Scientific Computing WS 2019/2020

Lecture 7

Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

Direct Solvers: Recap

Matrices from PDEs

- So far, we assumed that matrices are stored in a two-dimensional, $n \times n$ array of numbers
- This kind of matrices are also called *dense* matrices
- As we will see, matrices from PDEs (can) have a number of structural properties one can take advantage of when storing a matrix and solving the linear system

1D heat conduction

- v_L, v_R: ambient temperatures, α: heat transfer coefficient
- Second order boundary value problem in $\Omega = [0,1]$:

$$\begin{aligned} -u''(x) &= f(x) && \text{in}\Omega \\ -u'(0) + \alpha(u(0) - v_L) &= 0 \\ u'(1) + \alpha(u(1) - v_R) &= 0 \end{aligned}$$

- Let $h = \frac{1}{n-1}$, $x_i = x_0 + (i-1)h\ i = 1 \dots n$ be discretization points, let u_i approximations for $u(x_i)$ and $f_i = f(x_i)$
- Finite difference approximation:

$$\begin{aligned} -u'(0) + \alpha(u(0) - v_L) &\approx \frac{1}{h}(u_0 - u_1) + \alpha(u_0 - v_L) \\ -u''(x_i) - f(x_i) &\approx \frac{1}{h^2}(-u_{i+1} + 2u_i - u_{i-1}) - f_i \quad (i = 2 \dots n-1) \\ u'(1) + \alpha(u(1) - v_R) &\approx \frac{1}{h}(u_n - u_{n-1}) + \alpha(u_n - v_R) \end{aligned}$$

1D heat conduction: discretization matrix

- equations $2 \dots n-1$ multiplied by h
- only nonzero entries written

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

- Each row contains ≤ 3 elements
- Only $3n-2$ of n^2 elements are non-zero

General tridiagonal matrix

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

- To store matrix, it is sufficient to store only nonzero elements in three one-dimensional arrays for a_i, b_i, c_i, respectively

Gaussian elimination for tridiagonal systems

Gaussian elimination using arrays a, b, c as matrix storage ?

From what we have seen, this question arises in a quite natural way, and historically, the answer has been given several times

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

Progonka: derivation

- $a_i u_{i-1} + b_i u_i + c_i u_{i+1} = f_i \quad (i = 1 \dots n)$; $a_1 = 0$, $c_N = 0$
- 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}: $(a_i \alpha_i \alpha_{i+1} + b_i \alpha_{i+1} + c_i) u_{i+1} + a_i \alpha_i \beta_{i+1} + a_i \beta_i + b_i \beta_{i+1} - f_i = 0$
- This is true independently of u if

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

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

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

Progonka: realization

- Forward sweep:

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

  for $i = 2 \dots n-1$

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

- Backward sweep:

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

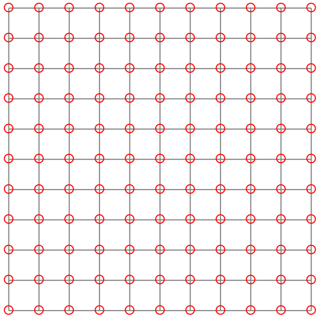
  for $n-1 \dots 1$:

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

Progonka: 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 difference grid



- Each discretization point has not more then 4 neighbours
- Matrix can be stored in five diagonals,
  LU factorization not anymore ≡ "fill-in"
- Certain iterative methods can take advantage of the regular and hierachical structure (multigrid) and are able to solve system in $O(n)$ operations
- Another possibility: fast Fourier transform with $O(n \log n)$ operations

Sparse matrices

- Tridiagonal and five-diagonal matrices can be seen as special cases of *sparse matrices*
- Generally they occur in finite element, finite difference and finite volume discretizations of PDEs on structured and unstructured grids
- Definition: 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: matrix-vector multiplication would use $O(n)$ instead of $O(n^2)$ operations

Sparse matrix questions

- What is a good storage 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

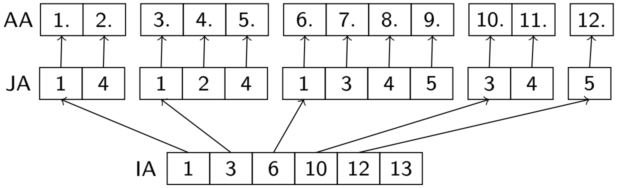
Y.Saad, Iterative Methods, p.92

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 `AA` and `JA` and `IA(n+1)=nnz+1`

