Numerical Analysis

Raphael Hauser
with thanks to Endre Süli

Oxford Mathematical Institute

HT 2019

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 $x = (x_1, x_2, \dots, x_n)$ for which

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned} \tag{5.1}$$

Lower-triangular matrices:

the matrix A is **lower triangular** if $a_{ij} = 0$ for all $1 \leq i < j \leq n$.

The system (5.1) is easy to solve if A is lower triangular.

Lower-triangular matrices:

the matrix A is **lower triangular** if $a_{ij} = 0$ for all $1 \le i < j \le n$.
The system (5.1) is easy to solve if A is lower triangular.

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

Lower-triangular matrices:

the matrix A is **lower triangular** if $a_{ij} = 0$ for all $1 \leq i < j \leq n$.
The system (5.1) is easy to solve if A is lower triangular.

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

This works if, and only if, $a_{ii} \neq 0$ for each i.

Lower-triangular matrices:

the matrix A is **lower triangular** if $a_{ij} = 0$ for all $1 \le i < j \le n$.
The system (5.1) is easy to solve if A is lower triangular.

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

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 multiply (or divide) and possibly add (or subtraction) as in $y = a * x + b$, where a, x, b and y are computer representations of real scalars.

Computational work estimate: one floating-point operation (flop) is one multiply (or divide) and possibly add (or subtraction) as in $y = a * x + b$, where a, x, b and y are computer representations of real scalars.

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

$$\sum_{i=1}^{n} i = \tfrac{1}{2}n(n+1) = \tfrac{1}{2}n^2 + \text{lower order terms} \quad \text{flops.}$$

We sometimes write this as $\frac{1}{2}n^2 + \mathcal{O}(n)$ flops or more crudely $\mathcal{O}(n^2)$ flops.

Upper-triangular matrices:

The matrix A is **upper triangular** if $a_{ij} = 0$ for all $1 \leq j < i \leq n$.

Once again, the system (5.1) is easy to solve if A is upper triangular.

Upper-triangular matrices:

The matrix A is **upper triangular** if $a_{ij} = 0$ for all $1 \leq j < i \leq n$.

Once again, the system (5.1) is easy to solve if A is upper triangular.

$$
\begin{aligned}
&\vdots \\
a_{ii}x_i + \cdots + a_{in-1}x_{n-1} + a_{1n}x_n &= b_i && \Longrightarrow && x_i = \frac{b_i - \displaystyle\sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \\
&\vdots \\
a_{n-1n-1}x_{n-1} + a_{n-1n}x_n &= b_{n-1} && \Longrightarrow && x_{n-1} = \frac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}} \\
a_{nn}x_n &= b_n && \Longrightarrow && x_n = \frac{b_n}{a_{nn}}
\end{aligned}
$$

Upper-triangular matrices:

The matrix A is **upper triangular** if $a_{ij} = 0$ for all $1 \leq j < i \leq n$.

Once again, the system (5.1) is easy to solve if A is upper triangular.

$$
\begin{aligned}
& \vdots \\
a_{ii}x_i + \cdots + a_{in-1}x_{n-1} + a_{1n}x_n &= b_i \qquad \Longrightarrow \quad x_i = \frac{b_i - \displaystyle\sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \\
& \vdots \\
a_{n-1n-1}x_{n-1} + a_{n-1n}x_n &= b_{n-1} \quad \Longrightarrow \quad x_{n-1} = \frac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}} \\
a_{nn}x_n &= b_n \qquad \Longrightarrow \quad x_n = \frac{b_n}{a_{nn}}
\end{aligned}
$$

Again, this works if, and only if, $a_{ii} \neq 0$ for each i.

Upper-triangular matrices:

The matrix A is **upper triangular** if $a_{ij} = 0$ for all $1 \leq j < i \leq n$.

Once again, the system (5.1) is easy to solve if A is upper triangular.

$$
\begin{aligned}
&\vdots \\
a_{ii}x_i + \cdots + a_{in-1}x_{n-1} + a_{1n}x_n &= b_i &&\implies \quad x_i = \frac{b_i - \displaystyle\sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \\
&\vdots \\
a_{n-1n-1}x_{n-1} + a_{n-1n}x_n &= b_{n-1} &&\implies \quad x_{n-1} = \frac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}} \\
a_{nn}x_n &= b_n &&\implies \quad x_n = \frac{b_n}{a_{nn}}
\end{aligned}
$$

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 $\frac{1}{2}n^2$ 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 $\implies$ Gauss(ian) elimination.

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(ian) elimination.

Example

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

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

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

Gaussian 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.

Gaussian Elimination

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

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

$$\begin{aligned} \texttt{row}\ i \quad &\leftarrow \quad \texttt{row}\ i - \frac{a_{ij}}{a_{jj}} * \texttt{row}\ j \\ b_i \quad &\leftarrow \quad b_i - \frac{a_{ij}}{a_{jj}} * b_j \end{aligned}$$

end

end

Example.

$$\begin{pmatrix} 3 & -1 & 2 \\ 1 & 2 & 3 \\ 2 & -2 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 12 \\ 11 \\ 2 \end{pmatrix} : \quad \text{represent as} \left(\begin{array}{ccc|c} 3 & -1 & 2 & 12 \\ 1 & 2 & 3 & 11 \\ 2 & -2 & -1 & 2 \end{array}\right)$$

$$\Longrightarrow \quad \begin{array}{l} \text{row } 2 \leftarrow \text{row } 2 - \frac{1}{3}\text{row } 1 \\ \text{row } 3 \leftarrow \text{row } 3 - \frac{2}{3}\text{row } 1 \end{array} \left(\begin{array}{ccc|c} 3 & -1 & 2 & 12 \\ 0 & \frac{7}{3} & \frac{7}{3} & 7 \\ 0 & -\frac{4}{3} & -\frac{7}{3} & -6 \end{array}\right)$$

$$\Longrightarrow \quad \text{row } 3 \leftarrow \text{row } 3 + \frac{4}{7}\text{row } 2 \left(\begin{array}{ccc|c} 3 & -1 & 2 & 12 \\ 0 & \frac{7}{3} & \frac{7}{3} & 7 \\ 0 & 0 & -1 & -2 \end{array}\right)$$

Back substitution:

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

Cost of Gaussian Elimination:

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`

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 $n - j$ flops as the **multiplier** a_{ij}/a_{jj} is calculated with just one flop; a_{jj} is called the **pivot**.

