LU Factorization:
Computing the Determinants

Yixuan Liu
Laura Vetter
Josh Cohen
Jae Yoon Chun

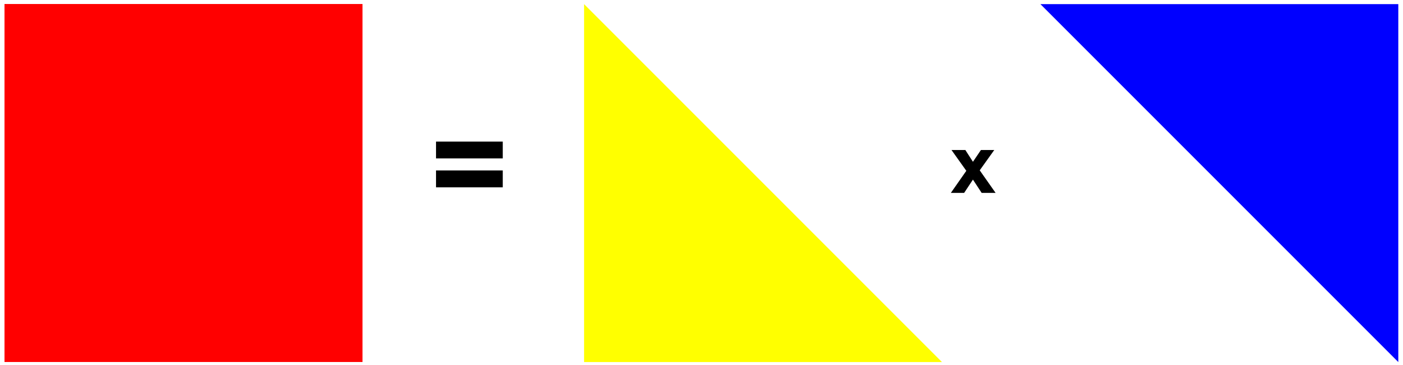
Questions to be answered…

- What is LU Factorization?
- How do we compute determinants from LU Factorization?
- Does LU Factorization always work for square matrices?
- Is it a more efficient way to calculate determinants compared to other methods?

LU Factorization

Definition:

If **A** is an $n \times n$ matrix over a field F ($A \in M_n(F)$) , then A is said to "have a LU factorization " if there exists a lower-triangular matrix **L** $\in M_n(F)$ and an upper triangular matrix **U** $\in M_n(F)$ such that **A=LU.**



This process is a by-product of Gaussian Elimination.

Finding LU factorization via Gaussian Elimination

Assuming row swaps are not required, then the Gaussian Elimination takes n-1 steps in total.

Let

$\mathbf{A^{(k)}}$: the k-th step of the Gaussian Elimination of A.

U: the reduced row echelon form of A (i.e. $\mathbf{U=A^{(n-1)}}$).

$\mathbf{M_1, M_2, \ldots, M_{n-1}}$: the elimination matrices.

Then we have

$A^{(1)} = M_1 * A$

$A^{(2)} = M_2 * A^{(1)} = M_2 * M_1 * A$

$\vdots$

$U = A^{(n-1)} = M_{n-1} * M_{n-2} * \ldots * M_1 * A$

Thus

$\mathbf{L = (M_{n-1} M_{n-2} \ldots M_1)^{-1} = M_1^{-1} M_2^{-1} \ldots M_{n-2}^{-1} M_{n-1}^{-1}}$ (i.e. the inverse of all elimination processes)

Permutation Matrix

Definition:
A permutation matrix, P, is obtained by permuting the rows of an $n \times n$ identity matrix according to some permutation of the number 1 to n.

Not all matrices can be written as LU, because it is sometimes necessary to swap rows during Gaussian Elimination. However, taking accounts of these swaps, we can find a permutation matrix P so that **A=PLU**.

