Linear algebra over a field

George Labahn and Arne Storjohann

Cheriton School of Computer Science
University of Waterloo

Recent Trends in Computer Algebra, Institut Henri Poincaré,

September 2023

Preliminaries: Setting and cost model

Preliminaries: Setting and cost model

- Let K be an abstract field; no assumptions on K

Preliminaries: Setting and cost model

- Let K be an abstract field; no assumptions on K
- Our inputs are matrices $\mathsf{K}^{n \times m}$ for arbitrary $n, m \geq 0$

Preliminaries: Setting and cost model

- Let K be an abstract field; no assumptions on K
- Our inputs are matrices $\mathsf{K}^{n \times m}$ for arbitrary $n, m \geq 0$
  - When n or m is zero, the matrix is empty
  - When $m = 1$ and $n \geq 1$ this resolves to a column vector
  - When $n = 1$ and $m \geq 1$ this resolves to a row vector

Preliminaries: Setting and cost model

- Let K be an abstract field; no assumptions on K
- Our inputs are matrices $\mathsf{K}^{n \times m}$ for arbitrary $n, m \geq 0$
  - When n or m is zero, the matrix is empty
  - When $m = 1$ and $n \geq 1$ this resolves to a column vector
  - When $n = 1$ and $m \geq 1$ this resolves to a row vector
- Algorithms analysed by counting arithmetic operations from K

$$\{+, -, \times, \text{“divide by nonzero”}\}$$

Preliminaries: Setting and cost model

- Let K be an abstract field; no assumptions on K
- Our inputs are matrices $\mathsf{K}^{n \times m}$ for arbitrary $n, m \geq 0$
  - When n or m is zero, the matrix is empty
  - When $m = 1$ and $n \geq 1$ this resolves to a column vector
  - When $n = 1$ and $m \geq 1$ this resolves to a row vector
- Algorithms analysed by counting arithmetic operations from K

$$\{+, -, \times, \text{“divide by nonzero”}\}$$

- Realistically, we should include the additional operation

$$\text{“is } c \in \mathsf{K} \text{ equal to 0?”}$$

Preliminaries: Setting and cost model

- Let K be an abstract field; no assumptions on K
- Our inputs are matrices $\mathsf{K}^{n \times m}$ for arbitrary $n, m \geq 0$
  - When n or m is zero, the matrix is empty
  - When $m = 1$ and $n \geq 1$ this resolves to a column vector
  - When $n = 1$ and $m \geq 1$ this resolves to a row vector
- Algorithms analysed by counting arithmetic operations from K

$$\{+, -, \times, \text{“divide by nonzero”}\}$$

- Realistically, we should include the additional operation

$$\text{“is } c \in \mathsf{K} \text{ equal to } 0\text{?”}$$

- We will focus on worst case complexity and use big-O notation
- Analysis will reflect reality when K is a finite field

Some initial classical problems

Some initial classical problems

Let $A \in K^{n \times m}$ be given. Classical problems are to compute

Some initial classical problems

Let $A \in \mathsf{K}^{n \times m}$ be given. Classical problems are to compute

- **Rank**: the rank r of A

Some initial classical problems

Let $A \in \mathsf{K}^{n \times m}$ be given. Classical problems are to compute

- **Rank**: the rank r of A
- **Column Rank Profile**: the lexicographically minimal list $[j_1, j_2, \ldots, j_r]$ of linearly independent columns of A

Some initial classical problems

Let $A \in \mathsf{K}^{n \times m}$ be given. Classical problems are to compute

- **Rank**: the rank r of A
- **Column Rank Profile**: the lexicographically minimal list $[j_1, j_2, \ldots, j_r]$ of linearly independent columns of A
- **Determinant**: the determinant $\det A$ if $n = m$

Some initial classical problems

Let $A \in \mathsf{K}^{n \times m}$ be given. Classical problems are to compute

- **Rank**: the rank r of A
- **Column Rank Profile**: the lexicographically minimal list $[j_1, j_2, \ldots, j_r]$ of linearly independent columns of A
- **Determinant**: the determinant $\det A$ if $n = m$
- **Inverse**: the inverse of a A if square and nonsingular

Some initial classical problems

Let $A \in \mathsf{K}^{n \times m}$ be given. Classical problems are to compute

- **Rank**: the rank r of A
- **Column Rank Profile**: the lexicographically minimal list $[j_1, j_2, \ldots, j_r]$ of linearly independent columns of A
- **Determinant**: the determinant $\det A$ if $n = m$
- **Inverse**: the inverse of a A if square and nonsingular
- **Linear System Solving**: given also a $b \in \mathsf{K}^{m \times 1}$, find an $x \in \mathsf{K}^{m \times 1}$ such that $Ax = b$ (or "inconsistent"; no x exists)

Some initial classical problems

Let $A \in \mathsf{K}^{n \times m}$ be given. Classical problems are to compute

- **Rank**: the rank r of A
- **Column Rank Profile**: the lexicographically minimal list $[j_1, j_2, \ldots, j_r]$ of linearly independent columns of A
- **Determinant**: the determinant $\det A$ if $n = m$
- **Inverse**: the inverse of a A if square and nonsingular
- **Linear System Solving**: given also a $b \in \mathsf{K}^{m \times 1}$, find an $x \in \mathsf{K}^{m \times 1}$ such that $Ax = b$ (or "inconsistent"; no x exists)
- **Right Nullspace**: a full column rank matrix $N \in \mathsf{K}^{m \times (m-r)}$ such that $AN = 0_{n \times (m-r)}$

Effective solution to all problems: Gaussian elimination

Effective solution to all problems: Gaussian elimination

- Transform A to a more useful shape while ...
  ... keeping the vector space $\{vA \mid v \in \mathsf{K}^{1\times n}\}$ invariant

Effective solution to all problems: Gaussian elimination

- Transform A to a more useful shape while ...
  ... keeping the vector space $\{vA \mid v \in \mathrm{K}^{1\times n}\}$ invariant
- Elementary (invertible!) row operations
  1. Interchange two rows
  2. Multiply a row by a nonzero element $c \in \mathrm{K}$
  3. Add the result of (2) to a different row.

