

# Determinants

## Group 12

Catterwell, A.   Smith, M.   Wang, R.   Watson, K.

University of Edinburgh

March 31, 2019

# Overview

- 1 Why calculate determinants?
  - Invertibility of matrices
  - Cramers Rule
  - Eigenvalues and Eigenvectors
- 2 Using LU decomposition to compute determinants
  - What is LU decomposition?
  - Is there always an LU decomposition?
  - PLU decomposition
  - How it helps us compute determinants
  - Runtime analysis
- 3 Other algorithms for computing determinants
  - Leibniz formula
  - Laplace expansion
  - Gaussian elimination
  - Bareiss algorithm
  - Bird's algorithm
- 4 Epilogue

# Invertibility of matrices

## Theorem

An  $n \times n$  square matrix  $A$  is invertible if and only if

$$\det(A) \neq 0$$

$i++i$

# Eigenvalues and Eigenvectors

## Definition

The Eigenvalues of a matrix  $A$  are the roots of the characteristic polynomial as defined

$$\chi_A = \det(A - \lambda I) = 0$$



i++i

# What is LU decomposition?

## Definition

An LU decomposition of an invertible matrix  $A$  is a factorization

$$A = LU$$

where  $L$  and  $U$  are lower and upper triangular matrices, respectively.

# Is there always an LU decomposition?

No.

An LU decomposition of  $A$  exists if and only if each of its *leading principle minors* (contiguous square submatrices in the top-left corner of  $A$ ), are also invertible.

## Example

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

This matrix is invertible but has no LU decomposition.

What can we do?



# PLU decomposition

## *Partial pivoting.*

We can pivot the matrix into the correct form by multiplication with an orthogonal, permutation matrix  $P$  (representing a permutation  $\sigma_P$ ) which gives us the PLU decomposition:

$$\sigma_P(A) = PA = LU$$

This technique works on *any* invertible matrix.

# How it helps us compute determinants

Now that we have  $PA = LU$ , it follows that

$$\begin{aligned} A &= P^{-1}LU \\ &= P^T LU \end{aligned}$$

since  $P^{-1} = P^T$  by the definition of orthogonal matrices.

## How it helps us compute determinants (cont.)

Now that we have  $A = P^T L U$ , it follows that

$$\begin{aligned}\det(A) &= \det(P^T L U) \\ &= \det(P^T) \cdot \det(L) \cdot \det(U) && \text{(Thm. 4.4.1)} \\ &= \det(P) \cdot \det(L) \cdot \det(U) && \text{(Lem. 4.4.4)}\end{aligned}$$

Given that

- the determinant of a triangular matrix is the product of its diagonal elements
- the determinant of a permutation matrix ( $P$ ) is the parity of the permutation it represents ( $\sigma_P$ )

it follows that

$$\det(A) = \operatorname{sgn}(\sigma_P) \cdot \left( \prod_{i=1}^n l_{i,i} \right) \left( \prod_{i=1}^n u_{i,i} \right)$$

# Runtime analysis

How quick is it?

- The PLU decomposition can be computed in  $\mathcal{O}(n^3)$  time.
- The determinants of the triangular matrices computed in  $\mathcal{O}(n)$  time.
- The parity of the permutation matrix in  $\mathcal{O}(n^2)$  time.

Therefore the total runtime for computing the determinant using the method is

$$\mathcal{O}(n^3) + \mathcal{O}(n^2) + \mathcal{O}(n) = \mathcal{O}(n^3)$$

What are some other methods to compute determinants?

# Leibniz formula

## Definition

The Leibniz formula defines the determinant of  $A \in \mathbb{M}(n)$  as

$$\det(A) = \sum_{\sigma \in \mathfrak{S}_n} \left( \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^n a_{i, \sigma(i)} \right)$$

where  $\mathfrak{S}_n$  is the set of permutations length  $n$ .

Computing the determinant using this method is slow with runtime  $\mathcal{O}((n+1)!)$ .