Example of a decomposition:
https://www.student.cs.uwaterloo.ca/~cs370/notes/LUExample2.pdf

Calculating Determinants using PLU Factorization

Suppose A has a PLU factorization, A = PLU. Then

det (A) = det(PLU)
= det(P) det(L) det(U)

Since det(L) = 1,

det (A) = (-1)^r det(U)
where r is the number of row swaps

Finding LU factorization via Gaussian Elimination

Assuming row swaps are not required, then the Gaussian Elimination takes n-1 steps in total.

Let

$A^{(k)}$: the k-th step of the Gaussian Elimination of A.

U: the reduced row echelon form of A (i.e. **$U=A^{(n-1)}$**).

$M_1, M_2, \ldots, M_{n-1}$: the elimination matrices.

$$\begin{bmatrix} 0 & 0 & 6 \\ 2 & 1 & 4 \\ 1 & 2 & 3 \end{bmatrix}$$

Then we have

$A^{(1)} = M_1 * A$

$A^{(2)} = M_2 * A^{(1)} = M_2 * M_1 * A$

.

.

.

$U = A^{(n-1)} = M_{n-1} * M_{n-2} * \ldots * M_1 * A$

Thus

$L = (M_{n-1} M_{n-2} \ldots M_1)^{-1} = M_1^{-1} M_2^{-1} \ldots M_{n-2}^{-1} M_{n-1}^{-1}$ (i.e. the inverse of all elimination processes)

U

$$\begin{bmatrix} 0 & 0 & 6 \\ 2 & 1 & 4 \\ 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & 0 & 6 \\ 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & 0 & 6 \\ 0 & \frac{3}{2} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & \frac{3}{2} & 1 \\ 0 & 0 & 6 \end{bmatrix}$$

P

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{row swap!}} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{row swap!}} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

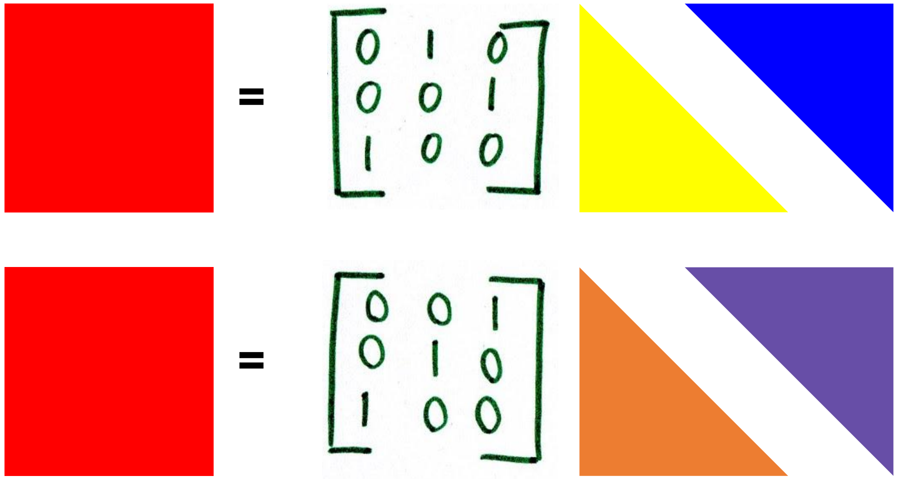
L

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{row add!}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

