. Then*

$$A^k \leq B^k, \quad \forall\ k \geq 0. \tag{1.42}$$

Proof. The proof is by induction. The inequality is clearly true for $k = 0$. Assume that (1.42) is true for k. According to the previous proposition, multiplying (1.42) from the left by A results in

$$A^{k+1} \leq AB^k. \tag{1.43}$$

Now, it is clear that if $B \geq 0$, then also $B^k \geq 0$, by Proposition 1.24. We now multiply both sides of the inequality $A \leq B$ by B^k to the right, and obtain

$$AB^k \leq B^{k+1}. \tag{1.44}$$

The inequalities (1.43) and (1.44) show that $A^{k+1} \leq B^{k+1}$, which completes the induction proof. □

Comparison of spectral radii of nonnegative matrices

Theorem 1.28 *Let A and B be two square matrices that satisfy the inequalities*

$$O \leq A \leq B. \tag{1.45}$$

Then

$$\rho(A) \leq \rho(B). \tag{1.46}$$

Proof. The proof is based on the following equality stated in Theorem 1.12

$$\rho(X) = \lim_{k\to\infty} \|X^k\|^{1/k}$$

for any matrix norm. Choosing the 1−norm, for example, we have from the last property in Proposition 1.24

$$\rho(A) = \lim_{k\to\infty} \|A^k\|_1^{1/k} \leq \lim_{k\to\infty} \|B^k\|_1^{1/k} = \rho(B)$$

which completes the proof. □

Nonnegative matrices in iterations

Theorem 1.29 *Let B be a nonnegative matrix. Then $\rho(B) < 1$ if and only if $I - B$ is nonsingular and $(I - B)^{-1}$ is nonnegative.*

Proof. Define $C = I - B$. If it is assumed that $\rho(B) < 1$, then by Theorem 1.11, $C = I - B$ is nonsingular and

$$C^{-1} = (I - B)^{-1} = \sum_{i=0}^{\infty} B^i. \tag{1.47}$$

In addition, since $B \geq 0$, all the powers of B as well as their sum in (1.47) are also nonnegative.

To prove the sufficient condition, assume that C is nonsingular and that its inverse is nonnegative. By the Perron-Frobenius theorem, there is a nonnegative eigenvector u associated with $\rho(B)$, which is an eigenvalue, i.e.,

$$Bu = \rho(B)u$$

or, equivalently,

$$C^{-1}u = \frac{1}{1 - \rho(B)}u.$$

Since u and C^{-1} are nonnegative, and $I - B$ is nonsingular, this shows that $1 - \rho(B) > 0$, which is the desired result. □

M-Matrices

Definition 1.30 *A matrix is said to be an M-matrix if it satisfies the following four properties:*

1. $a_{i,i} > 0$ *for* $i = 1, \dots, n$.
2. $a_{i,j} \le 0$ *for* $i \ne j,\ i, j = 1, \dots, n$.
3. *A is nonsingular.*
4. $A^{-1} \ge 0$.

- This matrix property plays an important role for discrtized PDEs:
  - convergence of iterative methods
  - nonnegativity of discrete solutions (e.g concentrations)
  - prevention of unphysical oscillations

Equivalent definition

Theorem 1.31 *Let a matrix A be given such that*

1. $a_{i,i} > 0$ *for* $i = 1, \dots, n$.
2. $a_{i,j} \le 0$ *for* $i \ne j,\ i, j = 1, \dots, n$.

Then A is an M-matrix if and only if

3. $\rho(B) < 1$, *where* $B = I - D^{-1}A$.

Proof. From the above argument, an immediate application of Theorem 1.29 shows that properties (3) and (4) of the above definition are equivalent to $\rho(B) < 1$, where $B = I - C$ and $C = D^{-1}A$. In addition, C is nonsingular iff A is and C^{-1} is nonnegative iff A is. □

Equivalent definition

Theorem 1.32 *Let a matrix A be given such that*

1. $a_{i,j} \le 0$ *for* $i \ne j,\ i, j = 1, \dots, n$.
2. *A is nonsingular.*
3. $A^{-1} \ge 0$.

Then

4. $a_{i,i} > 0$ *for* $i = 1, \dots, n$, *i.e., A is an M-matrix.*
5. $\rho(B) < 1$ *where* $B = I - D^{-1}A$.

Proof. Define $C \equiv A^{-1}$. Writing that $(AC)_{ii} = 1$ yields

$$\sum_{k=1}^{n} a_{ik} c_{ki} = 1$$

which gives

$$a_{ii} c_{ii} = 1 - \sum_{\substack{k=1 \\ k \ne i}}^{n} a_{ik} c_{ki}.$$

Since $a_{ik} c_{ki} \le 0$ for all k, the right-hand side is ≥ 1 and since $c_{ii} \ge 0$, then $a_{ii} > 0$. The second part of the result now follows immediately from an application of the previous theorem. □

Comparison criterion

Theorem 1.33 *Let A, B be two matrices which satisfy*

1. $A \le B$.
2. $b_{ij} \le 0$ *for all* $i \ne j$.

Then if A is an M-matrix, so is the matrix B.

