Simultaneous Linear Equations

Created: 2019-12-11 ∧ 15:32

Table of Contents

Learning Objectives

- Use the *Gaussian Elimination* method of solving sets of simultaneous linear equations.
- Select the correct ordering of rows to minimize floating point error.

The Problem

- We are given a series of equations of this form:

$$
\begin{aligned}
a_1x + b_1y + c_1z = d_1 \\
a_2x + b_2y + c_2z = d_2 \\
a_3x + b_3y + c_3z = d_3
\end{aligned}
$$

- We know the values of $a_i,\ b_i,\ \text{and}\ c_i$ for all i
- We want the values of $x, y,\ \text{and}\ z$.

- Solution: use a matrix formulation

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Solving The matrix equations

- Use Gaussian Elimination to solve this. Step 1: Forward elimination.
  - For each row i and row j where $i < j$, subtract off multiples of row i from row j to zero out column i.
  - The solution vector needs to participate in this as well: called the *augmented matrix*

$$\left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ 0 & b_2' & c_2' & d_2' \\ 0 & b_3' & c_3' & d_3' \end{array}\right] \Rightarrow \left[\begin{array}{c} a \\ 0 \\ 0 \end{array}\right.$$

- Step 2: Backward elimination.
  - For each row i and row j where $i > j$, subtract off multiples of row i from row j to zero out column j.
  - Result: a diagonal matrix. Then $z = d_3''/c_3''$, etc…

A Real Example, Step 1

- Want to solve:

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 10 & 30 \\ 3 & 12 & 20 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 30 \\ 50 \end{bmatrix}$$

- Step 1: forward substitution

$$\left[\begin{array}{ccc|c} 1 & 3 & 5 & 10 \\ 2 & 10 & 30 & 30 \\ 3 & 17 & 20 & 50 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 3 & 5 & 10 \\ 0 & 4 & 20 & 10 \\ 0 & 8 & 5 & 20 \end{array}\right] \Rightarrow \left[\begin{array}{c} 1 \\ 0 \\ 0 \end{array}\right.$$

Step 2

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 10 & 30 \\ 3 & 12 & 20 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 30 \\ 50 \end{bmatrix}$$

- Step 2: Backward substitution

$$\left[\begin{array}{ccc|c} 1 & 3 & 5 & 10 \\ 0 & 4 & 20 & 10 \\ 0 & 0 & -35 & 0 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 3 & 0 & 10 \\ 0 & 4 & 0 & 10 \\ 0 & 0 & -35 & 0 \end{array}\right] \Rightarrow \left[\begin{array}{c} 1 \\ 0 \\ 0 \end{array}\right.$$

- Solution: $x = 2.5, y = 1.25, z = 0$
- Tips:
  - If you end up with a all-zero row, the system is underspecified.
  - To reduce numerical error, sort rows by largest column value.

Source Code

- Gleefully stolen from Competitive Programming 3.

```
#define MAX_N 100
struct AugmentedMatrix { double mat[MAX_N][MAX_N +
struct ColumnVector { double vec[MAX_N]; };
ColumnVector GaussianElimination(int N, AugmentedMa
    // input: N, Augmented Matrix Aug, output: Colu
    int i, j, k, l; double t; ColumnVector X;
    // the forward elimination phase
    for (j = 0; j < N - 1; j++) {
       l = j;
       for (i = j + 1; i < N; i++) // which row has
          if (fabs(Aug.mat[i][j]) > fabs(Aug.mat[l]
       // swap this pivot row, reason: to minimize
       for (k = j; k <= N; k++)
           t = Aug.mat[j][k], Aug.mat[j][k] = Aug.m
        // the actual forward elimination phase
       for (i = j + 1; i < N; i++)
           for (k = N; k >= j; k--)
               Aug.mat[i][k] -= Aug.mat[j][k] * Aug
    }
    for (j = N - 1; j >= 0; j--) {// the back subst
       for (t = 0.0, k = j + 1; k < N; k++) t += Au
       X.vec[j] = (Aug.mat[j][N] - t) / Aug.mat[j][
    }
    return X;
}
```