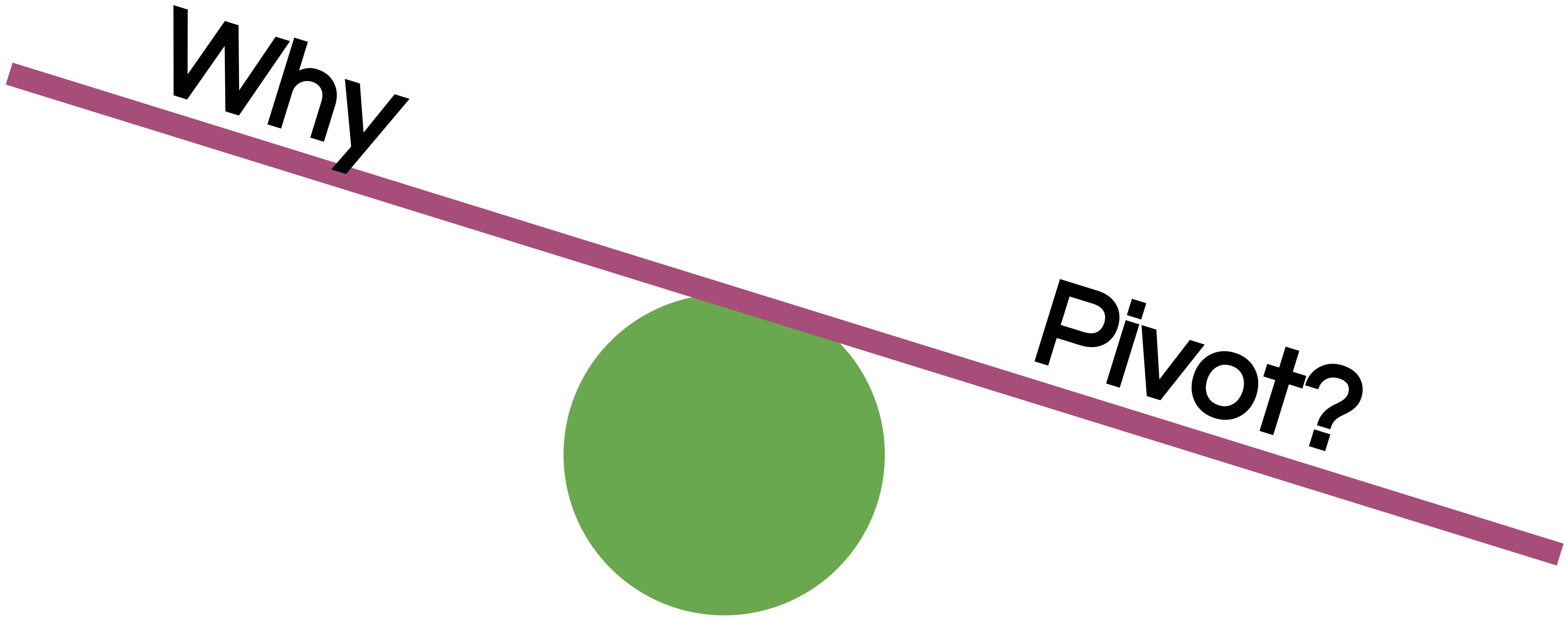
$A = P^T L U$!

Existence and Uniqueness

- The PLU factorisation always exists
- This factorisation is not unique



- However, for a fixed P, L and U are unique
- Sometimes, a particular choice of P is required to avoid serious rounding errors...



Why
Pivot?

U $\begin{bmatrix} 0 & 0 & 6 \\ 2 & 1 & 4 \\ 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & 0 & 6 \\ 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & 0 & 6 \\ 0 & \frac{3}{2} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & \frac{3}{2} & 1 \\ 0 & 0 & 6 \end{bmatrix}$

P $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{row swap!}} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{row swap!}} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$

L $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{row add!}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$A = P^T L U$!

Gaussian elimination

$$\begin{bmatrix} 2 & 1 & 4 \\ 0 & 0 & 6 \\ 1 & 2 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 4 \\ 0 & 0 & 6 \\ 0 & \frac{3}{2} & 1 \end{bmatrix}$$

$$R3 - \frac{U_{31}}{U_{11}} R1$$

$$= R3 - \frac{1}{2} R1$$

Roundoff errors

entry to be
eliminated

entry on the
diagonal

$$2 / 200 = 0.01$$

$$200 / 2 = 100$$

$$2.01 / 200 = 0.01005$$

$$200 / 2.01 =$$

$$99.502487562189054726368159203980$$
$$99502487562189054726$$

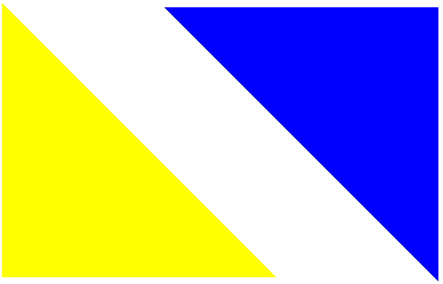
Pioneers

1948 - Alan Turing
Rounding of errors in matrix processes

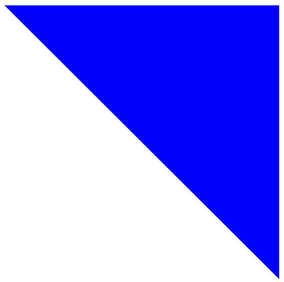
1961 - James Wilkinson
Error analysis of direct methods of matrix inversion

