SI231b: Matrix Computations

Lecture 22: Iterative Methods for Linear Systems

Yue Qiu

qiuyue@shanghaitech.edu.cn

School of Information Science and Technology

ShanghaiTech University

Nov. 30, 2021

Why Iterative Methods

We have learned in previous lectures (Lecture 5 – 8) to solve

$$\mathrm{Ax} = \mathrm{b},$$

using LU decomposition and its variants (LDLT/Cholesky factorization), where $\mathrm{A} \in \mathbb{R}^{n \times n}$ and is nonsingular.

Applying matrix factorization to solve linear systems belongs to the category of *direct methods*.

- returns exact solution $\mathrm{x} = \mathrm{A}^{-1}\mathrm{b}$ (assuming no round-off error)

Recall from Lecture 5 – 8 that the computational complexity of LU factorization and its variants is $\mathcal{O}(n^3)$, and the storage cost is $\mathcal{O}(n^2)$

- not affordable for large n

Why Iterative Methods

Thumbnail History of matrix computations in the 20th century:

- 1950, $n = 20$, J. H. Wilkinson

  Pilot ACE (first computer in UK)
- 1965, $n = 200$, G. Forsythe and C. Moler

  Computer Solution of Linear Algebraic Systems
- 1980, $n = 2,000$, LINPACK

  Written in Fortran, by J. Dongarra, J. Bunch, C. Moler, and G. Stewart
- 1995, $n = 20,000$, LAPACK (Linear Algebra PACKage)

  Widely used by Matlab/Python/TensorFlow/PyTorch $\cdots\cdots$

Iterative methods compute an approximate solution with less computational and storage cost.

To solve $Ax = b$, iterative methods generate a sequence of approximate solutions $\{x^{(k)}\}$ that converges to $A^{-1}b$,

- A is typically involved only in the context of matrix-vector multiplication
- attractive when A is large and sparse

Main concerns on iterative methods

- rate of convergence
- amount of computations per iteration
- required storage

Main Idea of Iterative Methods

Given a linear system

$$Ax = b, \tag{$\dagger$}$$

find another matrix B and a vector c such that

1. The matrix $I - B$ is nonsingular
2. The unique solution $A^{-1}b$ is identical to the solution of the system

$$x = Bx + c, \tag{$\ddagger$}$$

and starting from any vector x_0, compute the sequence $\{x^{(k)}\}$ via

$$x_{k+1} = Bx_k + c, \quad k \in \mathbb{N}. \tag{$\sharp$}$$

Under certain conditions, the sequence $\{x^{(k)}\}$ converges to the unique solution of $x = Bx + c$, and thus of $Ax = b$.

The matrix B is called the *iteration matrix*, and the iterative form ($\sharp$) is said to be *consistent* with ($\dagger$) if $c = (I - B)A^{-1}b$.

Consistency alone does not suffice to ensure the convergence of the iterative method (♯).

Example 1: to solve the linear system $2\mathsf{I}\mathsf{x} = \mathsf{b}$, consider the following iterative method

$$\mathsf{x}^{(k+1)} = -\mathsf{x}^{(k)} + \mathsf{b},$$

which is obviously consistent. This scheme is not convergent for any choice of the initial guess.

Example 2: for the same linear system $2\mathsf{I}\mathsf{x} = \mathsf{b}$, consider the following iterative method

$$\mathsf{x}^{(k+1)} = \frac{1}{2}\mathsf{x}^{(k)} + \frac{1}{4}\mathsf{b},$$

which is obviously consistent. This scheme is convergent for any choice of the initial guess.

The iteration matrix $\mathsf{B} = -\mathsf{I}$ for Example 1 and $\mathsf{B} = \frac{1}{2}\mathsf{I}$ for Example 2.

Theorem: Let ($\sharp$) be a consistent iterative method. The following statements are equivalent:

1. the iterative method is convergent.
2. the spectral radius of B denoted by $\rho(\mathrm{B})$ satisfies $\rho(\mathrm{B}) < 1$
3. $\|\mathrm{B}\| < 1$, for some subordinate matrix norm[1] $\|\cdot\|$.