# Laplace expansion

The Laplace (1st row) expansion for computing determinants is usually the first method taught for computing determinants of  $3 \times 3$  matrices and larger.

## Theorem

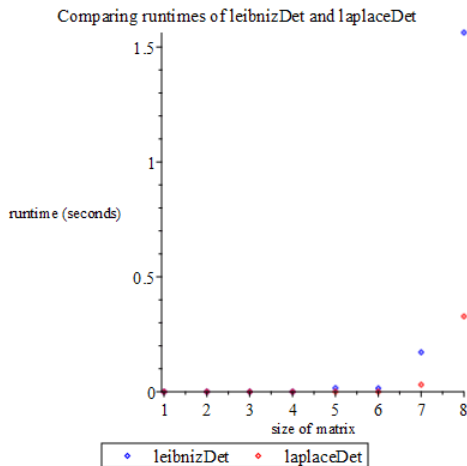
The formula for the (1st row) Laplace expansion of  $A \in \mathbb{M}(n)$  is given as:

$$\det(A) = \sum_{j=1}^n a_{1,j} C_{1,j}$$

where  $C_{i,j}$  is the  $(i,j)$  cofactor of  $A$ .

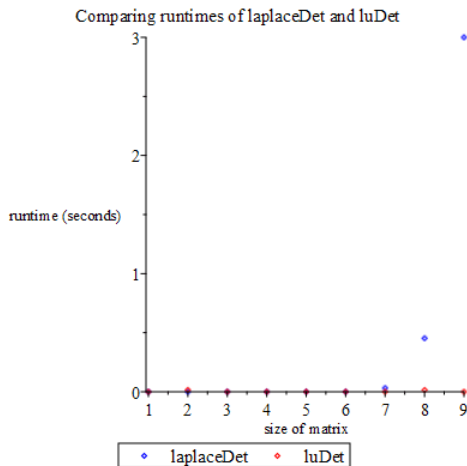
Its runtime complexity of  $\mathcal{O}(n!)$  is poor.

# Laplace expansion vs Leibniz formula



Runtimes are similar — both run in exponential time.

# Laplace expansion vs LU decomposition



The difference between the exponential and polynomial-time function is clear.

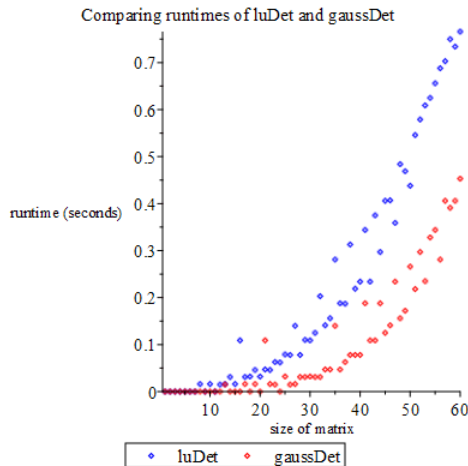


# Gaussian elimination

- The determinant of a triangular matrix can be computed by taking the product of its diagonal entries (which is a quick  $\mathcal{O}(n)$  operation).
- Any invertible square matrix can be transformed into echelon form by performing Gaussian elimination, which takes  $\mathcal{O}(n^3)$  time.

So how does it compare to LU decomposition?

# Gaussian elimination vs LU decomposition



The difference in runtimes is small (a constant factor).

# Gaussian elimination (cont.)

Conventional Gaussian elimination requires division. This has two problems:

- Over  $\mathbb{M}(n; \mathbb{R})$  solutions maybe inexact, so precision is lost.
- Division is not a ring operation, so would not necessarily work on matrices over a ring.

This is can be addressed by using...

# Bareiss algorithm

- Addresses the issue of precision-loss by performing *integer-preserving* Gaussian elimination on integer matrices.
- The runtime complexity is  $\mathcal{O}(n^3)$  which is the same as conventional Gaussian Elimination, whilst preserving exactness.