Solving linear systems Ax = b

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Roundoff errors

entry to be eliminated

entry on the diagonal

$$2 / 200 = 0.01$$

$$200 / 2 = 100$$

$$2.01 / 200 = 0.01005$$

$$200 / 2.01 =$$

$$99.502487562189054726368159203980995024875621890547 26$$

Condition numbers

The **condition number** (coined by turing) of a function with respect to an argument measures how much the output value of the function can change for a small change in the input argument

A problem with a low condition number is said to be **well-conditioned**, while a problem with a high condition number is said to be **ill-conditioned**

Pivoting

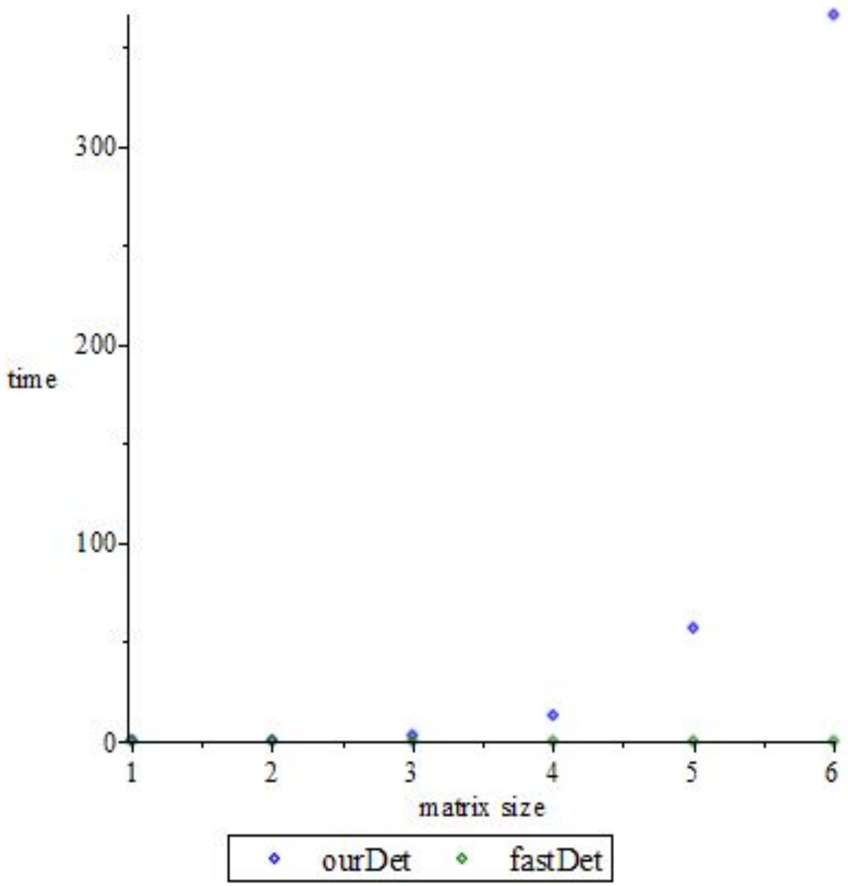
Partial Pivoting
Choose largest element in the column
LU = PA

$$\begin{bmatrix} a_{11} & a_{12} \cdots & a_{1n} \\ a_{21} & a_{22} \cdots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nn} \end{bmatrix}$$