The *spectral radius* of a square matrix is the largest absolute value of its eigenvalues, i.e., $\rho(\mathrm{A}) = \max|\lambda(\mathrm{A})|$. It determines the convergence rate. We can apply the following lemma to help to prove the theorem above.

Lemma: For any square matrix B, the following conditions are equivalent:

1. $\lim_{k\to\infty} \mathrm{B}^k = 0$.
2. $\lim_{k\to\infty} \mathrm{B}^k \mathrm{v} = 0$ for all vectors v.
3. $\rho(\mathrm{B}) < 1$
4. $\|\mathrm{B}\| < 1$, for some subordinate matrix norm $\|\cdot\|$.

[1] subordinate matrix norms are consistent with norms that induce them

Splitting Schemes

Many iterative methods to solve $\mathsf{A}\mathsf{x} = \mathsf{b}$ originate from the splitting $\mathsf{A} = \mathsf{M} - \mathsf{N}$, and can be written in the form

$$\mathsf{M}\mathsf{x}^{(k+1)} = \mathsf{N}\mathsf{x}^{(k)} + \mathsf{b},$$

with an initial guess/start $\mathsf{x}^{(0)}$.

To make the iterative methods practical, the matrix M should be easy to invert. The iteration matrix is given by $\mathsf{B} = \mathsf{M}^{-1}\mathsf{N}$.

Based on different splittings of A, we have the following iterative methods that will be introduced in this lecture.

- Jacobi Iteration
- Gauss-Seidel Iteration
- Successive Over-relaxation (SOR) Iteration

Jacobi Iteration

The Jacobi iteration splits the matrix A in the form

$$\mathsf{A} = \mathsf{D_A} - \mathsf{L_A} - \mathsf{U_A},$$

where

- $\mathsf{D_A}$ is the diagonal part of A
- $-\mathsf{L_A}$ is the strictly lower-triangular part of A
- $-\mathsf{U_A}$ is the strictly upper-triangular part of A

Then the Jacobi iteration takes the form

$$\mathsf{D_A}\mathsf{x}^{(k+1)} = (\mathsf{L_A} + \mathsf{U_A})\mathsf{x}^{(k)} + \mathsf{b},$$

and the iteration matrix $\mathsf{B} = \mathsf{D_A}^{-1}(\mathsf{L_A} + \mathsf{U_A})$

Key Insight of Jacobi Iteration

- assume $a_{ii} \neq 0$ for all i
- observe

$$\mathrm{b} = \mathrm{A}\mathrm{x} \quad \Leftrightarrow \quad b_i = a_{ii}x_i + \sum_{j \neq i} a_{ij}x_j, \quad i = 1, \ldots, n$$

$$\Leftrightarrow \quad x_i = \left(b_i - \sum_{j \neq i} a_{ij}x_j \right) / a_{ii}, \quad i = 1, \ldots, n \qquad (\natural)$$

- motivation: put $\mathrm{x}^{(k+1)}$ and $\mathrm{x}^{(k)}$ on the left and eight side of $(\natural)$, respectively, i.e.,

$$x_i^{(k+1)} = \left(b_i - \sum_{j \neq i} a_{ij}x_j^{(k)} \right) / a_{ii}, \quad i = 1, \ldots, n \qquad (\$)$$

- a compact form for $(\$)$ is given by $\mathrm{x}^{(k+1)} = \mathrm{D_A}^{-1} \Big((\mathrm{L_A} + \mathrm{U_A})\mathrm{x}^{(k)} + \mathrm{b} \Big)$.

- Each Jacobi iteration costs
  - $\mathcal{O}(n^2)$ for dense A
  - $\mathcal{O}(nnz(\mathrm{A}))$ for sparse A, here $nnz(\mathrm{A})$ represents the number of nonzero entries of A
- The Jacobi iteration can be computed in parallel or in a distributed fashion.
- Convergence of Jacobi iteration
  - does not converge in general
  - converges when A is strictly diagonal dominant (recall?)

Theorem: If $\mathrm{A} \in \mathbb{R}^{n\times n}$ is strictly diagonal dominant, then the Jacobi iteration converges to $\mathrm{x} = \mathrm{A}^{-1}\mathrm{b}$.