Proof. Assume that A is an M-matrix and let D_X denote the diagonal of a matrix X. The matrix D_B is positive because

$$D_B \ge D_A > 0.$$

Consider now the matrix $I - D_B^{-1}B$. Since $A \le B$, then

$$D_A - A \ge D_B - B \ge O$$

which, upon multiplying through by D_A^{-1}, yields

$$I - D_A^{-1}A \ge D_A^{-1}(D_B - B) \ge D_B^{-1}(D_B - B) = I - D_B^{-1}B \ge O.$$

Since the matrices $I - D_B^{-1}B$ and $I - D_A^{-1}A$ are nonnegative, Theorems 1.28 and 1.31 imply that

$$\rho(I - D_B^{-1}B) \le \rho(I - D_A^{-1}A) < 1.$$

This establishes the result by using Theorem 1.31 once again. □

Regular splittings

- $A = M - N$ is a regular splitting if
  - M is nonsingular
  - M^{-1}, N are nonnegative, i.e. have nonnegative entries
- Regard the iteration $u_{k+1} = M^{-1}Nu_k + M^{-1}b$.
- We have $I-M^{-1}A= $M^{-1}N$.

When does it converge ?

Convergence of iterations based on regular splittings

Theorem 4.4 *Let M, N be a regular splitting of a matrix A. Then $\rho(M^{-1}N) < 1$ if and only if A is nonsingular and A^{-1} is nonnegative.*

Proof. Define $G = M^{-1}N$. From the fact that $\rho(G) < 1$, and the relation

$$A = M(I - G) \tag{4.35}$$

it follows that A is nonsingular. The assumptions of Theorem 1.29 are satisfied for the matrix G since $G = M^{-1}N$ is nonnegative and $\rho(G) < 1$. Therefore, $(I - G)^{-1}$ is nonnegative as is $A^{-1} = (I - G)^{-1} M^{-1}$.

To prove the sufficient condition, assume that A is nonsingular and that its inverse is nonnegative. Since A and M are nonsingular, the relation (4.35) shows again that $I - G$ is nonsingular and in addition,

$$\begin{aligned} A^{-1}N &= \left(M(I - M^{-1}N)\right)^{-1} N \\ &= (I - M^{-1}N)^{-1}M^{-1}N \\ &= (I - G)^{-1}G. \end{aligned} \tag{4.36}$$

Clearly, $G = M^{-1}N$ is nonnegative by the assumptions, and as a result of the Perron-Frobenius theorem, there is a nonnegative eigenvector x associated with $\rho(G)$ which is an eigenvalue, such that

$$Gx = \rho(G)x.$$

Convergence of iterations based on regular splittings II

From this and by virtue of (4.36), it follows that

$$A^{-1}Nx = \frac{\rho(G)}{1 - \rho(G)}x.$$

Since x and $A^{-1}N$ are nonnegative, this shows that

$$\frac{\rho(G)}{1 - \rho(G)} \ge 0$$

and this can be true only when $0 \le \rho(G) \le 1$. Since $I - G$ is nonsingular, then $\rho(G) \ne 1$, which implies that $\rho(G) < 1$. □

This theorem establishes that the iteration (4.34) always converges, if M, N is a regular splitting and A is an M-matrix.

Regular splittings: example

- Jacobi
- Gauss-Seidel

Further methods for establishing convergence

- Theory for diagonally dominant matrices
- Theory for symmetric, positive definite matrices

Iterative methods so far

- main thread ("Roter Faden"):
  - Simple iterative methods converge if the spectral radius of the iteration matrix is less than one
  - If a matrix has the M-Property (positve main diagonal entries, nonpositive off diagonal entries, nonsingular, inverse nonnegative), then methods based regular splittings converge
  - But: how can we see that a matrix has the M-Property?
- This theory is useful in other contexts as well
- Main source: Varga, "Matrix Iterative Analysis"

The Gershgorin Circle Theorem

(everywhere, we assume $n \geq 2$)

Theorem Let A be an $n \times n$ (complex) matrix. Let

$$\Lambda_i = \sum_{\substack{j=1\dots n\\ j\neq i}} |a_{ij}|$$

If λ is an eigenvalue of A then there is r, $1 \leq r \leq n$ such that

$$|\lambda - a_{rr}| \leq \Lambda_r$$

Proof Assume λ is eigenvalue, x a corresponding eigenvector, normalized such that $\max_{i=1\dots n} |x_i| = |x_r| = 1$. From $Ax = \lambda x$ it follows that

$$(\lambda - a_{ii})x_i = \sum_{\substack{j=1\dots n\\ j\neq i}} a_{ij}x_j$$

$$|\lambda - a_{rr}| = |\sum_{\substack{j=1\dots n\\ j\neq r}} a_{rj}x_j| \leq \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}||x_j| \leq \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}| = \Lambda_r$$

Gershgorin Circle Corollaries

Corollary: Any eigenvalue of A lies in the union of the disks defined by the Gershgorin circles

$$\lambda \in \bigcup_{i=1\dots n} \{\mu \in \mathbb{C} : |\mu - |a_{ii}|| \leq \Lambda_i\}$$

Corollary:

$$\rho(A) \leq \max_{i=1\dots n} \sum_{j=1}^{n} |a_{ij}| = ||A||_\infty$$

$$\rho(A) \leq \max_{j=1\dots n} \sum_{i=1}^{n} |a_{ij}| = ||A||_1$$

Proof

$$|\mu - a_{ii}| \leq \Lambda_i \quad \Rightarrow \quad |\mu| \leq \Lambda_i + |a_{ii}| = \sum_{j=1}^{n} |a_{ij}|$$

Furthermore, $\sigma(A) = \sigma(A^T)$. □

Reducible and irreducible matrices

Definition A is *reducible* if there exists a permutation matrix P such that

$$PAP^T = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

A is *irreducible* if it is not reducible.