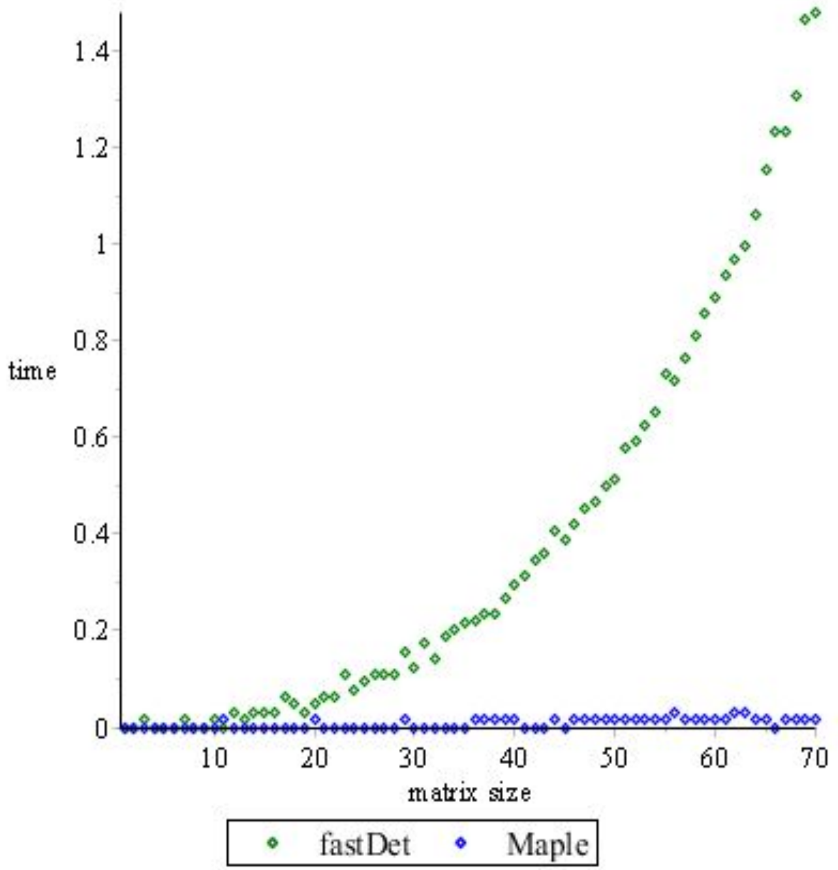
Complete Pivoting
Choose largest element in remaining portion of matrix
LU = PAQ

$$\begin{bmatrix} a_{11} & a_{12} \cdots & a_{1n} \\ a_{21} & a_{22} \cdots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} \cdots & a_{nn} \end{bmatrix}$$

Running Time Analysis



ourDet VS fastDet:
Computing the Determinant of a random 6 ×6 matrix



fastDet VS Maple:
Computing the Determinant of a random 70 × 70 matrix

Running Time

- A flop is a single (floating-point) operation x⊚y where ⊚ is any one of addition, subtraction, multiplication and division.
- Computation time increases because more calculations are required.
- The number of flops required by an algorithm to solve a problem is frequently a polynomial in the dimension(s) of the problem.
- Only the leading term of the polynomial is of interest, and generally only its degree

