

CS321-002

Introduction to Numerical Methods

Lecture 5

System of Linear Equations

Professor Jun Zhang

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046

October 26, 2000

System of Linear Equations

$$\begin{aligned} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n &= b_1 \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n &= b_2 \\ &\dots \dots \dots \\ a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nn} x_n &= b_n \end{aligned}$$

where a_{ij} are coefficients, x_i are unknowns, and b_i are right-hand sides. Written in a compact form is

$$\sum_{j=1}^n a_{ij} x_j = b_i, \quad i = 1, \dots, n$$

it can also be written in matrix form

$$Ax = b$$

where the matrix is

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

and $x = [x_1, x_2, \dots, x_n]^T$, $b = [b_1, b_2, \dots, b_n]^T$

Gaussian Elimination

linear systems are solved by Gaussian elimination, which involves repeated procedure of multiplying a row by a number and adding it to another row to eliminate a certain variable

for a particular step, this amounts to

$$a_{ij} \leftarrow a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj} \quad (k \leq j \leq n)$$
$$b_i \leftarrow b_i - \frac{a_{ik}}{a_{kk}} b_k$$

after this step, the variable x_k is eliminated in the $(k + 1)$ th and in the later equations

the Gaussian elimination modifies a matrix into an upper triangular form such that $a_{ij} = 0$ for all $i > j$. The solution of an upper triangular is then easily obtained by a back substitution procedure

Back Substitution

the obtained upper triangular system is

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\&\vdots \\a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n &= b_{n-1} \\a_{nn}x_n &= b_n\end{aligned}$$

we can compute

$$x_n = \frac{b_n}{a_{nn}}$$

from the last equation and substitute its value in other equations and repeat the process

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^n a_{ij} x_j \right)$$

for $i = n - 1, n - 2, \dots, 1$

Condition Number and Error

a quantity used to measure the quality of a matrix is called condition number, defined as

$$\rho(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\|$$

the condition number measures the transfer of error from the matrix \mathbf{A} to the right hand side vector \mathbf{b} . If \mathbf{A} has a large condition number, small error in \mathbf{A} may yield large error in the solution $\mathbf{x} = \mathbf{A}^{-1} \mathbf{b}$. Such a matrix is called ill-conditioned

the error e is defined as the difference between a computed solution and the exact solution

$$\mathbf{e} = \mathbf{x} - \tilde{\mathbf{x}}$$

since the exact solution is generally unknown, we measure the residual

$$\mathbf{r} = \mathbf{b} - \mathbf{A} \tilde{\mathbf{x}}$$

as an indicator of the size of the error

Small Pivot

$$\epsilon x_1 + x_2 = 1$$

$$x_1 + x_2 = 2$$

for some small ϵ . After the first step of Gaussian elimination

$$\begin{aligned} \epsilon x_1 + x_2 &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) x_2 &= 2 - \frac{1}{\epsilon} \end{aligned}$$

we have

$$\begin{aligned} x_2 &= \frac{2 - 1/\epsilon}{1 - 1/\epsilon} \\ x_1 &= \frac{1 - x_2}{\epsilon} \end{aligned}$$

for very small ϵ , the computer results will be $x_2 = 1$ and $x_1 = 0$. The correct results are

$$\begin{aligned} x_1 &= \frac{1}{1 - \epsilon} \approx 1 \\ x_2 &= \frac{1 - 2\epsilon}{1 - \epsilon} \approx 1 \end{aligned}$$

Scaled Partial Pivoting

we need to choose an element which is large relative to other elements of the same row as the pivot

let $L = (l_1, l_2, \dots, l_n)$ be an index array of integers. We first compute an array of scaling factor as $S = (s_1, s_2, \dots, s_n)$ where

$$s_i = \max_{1 \leq j \leq n} |a_{ij}| \quad (1 \leq i \leq n)$$

the first row i is choosing such that the ratio $|a_{i,1}|/s_i$ is the greatest. Suppose this index is l_1 , then appropriate multipliers of equation l_1 are subtracted from the other equations to eliminate x_1 from the other equations

suppose initially $L = (l_1, l_2, \dots, l_n) = (1, 2, \dots, n)$, if our first choice is l_j , we will interchange l_j and l_1 in the index set, not actually interchange the first and the l_j rows, to avoid moving data around the memory

Example

straightforward Gaussian elimination does not work well (not robust)

$$\begin{aligned}\epsilon x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2\end{aligned}$$

the scale factor will be computed as $S = \{1, 1\}$. In the first step, the scale factor ratio array $\{\epsilon, 1\}$. So the 2nd row is the pivoting row

after eliminating x_1 from the 1st equation, we have

$$\begin{aligned}(1 - \epsilon)x_2 &= 1 - 2\epsilon \\ x_1 + x_2 &= 2\end{aligned}$$

it follows that

$$\begin{aligned}x_2 &= \frac{1 - 2\epsilon}{1 - \epsilon} \approx 1 \\ x_1 &= 2 - x_2 \approx 1\end{aligned}$$

we computed correct results by using scaled partial pivoting strategy

Long Operation Count

we count the number of multiplications and divisions, ignore summations and subtractions

the 1st step, finding a pivoting costs n divisions.

