Gaussian Elimination in MATLAB

Introduction

Gaussian elimination is a fundamental algorithm in linear algebra for solving systems of linear equations, finding the inverse of a matrix, and computing determinants. MATLAB provides built-in tools (`rref`, `inv`, `det`, and the backslash operator \) but it's important to understand the computational steps.

Key Concepts

- Gaussian Elimination Steps
- Row operations
- Forward elimination
- Back substitution
- MATLAB Implementation

Definitions

- **Row Echelon Form (REF):** A matrix form obtained after forward elimination.
- **Reduced Row Echelon Form (RREF):** Further simplified form using back substitution.
- **Pivot:** The leading non-zero entry in a row, used to eliminate entries below (or above). stability.

Gaussian Elimination Steps

Given a system $Ax = b$, the algorithm follows:

$$A = \begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} \quad \longrightarrow \quad U = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 3 \end{bmatrix} \text{ (Forward Elimination)}$$

$$\begin{bmatrix} 2 & 1 & -1 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \text{(Back Substitution)}$$

Steps:

(i) Forward elimination: Zero out elements below the pivots.

(ii) Back substitution: Solve for variables starting from the last row.

Row Operations

In Gaussian elimination, we can edit our matrix in forward elimination using an **elimination matrix**. The elimination matrix simply does row operations on a matrix through multiplication:

Suppose we want to eliminate the entry in position $(2,1)$ of

$$C = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 10 \end{bmatrix}.$$

The elimination matrix to perform $R_2 \to R_2 - 2R_1$ is

$$E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

When we multiply:

$$E_{21}C = \begin{bmatrix} 1 & 4 & 7 \\ 0 & -3 & -6 \\ 3 & 6 & 10 \end{bmatrix},$$

the $(2,1)$ entry has been eliminated.

Forward Elimination

Forward elimination is the first stage of Gaussian elimination. The goal is to transform the system matrix A into an upper triangular matrix U by zeroing out entries below each pivot. This is achieved by applying a sequence of elimination matrices.

1. Choose the pivot a_{kk} in column k.
2. For each row $i > k$, compute the factor
$$f_{ik} = \frac{a_{ik}}{a_{kk}}.$$
3. Apply the row operation
$$R_i \;\to\; R_i - f_{ik}R_k,$$
which eliminates a_{ik}.

Eventually this will get us to the upper triangular matrix U (see above).

Back Substitution

Once forward elimination reduces the system to an upper triangular matrix U, the next step is **back substitution**. This solves for the unknowns starting from the last equation and working upwards.

Say we have:

$$Ux = c, \quad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix},$$

the equations are:

$$u_{nn}x_n = c_n,$$

$$u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = c_{n-1},$$

and so on, up to the first row. It's hard to read, but we just want to stop with the bottom row that has a trivial solution, then slowly work our way up using the previous solution.

MATLAB Implementation

Manual Implementation

Example MATLAB function for Gaussian elimination:

```
function x = gaussElim(A,b)
% x = gaussElim(A,b)
%
% This function solves a linear system using Gaussian elimination.
%
% InputS;
% A - coefficient matrix (n,n)
% b - right hand side vector (n,1)
%
% Output:
% x - solution vector (n,1)
%
% input validity checks
[m,n] = size(A);
if ~(m == n)
    error('gaussElim: A matrix must be square')
end
[mb,nb] = size(b);
if ~((nb == 1) && (mb == n))
    error('gaussElim: b vector is wrong size')
end
% form the augmented matrix
Aug = [A b];
% initialize variables
x = zeros(n,1);
I = eye(n);
% forward elimination
for j = 1:n-1
    for i = j+1:n
        factor = -Aug(i,j)/Aug(j,j);
        E = I;
        E(i,j) = factor;
        Aug = E*Aug;
    end
end
% back substitution
U = Aug(:,1:n);
bp = Aug(:,n+1);
x(n) = bp(n)/U(n,n);
for i = n-1:-1:1
    x(i) = (bp(i) - U(i,i+1:n)*x(i+1:n))/U(i,i);
    end
end
```

Built-in MATLAB Tools

- Solve directly: `x = A\b;`
- Reduced row echelon form: `rref([A b])`
- Matrix inverse (not recommended for solving): `inv(A)*b`

Numerical Considerations

- Division by small pivots can cause numerical instability.
- Partial pivoting helps reduce round-off errors.
- MATLAB's \ operator automatically chooses stable algorithms (LU, QR, or more advanced factorizations).

Example Problem

Solve the system:

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}$$

(a) Using the custom function:

```
A = [2 1 -1; -3 -1 2; -2 1 2];
b = [8; -11; -3];
x = gauss_elim(A, b)
```

(b) Using MATLAB built-in solver:

```
x = A \ b
```

(c) Both approaches give:

$$x = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}$$

Recap

Gaussian elimination transforms systems into a form that's easy to solve by back substitution. While MATLAB automates this, understanding the process clarifies numerical stability, algorithmic efficiency, and when to trust results.