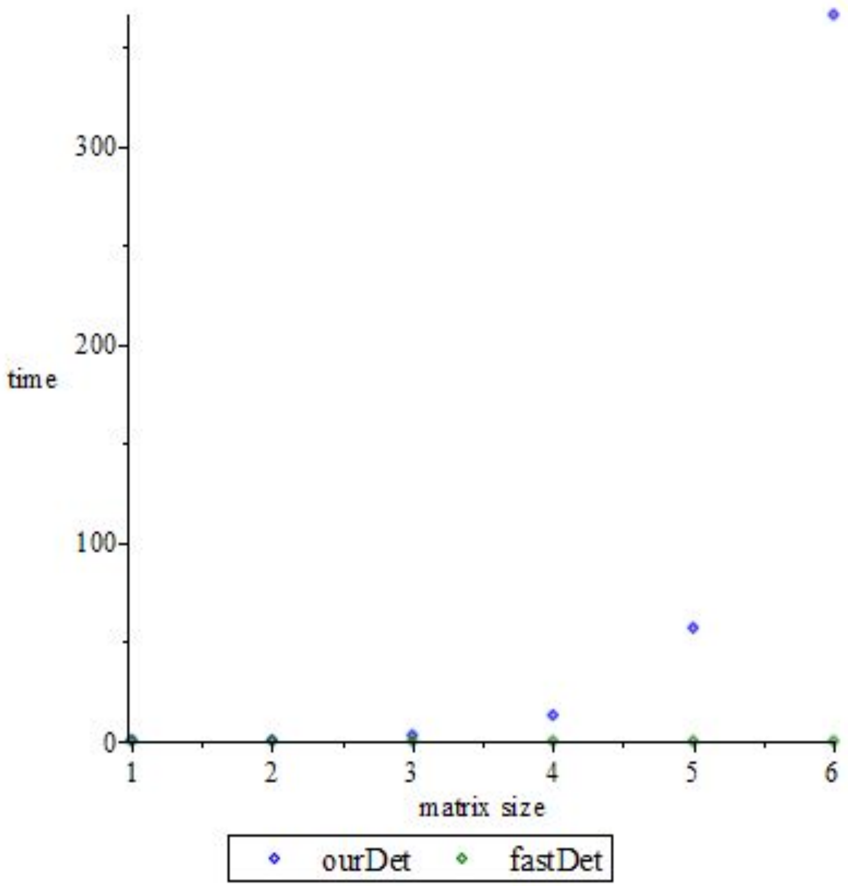
$PA = LU$: Costs and benefits

In general, considering terms in row i of *LU* = *PA*,

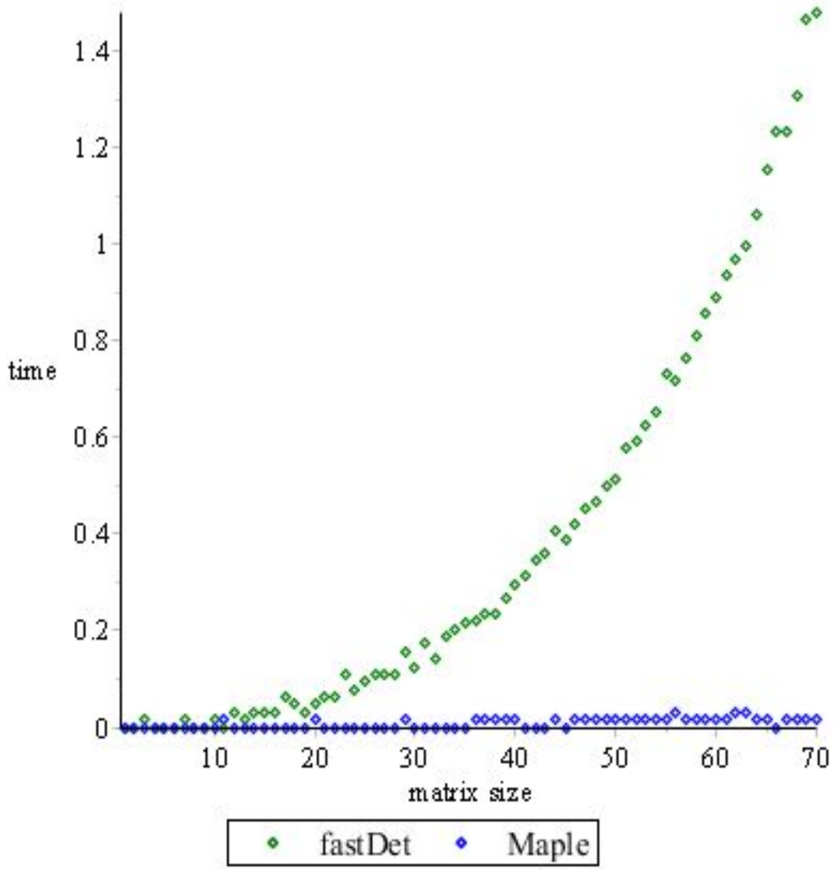
$$
\begin{aligned}
l_{ij} &= \left([PA]_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}\right)/u_{jj} \quad j = 1, \ldots, i-1 \quad \sum_{j=1}^{i-1} 2(j-1) + 1 \text{ flops} \\
u_{ij} &= [PA]_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad j = i, \ldots, n \qquad \sum_{j=i}^{n} 2(i-1) \text{ flops}
\end{aligned}
$$