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

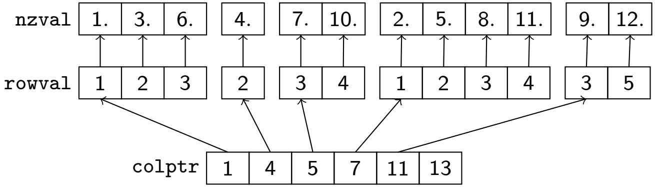


- Used in most sparse matrix solver packages
- CSC (Compressed Column Storage) uses similar principle but stores the matrix column-wise.

Compressed Sparse Column (CSC) format (Julia SparseMatrixCSC)

- real array `nzval`, length nnz: nonzero elements column by column
- integer array `rowval`, length nnz: row indices of the elements of `nzval`
- integer array `colptr`, length n+1: start indizes of each column in the arrays `rowval` and `colptr[n+1]=nnz+1`

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



$O(N)$ Sparse matrix - vector multiplication:

```
for icol=1:length(colptr)-1
    for rowind=colptr[icol]:colptr[icol+1]-1
        f[rowval[rowind]]+=nzval[rowind]*u[icol]
     end
 end
```

Sparse direct solvers

- Sparse direct solvers implement Gaussian elimination with different pivoting strategies
  - UMFPACK
  - Pardiso (omp + MPI parallel)
  - SuperLU (omp parallel)
  - MUMPS (MPI parallel)
  - Pastix
- Quite efficient for 1D/2D problems
- Essentially they implement the LU factorization algorithm
- They suffer from *fill-in*, especially for 3D problems: Let $nz(M)$ be the number of nonzero elements of a matrix A. Let $A = LU$ be an LU-Factorization. Then, as a rule, $nz(L + U) > NZ(A)$.
  - $\Rightarrow$ increased memory usage to store L,U
  - $\Rightarrow$ high operation cout

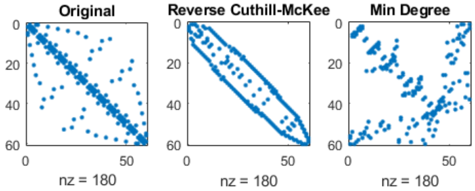
Sparse direct solvers: solution steps (Saad Ch. 3.6)

1. Pre-ordering
   - Decrease amount of non-zero elements generated by fill-in 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
   - Moderately 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

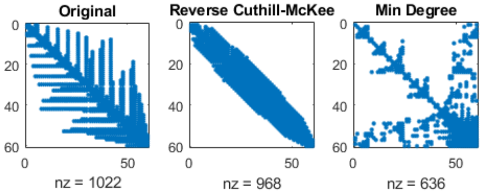
Sparse direct solvers: influence of reordering

- Sparsity patterns for original matrix with three different orderings of unknowns – number of nonzero elements (of course) independent of ordering:



https://de.mathworks.com

- Sparsity patterns for corresponding LU factorizations – number of nonzero elements depend original ordering!



https://de.mathworks.com

Sparse direct solvers: Complexity

- Complexity estimates depend on storage scheme, reordering etc.
- Sparse matrix - vector multiplication has complexity $O(N)$
- Some estimates can be given for from graph theory for discretizations of heat equation with $N = n^d$ unknowns on close to cubic grids in space dimension d
  - sparse LU factorization:

d	work	storage
1	$O(N) \mid O(n)$	$O(N) \mid O(n)$
2	$O(N^{\frac{3}{2}}) \mid O(n^3)$	$O(N \log N) \mid O(n^2 \log n)$
3	$O(N^2) \mid O(n^6)$	$O(N^{\frac{4}{3}}) \mid O(n^4)$

  - triangular solve: work dominated by storage complexity

d	work
1	$O(N) \mid O(n)$
2	$O(N \log N) \mid O(n^2 \log n)$
3	$O(N^{\frac{4}{3}}) \mid O(n^4)$

Source: J. Poulson, PhD thesis, http://hdl.handle.net/2152/ETD-UT-2012-12-6622

Iterative Solvers

Elements of iterative methods (Saad Ch.4)

Let $V = \mathbb{R}^n$ be equipped with the inner product $(\cdot, \cdot)$. Let A be an $n \times n$ nonsingular matrix.