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 $n - j$ 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

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

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 $n - j$ 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

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

The calculations involving b are

$$\sum_{j=1}^{n-1} (n-j) = \sum_{l=1}^{n-1} l = \frac{n(n-1)}{2} = \frac{1}{2}n^2 + \mathcal{O}(n) \quad \text{flops,}$$

just as for the triangular substitution.

LU Factorization

The basic operation of Gaussian Elimination, row $i \leftarrow$ row $i + \lambda *$ row j can be achieved by pre-multiplication by a special lower-triangular matrix

$$M(i,j,\lambda) = I + \begin{array}{cc} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{array}{c} \\ \leftarrow i \\ \\ \end{array} \\ \begin{array}{ccc} & \uparrow & \\ & j & \end{array} & \end{array}$$

where I is the identity matrix.

Example: $n = 4$,

$$M(3,2,\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad M(3,2,\lambda)\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ \lambda b + c \\ d \end{pmatrix},$$

i.e., $M(3,2,\lambda)A$ performs: row 3 of $A \leftarrow$ row 3 of $A + \lambda *$ row 2 of A and similarly $M(i,j,\lambda)A$ performs: row i of $A \leftarrow$ row i of $A + \lambda *$ row j of A.

Example: $n = 4$,

$$M(3,2,\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } M(3,2,\lambda) \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ \lambda b + c \\ d \end{pmatrix},$$

i.e., $M(3,2,\lambda)A$ performs: row 3 of $A \leftarrow$ row 3 of $A + \lambda *$ row 2 of A and similarly $M(i,j,\lambda)A$ performs: row i of $A \leftarrow$ row i of $A + \lambda *$ row j of A.

So GE for e.g., $n = 3$ is

$$\underset{l_{32} = \frac{a_{32}}{a_{22}}}{M(3,2,-l_{32})} \cdot \underset{l_{31} = \frac{a_{31}}{a_{11}}}{M(3,1,-l_{31})} \cdot \underset{l_{21} = \frac{a_{21}}{a_{11}}}{M(2,1,-l_{21})} \cdot A = U = \underset{\text{upper triangular}}{(\ \diagdown\!\!\rceil\ )}.$$

Example: $n = 4$,

$$M(3,2,\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } M(3,2,\lambda) \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ \lambda b + c \\ d \end{pmatrix},$$

i.e., $M(3,2,\lambda)A$ performs: row 3 of $A \leftarrow$ row 3 of $A + \lambda *$ row 2 of A and similarly $M(i,j,\lambda)A$ performs: row i of $A \leftarrow$ row i of $A + \lambda *$ row j of A.

So GE for e.g., $n = 3$ is

$$\underset{l_{32} = \frac{a_{32}}{a_{22}}}{M(3,2,-l_{32})} \cdot \underset{l_{31} = \frac{a_{31}}{a_{11}}}{M(3,1,-l_{31})} \cdot \underset{l_{21} = \frac{a_{21}}{a_{11}}}{M(2,1,-l_{21})} \cdot A = U = (\quad).$$

upper

triangular

The l_{ij} are the **multipliers**.

Be careful:

each multiplier l_{ij} uses the data a_{ij} and a_{ii} that *results from the transformations already applied*, not data from the original matrix.

So l_{32} uses a_{32} and a_{22} that result from the previous transformations $M(2, 1, -l_{21})$ and $M(3, 1, -l_{31})$.

Lemma

If $i \neq j$*,* $(M(i,j,\lambda))^{-1} = M(i,j,-\lambda)$*.*

Proof. Exercise.

Outcome: for $n = 3$, $A = M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) \cdot U$, where

$$M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} = L = \left(◺ \right).$$

lower

triangular

Outcome: for $n = 3$, $A = M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) \cdot U$, where

$$M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} = L = \underbrace{(\,\triangle\,)}_{\text{lower triangular}}.$$

This is true for general n.

Outcome: for $n = 3$, $A = M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) \cdot U$, where

$$M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} = L = (\,\text{◺}\,).$$

lower

triangular

This is true for general n.

Theorem

For any dimension n, GE can be expressed as $A = LU$, where $U = (\,\text{◹}\,)$ is upper triangular resulting from GE, and $L = (\,\text{◺}\,)$ is unit lower triangular (lower triangular with ones on the diagonal) with $l_{ij} =$ multiplier used to create the zero in the (i,j)th position.

Most implementations of GE therefore, rather than doing GE as above,

$$
\begin{aligned}
\text{factorize} \quad & A = LU && \text{(takes } \approx \tfrac{1}{3}n^3 \text{ flops)} \\
\text{and then solve} \quad & Ax = b && \\
\text{by solving} \quad & Ly = b && \text{(forward substitution)} \\
\text{and then} \quad & Ux = y && \text{(back substitution)}
\end{aligned}
$$

Note: this is much more efficient if we have many different right-hand sides b but the same A.