Proof ?

Gauss-Seidel Iteration

The Jacobi iteration splits the matrix A in the form

$$A = D_A + L_A - U_A,$$

where

- D_A is the diagonal part of A
- L_A is the strictly lower-triangular part of A
- $-U_A$ is the strictly upper-triangular part of A

Then the Gauss-Seidel iteration takes the form

$$(D_A + L_A)\,x^{(k+1)} = U_A x^{(k)} + b,$$

and the iteration matrix $B = (D_A + L_A)^{-1} U_A$.

Motivation of Gauss-Seidel Iteration

Recall the Jacobi iteration

$$x_i^{(k+1)} = \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right) / a_{ii}, \quad i = 1, \ldots, n$$

While computing $x_i^{(k+1)}$, the results $x_j^{(k+1)}$ $(j < i)$ are already available.

Modification of the Jacobi iteration

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

This in turn gives the Gauss-Seidel iteration

$$(\mathrm{D_A} + \mathrm{L_A})\, \mathrm{x}^{(k+1)} = \mathrm{U_A} \mathrm{x}^{(k)} + \mathrm{b}.$$

More on Gauss-Seidel Iteration

- Gauss-Seidel iteration is computationally more expensive than Jacobi iteration.
  - $\mathcal{O}(n^2)$ for dense A
  - $\mathcal{O}(nnz(\mathrm{A}))$ for sparse A
- Convergence of Gauss-Seidel iteration
  - does not converge in general
  - if the Jacobi method converges, Gauss-Seidel often converges faster. However, there are examples where Jacobi converges faster than Gauss-Seidel.
  - converges when A is symmetric positive definite (recall?)

Theorem: If $\mathrm{A} \in \mathbb{R}^{n \times n}$ is symmetric positive definite, then the Gauss-Seidel iteration converges to $\mathrm{x} = \mathrm{A}^{-1}\mathrm{b}$ for any $\mathrm{x}^{(0)}$.

Proof: cf. Theorem 11.2.3 in [**Golub & van Loan 13'**].

SOR Iteration

In the Gauss-Seidel iteration, we split $A = D_A + L_A - U_A$. The spectral radius of $(D_A + L_A)^{-1}U_A$ may be close to 1, which results in slow convergence.

To accelerate the convergence, splitting A in the following way

$$A = \left(\frac{1}{\omega}D_A + L_A\right) - \left((\frac{1}{\omega} - 1)D_A + U_A\right).$$

This defines the *successive over-relaxation* (SOR) iteration

$$\left(\frac{1}{\omega}D_A + L_A\right)x^{(k+1)} = \left((\frac{1}{\omega} - 1)D_A + U_A\right)x^{(k)} + b$$

- $\omega = 1$, turns into Gauss-Seidel iteration
- motivation of SOR is to minimize the spectral radius of the iteration matrix

$$\left(\frac{1}{\omega}D_A + L_A\right)^{-1}\left((\frac{1}{\omega} - 1)D_A + U_A\right)$$

Key Insight of SOR Iteration

The SOR iteration update:

$$x_i^{(k+1)} = \omega \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right) / a_{ii} + (1-\omega) x_i^{(k)}, \quad i = 1, \dots, n$$

- a combination of Gauss-Seidel update and previous iteration update

When A is symmetric positive definite, the SOR turns into symmetric SOR (SSOR).

Convergence of SOR is more difficult to analyze.

- If ω is real, SOR does not converge when $\omega < 0$ or $\omega > 2$.
- For ω being complex, SOR does not converge when $|\omega - 1| > 1$.

For more results on the convergence analysis, cf. Chapter 11.2.7 of [**Golub & van Loan 13'**], and **CIS 515 at UPenn**

https://www.cis.upenn.edu/~cis515/cis515-20-sl5.pdf

https://www.cis.upenn.edu/~cis515/

Readings

You are supposed to read

- Gene H. Golub and Charles F. Van Loan. Matrix Computations, *Johns Hopkins University Press*, 2013.

  Chapter 11.2.