Solve $Au = b$ iteratively. For this purpose, two components are needed:

- **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

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

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 $k = k + 1$, 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
- Preconditioner: $M = D$, where D is the main diagonal of $A \Rightarrow$

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

- Equivalent to the succesive (row by row) solution of

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

- Already calculated results not taken into account
- Alternative formulation with $A = M - N$:

$$\begin{aligned} u_{k+1} &= D^{-1}(E+F)u_k + D^{-1}b \\ &= M^{-1}Nu_k + M^{-1}b \end{aligned}$$

- Variable ordering does not matter

The Gauss-Seidel method

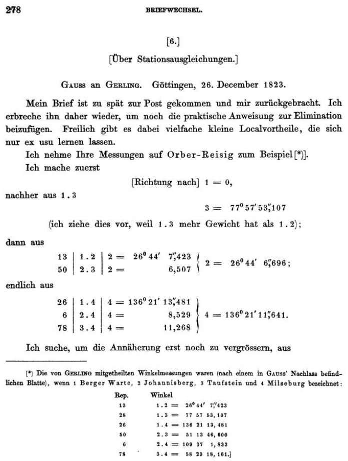
- Solve for main diagonal element row by row
- Take already calculated results into account

$$a_{ii}u_{k+1,i} + \sum_{j<i} a_{ij}u_{k+1,j} + \sum_{j>i} a_{ij}u_{k,j} = b_i \qquad (i = 1 \dots n)$$
$$(D-E)u_{k+1} - Fu_k = b$$

- May be it is faster
- Variable order probably matters
- Preconditioners: forward $M = D - E$, backward: $M = D - F$
- Splitting formulation: $A = M - N$
  forward: $N = F$, backward: $M = E$
- Forward case:

$$\begin{aligned} u_{k+1} &= (D-E)^{-1}Fu_k + (D-E)^{-1}b \\ &= M^{-1}Nu_k + M^{-1}b \end{aligned}$$

Gauss an Gerling I



[6.]

[Über Stationsausgleichungen.]

GAUSS an GERLING. Göttingen, 26. December 1823.

Mein Brief ist zu spät zur Post gekommen und mir zurückgebracht. Ich erbreche ihn daher wieder, um noch die praktische Anweisung zur Elimination beizufügen. Freilich gibt es dabei vielfache kleine Localvortheile, die sich nur ex usu lernen lassen.

Ich nehme Ihre Messungen auf Orber-Reisig zum Beispiel[*].

Ich mache zuerst

$$[\text{Richtung nach}]\ 1 = 0,$$

nachher aus 1.3

$$3 = 77^0\,57'\,53'',107$$

(ich ziehe dies vor, weil 1.3 mehr Gewicht hat als 1.2);

dann aus

$$\left.\begin{array}{r|c|ll} 13 & 1.2 & 2 = & 26^0\,44'\ \ 7'',423 \\ 50 & 2.3 & 2 = & \qquad\quad 6,507 \end{array}\right\}\ 2 = 26^0\,44'\ \ 6'',696;$$

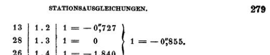
endlich aus

$$\left.\begin{array}{r|c|ll} 26 & 1.4 & 4 = & 136^0\,21'\,13'',481 \\ 6 & 2.4 & 4 = & \qquad\qquad 8,529 \\ 78 & 3.4 & 4 = & \qquad\quad\ 11,268 \end{array}\right\}\ 4 = 136^0\,21'\,11'',641.$$

