PART IV - Systems of Equations

Lecture 20 - *Gaussian Elimination*

OBJECTIVE:
We all know about Gaussian elimination.
It is the simplest way to solve linear systems of equations by hand, and a standard, systematic way to solve them on a computer.
In this lecture we describe Gaussian elimination in its pure form.
In the next lecture we will add the concept of row pivoting, which is essential to the stability of the algorithm.

◇ LU FACTORIZATION

Gaussian elimination can be viewed as transforming a full linear system into an upper-triangular one by transformations on the left.

It is similar to Householder triangularization, except now the transformations are not orthogonal.

Let $\mathbf{A} \in \mathbb{R}^{m \times m}$.
(Gaussian elimination is rarely applied to rectangular matrices.)

The idea is to introduce zeros below the diagonal one column at a time (like Householder triangularization).

This is accomplished by subtracting multiples of each row from subsequent rows.

The elimination process is equivalent to multiplying $\mathbf{A}$ by a sequence of lower-triangular matrices $\mathbf{L}_k$ on the left

$$\mathbf{L}_{m-1}\mathbf{L}_{m-2}\dots\mathbf{L}_1\mathbf{A} = \mathbf{U}$$

Setting

$$\mathbf{L} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\dots\mathbf{L}_{m-1}^{-1}$$

gives an *LU factorization* of $\mathbf{A}$

$$\mathbf{A} = \mathbf{L}\mathbf{U}$$

where $\mathbf{L}$ is lower-triangular and $\mathbf{U}$ is upper-triangular.

In practice we can choose $\mathbf{L}$ to be *unit lower-triangular*, which means all of its diagonal entries are equal to 1.

Example: Suppose $\mathbf{A}$ is 4×4

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{\mathbf{L}_1} \begin{bmatrix} \times & \times & \times & \times \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}$$

$\mathbf{A}$ $\qquad$ $\mathbf{L}_1\mathbf{A}$

$$\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix} \xrightarrow{\mathbf{L}_2} \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & 0 & * & * \\ & 0 & * & * \end{bmatrix}$$

$\mathbf{L}_1\mathbf{A}$ $\qquad$ $\mathbf{L}_2\mathbf{L}_1\mathbf{A}$

$$\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & \times & \times \end{bmatrix} \xrightarrow{\mathbf{L}_3} \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & 0 & * \end{bmatrix}$$

$\mathbf{L}_2\mathbf{L}_1\mathbf{A}$ $\qquad$ $\mathbf{L}_3\mathbf{L}_2\mathbf{L}_1\mathbf{A}$

$\longrightarrow$ compare and contrast with QR

The k^{th} transformation $\mathbf{L}_k$ introduces zeros below the diagonal in column k by subtracting multiples of row k from rows $k+1, k+2, \ldots, m$.

Because the first $k-1$ entries of row k are already zero, previous zeros are not destroyed (or *filled-in*).

So,

$$\text{Gaussian elimination} \;=\; \text{triangular triangularization}$$

$\diamond$ EXAMPLE

Let

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}$$

(This $\mathbf{A}$ was chosen because it has a nice LU decomposition.)

So,

$$\mathbf{L}_1\mathbf{A} = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ -4 & & 1 & \\ -3 & & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix}$$

(Subtract 2 $\times$ row 1 from row 2, 4 $\times$ row 1 from row 3, and 3 $\times$ row 1 from row 4)

Further,

$$\mathbf{L}_2\mathbf{L}_1\mathbf{A} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & -3 & 1 & \\ & -4 & & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix}$$

(Subtract 3 $\times$ row 2 from row 3 and 4 $\times$ row 2 from row 4)

Finally,

$$\begin{aligned} \mathbf{L}_3\mathbf{L}_2\mathbf{L}_1\mathbf{A} &= \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & & 2 \end{bmatrix} = \mathbf{U} \end{aligned}$$

(Subtract row 3 from row 4)

To obtain $\mathbf{A} = \mathbf{LU}$, we need to form

$$\mathbf{L} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\mathbf{L}_3^{-1}.$$

This turns out to be trivial, thanks to two happy surprises.

$\mathbf{L}_1^{-1}$ can be obtained from $\mathbf{L}_1$ by negating the off-diagonal entries:

$$\mathbf{L}_1^{-1} = \begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ -4 & & 1 & \\ -3 & & & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & & 1 & \\ 3 & & & 1 \end{bmatrix}$$

(Similarly, the same is true for $\mathbf{L}_2$ and $\mathbf{L}_3$.)