Directed matrix graph:

- Nodes: $\mathcal{N} = \{N_i\}_{i=1\dots n}$
- Directed edges: $\mathcal{E} = \{\vec{N_k N_l} | a_{kl} \neq 0\}$

A is irreducible $\Leftrightarrow$ the matrix graph is connected, i.e. for each *ordered* pair N_i, N_j there is a path consisting of directed edges, connecting them.

Equivalently, for each i, j there is a sequence of nonzero matrix entries $a_{ik_1}, a_{k_1k_2}, \dots, a_{k_rj}$.

Taussky theorem

Theorem Let A be irreducible. Assume that the eigenvalue λ is a boundary point of the union of all the disks

$$\lambda \in \partial \bigcup_{i=1\dots n} \{\mu \in \mathbb{C} : |\mu - a_{ii}| \leq \Lambda_i\}$$

Then, all n Gershgorin circles pass through λ, i.e. for $i = 1 \dots n$,

$$|\lambda - a_{ii}| = \Lambda_i$$

Taussky theorem proof

Proof Assume λ is eigenvalue, x a corresponding eigenvector, normalized such that $\max_{i=1\dots n} |x_i| = |x_r| = 1$. From $Ax = \lambda x$ it follows that

$$|\lambda - a_{rr}| \leq \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}| \cdot |x_j| \leq \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}| = \Lambda_r \qquad (*)$$

Boundary point $\Rightarrow |\lambda - a_{rr}| = \Lambda_r$

$\Rightarrow$ For all $l \neq r$ with $a_{r,p} \neq 0$, $|x_p| = 1$.

Due to irreducibility there is at least one such p. For this p, equation $(*)$ is valid $\Rightarrow |\lambda - a_{pp}| = \Lambda_p$

Due to irreducibility, this is true for all $p = 1 \dots n$ □

Diagonally dominant matrices

Definition

- A is *diagonally dominant* if for $i = 1 \dots n$,

$$|a_{ii}| \geq \sum_{\substack{j=1\dots n\\ j\neq i}} |a_{ij}|$$

- A is *strictly diagonally dominant* (sdd) if for $i = 1 \dots n$,

$$|a_{ii}| > \sum_{\substack{j=1\dots n\\ j\neq i}} |a_{ij}|$$

- A is *irreducibly diagonally dominant* (idd) if A is irreducible, for $i = 1 \dots n$,

$$|a_{ii}| \geq \sum_{\substack{j=1\dots n\\ j\neq i}} |a_{ij}|$$

  and for at least one r, $1 \leq r \leq n$,

$$|a_{rr}| > \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}|$$

A very practical nonsingularity criterion

Theorem: Let A be strictly diagonally dominant or irreducibly diagonally dominant. Then A is nonsingular.

If in addition, if $a_{ii} > 0$ for $i = 1 \dots n$, then all real parts of the eigenvalues of A are positive:

$$\mathrm{Re}\lambda_i > 0, \quad i = 1 \dots n$$

Proof:

Assume A strictly diagonally dominant. Then the union of the Gershgorin disks does not contain 0 and $\lambda = 0$ cannot be an eigenvalue.

As for the real parts, the union of the disks is

$$\bigcup_{i=1\dots n} \{\mu \in \mathbb{C} : |\mu - a_{ii}| \leq \Lambda_i\}$$

and $\mathrm{Re}\mu$ must be larger than zero if it should be contained.

A very practical nonsingularity criterion II

Assume A irreducibly diagonally dominant. Then, if 0 is an eigenvalue, by the Taussky theorem, we have $|a_{ii}| = \Lambda_i$ for all $i = 1 \dots n$. This is a contradiction as by definition there is at least one i such that $|a_{ii}| > \Lambda_i$

Obviously, all real parts of the eigenvalues must be ≥ 0. Therefore, if a real part is 0, it lies on the boundary of one disk. So by Taussky it must be contained in the boundary of all the disks and the imaginary axis. But there is at least one disk which does not touch the imaginary axis. □

Corollary

Theorem: If A is symmetric, sdd or idd, with positive diagonal entries, it is positive definite.

Proof: All eigenvalues of A are real, and due to the nonsingularity criterion, they must be positive, so A is positive definite. □.

Theorem on Jacobi matrix

Theorem: Let A be sdd or idd, and D its diagonal. Then

$$\rho(|I - D^{-1}A|) < 1$$

Proof: Let $B = (b_{ij}) = I - D^{-1}A$. Then

$$b_{ij} = \begin{cases} 0, & i = j \\ -\frac{a_{ij}}{a_{ii}}, & i \neq j \end{cases}$$

If A is sdd, then for $i = 1 \dots n$,

$$\sum_{j=1\dots n} |b_{ij}| = \sum_{\substack{j=1\dots n \\ j\neq i}} |\frac{a_{ij}}{a_{ii}}| = \frac{\Lambda_i}{|a_{ii}|} < 1$$

Therefore, $\rho(|B|) < 1$.