- **Our method:** **flops = $2n^3/3$**
- **The method in lab 3:** **flops = $2n^3$**

Running Time Analysis



ourDet VS fastDet:
Computing the Determinant of a random 6 ×6 matrix



fastDet VS Maple:
Computing the Determinant of a random 70 × 70 matrix

Example: Matrix-vector product

- **Form $\underline{y} = A\underline{x}$ for** $A \in \mathbb{R}^{m \times n}$
- **"Algorithm" is**

$$\text{For } i = 1, \ldots, m \quad y_i = \sum_{j=1}^{n} a_{ij} x_j$$

- **Analysis:**
  - **2 flops for each term in the summation**
  - ***n* terms in the summation: total 2*n* flops per summation**
  - ***m* summations: total 2*mn* flops**
- **The flop count of 2*mn***
  - **Double either dimension and the count doubles**
  - **Each entry in *A* is used exactly once**
- **If $m = n$ then the flop count is $2n^2$.**

Example: Matrix-Matrix product

- Form $C = AB$ for $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$
- "Algorithm" is

$$\text{For } \begin{matrix} i = 1, \dots, m \\ j = 1, \dots, n \end{matrix} \quad c_{ij} = \sum_{k=1}^{r} a_{ik} b_{kj}$$

- Analysis:
  - 2 flops for each term in the summation
  - r terms in the summation: total $2r$ flops per summation
  - mn summations: total $2mnr$ flops
- The flop count of $2mnr$ is mnr
- If $r = m = n$ then the flop count is $2n^3$.

Cholesky decomposition: Computational cost

- **Cholesky decomposition general form A = LL^T**

$$\begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \dots & l_{n,n-1} & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \dots & l_{n1} \\ & l_{22} & \dots & l_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{21} & a_{22} & \dots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

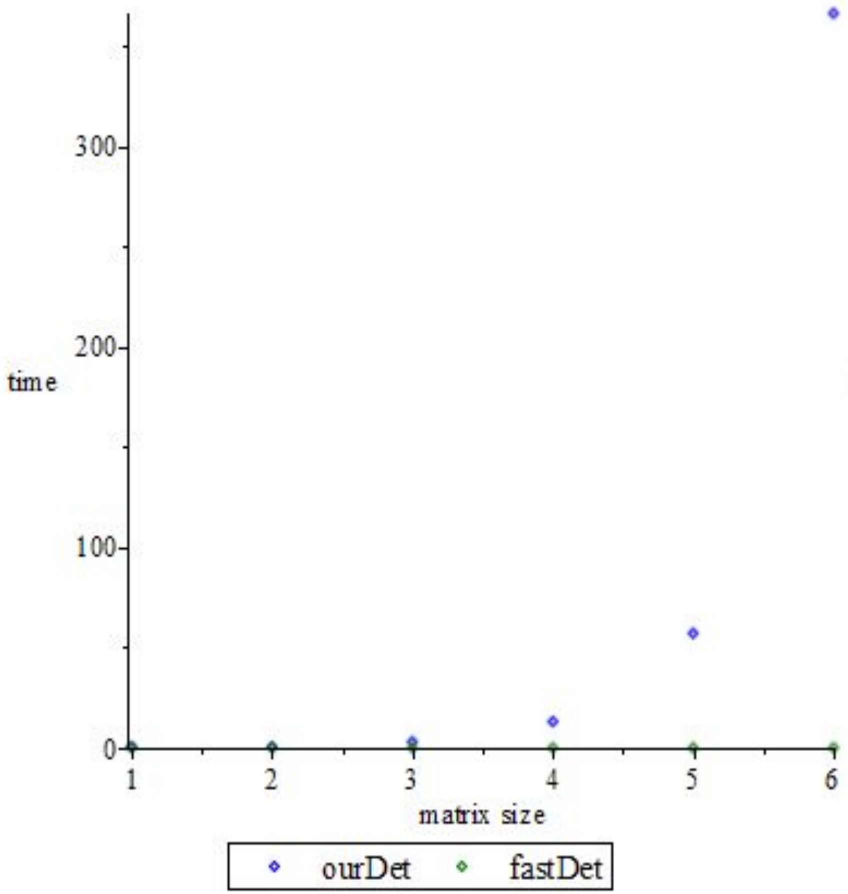
- **$n(n+1)/2$ nonlinear equations in $n(n+1)/2$ unknowns**