Also $\mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\mathbf{L}_3^{-1}$ is obtained by "merging" the individual non-zero elements.

i.e.,

$$\mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\mathbf{L}_3^{-1} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & 3 & 1 & \\ 3 & 4 & 1 & 1 \end{bmatrix}$$

Thus,

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} \\ &= \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & 3 & 1 & \\ 3 & 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & & 2 \end{bmatrix} = \mathbf{LU}\end{aligned}$$

◇ GENERAL FORMULAS AND TWO STROKES OF LUCK

For a general $m \times m$ matrix, let $\mathbf{x}_k$ denote the k^{th} column of the matrix at the beginning of step k.

Then $\mathbf{L}_k$ must be chosen so that

$$\mathbf{x}_k = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{kk} \\ x_{k+1,k} \\ \vdots \\ x_{mk} \end{bmatrix} \qquad \overrightarrow{\mathbf{L}_k} \qquad \mathbf{L}_k\mathbf{x}_k = \begin{bmatrix} x_{1k} \\ \vdots \\ x_{kk} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

To do this, we subtract $l_{jk} \times$ row k from row j:

$$l_{jk} = \frac{x_{jk}}{x_{kk}} \qquad j = k+1, k+2, \ldots, m$$

(l_{jk} is known as the *multiplier*)

Then $\mathbf{L}_k$ takes the form

$$\mathbf{L}_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -l_{mk} & & & 1 \end{bmatrix}$$

In the example, we noted two strokes of luck

1. $\mathbf{L}_k$ can be inverted by negating the off-diagonal elements $-l_{jk}$,

2. $\mathbf{L}$ can be formed by merging the entries l_{jk}.

We can explain these bits of good fortune as follows:

Let

$$\mathbf{l}_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ l_{k+1,k} \\ \vdots \\ l_{mk} \end{bmatrix}$$

Then

$$\mathbf{L}_k = \mathbf{I} - \mathbf{l}_k \mathbf{e}_k^T \qquad \textit{(verify!)}$$

By construction,

$$\mathbf{e}_k^T \mathbf{l}_k = 0 \qquad \textit{(verify!)}$$

Thus

$$(\underbrace{\mathbf{I} - \mathbf{l}_k \mathbf{e}^T}_{\mathbf{L}_k})(\mathbf{I} + \mathbf{l}_k \mathbf{e}^T) = \mathbf{I} - \mathbf{l}_k \mathbf{e}^T \mathbf{l}_k \mathbf{e}_k^T = \mathbf{I}$$

i.e.,

$$\mathbf{L}_k^{-1} = \mathbf{I} + \mathbf{l}_k \mathbf{e}^T$$

($= \mathbf{L}$ with all the entries in $\mathbf{l}_k$ negated)

Now consider for example $\mathbf{L}_k^{-1}\mathbf{L}_{k+1}^{-1}$.

By construction,

$$\mathbf{e}_k^T \mathbf{l}_{k+1} = 0 \qquad \textit{(verify!)}$$

So,

$$\begin{aligned} \mathbf{L}_k^{-1}\mathbf{L}_{k+1}^{-1} &= (\mathbf{I} + \mathbf{l}_k \mathbf{e}_k^T)(\mathbf{I} + \mathbf{l}_{k+1}\mathbf{e}_{k+1}^T) \\ &= \mathbf{I} + \mathbf{l}_k \mathbf{e}_k^T + \mathbf{l}_{k+1}\mathbf{e}_{k+1}^T \end{aligned}$$

($= \mathbf{L}$ with entries of $\mathbf{L}_k^{-1}, \mathbf{L}_{k+1}^{-1}$ merged)

This of course can be generalized m times to obtain

$$\mathbf{L} = \mathbf{L}_1^{-1}\mathbf{L}_2^{-1}\dots\mathbf{L}_{m-1}^{-1} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{m1} & l_{m2} & \dots & l_{m,m-1} & 1 \end{bmatrix}$$

Note 1. *We used the analogue of this result for upper-triangular matrices when interpreting the modified Gram-Schmidt process as a sequence of right-multiplications by a sequence of upper-triangular matrices* $\mathbf{R}_k$*.*

Of course, in practice, the matrices $\mathbf{L}_k$ are never formed and multiplied explicitly
$\longrightarrow$ only the multipliers are computed and stored, often in the lower triangle of $\mathbf{A}$, which would otherwise contain zeros upon reduction to $\mathbf{U}$

ALGORITHM 20.1: GAUSSIAN ELIMINATION WITHOUT PIVOTING

$\mathbf{U} = \mathbf{A}$
$\mathbf{L} = \mathbf{I}$
for $k = 1$ to $m - 1$ **do**

  for $j = k + 1$ to m **do**
    $l(j, k) = u(j, k)/u(k, k)$
    $u(j, k : m) = u(j, k : m) - l(j, k) * u(k, k : m)$
  end for
end for

◇ OPERATION COUNT

The dominant expense is the line

$$u(j, k : m) = u(j, k : m) - l(j, k) * u(k, k : m)$$

→ one scalar-vector multiplication, and one vector subtraction.