# Bird's algorithm

Define  $\mu : \mathbb{M}(n) \rightarrow \mathbb{M}(n)$ :

$$\mu(X) = \begin{pmatrix} \mu_{2,2} - x_{2,2} & x_{1,2} & \cdots & x_{1,n-1} & x_{1,n} \\ 0 & \mu_{3,3} - x_{3,3} & \cdots & x_{2,n-1} & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mu_{n,n} - x_{n,n} & x_{n-1,n} \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

and  $F_A : \mathbb{M}(n) \rightarrow \mathbb{M}(n)$ , with  $A \in \mathbb{M}(n)$

$$F_A(X) = \mu(X) \cdot A$$

$$F_A^2(X) = \mu(F_A(X)) \cdot A$$

$$\vdots$$

$$F_A^n(X) = \mu(F_A^{n-1}(X)) \cdot A$$

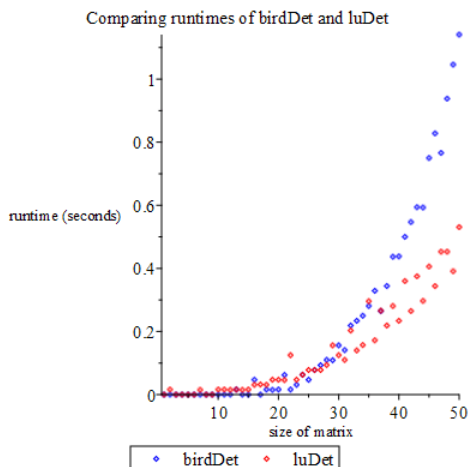
# Bird's algorithm (cont.)

## Bird's Theorem

$$F_A^{n-1}(A) = \begin{pmatrix} d & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ with } d = \begin{cases} \det(A) & \text{odd } n \\ -\det(A) & \text{even } n \end{cases}$$

- Enables the *division-free* computation of determinants in  $\mathcal{O}(n \cdot M(n))$  where  $M(n)$  is the runtime complexity of the matrix multiplication algorithm used.
- If the conventional  $\mathcal{O}(n^3)$  matrix multiplication algorithm is used, then Bird's algorithm will run in  $\mathcal{O}(n^4)$  time.
- But this can be reduced to  $\mathcal{O}(n^{3.8})$  by using the *Strassen algorithm* for matrix multiplication.

# Bird's algorithm vs LU decomposition



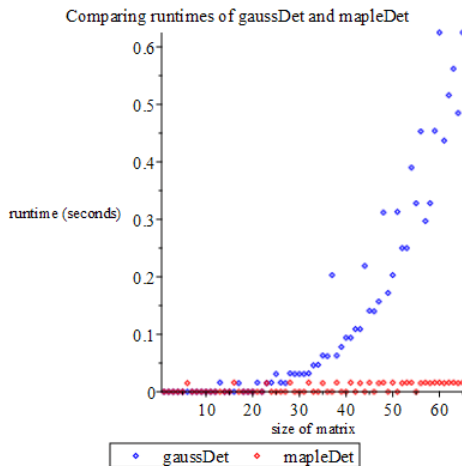
Bird's runtimes increase noticeably more rapidly than LU decomposition, but it's still polynomial.

# Summary of determinant algorithms

<i>Algorithm</i>	<i>Runtime</i>	<i>Exact?</i>
Leibniz formula	$\mathcal{O}((n+1)!)$	Yes
Laplace expansion	$\mathcal{O}(n!)$	Yes
LU decomposition	$\mathcal{O}(n^3)$	No
Gaussian elimination	$\mathcal{O}(n^3)$	No
Bareiss algorithm	$\mathcal{O}(n^3)$	Yes
Bird's algorithm	$\mathcal{O}(n^{3.8})$	Yes



# How fast is Maple's built-in determinant function?



Very. Maple's optimisation means a fair comparison cannot be made.

# Thanks!