additional n operations are needed to multiply a factor to the pivoting row for each of the $n - 1$ eliminations. The cost is $n(n - 1)$ operations. The total cost of this step is n^2 operations

the computation is repeated on the remaining $(n - 1)$ equations. The total costs of Gaussian elimination with scaled partial pivoting is

$$n^2 + (n - 1)^2 + \dots + 4^2 + 3^2 + 2^2 = \frac{n(n + 1)(2n + 1)}{6} - 1 \approx \frac{n^3}{3}$$

back substitution costs $n(n - 1)/2$ operations

Tridiagonal Systems

the back substitution is straightforward

$$x_n = \frac{b_n}{d_n}$$

$$x_i = \frac{b_i - c_i x_{i+1}}{d_i} \quad (i = n - 1, \dots, 1)$$

no pivoting is performed, otherwise the procedure will be quite different due to the fill-in

diagonal dominance: a matrix $A = (a_{ij})_{n \times n}$ is diagonally dominant if

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}| \quad (1 \leq i \leq n)$$

for a diagonally dominant tridiagonal system, no pivoting is needed, i.e., no division by zero

we want to show Gaussian elimination preserves diagonal dominance, i.e., $|d_i| > |a_{i-1}| + |c_i|$

Iterative Solution Methods

obtain an approximate solution to a linear system iteratively

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ &\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_{nn} &= b_n \end{aligned}$$

create a single equation for each variable

$$\begin{aligned} x_1 &= \frac{b_1 - (a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n)}{a_{11}} \\ x_2 &= \frac{b_2 - (a_{21}x_1 + a_{23}x_3 + \cdots + a_{2n}x_n)}{a_{22}} \\ &\dots\dots\dots \\ x_{nn} &= \frac{b_n - (a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots)}{a_{nn}} \end{aligned}$$

we assign an arbitrary value to each of the variables in the right hand side to start iteration

Jacobi Iteration

we repeatedly use each set of computed values as the next iterates, and hope the process will converge to the true solution. (May stop early if necessary.) This iteration procedure is called Jacobi iteration method. Convergence is not guaranteed for general matrices

$$x_1^{(k)} = \frac{b_1 - (a_{12}x_2^{(k-1)} + a_{13}x_3^{(k-1)} + \dots + a_{1n}x_n^{(k-1)})}{a_{11}}$$

$$x_2^{(k)} = \frac{b_2 - (a_{21}x_1^{(k-1)} + a_{23}x_3^{(k-1)} + \dots + a_{2n}x_n^{(k-1)})}{a_{22}}$$

.....

$$x_n^{(k)} = \frac{b_n - (a_{n1}x_1^{(k-1)} + a_{n2}x_2^{(k-1)} + a_{n3}x_3^{(k-1)} + \dots)}{a_{nn}}$$

Gauss-Seidel Iteration

if the iteration process converges, it may be faster if we use the new values as soon as they are available. This gives the Gauss-Seidel iteration method

$$x_1^{(k)} = \frac{b_1 - (a_{12}x_2^{(k-1)} + a_{13}x_3^{(k-1)} + \dots + a_{1n}x_n^{(k-1)})}{a_{11}}$$

$$x_2^{(k)} = \frac{b_2 - (a_{21}x_1^{(k)} + a_{23}x_3^{(k-1)} + \dots + a_{2n}x_n^{(k-1)})}{a_{22}}$$

.....

$$x_n^{(k)} = \frac{b_n - (a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + a_{n3}x_3^{(k)} + \dots)}{a_{nn}}$$

in general, Gauss-Seidel is faster than Jacobi, but the later is more attractive to run on parallel computers. No convergence is guaranteed

SOR Method

around 1950, David Young at Harvard University found that iteration can be accelerated by using a weighted average of the successive approximate solutions. Given some parameter $0 < \omega < 2$, we perform

$$x_i^{(k)} = \omega \tilde{x}_i^{(k)} + (1 - \omega) x_i^{(k-1)} \quad i = 1, 2, \dots, n$$

this step symbolizes the beginning of modern iterative methods

with an arbitrary initial guess, convergence of these basic iterative methods is guaranteed if

- 1.) the matrix is strictly diagonally dominant;
- 2.) the matrix is symmetric positive definite, i.e., $A = A^T$ and $x^T A x > 0$