Suppose $l = m - k$ is the length of the row vectors being manipulated.

Then the number of flops is

$$l \text{ multiplications} + l \text{ subtractions} = 2l \text{ flops}$$

or 2 flops per entry.

So total flops

$$\begin{aligned}
&\sim \sum_{k=1}^{m-1} \sum_{j=k+1}^{m} (m-k) \qquad \text{entries } \times 2 \text{ flops per entry} \\
&\sim 2 \sum_{k=1}^{m-1} (m-k)^2 \\
&\sim 2 \sum_{k=1}^{m-1} m^2 + k^2 - 2mk \\
&= 2 \left[m^2(m-1) + \frac{m(m-1)(2(m-1)+1)}{6} - 2m\frac{m(m-1)}{2} \right] \\
&\sim 2 \left[m^3 + \frac{m^3}{3} - m^3 \right] \\
&= \frac{2m^3}{3}
\end{aligned}$$

◇ SOLUTION OF $\mathbf{Ax} = \mathbf{b}$ BY LU FACTORIZATION

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{LUx} &= \mathbf{b} \end{aligned}$$

This can be solved by solving *two triangular systems*.

First solve $\mathbf{Ly} = \mathbf{b}$ for an intermediate variable $\mathbf{y}$ (forward substitution).

Then solve $\mathbf{Ux} = \mathbf{y}$ for the unknown variable $\mathbf{x}$ (back substitution).

The initial factorization of $\mathbf{A}$ into $\mathbf{LU}$ costs $\sim \frac{2}{3}m^3$ flops.

The forward and back substitutions each require $\sim m^2$ flops.

$\rightarrow$ the total work to solve $\mathbf{Ax} = \mathbf{b}$ by LU decomposition is $\sim \frac{2}{3}m^3$ flops, half the amount required by Householder triangularization.

This factor of 2 is certainly one reason why Gaussian elimination is more standard than QR factorization to solve (square) systems of equations.

The other reason may largely be historical: Gaussian elimination has been known for centuries, while QR only appeared with the invention of computers.

◇ INSTABILITY OF GAUSSIAN ELIMINATION WITHOUT PIVOTING

Unfortunately, Gaussian elimination as presented in its pure form is unusable for solving general linear systems because it is not backward stable.

This instability is related to the more obvious difficulty that

Gaussian elimination breaks down if $x_{kk} = 0$*!*

e.g. Consider

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

This matrix has full rank and is well-conditioned:

$$\kappa(\mathbf{A}) = \frac{3+\sqrt{5}}{2} \approx 2.618$$

Perturb $\mathbf{A}$ slightly to

$$\mathbf{A} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}.$$

Now the process does not fail.

But, it yields

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{+20} \end{bmatrix}$$ *(verify!)*

In a normal computing environment with $\epsilon_{\text{machine}} = 10^{-16}$,

$$1 - 10^{20} = -10^{20}.$$

$$\therefore \qquad \tilde{\mathbf{L}} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \qquad \tilde{\mathbf{U}} \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{+20} \end{bmatrix}$$

This doesn't look like much rounding off...however,

$$\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix}$$

This is not close to $\mathbf{A}$*!*

$\rightarrow$ if we now solve $\mathbf{Ax} = \mathbf{b}$, our solution will be completely off.

e.g., with $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, we get $\tilde{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

But, the correct answer is $\mathbf{x} \approx \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ *(verify!)*

$\rightarrow$ Gaussian elimination has computed $\mathbf{L}$ and $\mathbf{U}$ stably ($\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$ are close to the true factors $\mathbf{L}$ and $\mathbf{U}$).

But we have not been able to use $\tilde{\mathbf{L}}, \tilde{\mathbf{U}}$ to solve $\mathbf{Ax} = \mathbf{b}$ stably.

The reason is that LU factorization (although stable) is <u>not</u> backward stable.

In general, if one step of an algorithm is stable but not backward stable, the stability of the overall calculation may be in jeopardy.

In fact, in general, Gaussian elimination is neither backward stable nor stable as a general algorithm for LU factorization.

$\longrightarrow$ the triangular matrices may have condition numbers that are arbitrarily larger than that of $\mathbf{A}$ itself, leading to instabilities in the forward and back substitution phases of solving $\mathbf{Ax} = \mathbf{b}$.