Ich suche, um die Annäherung erst noch zu vergrössern, aus

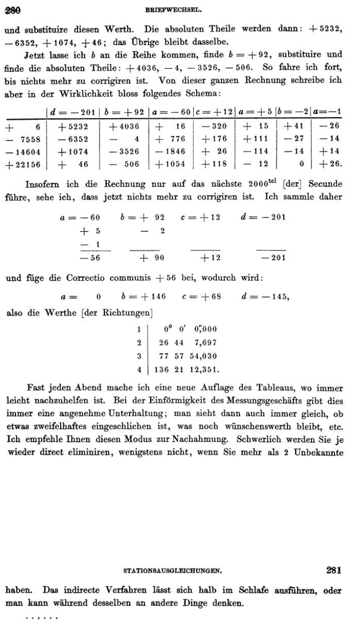
[*) Die von GERLING mitgetheilten Winkelmessungen waren (nach einem in GAUSS' Nachlass befindlichen Blatte), wenn 1 Berger Warte, 2 Johannisberg, 3 Taufstein und 4 Milseburg bezeichnet:

Rep.	Winkel
13	1.2 = 26° 44′ 7″,423
28	1.3 = 77 57 53, 107
26	1.4 = 136 21 13, 481
50	2.3 = 51 13 46, 600
6	2.4 = 109 37 1, 833
78	3.4 = 58 23 18, 161.]



$$\left.\begin{array}{r|c|ll} 13 & 1.2 & 1 = & -0'',727 \\ 28 & 1.3 & 1 = & \quad 0 \\ 26 & 1.4 & 1 = & -1,840 \end{array}\right\}\ 1 = -0'',855.$$

Gauss an Gerling II



und substituire diesen Werth. Die absoluten Theile werden dann: $+5232$, -6352, $+1074$, $+46$; das Übrige bleibt dasselbe.

Jetzt lasse ich b an die Reihe kommen, finde $b = +92$, substituire und finde die absoluten Theile: $+4036$, -4, -3526, -506. So fahre ich fort, bis nichts mehr zu corrigiren ist. Von dieser ganzen Rechnung schreibe ich aber in der Wirklichkeit bloss folgendes Schema:

	$d = -201$	$b = +92$	$a = -60$	$c = +12$	$a = +5$	$b = -2$	$a = -1$
$+6$	$+5232$	$+4036$	$+16$	-320	$+15$	$+41$	-26
-7558	-6352	-4	$+776$	$+176$	$+111$	-27	-14
-14604	$+1074$	-3526	-1846	$+26$	-114	-14	$+14$
$+22156$	$+46$	-506	$+1054$	$+118$	-12	0	$+26.$

Insofern ich die Rechnung nur auf das nächste 2000^{tel} [der] Secunde führe, sehe ich, dass jetzt nichts mehr zu corrigiren ist. Ich sammle daher

$$\begin{array}{rrrr} a = -60 & b = +92 & c = +12 & d = -201 \\ +5 & -2 & & \\ -1 & & & \\ \hline -56 & +90 & +12 & -201 \end{array}$$

und füge die Correctio communis $+56$ bei, wodurch wird:

$$a = 0 \qquad b = +146 \qquad c = +68 \qquad d = -145,$$

also die Werthe [der Richtungen]

1	$0^0\ 0'\ 0'',000$
2	$26\ 44\ 7,697$
3	$77\ 57\ 54,030$
4	$136\ 21\ 12,351.$

Fast jeden Abend mache ich eine neue Auflage des Tableaus, wo immer leicht nachzuhelfen ist. Bei der Einförmigkeit des Messungsgeschäfts gibt dies immer eine angenehme Unterhaltung; man sieht dann auch immer gleich, ob etwas zweifelhaftes eingeschlichen ist, was noch wünschenswerth bleibt, etc. Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigstens nicht, wenn Sie mehr als 2 Unbekannte

haben. Das indirecte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken.

. . . . . .

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

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

- Preconditioner:

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

- 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 $Au = b$.
- Let $e_k = u_k - \hat{u}$ be the error of the k-th iteration step

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

  resulting in

$$e_{k+1} = (I - M^{-1}A)^k e_0$$

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

Jordan canonical form of a matrix A

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

Theorem (Saad, Th. 1.8) B can be transformed to a block diagonal matrix consisting of p diagonal blocks $D_1 \dots D_p$, each associated with a distinct eigenvalue λ_i.

- Each of the diagonal blocks D_i has itself a block diagonal structure consisting of γ_i *Jordan blocks* $J_{i,1} \dots J_{i,\gamma_i}$.
- 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

$$X^{-1}BX = J = \begin{pmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_p \end{pmatrix}$$

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

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

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

Spectral radius and convergence

Definition The spectral radius $\rho(B)$ is the largest absolute value of any eigenvalue of B: $\rho(B) = \max_{\lambda \in \sigma(B)} |\lambda|$.

