CSC349A Numerical Analysis

Lecture 11

Rich Little

University of Victoria

2018

Table of Contents I

Systems of Linear Equations

- Math background pg. 231-241
- Review if necessary
  - matrix notation
  - transpose
  - matirx operations - addition and multiplication
  - inverses, determinants
  - triangular matirces
- For motivations in Engineering see Chapter 12 Case Studies - pg. 319-330
  - Steady-state analysis of systems of reactors
  - Analysis of a statically determinate truss
  - Currents and voltages in resistor circuits
  - Spring-mass systems

Naive Gaussian Elimination

Notation for a system of n linear equations in n unknowns:

$$Ax = b$$

where

- A is an $n \times n$ nonsingular matirx, and
- x and b are column vectors with n entries.

Naive Gaussian Elimination

That is, given A and b, solve for x in

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

or

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Theoretical basis for Gaussian Elimination

You can apply any of the following 3 elementary row operations to $Ax = b$ without changing the solution.

i multiply any equation E_i by a nonzero constant λ

ii replace equation E_i by $E_i + \lambda E_j$

iii interchange any two equations E_i and E_j

Naive Gaussian Elimination uses operation ii, only, to do the following:

1. reduce the coefficient matrix A to upper triangular form - **forward elimination**
2. solve the reduced matrix for x - **back-substitution**

In General

After forward elimination we have the augmented matrix

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right]$$

and then we back-substitute

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}, x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}}$$

Notes

1. When implemented we overwrite A and b, thus no need for the superscripts.
2. Don't need to calculate, nor store the lower 0's.
3. The entries $a_{11}, a_{22}^{(1)}, a_{33}^{(2)}, \ldots, a_{n-1,n-1}^{(n-2)}$ are called the <u>pivots</u>. They are the denominators in the multipliers - <u>must be nonzero</u>.

Table of Contents I

Forward Elimination Pseudocode

Algorithm 1 pseudocode for the calculation of forward elimination

```
1: for $k = 1$ to $n - 1$ do
2:     for $i = k + 1$ to $n$ do
3:         $factor = a_{i,k} / a_{k,k}$
4:         for $j = k + 1$ to $n$ do
5:             $a_{i,j} = a_{i,j} - factor \times a_{k,j}$
6:         end for
7:         $b_i = b_i - factor \times b_k$
8:     end for
9: end for
```

Backward substitution pseudocode

Algorithm 2 pseudocode for the calculation of back substitution

```
1: $x_n = b_n / a_{n,n}$
2: for $i = n - 1$ to $1$ do
3:     $sum = b_i$
4:     for $j = i + 1$ to $n$ do
5:         $sum = sum - a_{i,j} \times x_j$
6:     end for
7:     $x_i = sum / a_{i,i}$
8: end for
```

Table of Contents I

Floating-point operations

The most common measure of the efficiency of algorithms for solving linear systems $Ax = b$ is a floating-point operation (flop) count. This is determined as a function of n, the order of the matrix A.

The analysis uses the following identities:

$$\sum_{i=1}^{m} i = 1 + 2 + 3 + \cdots + i = \frac{m(m+1)}{2}$$

$$\sum_{i=1}^{m} i^2 = 1^2 + 2^2 + 3^2 + \cdots + i^2 = \frac{m(m+1)(2m+1)}{6}$$

Analysis of the Naive Gaussian Elimination algorithm: count the number of floating point add/subtracts and multiplies/divides.

Forward Elimination I

The forward elimination iterates over the rows of the matrix to calculate the factors (or multipliers) and each time there is one less row to consider. Therefore the number of divides is:

$$(n-1) + (n-2) + \cdots + 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$$

For each one of these multipliers all the entries in the corresponding row (except the zeros) are subtracted and multiplies. Therefore the number of subtractions and the number of multiplications are both:

$$(n-1)^2 + (n-2)^2 + \cdots + 1^2 = \sum_{i=1}^{n-1} i^2 = \frac{(n-1)(n)(2n-1)}{6}$$

Foward Elimination II

We also have to do the corresponding subtracts and multiplies for *b* both of which are:

$$(n-1)+(n-2)+\cdots+(1)=\frac{(n-1)n}{2}$$

Therefore the forward elimination totals are:

$$\text{number of mult/div}=\frac{2n^3+3n^2-5n}{6}$$
$$\text{number of add/sub}=\frac{2n^3-2n}{6}$$

Back substitution

We have one initial division followed by:

$$\text{number of subtracts} = 1 + 2 + \cdots + (n-1) = \frac{(n-1)n}{2}$$
$$\text{number of multiplies} = 1 + 2 + \cdots + (n-1) = \frac{(n-1)n}{2}$$

with the last number of divides being $n-1$.
Therefore for back substituion the total is:

$$\begin{aligned} \text{number of mult/div} &= \frac{n^2+n}{2} \\ \text{number of add/sub} &= \frac{n^2-n}{2} \end{aligned}$$

Discussion

The purpose of the flop count is to give a measure of how computation time increases as the problem size n increases. For example, it can help us determine that the forward elimination is much more time consuming than the back substitution.

For **Forward elimination** we have

$$\approx \frac{n^3}{3} + \frac{n^3}{3} = \frac{2n^3}{3} \text{flops}$$

The notation $O(n^3)$ is used to indicate the order of increase with problem size without taking into account the constants.

Discussion II

For **Back substitution** we have

$$\frac{n^2+n}{2}+\frac{n^2-n}{2}=n^2$$

For example, a $2n \times 2n$ linear system requires about 8 times the computation time than an $n \times n$ linear system as the cost of the forward elimination step would be

$$\frac{2(2n)^3}{3}=8(\frac{2n^3}{3}).$$