SI231b: Matrix Computations

Lecture 21: Iterative Methods for Linear Systems

Yue Qiu

qiuyue@shanghaitech.edu.cn

School of Information Science and Technology

ShanghaiTech University

Nov. 21, 2022

Why Iterative Methods

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

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

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

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

- returns exact solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ (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.

Basic Principles of Iterative Methods

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

- $\mathbf{A}$ is typically involved only in the context of matrix-vector multiplication
- attractive when $\mathbf{A}$ 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

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{$\dagger$}$$

find another matrix $\mathbf{B}$ and a vector $\mathbf{c}$ such that

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

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{c}, \tag{$\ddagger$}$$

and starting from any vector $\mathbf{x}_0$, compute the sequence $\{\mathbf{x}^{(k)}\}$ via

$$\mathbf{x}_{k+1} = \mathbf{B}\mathbf{x}_k + \mathbf{c}, \quad k \in \mathbb{N}. \tag{$\sharp$}$$

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

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

Convergence of Iterative Methods

Consistency alone does not suffice to ensure the convergence of the iterative method ($\sharp$).

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

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

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

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

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

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

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

Convergence of Iterative Methods

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

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

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

Lemma: For any square matrix $\mathbf{B}$, the following conditions are equivalent:

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

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

Splitting Schemes

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

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

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

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

Based on different splittings of $\mathbf{A}$, 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 $\mathbf{A}$ in the form

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

where

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

Then the Jacobi iteration takes the form

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

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

Key Insight of Jacobi Iteration

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

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

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

- motivation: put $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$ 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 $\mathbf{x}^{(k+1)} = {\mathbf{D}_\mathbf{A}}^{-1} \Big((\mathbf{L}_\mathbf{A} + \mathbf{U}_\mathbf{A})\mathbf{x}^{(k)} + \mathbf{b} \Big)$.

More on Jacobi Iteration

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

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

Proof ?

Gauss-Seidel Iteration

The Jacobi iteration splits the matrix $\mathbf{A}$ in the form

$$\mathbf{A} = \mathbf{D_A} + \mathbf{L_A} - \mathbf{U_A},$$

where

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

Then the Gauss-Seidel iteration takes the form

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

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

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

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

More on Gauss-Seidel Iteration

- Gauss-Seidel iteration is computationally more expensive than Jacobi iteration.
  - $\mathcal{O}(n^2)$ for dense $\mathbf{A}$
  - $\mathcal{O}(nnz(\mathbf{A}))$ for sparse $\mathbf{A}$
- 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 $\mathbf{A}$ is symmetric positive definite (recall?)

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

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

SOR Iteration

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

To accelerate the convergence, splitting $\mathbf{A}$ in the following way

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

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

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

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

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

Key Insight of SOR Iteration

The SOR iteration update:

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

- 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.