Theorem on Jacobi matrix II

If A is idd, then for $i = 1 \dots n$,

$$\sum_{j=1\dots n} |b_{ij}| = \sum_{\substack{j=1\dots n \\ j\neq i}} |\frac{a_{ij}}{a_{ii}}| = \frac{\Lambda_i}{|a_{ii}|} \leq 1$$

$$\sum_{j=1\dots n} |b_{rj}| = \frac{\Lambda_r}{|a_{rr}|} < 1 \text{ for at least one } r$$

Therefore, $\rho(|B|) <= 1$. Assume $\rho(|B|) = 1$ By Perron-Frobenius, 1 is an eigenvalue. As it is in the union of the Gershgorin disks

$$|\lambda| = 1 \leq \frac{\Lambda_i}{|a_{ii}|} \leq 1$$

it must lie on the boundary of this union, and by Taussky one has for all i

$$|\lambda| = 1 \leq \frac{\Lambda_i}{|a_{ii}|} = 1$$

which contradicts the idd condition. □

Jacobi method convergence

Corollary: Let A be sdd or idd, and D its diagonal. Assume that $a_{ii} > 0$ and $a_{ij} \leq 0$ for $i \neq j$. Then $\rho(I - D^{-1}A) < 1$, i.e. the Jacobi method converges.

Proof In this case, $|B| = B$. □.

Main Practical M-Matrix Criterion

Corollary: Let A be sdd or idd. Assume that $a_{ii} > 0$ and $a_{ij} \leq 0$ for $i \neq j$. Then A is an M-Matrix, i.e. A is nonsingular and $A^{-1} \geq 0$.
Proof: Let $B = \rho(I - D^{-1}A)$. Then $\rho(B) < 1$, therefore $I - B$ is nonsingular.

We have for $k > 0$:

$$I - B^{k+1} = (I - B)(I + B + B^2 + \dots + B^k)$$
$$(I - B)^{-1}(I - B^{k+1}) = (I + B + B^2 + \dots + B^k)$$

The left hand side for $k \to \infty$ converges to $(I - B)^{-1}$, therefore

$$(I - B)^{-1} = \sum_{k=0}^{\infty} B^k$$

As $B \geq 0$, we have $(I - B)^{-1} = A^{-1}D \geq 0$. As $D > 0$ we must have $A^{-1} \geq 0$. □

Regular splittings

- $A = M - N$ is a regular splitting if
  - M is nonsingular
  - M^{-1}, N are nonnegative, i.e. have nonnegative entries
- Regard the iteration $u_{k+1} = M^{-1}Nu_k + M^{-1}b$.
- We have $I - M^{-1}A = M^{-1}N$.

Convergence theorem for regular splitting

Theorem: Assume A is nonsingular, $A^{-1} \geq 0$, and $A = M - N$ is a regular splitting. Then $\rho(M^{-1}N) < 1$.

Proof: Let $G = M^{-1}N$. Then $A = M(I - G)$, therefore $I - G$ is nonsingular.

In addition

$$A^{-1}N = (M(I - M^{-1}N))^{-1}N = (I - M^{-1}N)^{-1}M^{-1}N = (I - G)^{-1}G$$

By Perron-Frobenius, there $\rho(G)$ is an eigenalue with a nonnegative eigenvector x . Thus,

$$0 \leq A^{-1}Nx = \frac{\rho(G)}{1 - \rho(G)}x$$

Therefore $0 \leq \rho(G) \leq 1$. As $I - G$ is nonsingular, $\rho(G) < 1$ $\square$.

Convergence rate

Corollary: $\rho(M^{-1}N) = \frac{\tau}{1+\tau}$ where $\tau = \rho(A^{-1}N)$.

Proof: Rearrange $\tau = \frac{\rho(G)}{1-\rho(G)}$ $\square$

Corollary: Let $A \geq 0$, $A = M_1 - N_1$ and $A = M_2 - N_2$ be regular splittings. If $N_2 \geq N_1 \geq 0$, then $1 > \rho(M_2^{-1}N_2) \geq \rho(M_1^{-1}N_1)$.

Proof: $\tau_2 = \rho(A^{-1}N_2) \geq \rho(A^{-1}N_1) = \tau_1$, $\frac{\tau}{1+\tau}$ is strictly increasing.

Application

Let A be an M-Matrix. Assume $A = D - E - F$.

- Jacobi method: $M = D$ is nonsingular, $M^{-1} \geq 0$. $N = E + F$ nonnegative $\Rightarrow$ convergence
- Gauss-Seidel: $M = D - E$ is an M-Matrix as $A \leq M$ and M has non-positive off-digonal entries. $N = F \geq 0$. $\Rightarrow$ convergence
- Comparison: $N_J \geq N_{GS}$ $\Rightarrow$ Gauss-Seidel converges faster.

Intermediate Summary

- Given some matrix, we now have some nice recipies to establish nonsingularity and iterative method convergence:
- **Check if the matrix is irreducible.**
  This is mostly the case for elliptic and parabolic PDEs.
- **Check for if matrix is strictly or irreducibly diagonally dominant**.
  If yes, it is in addition nonsingular.
- **Check if main diagonal entries are positive and off-diagonal entries are nonpositive.**
  If yes, in addition, the matrix is an M-Matrix, its inverse is nonnegative, and elementary iterative methods converge.

