Numerical Analysis Hilary Term 2018
Lecture 4: Gaussian Elimination

Setup: given a square n by n matrix A and vector with n components b, find x such that

$$Ax = b.$$

Equivalently find $x = (x_1, x_2, \dots, x_n)^{\mathrm{T}}$ for which

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

Lower-triangular matrices: the matrix A is **lower triangular** if $a_{ij} = 0$ for all $1 \le i < j \le n$. The system (1) is easy to solve if A is lower triangular.

$$\begin{array}{lclcll} a_{11}x_1 & = b_1 & \Longrightarrow & x_1 = \dfrac{b_1}{a_{11}} & \Downarrow \\ a_{21}x_1 + a_{22}x_2 & = b_2 & \Longrightarrow & x_2 = \dfrac{b_2 - a_{21}x_1}{a_{22}} & \Downarrow \\ \quad\vdots & & & & \Downarrow \\ a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ii}x_i & = b_i & \Longrightarrow & x_i = \dfrac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}} & \Downarrow \\ \quad\vdots & & & & \Downarrow \end{array}$$

This works if, and only if, $a_{ii} \neq 0$ for each i. The procedure is known as **forward substitution**.

Computational work estimate: one floating-point operation (flop) is one scalar multiply/division/addition/subtraction as in $y = a * x$ where a, x and y are computer representations of real scalars.[1]

Hence the work in forward substitution is 1 flop to compute x_1 plus 3 flops to compute x_2 plus ... plus $2i - 1$ flops to compute x_i plus ... plus $2n - 1$ flops to compute x_n, or in total

$$\sum_{i=1}^{n}(2i-1) = 2\left(\sum_{i=1}^{n} i\right) - n = 2\left(\tfrac{1}{2}n(n+1)\right) - n = n^2 + \text{lower order terms}$$

flops. We sometimes write this as $n^2 + O(n)$ flops or more crudely $O(n^2)$ flops.

Upper-triangular matrices: the matrix A is **upper triangular** if $a_{ij} = 0$ for all $1 \le j < i \le n$. Once again, the system (1) is easy to solve if A is upper triangular.

[1] This is an abstraction: e.g., some hardware can do $y = a * x + b$ in one FMA flop ("Fused Multiply and Add") but then needs several FMA flops for a single division. For a trip down this sort of rabbit hole, look up the "Fast inverse square root" as used in the source code of the video game "Quake III Arena".

$$
\begin{array}{rlcll}
 & \vdots & & & \Uparrow \\
a_{ii}x_i + \cdots + a_{in-1}x_{n-1} + a_{1n}x_n & = b_i & \Longrightarrow & x_i = \dfrac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} & \Uparrow \\
 & \vdots & & & \Uparrow \\
a_{n-1n-1}x_{n-1} + a_{n-1n}x_n & = b_{n-1} & \Longrightarrow & x_{n-1} = \dfrac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}} & \Uparrow \\
a_{nn}x_n & = b_n & \Longrightarrow & x_n = \dfrac{b_n}{a_{nn}}. & \Uparrow
\end{array}
$$

Again, this works if, and only if, $a_{ii} \neq 0$ for each i. The procedure is known as **backward** or **back substitution**. This also takes approximately n^2 flops.

For computation, we need a reliable, systematic technique for reducing $Ax = b$ to $Ux = c$ with the same solution x but with U (upper) triangular $\Longrightarrow$ Gauss elimination.

Example

$$
\begin{bmatrix} 3 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 12 \\ 11 \end{bmatrix}.
$$

Multiply first equation by 1/3 and subtract from the second $\Longrightarrow$

$$
\begin{bmatrix} 3 & -1 \\ 0 & \frac{7}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 12 \\ 7 \end{bmatrix}.
$$

Gauss(ian) Elimination (GE): this is most easily described in terms of overwriting the matrix $A = \{a_{ij}\}$ and vector b. At each stage, it is a systematic way of introducing zeros into the lower triangular part of A by subtracting multiples of previous equations (i.e., rows); such (elementary row) operations do not change the solution.

for columns $j = 1, 2, \ldots, n-1$

for rows $i = j+1, j+2, \ldots, n$

$$\begin{array}{rcl} \text{row } i & \leftarrow & \text{row } i - \dfrac{a_{ij}}{a_{jj}} * \text{row } j \\ b_i & \leftarrow & b_i - \dfrac{a_{ij}}{a_{jj}} * b_j \end{array}$$

end

end

Example.

$$\begin{bmatrix} 3 & -1 & 2 \\ 1 & 2 & 3 \\ 2 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ 11 \\ 2 \end{bmatrix} : \text{ represent as } \left[\begin{array}{rrr|r} 3 & -1 & 2 & 12 \\ 1 & 2 & 3 & 11 \\ 2 & -2 & -1 & 2 \end{array}\right]$$

$$\Longrightarrow \quad \begin{array}{l} \text{row } 2 \leftarrow \text{row } 2 - \frac{1}{3}\text{row } 1 \\ \text{row } 3 \leftarrow \text{row } 3 - \frac{2}{3}\text{row } 1 \end{array} \left[\begin{array}{rrr|r} 3 & -1 & 2 & 12 \\ 0 & \frac{7}{3} & \frac{7}{3} & 7 \\ 0 & -\frac{4}{3} & -\frac{7}{3} & -6 \end{array}\right]$$

$$\Longrightarrow \quad \text{row } 3 \leftarrow \text{row } 3 + \tfrac{4}{7}\text{row } 2 \left[\begin{array}{rrr|r} 3 & -1 & 2 & 12 \\ 0 & \frac{7}{3} & \frac{7}{3} & 7 \\ 0 & 0 & -1 & -2 \end{array}\right]$$

Back substitution:

$$\begin{array}{rcl} x_3 & = & 2 \\ x_2 & = & \dfrac{7 - \frac{7}{3}(2)}{\frac{7}{3}} = 1 \\ x_1 & = & \dfrac{12 - (-1)(1) - 2(2)}{3} = 3. \end{array}$$

Cost of Gaussian Elimination: note, row $i \leftarrow$ row $i - \dfrac{a_{ij}}{a_{jj}} *$ row j is

for columns $k = j+1, j+2, \ldots, n$

$$a_{ik} \leftarrow a_{ik} - \frac{a_{ij}}{a_{jj}} a_{jk}$$

end

This is approximately $2(n-j)$ flops as the **multiplier** a_{ij}/a_{jj} is calculated with just one flop; a_{jj} is called the **pivot**. Overall therefore, the cost of GE is approximately

$$\sum_{j=1}^{n-1} 2(n-j)^2 = 2\sum_{l=1}^{n-1} l^2 = 2\frac{n(n-1)(2n-1)}{6} = \frac{2}{3}n^3 + O(n^2)$$

flops. The calculations involving b are

$$\sum_{j=1}^{n-1} 2(n-j) = 2\sum_{l=1}^{n-1} l = 2\frac{n(n-1)}{2} = n^2 + O(n)$$

flops, just as for the triangular substitution.