L3

Linear Equations

Number of operations

Gauss –Jordan Elimination

Camille Jordan

Linear Systems

Gabriel Cramer

LU Factorization

Iterative Methods

Gauss Elimination

Carl Friedric Gauss

Norms and Conditions

Gauss Elimination

To automate elimination method with pivoting and scaling with coefficient matrix A and inhomogeneous vector b, define an order vector o $(n \times 1)$ for rows which is initially $(1,2,3, \dots, n)$.

1- In column 1, the largest coefficient be row k. Pivot row k to be the first row, with order vector will be $o = (k, 2,3, .., 1, .. n)$

2- Eliminate x_k from $(n-1)$ equations

3- In modified $(n-1)$ equations, pivot and eliminate $x_{k'}$, and $o = (k, k', 3, .., 1, .. n)$.

4- Upon completion, matrix A is transformed to an upper triangular matrix that could be solved for x direct backward substitution.

5- Number of operations(?)

Gauss –Jordan Elimination

This elimination procedure will find A^{-1} and x

1- Form an augmented matrix in the form

$$[A \mid b \mid I]$$

2- By elimination, transform A to I so that we have

$$[A \mid b \mid I] \to [I \mid b' \mid A']$$

3- $A' = A^{-1}$ and $b' = x$

4- No. of operations (?).

LU Factorization

$$L = \begin{bmatrix} a_{11} & 0 & 0 & 0 \ldots & \ldots .0 \\ a_{21} & a_{22} & 0 & 0 \ldots \ldots & .0 \\ a_{31} & a_{32} & a_{33} & 0, \ldots \ldots & .0 \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ a_{n1} & a_{n2} & a_{n3} & \ldots & \ldots . a_{nn} \end{bmatrix} \quad U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \ldots \ldots . & a_{1n} \\ 0 & a_{22} & a_{23} & a_{24} \ldots \ldots . & a_{2n} \\ 0 & 0 & a_{33} & a_{34} \ \ldots & . a_{3n} \\ 0 & 0 & 0 & a_{44} \ a_{45} & a_{4n} \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ 0 & 0 & 0 & 0 \ldots & \ldots . a_{nn} \end{bmatrix}$$

For linear equation $Ax = b$, decompose matrix A as product of L and U

$$A = LU$$
$$LUx = b$$

Multiply both sides with L^{-1}

$$L^{-1}LUx = L^{-1}b \ \rightarrow IUx = L^{-1}b$$

With $IU = U$ and set $b' = L^{-1}b$

$$Ux = b'$$

This equation solved for x with direct backward substitution.

Factorization Procedure

For $Ax = b$, use Gauss elimination to transform matrix A into U.

In this transformation, keep the elimination multipliers m_{ij} and

$l_{ij} = m_{ij}$.

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \ldots & \ldots .0 \\ m_{21} & 1 & 0 & 0 \ldots \ldots & .0 \\ m_{31} & m_{32} & 1 & 0, \ldots \ldots & .0 \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots . \\ m_{n1} & m_{n2} & m_{n3} & \ldots & \ldots .1 \end{bmatrix}$$

This procedure is Doolittle factorization and with steps on the left, x can be determined.

MatLab

For A is defined $[l, u] = lu(A)$

For A and P defined $[l, u, P] = lu(A)$
with $PA = LU$

More on determinants

$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

For which del $[A]$ = $|A| = a_{11}\ a_{12} - \ a_{12}\, a_{21}$

1- Pivot matrix A to A^p

$A^p = \begin{bmatrix} a_{21} & a_{22} \\ a_{11} & a_{12} \end{bmatrix}$.

$|A^p|$ = $a_{12}a_{21} \ - a_{11}\ a_{12} = -|A|$

Pivoting by i rows, $|A^p| = (-1)^i|A|$

2- Multiply A by a constant α

$\alpha A = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} \\ a_{21} & a_{22} \end{bmatrix} \rightarrow |\alpha A| = \ \alpha|A|$

3- Set $R_1' = R_1 + \lambda R_2$

$R_1' = \begin{bmatrix} a_{11} + \lambda a_{21} & a_{12} + \lambda a_{21} \\ a_{21} & a_{22} \end{bmatrix}$

$|R_1'| = |A|$

Solving Tridiagonal Matrix

$$T = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \ \dots & \dots\ .0 \\ a_{21} & a_{22} & a_{23} & 0 \ \dots \dots & .0 \\ 0 & a_{32} & a_{33} & a_{34}, \dots & \dots .0 \\ & \dots \dots \dots \dots \dots \dots \dots \dots \dots . & & & \\ & \dots \dots \dots \dots \dots \dots \dots \dots \dots . & & & \\ & \dots \dots \dots \dots \dots \dots \dots \dots \dots . & & & \\ 0 \quad 0 & 0 & & a_{(n-1)n} & a_{nn} \end{bmatrix}$$

1- Gauss elimination to make it zero the lowest diagonal elements to make T a U. Back substitution to find x. In this transformation, the vector b should also be modified.

2- Use Gauss elimination to calculate em's and decompose T to L and U.

3- No. of operations (?)