Example: 1D finite volume matrix:

We assume $\alpha > 0$.

$$\begin{pmatrix} \alpha + \frac{1}{h} & -\frac{1}{h} & & & & \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & \\ & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & \\ & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ & & & & -\frac{1}{h} & \frac{1}{h} + \alpha \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} \frac{h}{2} f_1 + \alpha v_1 \\ hf_2 \\ hf_3 \\ \vdots \\ hf_{N-2} \\ hf_{N-1} \\ \frac{h}{2} f_N + \alpha v_n \end{pmatrix}$$

- idd
- main diagonal entries are positive and off-diagonal entries are nonpositive

So this matrix is nonsingular, has the M-property, and we can e.g. apply the Jacobi iterative method to solve it.

Moreover, due to $A^{-1} \geq 0$, for $f \geq 0$ and $v \geq 0$ it follows that $u \geq 0$.

Incomplete LU factorizations (ILU)

Idea (Varga, Buleev, 1960):

- fix a predefined zero pattern
- apply the standard LU factorization method, but calculate only those elements, which do not correspond to the given zero pattern
- Result: incomplete LU factors L, U, remainder R:

$$A = LU - R$$

- Problem: with complete LU factorization procedure, for any nonsingular matrix, the method is stable, i.e. zero pivots never occur. Is this true for the incomplete LU Factorization as well ?

Stability of ILU

Theorem (Saad, Th. 10.2): If A is an M-Matrix, then the algorithm to compute the incomplete LU factorization with a given nonzero pattern

$$A = LU - R$$

is stable. Moreover, $A = LU - R$ is a regular splitting.

ILU(0)

- Special case of ILU: ignore any fill-in.
- Representation:

$$M = (\tilde{D} - E)\tilde{D}^{-1}(\tilde{D} - F)$$

- $\tilde{D}$ is a diagonal matrix (wich can be stored in one vector) which is calculated by the incomplete factorization algorithm.
- Setup:

```
for i=1...n do
    d(i)=a(i,i)
end

for i=1...n do
     d(i)=1.0/d(i)
     for j=i+1 ... n do
         d(j)=d(j)-a(i,j)*d(i)*a(j,i)
     end
end
```

ILU(0)

Solve $Mu = v$

```
for i=1...n do
    x=0
    for j=1 ... i-1 do
       x=x+a(i,j)*u(j)
    end
    u(i)=d(i)*(v(i)-x)
end

for i=n...1 do
    x=0
    for j=i+1...n do
       x=x+a(i,j)*u(j)
    end
    u(i)=u(i)-d(i)*x
```


ILU(0)

- Generally better convergence properties than Jacobi, Gauss-Seidel
- One can develop block variants
- Alternatives:
  - ILUM: ("modified"): add ignored off-diagonal entries to $\tilde{D}$
  - ILUT: zero pattern calculated dynamically based on drop tolerance
- Dependence on ordering
- Can be parallelized using graph coloring
- Not much theory: experiment for particular systems
- I recommend it as the default initial guess for a sensible preconditioner
- Incomplete Cholesky: symmetric variant of ILU

Preconditioners

- Leave this topic for a while now
- Hopefully, we well be able to discuss
  - Multigrid: gives $O(n)$ complexity in optimal situations
  - Domain decomposition: Structurally well suited for large scale parallelization

~

More general iteration schemes

Generalization of iteration schemes

- Simple iterations converge slowly
- For most practical purposes, Krylov subspace methods are used.
- We will introduce one special case and give hints on practically useful more general cases
- Material after J. Shewchuk: !An Introduction to the Conjugate Gradient Method Without the Agonizing Pain"

Solution of SPD system as a minimization procedure

Regard $Au = f$,where A is symmetric, positive definite. Then it defines a bilinear form $a : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$$a(u, v) = (Au, v) = v^T Au = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} v_i u_j$$

As A is SPD, for all $u \neq 0$ we have $(Au, u) > 0$.

For a given vector b, regard the function

$$f(u) = \frac{1}{2} a(u, u) - b^T u$$

What is the minimizer of f ?

$$f'(u) = Au - b = 0$$

- Solution of SPD system $\equiv$ minimization of f.

Method of steepest descent

- Given some vector u_i look for a new iterate u_{i+1}.
- The direction of steepest descend is given by $-f'(u_i)$.
- So look for u_{i+1} in the direction of $-f'(u_i) = r_i = b - Au_i$ such that it minimizes f in this direction, i.e. set $u_{i+1} = u_i + \alpha r_i$ with α choosen from

$$\begin{aligned}
0 &= \frac{d}{d\alpha} f(u_i + \alpha r_i) = f'(u_i + \alpha r_i) \cdot r_i \\
&= (b - A(u_i + \alpha r_i), r_i) \\
&= (b - Au_i, r_i) - \alpha(Ar_i, r_i) \\
&= (r_i, r_i) - \alpha(Ar_i, r_i) \\
\alpha &= \frac{(r_i, r_i)}{(Ar_i, r_i)}
\end{aligned}$$

Method of steepest descent: iteration scheme

$$\begin{aligned}
r_i &= b - Au_i \\
\alpha_i &= \frac{(r_i, r_i)}{(Ar_i, r_i)} \\
u_{i+1} &= u_i + \alpha_i r_i
\end{aligned}$$

Let $\hat{u}$ the exact solution. Define $e_i = u_i - \hat{u}$. Let $||u||_A = (Au, u)^{\frac{1}{2}}$ be the *energy norm* wrt. A.