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

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

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

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

Spectral radius and convergence II

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

$$J_i^k = \sum_{j=0}^{l_{i-1}} \binom{k}{j} \lambda^{k-j} E^j$$

$$||J_i||^k \leq \sum_{j=0}^{l_{i-1}} \binom{k}{j} |\lambda|^{k-j} ||E||^j$$

But $\binom{k}{j} = \frac{k!}{j!(k-j)!} = \sum_{i=0}^{j} \begin{bmatrix} j \\ i \end{bmatrix} \frac{k^i}{j!} = p_j(k)$ is a polynomial of degree j in k
where 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, $p_j(k)|\lambda|^{k-j} \to 0$ $(k \to \infty)$ as exponential decay beats polynomial growth
$\square$.

Corollary from proof

Theorem (Saad, Th. 1.12)

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

□

Back to iterative methods

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

Convergence rate

Assume λ with $|\lambda| = \rho(I - M^{-1}A) < 1$ is the largest eigenvalue and has a single Jordan block of size l. Then the convergence rate is dominated by this Jordan block, and therein by the term with the lowest possible power in λ which due to $E^l = 0$ is

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

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

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, sufficient criterion for convergence

Assume A has positive real eigenvalues $0 < \lambda_{min} \leq \lambda_i \leq \lambda_{max}$.
E.g. A is symmetric, positive definite (spd).

- Let $\alpha > 0$, $M = \frac{1}{\alpha} I \Rightarrow I - M^{-1}A = I - \alpha A$
- Then for the eigenvalues μ_i of $I - \alpha A$ one has:

$$1 - \alpha\lambda_{max} \leq \mu_i \leq 1 - \alpha\lambda_{min}$$
$$\mu_i < 1$$

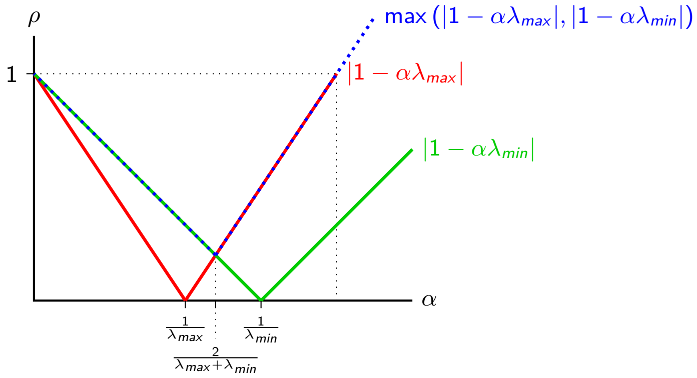
- We also need $1 - \alpha\lambda_{max} > -1$, so we must have $0 < \alpha < \frac{2}{\lambda_{max}}$.

Theorem. The Richardson iteration converges for any α with $0 < \alpha < \frac{2}{\lambda_{max}}$.

The convergence rate is $\rho = \max\left(|1 - \alpha\lambda_{max}|, |1 - \alpha\lambda_{min}|\right)$.

$\square$

Richardson iteration, choice of optimal parameter



- Due to $-(1-\alpha\lambda_{max}) > -(1-\alpha\lambda_{min})$ and $+(1-\alpha\lambda_{min}) > +(1-\alpha\lambda_{max})$,

$$\begin{aligned}\rho &= \max\left(|1-\alpha\lambda_{max}|, |1-\alpha\lambda_{min}|\right)\\ &= \max\left((1-\alpha\lambda_{max}), -(1-\alpha\lambda_{min})\right)\end{aligned}$$

- $1-\alpha\lambda_{max}$ is monotonically decreasing, the $-(1-\alpha\lambda_{min})$ increases, so the minimum must be at the intersection

$$1-\alpha\lambda_{max} = -1+\alpha\lambda_{min} \quad \Rightarrow \quad 2 = \alpha(\lambda_{max}+\lambda_{min})$$