4- T can be stored as a $n \times 3$ matrix to save memory space.

$$T = \begin{bmatrix} - & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} \\ a_{32} & a_{33} & a_{34} \\ & \dots \dots \dots \dots & \\ & \dots \dots \dots \dots & \\ & \dots \dots \dots \dots & \\ a_{(n-1)n} & a_{nn} & - \end{bmatrix}$$

Norms and conditions

For A a matrix and b a vector

Norm of $A = \|A\|$
Norm of $b = \|b\|$

For b

$\|b\|_1 = \sum_{i=1}^{n}|x_i|$
$\|b\|_2 = \left[\sum_{i=1}^{n} x_i^2\right]^{1/2}$
$\|b\|_\infty = \max\limits_{i=1,n} x_i$

For A

$\|A\|_1 = \max\limits_{j=1,n} \sum_{i=1}^{n}|a_{i,j}|$ columns
$\|A\|_2 = \max\limits_{i=1,n} \sum_{i=1}^{n}|a_{i,j}|$ rows
$\|A\|_3 = \min\limits_{i=1,n} \lambda_i$ Special norm
$\|A\|_e = \left[\sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}^2\right]^{1/2}$ Euclidean norm

$\lambda_i = n$ eigenvalues of A

Norm Properties

$\|A\| \geq 0$ zero when $A = 0$
$\|kA\| = k\|A\|$
$\|A + B\| \leq \|A\| + \|B\|$
$\|AB\| \leq \|A\| \times \|B\|$

Conditioning number $C(A)$

For linear problem $Ax = b$, conditioning number defined as

$C(A) = \|A\| \, \|A^{-1}\|$

A variation of x as δx in terms of variation of matrix A as δA

$$\frac{\|\delta x\|}{\|x\|} \leq C(A) \frac{\|\delta A\|}{\|A\|}$$

$C(A)$ scales δx with δA

$C(A) << 1$ problem is well-conditioned
$C(A) \gg 1$ problem is ill-conditioned

Iterative Compensation

For linear system $Ax = b$,

The exact solution x

Your numerical solution z

your error $\delta x = (x - z)$

Recover $\delta b = Az - b$

Solve for δx $A\,\delta x = \delta b$

Improved solution $(z + \delta x)$

- This procedure can be continued iteratively for better accuracy.
- Convergence checked at every iteration
- Use of LU factorization will reduce no. of operations (?).

Accuracy and Convergence

Two subsequent iterations k and $k+1$

x- vector values x^k and x^{k+1}

Error at step $k+1$ $\Delta x^k = x^{k+1} - x^k$

Accuracy – Choose ε to be much smaller than unity.

Absolute accuracy

$$\left|\Delta x_i^k\right|_{max} \leq \varepsilon$$

$$\sum_{i=1}^{n}\left|\Delta x_i^k\right| \leq \varepsilon$$

$$\left[\sum_{i=1}^{n}\left(\Delta x_i^k\right)^2\right]^{1/2} \leq \varepsilon$$

Relative accuracy

$$\left|\frac{\left|\Delta x_i^k\right|_{max}}{x_i}\right| \leq \varepsilon$$

$$\sum_{i=1}^{n}\left|\frac{\Delta x_i^k}{x_i}\right| \leq \varepsilon$$

$$\left[\sum_{i=1}^{n}\left(\frac{\Delta x_i^k}{x_i}\right)^2\right]^{1/2} \leq \varepsilon$$

Iterative Methods

For $Ax = b$ or $\sum_{j=1}^{n} a_{ij}x_j = b_i$

$$a_{11}\,x_1 + a_{12}x_2 + a_{13}x_3 - b_1 = R_1$$
$$a_{21}\,x_1 + a_{22}x_2 + a_{23}x_3 - b_2 = R_2$$
$$a_{31}\,x_1 + a_{32}x_2 + a_{33}x_3 - b_3 = R_3$$

Residual values for row $i = R_i$

Jacoby Method

At iteration k, solve for x_i in row i

$$x_1^{k+1} = \frac{R_1^k + b_1 - a_{12}x_2^k - a_{13}x_3^k}{a_{11}} = x_1^k - \frac{R_1^k}{a_{11}}$$
$$x_2^{k+1} = \frac{R_2^k + b_2 - a_{21}x_1^k - a_{13}x_3^k}{a_{11}} = x_2^k - \frac{R_2^k}{a_{22}}$$
$$x_3^{k+1} = x_3^k - \frac{R_3^k}{a_{33}}$$

Initial guess x_i^0 as x_1^0, x_2^0, x_3^0

Use recurrence equations above to solve for x_1^1, x_2^1, x_3^1 and beyond, until desired convergence.

Gauss-Seidel Method

Jacoby method with updated values for $k+1$ iteration known as successive iteration

$$x_1^{k+1} = \frac{R_1^k + b_1 - a_{12}x_2^k - a_{13}x_3^k}{a_{11}}$$
$$x_2^{k+1} = \frac{R_2^k + b_2 - a_{21}x_1^{k+1} - a_{13}x_3^k}{a_{11}}$$
$$x_3^{k+1} = \frac{R_3^k + b_3 - a_{31}x_1^{k+1} - a_{32}x_2^{k+1}}{a_{11}}$$

Faster convergence than Jacoby method.

Successive Over Relaxation- SOR

Recurrence equations can be modified for faster convergence. Use

Gauss-Seidel method with

$$x_i^{k+1} = x_i^k - \omega\frac{R_i^k}{a_{ii}}$$

- ω is over-relaxation factor and for most systems $1 < \omega < 2$.
- Optimal ω, ω_{opt} is determined by numerical experimentation.
- For marginally stable systems, SOR can diverge the calculations
- For $0 < \omega < 1$, successive under relaxation. Slows the calculations and can stabilize the calculations for complex systems.

$$Ax = b$$

A = [1.002, 1; 1., 0.998]

b = [2.002; 1.998]

X= ?

B2 = [2.0021; 1.998]

X2 =?