Theorem The convergence rate of the method is

$$||e_i||_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^i ||e_0||_A$$

Conjugate directions

For steepest descent, there is no guarantee that a search direction $d_i = r_i = Ae_i$ is not used several times. If all search directions would be orthogonal, or, indeed, A-orthogonal, one could control this situation.

So, let $d_0, d_1 \dots d_{n-1}$ be a series of A-orthogonal (or conjugate) search directions, i.e. $(Ad_i, d_j) = 0,\ i \neq j$.

- Look for u_{i+1} in the direction of d_i such that it minimizes f in this direction, i.e. set $u_{i+1} = u_i + \alpha d_i$ with α choosen from

$$
\begin{aligned}
0 &= \frac{d}{d\alpha} f(u_i + \alpha d_i) = f'(u_i + \alpha d_i) \cdot d_i \\
&= (b - A(u_i + \alpha d_i), d_i) \\
&= (b - Au_i, d_i) - \alpha(Ad_i, d_i) \\
&= (r_i, d_i) - \alpha(Ad_i, d_i) \\
\alpha &= \frac{(r_i, d_i)}{(Ad_i, d_i)}
\end{aligned}
$$

Conjugate directions II

$e_0 = u_0 - \hat{u}$ (such that $Ae_0 = -r_0$) can be represented in the basis of the search directions:

$$e_0 = \sum_{i=0}^{n-1} \delta_j d_j$$

Projecting onto d_k in the A scalar product gives

$$
\begin{aligned}
(Ae_0, d_k) &= \sum_{i=0}^{n-1} \delta_j (Ad_j, d_k) \\
(Ae_0, d_k) &= \delta_k (Ad_k, d_k) \\
\delta_k &= \frac{(Ae_0, d_k)}{(Ad_k, d_k)} = \frac{(Ae_0 + \sum_{i<k} \alpha_i d_i, d_k)}{(Ad_k, d_k)} = \frac{(Ae_k, d_k}{(Ad_k, d_k)} \\
&= \frac{(r_k, d_k)}{(Ad_k, d_k)} \\
&= -\alpha_k
\end{aligned}
$$

Conjugate directions III

Then,

$$
\begin{aligned}
e_i &= e_0 + \sum_{j=0}^{i-1} \alpha_j d_j \\
&= -\sum_{j=0}^{n-1} \alpha_j d_j + \sum_{j=0}^{i-1} \alpha_j d_j \\
&= -\sum_{j=i}^{n-1} \alpha_j d_j
\end{aligned}
$$

So, the iteration consists in component-wise suppression of the error, and it must converge after n steps.

But by what magic we can obtain these d_i?

Conjugate directions V

Furthermore, we have

$$
\begin{aligned}
u_{i+1} &= u_i + \alpha_i d_i \\
e_{i+1} &= e_i + \alpha_i d_i \\
Ae_{i+1} &= Ae_i + \alpha_i Ad_i \\
r_{i+1} &= r_i - \alpha_i Ad_i
\end{aligned}
$$

Gram-Schmidt Orthogonalization

- Assume we have been given some linearly independent vectors $v_0, v_1 \dots v_{n-1}$.
- Set $d_0 = v_0$
- Define

$$d_i = v_i + \sum_{k=0}^{i-1} \beta_{ik} d_k$$

- For $j < i$, A-project onto d_j and require orthogonality:

$$
\begin{aligned}
(Ad_i, d_j) &= (Av_i, d_j) + \sum_{k=0}^{i-1} \beta_{ik} (Ad_k, d_j) \\
0 &= (Av_i, d_j) + \beta_{ij}(Ad_j, d_j) \\
\beta_{ij} &= -\frac{(Av_i, d_j)}{(Ad_j, d_j)}
\end{aligned}
$$

- If v_i are the coordinate unit vectors, this is Gaussian elimination!
- If v_i are arbitrary, they all must be kept in the memory

Conjugate gradients (Hestenes, Stiefel, 1952)

As Gram-Schmidt builds up d_i from d_j, $j < i$, we can choose $v_i = r_i$ – the residuals built up during the conjugate direction process.

Let $\mathcal{K}_i = \mathrm{span}\{d_0 \dots d_{i-1}\}$. Then, $r_i \perp \mathcal{K}_i$

But d_i are built by Gram-Schmidt from the residuals, so we also have $\mathcal{K}_i = \mathrm{span}\{r_0 \dots r_{i-1}\}$ and $(r_i, r_j) = 0$ for $j < i$.