- **Total Cost:**

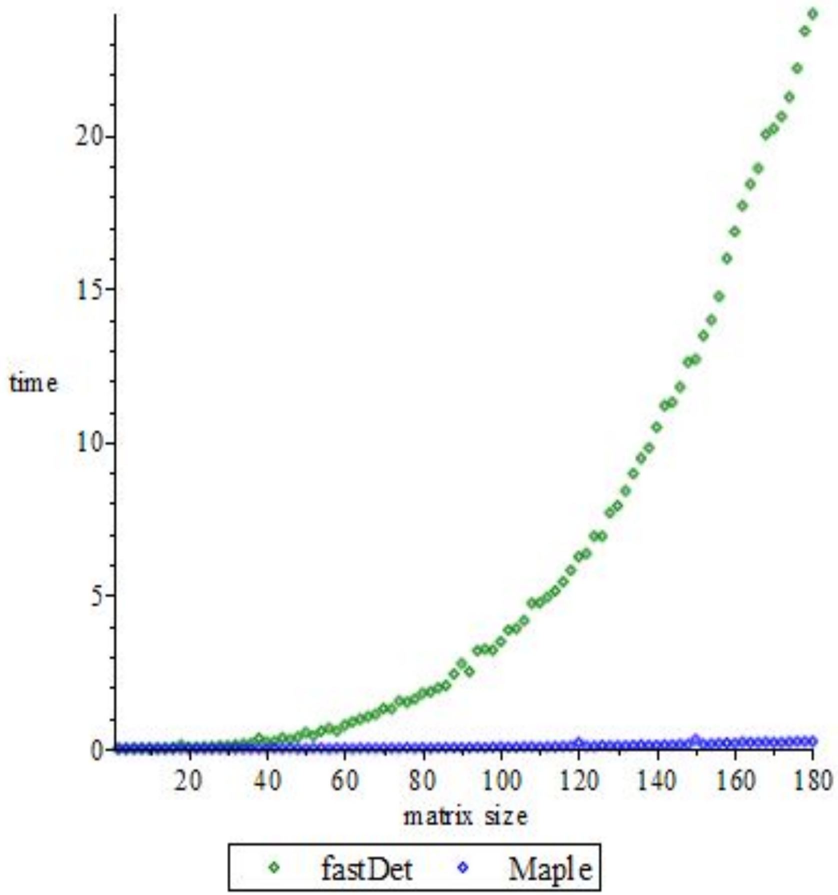
$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + O(n^2)$$

- **Cholesky decomposition is half the cost of Gaussian elimination for general A**

Running Time Analysis



ourDet VS fastDet:
Computing the Determinant of a random 6 ×6 matrix



fastDet VS Maple:
Computing the Determinant of a random 70 × 70 matrix


Any questions?

References

https://vismor.com/documents/network_analysis/matrix_algorithms/S4.php

http://math.stackexchange.com/questions/186972/how-can-lu-factorization-be-used-in-non-square-matrix

http://arxiv.org/pdf/math/0506382v1.pdf

http://mathforcollege.com/nm/simulations/nbm/04sle/nbm_sle_sim_inversecomptime.pdf

http://math.stackexchange.com/questions/160784/what-is-the-importance-of-determinants-in-linear-algebra

http://gauss.uc3m.es/web/personal_web/fdopico/papers/arbor-2011-turing.pdf