Richardson iteration, choice of optimal parameter

Theorem. The optimal parameter is $\alpha_{opt} = \frac{2}{\lambda_{min}+\lambda_{max}}$.
For this parameter, the convergence factor is

$$\rho_{opt} = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}} = \frac{\kappa - 1}{\kappa + 1}$$

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

Spectral equivalence

Theorem. M, A spd. Assume the *spectral equivalence estimate*

$$0 < \gamma_{min}(Mu, u) \leq (Au, u) \leq \gamma_{max}(Mu, u)$$

Then for the eigenvalues μ_i of $M^{-1}A$ we have

$$\gamma_{min} \leq \mu_{min} \leq \mu_i \leq \mu_{max} \leq \gamma_{max}$$

and $\kappa(M^{-1}A) \leq \frac{\gamma_{max}}{\gamma_{\min}}$

Proof. Let the inner product $(\cdot, \cdot)_M$ be defined via $(u, v)_M = (Mu, v)$. In this inner product, $C = M^{-1}A$ is self-adjoint:

$$\begin{aligned}(Cu, v)_M &= (MM^{-1}Au, v) = (Au, v) = (M^{-1}Mu, Av) = (Mu, M^{-1}Av)\\ &= (u, M^{-1}A)_M = (u, Cv)_M\end{aligned}$$

Minimum and maximum eigenvalues can be obtained as Ritz values in the $(\cdot, \cdot)_M$ scalar product

$$\begin{aligned}\mu_{min} &= \min_{u\neq 0} \frac{(Cu, u)_M}{(u, u)_M} = \min_{u\neq 0} \frac{(Au, u)}{(Mu, u)} \geq \gamma_{min}\\ \mu_{max} &= \max_{u\neq 0} \frac{(Cu, u)_M}{(u, u)_M} = \max_{u\neq 0} \frac{(Au, u)}{(Mu, u)} \leq \gamma_{max}\end{aligned}$$

□

Matrix preconditioned Richardson iteration

M, A spd.

- Scaled Richardson iteration with preconditoner M

$$u_{k+1} = u_k - \alpha M^{-1}(Au_k - b)$$

- Spectral equivalence estimate

$$0 < \gamma_{min}(Mu, u) \leq (Au, u) \leq \gamma_{max}(Mu, u)$$

- $\Rightarrow \gamma_{min} \leq \lambda_i \leq \gamma_{max}$
- $\Rightarrow$ optimal parameter $\alpha = \frac{2}{\gamma_{max}+\gamma_{min}}$
- Convergence rate with optimal parameter: $\rho \leq \frac{\kappa(M^{-1}A)-1}{\kappa(M^{-1}A)+1}$
- This is one possible way for convergence analysis which at once gives convergence rates
- But ... how to obtain a good spectral estimate for a particular problem ?

1D heat conduction: spectrum

- Regard the $n \times n$ 1D heat conduction matrix with $h = \frac{1}{n-1}$ and $\alpha = \frac{1}{h}$ (easier to analyze).

$$A = \begin{pmatrix} \frac{2}{h} & -\frac{1}{h} & & & & \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & \\ & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & \\ & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ & & & & -\frac{1}{h} & \frac{2}{h} \end{pmatrix}$$

- Eigenvalues (tri-diagonal Toeplitz matrix):

$$\lambda_i = \frac{2}{h}\left(1 + \cos\left(\frac{i\pi}{n+1}\right)\right) \quad (i = 1 \dots n)$$

Source: A. Böttcher, S. Grudsky: Spectral Properties of Banded Toeplitz Matrices. SIAM,2005

- Express them in h: $n + 1 = \frac{1}{h} + 2 = \frac{1+2h}{h} \Rightarrow$

$$\lambda_i = \frac{2}{h}\left(1 + \cos\left(\frac{ih\pi}{1+2h}\right)\right) \quad (i = 1 \dots n)$$

1D heat conduction: spectral bounds estimate

- For $i = 1 \dots n$, the argument of cos is in $(0, \pi)$
- cos is monotonically decreasing in $(0, \pi)$, so we get λ_{max} for $i = 1$ and λ_{min} for $i = n = \frac{1+h}{h}$
- Therefore:

$$\lambda_{max} = \frac{2}{h}\left(1 + \cos\left(\pi \frac{h}{1+2h}\right)\right) \approx \frac{2}{h}\left(2 - \frac{\pi^2 h^2}{2(1+2h)^2}\right)$$

$$\lambda_{min} = \frac{2}{h}\left(1 + \cos\left(\pi \frac{1+h}{1+2h}\right)\right) \approx \frac{2}{h}\left(\frac{\pi^2 h^2}{2(1+2h)^2}\right)$$

