Chapter 5: Iterative Methods For Large Linear Systems

Michael Hanke

Mathematical Models, Analysis and Simulation, Part I

Model Problems

Read: Strang, p. 283–285

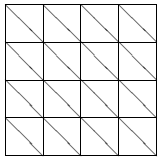
Use the Poisson equation with homogeneous boundary conditions:

$$-\Delta u = f, \quad x \in \Omega, \text{ subject to } u = 0 \text{ on } \Gamma = \partial\Omega.$$

1D: $\Omega = [0,1]$. Use linear finite elements on equidistant subintervals with step size $h = 1/(N+1)$. $\mathbf{A}$ becomes a tridiagonal $N \times N$-matrix,

$$\mathbf{A} = \frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{pmatrix}.$$

2D: $\Omega = [0,1]^2$. Use linear finite elements on the following triangulation:



Model Problems (cont.)

Using an equidistant subdivision with $h = 1/(N+1)$ in both x- and y-direction, then $\mathbf{A}$ becomes a block tridiagonal $n \times n$-matrix with $n = N^2$:

$$\mathbf{A} = \begin{pmatrix} \mathbf{B} & -\mathbf{I} & 0 & \dots & \dots & 0 \\ -\mathbf{I} & \mathbf{B} & -\mathbf{I} & 0 & \dots & 0 \\ 0 & -\mathbf{I} & \mathbf{B} & -\mathbf{I} & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & -\mathbf{I} & \mathbf{B} & -\mathbf{I} \\ 0 & \dots & \dots & 0 & -\mathbf{I} & \mathbf{B} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 4 & -1 & 0 & \dots & \dots & 0 \\ -1 & 4 & -1 & 0 & \dots & 0 \\ 0 & -1 & 4 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & -1 & 4 & -1 \\ 0 & \dots & \dots & 0 & -1 & 4 \end{pmatrix}.$$

Optimal Computational Complexity

Assume that our solution is an n-dimensional vextor $\mathbf{x}$.

An algorithm has optimal computational complexity if

#flops $\sim n$.

Direct Methods: Cholesky Factorization

- In both cases, $\mathbf{A}$ is symmetric and positive definite (spd).
- No pivotization necessary!
- *Computational complexity* for an $n \times n$-matrix with (half) band width w:

$$\boxed{O(w^2 n)}$$

- **1D:** $w = 1, n = N$: Complexity $O(n)$
  2D: $w = N, n = N^2$: Complexity $O(N^4) = O(n^2)$
- Cholesky factorization is optimal in 1D, but too expensive for large 2D problems. It becomes even worse in 3D.

Exercise: Assume that the number of grid points in x- and y-direction is N and M, respectively, with $N \neq M$. What is the computational complexity? Does it depend on the ordering?

Iterative Methods

Read: Strang, p 563–570, 592–594, (586–591)

- Consider the system $\mathbf{Ax} = \mathbf{b}$ with the exact solution $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.
- *Idea*: Try to construct a sequence $\mathbf{x^k}$ such that

$$\lim_{k\to\infty} \mathbf{x^k} = \mathbf{x}^*$$

  as fast as possible.
- Justification: $\mathbf{x}^*$ is only a discrete approximation to a continuous function u.
- Construction principles:
  - The classical way: Matrix splitting
  - The dynamic interpretation: Relaxation methods
  - The modern way: Minimization methods

Relaxation

- Observation: For any spd (e.g., mass) matrix $\mathbf{M}$ and any $\mathbf{x}^0 \in \mathbb{R}^n$,

$$\boxed{\mathbf{x}^* = \lim_{t\to\infty} \mathbf{x}(t) \text{ where } \mathbf{M}\frac{d\mathbf{x}}{dt} = \mathbf{b} - \mathbf{Ax}, \quad \mathbf{x}(0) = \mathbf{x}^0}$$

- *Damped Richardson iteration*: Let $\mathbf{M} = \mathbf{I}$, use forward Euler with time step $\Delta t = \omega$:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \omega(\mathbf{b} - \mathbf{Ax}^k), k = 0, 1, \ldots$$

- Error estimation:
  - Error: $\mathbf{e}^k := \mathbf{x}^k - \mathbf{x}^*$
  - How many iterations K must be taken such that $\|\mathbf{e}^K\| \leq \texttt{tol}$?
  - What is the optimal ω?

Convergence Analysis

- Let $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of $\mathbf{A}$ and $\mathbf{v}^1, \ldots, \mathbf{v}^n$ the eigenvectors.
- Define

$$\mathbf{e}^k = \sum_{j=1}^{n} c_{kj}\mathbf{v}^j.$$

  Consequently, (Prove!)

$$c_{k+1,j} = \underbrace{(1 - \omega\lambda_j)}_{g_j} c_{kj}.$$

- Convergence:

$$\lim_{k\to\infty} \mathbf{x^k} = \mathbf{x}^* \text{ iff } \max_{j=1,\ldots,n} |g_j| < 1$$

$$\text{iff } \omega < \frac{2}{\lambda_n}$$

- $g_{\max} = \max_{j=1,\ldots,n} |g_j|$ measures the speed of convergence: The smaller $g_{\max}$, the faster the convergence.