From $r_i = r_{i-1} - \alpha_{i-1} Ad_{i-1}$ we obtain

$\mathcal{K}_i = \mathcal{K}_{i-1} \cup \mathrm{span}\{Ad_{i-1}\}$

This gives two other representations of $\mathcal{K}_i$:

$$
\begin{aligned}
\mathcal{K}_i &= \mathrm{span}\{d_0, Ad_0, A^2 d_0, \dots, A^{i-1} d_0\} \\
&= \mathrm{span}\{r_0, Ar_0, A^2 r_0, \dots, A^{i-1} r_0\}
\end{aligned}
$$

Such type of subspace of $\mathbb{R}^n$ is called *Krylov subspace*, and orthogonalization methods are more often called *Krylov subspace methods*.

Conjugate gradients II

Look at Gram-Schmidt under these conditions. The essential data are (setting $v_i = r_i$ and using $j < i$) $\beta_{ij} = -\frac{(Ar_i, d_j)}{(Ad_j, d_j)} = -\frac{(Ad_j, r_i)}{(Ad_j, d_j)}$.

Then, for $j < i$:

$$
\begin{aligned}
r_{j+1} &= r_j - \alpha_j Ad_j \\
(r_{j+1}, r_i) &= (r_j, r_i) - \alpha_j (Ad_j, r_i) \\
\alpha_j (Ad_j, r_i) &= (r_j, r_i) - (r_{j+1}, r_i) \\
(Ad_j, r_i) &= \begin{cases} -\frac{1}{\alpha_j}(r_{j+1}, r_i), & j+1 = i \\ \frac{1}{\alpha_j}(r_j, r_i), & j = i \\ 0, & \text{else} \end{cases} = \begin{cases} -\frac{1}{\alpha_{i-1}}(r_i, r_i), & j+1 = i \\ \frac{1}{\alpha_i}(r_i, r_i), & j = i \\ 0, & \text{else} \end{cases} \\
\beta_{ij} &= \begin{cases} \frac{1}{\alpha_{i-1}} \frac{(r_i, r_i)}{(Ad_{i-1}, d_{i-1})}, & j+1 = i \\ 0, & \text{else} \end{cases}
\end{aligned}
$$

Conjugate gradients III

For Gram-Schmidt we defined (replacing v_i by r_i):

$$
\begin{aligned}
d_i &= r_i + \sum_{k=0}^{i-1} \beta_{ik} d_k \\
&= r_i + \beta_{i,i-1} d_{i-1}
\end{aligned}
$$

So, the new orthogonal direction depends only on the previous orthogonal direction and the current residual. We don't have to store old residuals or search directions. In the sequel, set $\beta_i := \beta_{i,i-1}$.

We have

$$
\begin{aligned}
d_{i-1} &= r_{i-1} + \beta_{i-1} d_{i-2} \\
(d_{i-1}, r_{i-1}) &= (r_{i-1}, r_{i-1}) + \beta_{i-1}(d_{i-2}, r_{i-1}) \\
&= (r_{i-1}, r_{i-1}) \\
\beta_i &= \frac{1}{\alpha_{i-1}} \frac{(r_i, r_i)}{(Ad_{i-1}, d_{i-1})} = \frac{(r_i, r_i)}{(d_{i-1}, r_{i-1})} \\
&= \frac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}
\end{aligned}
$$

Conjugate gradients IV - The algorithm

Given initial value u_0, spd matrix A, right hand side b.

$$
\begin{aligned}
d_0 &= r_0 = b - Au_0 \\
\alpha_i &= \frac{(r_i, r_i)}{(Ad_i, d_i)} \\
u_{i+1} &= u_i + \alpha_i d_i \\
r_{i+1} &= r_i - \alpha_i A d_i \\
\beta_{i+1} &= \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)} \\
d_{i+1} &= r_{i+1} + \beta_{i+1} d_i
\end{aligned}
$$

At the i-th step, the algorithm yields the element from $e_0 + \mathcal{K}_i$ with the minimum energy error.

Theorem The convergence rate of the method is

$$||e_i||_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^i ||e_0||_A$$

where $\kappa = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is the spectral condition number.

Preconditioning

We discussed all these nice preconditioners - GS, Jacobi, ILU, may be there are more of them. Are they of any help here ?

Let M be spd. We can try to solve $M^{-1}Au = M^{-1}b$ instead of the original system.

But in general, $M^{-1}A$ is neither symmetric, nor definite. But there is a trick:

Let E be such that $M = EE^T$, e.g. its Cholesky factorization. Then, $\sigma(M^{-1}A) = \sigma(E^{-1}AE^{-T})$:

Assume $M^{-1}Au = \lambda u$. We have

$$(E^{-1}AE^{-T})(E^T u) = (E^T E^{-T})E^{-1}Au = E^T M^{-1} Au = \lambda E^T u$$

$\Leftrightarrow$ $E^T u$ is an eigenvector of $E^{-1}AE^{-T}$ with eigenvalue λ.

Good preconditioner: $M \approx A$ in the sense that $\kappa(M^{-1}A) << \kappa(A)$.