Here, we used the Taylor expansion

$$cos(\delta) = 1 - \frac{\delta^2}{2} + O(\delta^4) \quad (\delta \to 0)$$

$$cos(\pi - \delta) = -1 + \frac{\delta^2}{2} + O(\delta^4) \quad (\delta \to 0)$$

and $\frac{1+h}{1+2h} = \frac{1+2h}{1+2h} - \frac{h}{1+2h} = 1 - \frac{h}{1+2h}$

Jacobi preconditioned Richardson for 1D heat conduction

- The Jacobi preconditioner just multiplies by $\frac{h}{2}$, therefore for $M^{-1}A$:

$$\mu_{max} \approx 2 - \frac{\pi^2 h^2}{2(1+2h)^2}$$
$$\mu_{min} \approx \frac{\pi^2 h^2}{2(1+2h)^2}$$

- Optimal parameter: $\alpha = \frac{2}{\lambda_{max}+\lambda_{min}} \approx 1 \ (h \to 0)$
- Good news: this is independent of h resp. n
- No need for spectral estimate in order to work with optimal parameter.
- Is this true beyond this special case ?

Jacobi for 1D heat conduction: convergence factor

- Condition number + spectral radius

$$\kappa(M^{-1}A) = \kappa(A) = \frac{4(1+2h)^2}{\pi^2 h^2} - 1$$

$$\rho(I - M^{-1}A) = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{\pi^2 h^2}{2(1+2h)^2}$$

- Bad news: $\rho \to 1 \quad (h \to 0)$
- Typical situation with second order PDEs:

$$\kappa(A) = O(h^{-2}) \quad (h \to 0)$$

$$\rho(I - D^{-1}A) = 1 - O(h^2) \quad (h \to 0)$$

- Mean square error of approximation $||u - u_h||_2 < h^\gamma$, in the simplest case $\gamma = 2$.

Iterative solver complexity I

- Solve linear system iteratively until $||e_k|| = ||(I - M^{-1}A)^k e_0|| \leq \epsilon$

$$
\begin{aligned}
\rho^k e_0 &\leq \epsilon \\
k \ln \rho &< \ln \epsilon - \ln e_0 \\
k \geq k_\rho &= \left\lceil \frac{\ln e_0 - \ln \epsilon}{\ln \rho} \right\rceil
\end{aligned}
$$

- $\Rightarrow$ we need at least k_ρ iteration steps to reach accuracy ϵ
- Optimal iterative solver complexity - assume:
  - $\rho < \rho_0 < 1$ independent of h resp. N
  - A sparse ($A \cdot u$ has complexity $O(N)$)
  - Solution of $Mv = r$ has complexity $O(N)$.

$\Rightarrow$ Number of iteration steps k_ρ independent of N
$\Rightarrow$ Overall complexity $O(N)$

Iterative solver complexity II

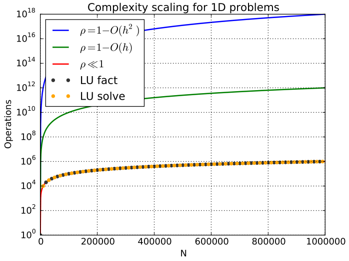
- Assume
  - $\rho = 1 - h^\delta \Rightarrow \ln \rho \approx -h^\delta \rightarrow k_\rho = O(h^{-\delta})$
  - d: space dimension $\Rightarrow h \approx N^{-\frac{1}{d}} \Rightarrow k_\rho = O(N^{\frac{\delta}{d}})$
  - $O(N)$ complexity of one iteration step (e.g. Jacobi, Gauss-Seidel)

