Many software libraries contain **no provision** for Gauss Elimination — WHY NOT?

if you solve $\mathbf{Ax} = \mathbf{b}_1$ ($\approx \frac{1}{3}n^3$ operations) and, some time later, solve $\mathbf{Ax} = \mathbf{b}_2$ ($\approx \frac{1}{3}n^3$ operations) most of these operations have been done before! (all but n^2 operations)

$\implies$ try to store information so you don't waste this effort.

In practice, we store the multipliers l_{ij} in a lower triangular matrix

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix}$$

$$\mathbf{L} = \begin{cases} l_{ij} & i > j \\ 1 & i = j \\ 0 & i < j \end{cases}$$

LU factorization

Then, if no pivoting was required

$$\mathbf{A} = \mathbf{LU}$$

the **LU factorization of A**

where $\mathbf{U}$ is the upper triangular matrix resulting from Gauss Elimination

Proof.

Use elementary matrices representing each row op $\square$

Note: the LU factorization is not unique, since

$$\mathbf{LU} = (\mathbf{LD})(\mathbf{D}^{-1}\mathbf{U}) = \bar{\mathbf{L}}\bar{\mathbf{U}}$$

where $\mathbf{D}$ is any nonsingular diagonal matrix.

Using the LU factors

Now our strategy becomes: to solve $\mathbf{A}\mathbf{x} = \mathbf{b}_1$

1. factorize $\mathbf{A} = \mathbf{L}\mathbf{U}$ ($\approx \frac{1}{3}n^3$ operations)

2. to solve $\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b}_1$; solve $\mathbf{L}\mathbf{y} = \mathbf{b}_1$ by forward substitution ($\approx \frac{1}{2}n^2$ operations) where $\mathbf{y} = \mathbf{U}\mathbf{x}$

3. solve $\mathbf{U}\mathbf{x} = \mathbf{y}$ by back substitution ($\approx \frac{1}{2}n^2$ operations)

Total: $\approx \frac{1}{3}n^3 + n^2$ operations
as before!

Re-using the LU factors

then, to solve $\mathbf{A}\mathbf{x} = \mathbf{b}_2$ (same $\mathbf{A}$, different RHS)

1. $\mathbf{L}, \mathbf{U}$ known, so don't factorize again
2. solve $\mathbf{L}\mathbf{y} = \mathbf{b}_2$ by forward substitution ($\approx \frac{1}{2}n^2$ operations)
3. solve $\mathbf{U}\mathbf{x} = \mathbf{y}$ by back substitution ($\approx \frac{1}{2}n^2$ operations)

$\implies$ next system with same $\mathbf{A}$ takes $\approx n^2$ operations

This is why most libraries have an LU factorization procedure plus an LU solve procedure instead of Gauss elimination

If row interchanges are required

$$\mathbf{PA} = \mathbf{LU}$$

the **LU factorization of row-permuted A**

where **P** is a permutation matrix describing the row interchanges

Example

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

is a matrix that swaps the 1st and 3rd rows of another matrix.

i.e. **PA** is **A** but with row interchanges required during Gauss Elimination done in advance

Using permuted LU

then to solve $\mathbf{Ax} = \mathbf{b}$:

1. $$\mathbf{PAx} = \mathbf{Pb}$$

   so that

   $$\mathbf{LUx} = \mathbf{Pb}$$

2. solve $\mathbf{Ly} = \mathbf{Pb}$
3. solve $\mathbf{Ux} = \mathbf{y}$

Must permute RHS as well!

LU factorization + PP does the same as GEPP, just with steps re-organized

- **Demo**

Matrices with special structure

Gauss Elimination and LU factorization apply to general dense matrices.
If the matrix has any special structure, it's often possible to exploit this.
We touch on 2 examples.

1. **Positive definite matrices**

Definition

A is **positive definite** iff

$$\mathbf{x}^T \mathbf{A} \mathbf{x} > 0 \ \forall \mathbf{x} \neq 0$$

usually consider only symmetric positive definite matrices

- arise naturally in some applications e.g. Least Squares fitting
- same as having all positive eigenvalues

Choleski factorization

Theorem

$\mathbf{A}$ is **symmetric positive definite** *is equivalent to*
$\mathbf{A}$ has a symmetric triangular factorization

$$\mathbf{A} = \mathbf{R}^T\mathbf{R}$$

where $\mathbf{R}$ is a nonsingular upper (right) triangular matrix.

Called the **Choleski factorization** of $\mathbf{A}$ after Choleski: †1918
$\mathbf{R}^T$ plays the same role as $\mathbf{L}$; $\mathbf{R}$ plays the same role as $\mathbf{U}$.
The Choleski factorization is unique.

Proof.

To show $\mathbf{A} = \mathbf{R}^T\mathbf{R} \implies \mathbf{A}$ is positive definite:
$\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{R}^T\mathbf{R}\mathbf{x} = (\mathbf{R}\mathbf{x})^T\mathbf{R}\mathbf{x} = ||\mathbf{R}\mathbf{x}||^2 > 0 \ \forall \mathbf{x} \neq 0$
To show $\mathbf{A} = \mathbf{R}^T\mathbf{R} \implies \mathbf{A}$ is symmetric:
$\mathbf{A}^T = (\mathbf{R}^T\mathbf{R})^T = \mathbf{R}^T(\mathbf{R}^T)^T = \mathbf{R}^T\mathbf{R} = \mathbf{A}$
To show converse: by induction. $\square$

This factorization takes $\approx \frac{1}{6}n^3 \times/\div + n\sqrt{}$ operations

Moral: If you know $\mathbf{A}$ is symmetric positive definite, can save 50% of work by Choleski factorization.

Turns out that Choleski factorization is not vulnerable to subtractive cancellation, so don't need to pivot.

Banded matrices

A matrix is banded if it has nonzeroes only near the main diagonal:

$$A_{ij} = 0 \text{ for } \mid i - j \mid > k$$

bandwidth of matrix $= 2k + 1$
i.e. a regular pattern of zeroes, a generalization of diagonal matrices.

Example

- $k = 1$ **tridiagonal matrices**
- $k = 2$ **pentadiagonal matrices**

Fast factorization

If no pivoting is required, the **LU** factors inherit this banded structure
$\rightarrow$ can be factored very fast

Example

the LU factors of a tridiagonal matrix are bidiagonal
hence a tridiagonal matrix can be factored without pivoting in $\approx 2n \times / \div$ operations

VERY FAST!

Use any special structure of the matrix to save time/storage.

If pivoting is required, it destroys the banded structure, so the factors are not banded!

What does MATLAB's backslash do?

To solve the system $\mathbf{Ax} = \mathbf{b}$, in MATLAB just type

```
x=A\b;
```

to remember this, think of it as premultiplying by the inverse: $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

This **backslash** command (amongst other things):

- solves by substitution if $\mathbf{A}$ is triangular
- attempts Cholesky factorization if $\mathbf{A}$ is symmetric
- does GEPP by LU factorization if $\mathbf{A}$ is a general dense matrix

So, easiest way to solve with LU if you want to keep the LU factors is

```
[L,U,P]=lu(A); x=U\(L\(P*b))
```

similarly for Cholesky:

```
R=chol(A); x=R\(R'\b)
```

End of Week 6