Preconditioned CG I

Now we can use the CG algorithm for the preconditioned system

$$E^{-1}AE^{-T}\tilde{x} = E^{-1}b$$

with $\tilde{u} = E^T u$

$$
\begin{aligned}
\tilde{d}_0 &= \tilde{r}_0 = E^{-1}b - E^{-1}AE^{-T}u_0 \\
\alpha_i &= \frac{(\tilde{r}_i, \tilde{r}_i)}{(E^{-1}AE^{-T}\tilde{d}_i, \tilde{d}_i)} \\
\tilde{u}_{i+1} &= \tilde{u}_i + \alpha_i \tilde{d}_i \\
\tilde{r}_{i+1} &= \tilde{r}_i - \alpha_i E^{-1}AE^{-T}\tilde{d}_i \\
\beta_{i+1} &= \frac{(\tilde{r}_{i+1}, \tilde{r}_{i+1})}{(\tilde{r}_i, \tilde{r}_i)} \\
\tilde{d}_{i+1} &= \tilde{r}_{i+1} + \beta_{i+1}\tilde{d}_i
\end{aligned}
$$

Not very practical as we need E

Preconditioned CG II

Assume $\tilde{r}_i = E^{-1}r_i$, $\tilde{d}_i = E^T d_i$, we get the equivalent algorithm

$$
\begin{aligned}
r_0 &= b - Au_0 \\
d_0 &= M^{-1}r_0 \\
\alpha_i &= \frac{(M^{-1}r_i, r_i)}{(Ad_i, d_i)} \\
u_{i+1} &= u_i + \alpha_i d_i \\
r_{i+1} &= r_i - \alpha_i A d_i \\
\beta_{i+1} &= \frac{(M^{-1}r_{i+1}, r_{i+1})}{(r_i, r_i)} \\
d_{i+1} &= M^{-1}r_{i+1} + \beta_{i+1} d_i
\end{aligned}
$$

It relies on the solution of the preconditioning system, the calculation of the matrix vector product and the calculation of the scalar product.

A few issues

Usually we stop the iteration when the residual r becomes small. However during the iteration, floating point errors occur which distort the calculations and lead to the fact that the accumulated residuals

$$r_{i+1} = r_i - \alpha_i A d_i$$

give a much more optimistic picture on the state of the iteration than the real residual

$$r_{i+1} = b - Au_{i+1}$$

C++ implementation

```
template < class Matrix, class Vector, class Preconditioner, class Real >
int  CG(const Matrix &A, Vector &x, const Vector &b,
   const Preconditioner &M, int &max_iter, Real &tol)
{ Real resid;
  Vector p, z, q;
  Vector alpha(1), beta(1), rho(1), rho_1(1);
  Real normb = norm(b);
  Vector r = b - A*x;
  if (normb == 0.0)   normb = 1;
  if ((resid = norm(r) / normb) <= tol) {
    tol = resid;
    max_iter = 0;
    return 0;
  }
  for (int i = 1; i <= max_iter; i++) {
    z = M.solve(r);
    rho(0) = dot(r, z);
    if (i == 1)
      p = z;
    else {
      beta(0) = rho(0) / rho_1(0);
      p = z + beta(0) * p;
    }
    q = A*p;
    alpha(0) = rho(0) / dot(p, q);
    x += alpha(0) * p;
    r -= alpha(0) * q;
    if ((resid = norm(r) / normb) <= tol) {
      tol = resid;
      max_iter = i;
      return 0;
    }
    rho_1(0) = rho(0);
  }
  tol = resid;   return 1;
}
```


C++ implementation II

- Available from `http://www.netlib.org/templates/cpp//cg.h`
- Slightly adapted for numcxx
- Available in numxx in the namespace netlib.

Unsymmetric problems

- By definition, CG is only applicable to symmetric problems.
- The biconjugate gradient (BICG) method provides a generalization:

Choose initial guess x_0, perform

$$
\begin{aligned}
r_0 &= b - A x_0 & \hat{r}_0 &= \hat{b} - \hat{x}_0 A^T \\
p_0 &= r_0 & \hat{p}_0 &= \hat{r}_0 \\
\alpha_i &= \frac{(\hat{r}_i, r_i)}{(\hat{p}_i, Ap_i)} & & \\
x_{i+1} &= x_i + \alpha_i p_i & \hat{x}_{i+1} &= \hat{x}_i + \alpha_i \hat{p}_i \\
r_{i+1} &= r_i - \alpha_i A p_i & \hat{r}_{i+1} &= \hat{r}_i - \alpha_i \hat{p}_i A^T \\
\beta_i &= \frac{(\hat{r}_{i+1}, r_{i+1})}{(\hat{r}_i, r_i)} & & \\
p_{i+1} &= r_{i+1} + \beta_i p_i & \hat{p}_{i+1} &= \hat{r}_{i+1} + \beta_i \hat{p}_i
\end{aligned}
$$

The two sequences produced by the algorithm are biorthogonal, i.e., $(\hat{p}_i, Ap_j) = (\hat{r}_i, r_j) = 0$ for $i \neq j$.

Unsymmetric problems II

- BiCG is very unstable an additionally needs the transposed matrix vector product, it is seldomly used in practice
- There is as well a preconditioned variant of BiCG which also needs the transposed preconditioner.
- Main practical approaches to fix the situation:
  - "Stabilize" BiCG $\to$ BiCGstab
  - tweak CG $\to$ Conjugate gradients squared (CGS)
  - Error minimization in Krylov subspace $\to$ Generalized Minimum Residual (GMRES)
- Both CGS and BiCGstab can show rather erratic convergence behavior
- For GMRES one has to keep the full Krylov subspace, which is not possible in practice $\Rightarrow$ restart strategy.
- From my experience, BiCGstab is a good first guess