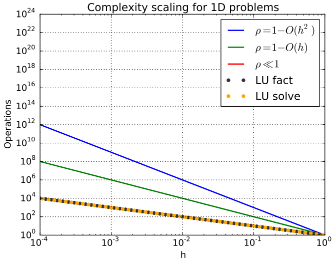
  $\Rightarrow$ Overall complexity $O(N^{1+\frac{\delta}{d}}){=}O(N^{\frac{d+\delta}{d}})$
- Jacobi: $\delta = 2$
- Hypothetical "Improved iterative solver" with $\delta = 1$?
- Overview on complexity estimates

dim	$\rho = 1 - O(h^2)$	$\rho = 1 - O(h)$	LU fact.	LU solve
1	$O(N^3)$	$O(N^2)$	$O(N)$	$O(N)$
2	$O(N^2)$	$O(N^{\frac{3}{2}})$	$O(N^{\frac{3}{2}})$	$O(N \log N)$
3	$O(N^{\frac{5}{3}})$	$O(N^{\frac{4}{3}})$	$O(N^2)$	$O(N^{\frac{4}{3}})$

Solver complexity scaling for 1D problems

dim	$\rho = 1 - O(h^2)$	$\rho = 1 - O(h)$	LU fact.	LU solve
1	$O(N^3)$	$O(N^2)$	$O(N)$	$O(N)$

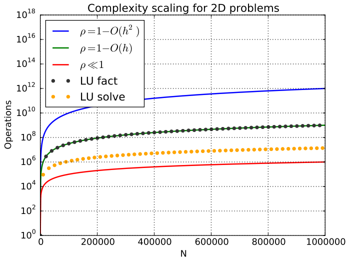


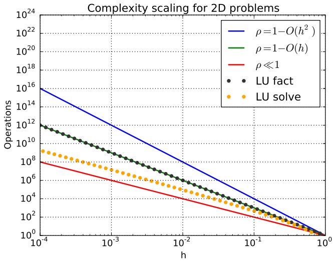


- Direct solvers significantly better than iterative ones

Solver complexity scaling for 2D problems

dim	$\rho = 1 - O(h^2)$	$\rho = 1 - O(h)$	LU fact.	LU solve
2	$O(N^2)$	$O(N^{\frac{3}{2}})$	$O(N^{\frac{3}{2}})$	$O(N \log N)$

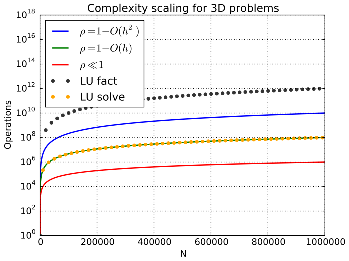


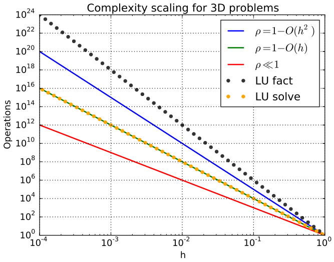


- Direct solvers better than simple iterative solvers (Jacobi etc.)
- Direct solves on par with improved iterative solvers

Solver complexity scaling for 3D problems

dim	$\rho = 1 - O(h^2)$	$\rho = 1 - O(h)$	LU fact.	LU solve
3	$O(N^{\frac{5}{3}})$	$O(N^{\frac{4}{3}})$	$O(N^2)$	$O(N^{\frac{4}{3}})$





- LU factorization is expensive
- LU solve on par with improved iterative solvers

What could be done ?

- Find optimal iterative solver with $O(N)$ complexity
- Find "improved preconditioner" with $\kappa(M^{-1}A) = O(h^{-1}) \Rightarrow \delta = 1$
- Find "improved iterative scheme": with $\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$:

  For Jacobi, we had $\kappa = X^2 - 1$ where $X = \frac{2(1+2h)}{\pi h} = O(h^{-1})$.

$$\begin{aligned}
\rho &= 1 + \frac{\sqrt{X^2-1}-1}{\sqrt{X^2-1}+1} - 1\\
&= 1 + \frac{\sqrt{X^2-1}-1-\sqrt{X^2-1}-1}{\sqrt{X^2-1}+1}\\
&= 1 - \frac{1}{\sqrt{X^2-1}+1}\\
&= 1 - \frac{1}{X\left(\sqrt{1-\frac{1}{X^2}}+\frac{1}{X}\right)}\\
&= 1 - O(h)
\end{aligned}$$

$\Rightarrow \delta = 1$