Convergence Analysis (cont.)

- Fastest convergence:

$$\boxed{\omega = \frac{2}{\lambda_1 + \lambda_n}, g_{\max} = \frac{\kappa_2(\mathbf{A}) - 1}{\kappa_2(\mathbf{A}) + 1}}$$

  where $\kappa_2(\mathbf{A}) = \frac{\lambda_n}{\lambda_1}$ is the Euclidean condition number of $\mathbf{A}$.
- For our model example, the eigenvalues are known. In 1D:

$$\lambda_j = \frac{1}{h}(2 - 2\cos j\pi h)$$

  such that

$$\lambda_n \approx \frac{4}{h}, \lambda_1 \approx h\pi^2$$

$$\Rightarrow \boxed{\kappa_2(\mathbf{A}) = O(h^{-2})}$$

- A similar estimate holds true in 2D and 3D.
- Conclusion: Number of iterations is

$$\boxed{K = O(\kappa_2(\mathbf{A}) \log \mathtt{tol}) = O(h^{-2} \log \mathtt{tol}).}$$

- *This is much too slow!!*
- Note: For special choices of ω, we obtain the Jacobi iteration.

Minimization: Steepest Descent

- Property:

$$\boxed{\mathbf{x}^* \text{ minimizes } P(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}}$$

- Minimization scheme:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$$

  with $\mathbf{d}^k$ (search direction) and $\alpha_k > 0$ (step size) constructed appropriately.
- *Direction of steepest descent* $\mathbf{d}^k$:
  $P(\mathbf{x}^k + \alpha\mathbf{d}^k)$ should be decrease as fast as possible in direction $\mathbf{d}^k$:

$$\frac{1}{\|\mathbf{d}^k\|} \frac{dP(\mathbf{x}^k + \alpha\mathbf{d}^k)}{d\alpha}\bigg|_{\alpha=0} \to \min! \text{ wrt } \mathbf{d}^k.$$

  Solution: $\mathbf{d}^k = -(\nabla P(\mathbf{x}^k))^T$. In our case:

$$\boxed{\mathbf{d}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k = \mathbf{r}^k} \text{ (residual)}$$

- Exercise: Prove these results!

Steepest Decent (cont.)

- Optimal step size

$$\alpha_k = \arg\min P(\mathbf{x}^k + \alpha\mathbf{d}^k)$$

  Solution:

$$\boxed{\alpha_k = \frac{\mathbf{r}^{kT}\mathbf{d}^k}{\mathbf{d}^{kT}\mathbf{A}^k\mathbf{d}^k}}$$

- Exercise: Prove this!
- Speed of convergence: (Proof: Luenberger, p 152)

$$\boxed{\|\mathbf{e}^{k+1}\|_E \le \frac{\kappa_2(\mathbf{A}) - 1}{\kappa_2(\mathbf{A}) + 1}\|\mathbf{e}^k\|_E}$$

  This is as slow as Richardson iteration!
- Note: The SOR method can be obtained with appropriately chosen search directions.

Speeding Up Steepest Descent: Conjugate Gradients

- Idea: Try to find a new search direction which uses also previous informations,

$$\mathbf{d}^{k+1} = -\mathbf{r}^{k+1} + \beta_k \mathbf{d}^k$$

  for a well-chosen β_k.
- Note: $\beta_k = 0$ is the steepest descent direction.
- The optimal choice is determined by

$$\boxed{\mathbf{d}^{iT}\mathbf{A}\mathbf{d}^j = 0 \text{ for } i \ne j} \text{ ($\mathbf{A}$-conjugacy)}$$

- The resulting algorithm can be implemented very efficiently.

Speed of Convergence

- Speed of convergence:

$$\|\mathbf{e}^{k+1}\|_E \leq \frac{\sqrt{\kappa_2(\mathbf{A})}-1}{\sqrt{\kappa_2(\mathbf{A})}+1}\|\mathbf{e}^k\|_E$$

- In our model example:

$$K = O(\sqrt{\kappa_2(\mathbf{A})}\log \texttt{tol}) = O(h^{-1}\log \texttt{tol}).$$

  This is faster, but slow.
- The real way out is *preconditioning*: Reformulate the problem such that the spectral radius is reduced.
- *The method of conjugate gradients is one of the most successful iterative methods for discretized partial differential equations.*

Comparison of Some Iterative Methods

For our model example in 2D, the computational complexity can be characterized as follows:

method	complexity
Cholesky factorization	$O(n^2)$
Jacobi	$O(n^2 \log n)$
Gauss-Seidel	$O(n^2 \log n)$
SOR (with ω_{opt})	$O(n^{3/2} \log n)$
Conjugate Gradients	$O(n^{3/2} \log n)$
Preconditioned CG	$O(n^{5/4} \log n)$
WISH	$O(n)$

Here, $n = N^2 \approx h^{-2}$.

Note: For our model example there exist especially adapted methods which obtain (nearly) optimal computational complexity, so-called *fast Poisson solvers*.

Q: Can one do better?

A: Yes. An example is Multigrid Methods. Will be considered in other courses.