Effective solution to all problems: Gaussian elimination

- Transform A to a more useful shape while ...
  ... keeping the vector space $\{vA \mid v \in \mathsf{K}^{1\times n}\}$ invariant
- Elementary (invertible!) row operations
  1. Interchange two rows
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.
- Can transform input matrix to row echelon form
  Example is for an input matrix $A \in \mathsf{K}^{6\times 9}$:

$$A = \begin{bmatrix} * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet & * & * & * & * & * & * & * & * \\ & & & \bullet & * & * & * & * & * \\ & & & & \bullet & * & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} = R.$$

Convention: $*$ possibly nonzero; $\bullet$ necessarily nonzero; blank is zero

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix}$$

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{array}\right]$$

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{array}\right] \xrightarrow{(1)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ & & & \bullet & * \\ \hline \end{array}\right] = R$$

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \begin{bmatrix} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{bmatrix} \xrightarrow{(1)} \begin{bmatrix} & \bullet & * & * & * \\ & & & \bullet & * \\ \hline & & & & \end{bmatrix} = R$$

- We conclude that A has rank 2 and column rank profile $[2, 4]$

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \begin{bmatrix} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{bmatrix} \xrightarrow{(1)} \begin{bmatrix} & \bullet & * & * & * \\ & & & \bullet & * \\ \hline & & & & \end{bmatrix} = R$$

- We conclude that A has rank 2 and column rank profile $[2, 4]$
- First two rows of R give a basis for the rowspace of A

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{array}\right] \xrightarrow{(1)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ & & & \bullet & * \\ \hline \end{array}\right] = R$$

- We conclude that A has rank 2 and column rank profile $[2, 4]$
- First two rows of R give a basis for the rowspace of A
- Cost to compute an R for an $A \in \mathsf{K}^{n\times m}$: $O(nmr)$

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{array}\right] \xrightarrow{(1)} \left[\begin{array}{ccccc} & \bullet & * & * & * \\ & & & \bullet & * \\ \hline \end{array}\right] = R$$

- We conclude that A has rank 2 and column rank profile $[2, 4]$
- First two rows of R give a basis for the rowspace of A
- Cost to compute an R for an $A \in \mathsf{K}^{n\times m}$: $O(nmr)$
  - Leading constant in $O(nmr)$ is ≤ 1

Example: Computing an echelon form

- From previous slide: Elementary row operations
  1. Interchange two rows (here, used for "pivoting")
  2. Multiply a row by a nonzero element $c \in \mathsf{K}$
  3. Add the result of (2) to a different row.

Example is for an $A \in \mathsf{K}^{3\times 5}$

$$A = \begin{bmatrix} \circ & \bullet & * & * & * \\ \circ & * & * & * & * \\ \circ & * & * & * & * \end{bmatrix} \xrightarrow{(3)} \begin{bmatrix} & \bullet & * & * & * \\ \hline & & \circ & \circ & \circ \\ & & \circ & \bullet & * \end{bmatrix} \xrightarrow{(1)} \begin{bmatrix} & \bullet & * & * & * \\ & & & \bullet & * \\ \hline & & & & \end{bmatrix} = R$$

- We conclude that A has rank 2 and column rank profile $[2, 4]$
- First two rows of R give a basis for the rowspace of A
- Cost to compute an R for an $A \in \mathsf{K}^{n\times m}$: $O(nmr)$
  - Leading constant in $O(nmr)$ is ≤ 1
  - Leading constant if $n = m = r$ is $1/3$

Computing the determinant of an $A \in \mathsf{K}^{n \times n}$

- Compute the row echelon form
- Keep track of the number p of row interchanges
- If rank is $< n$ then determinant is zero...
  ... otherwise

Computing the determinant of an $A \in \mathsf{K}^{n \times n}$

- Compute the row echelon form
- Keep track of the number p of row interchanges
- If rank is $< n$ then determinant is zero...
  ... otherwise

Example is for $A \in \mathsf{K}^{3 \times 3}$

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet_1 & * & * \\ & \bullet_2 & * \\ & & \bullet_3 \end{bmatrix}$$

Computing the determinant of an $A \in \mathsf{K}^{n \times n}$

- Compute the row echelon form
- Keep track of the number p of row interchanges
- If rank is $< n$ then determinant is zero...
  ... otherwise

Example is for $A \in \mathsf{K}^{3 \times 3}$

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet_1 & * & * \\ & \bullet_2 & * \\ & & \bullet_3 \end{bmatrix}$$

- $\det A = (-1)^p \bullet_1 \bullet_2 \bullet_3$

Computing the determinant of an $A \in \mathsf{K}^{n \times n}$

- Compute the row echelon form
- Keep track of the number p of row interchanges
- If rank is $< n$ then determinant is zero...
  ... otherwise

Example is for $A \in \mathsf{K}^{3 \times 3}$

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet_1 & * & * \\ & \bullet_2 & * \\ & & \bullet_3 \end{bmatrix}$$

- $\det A = (-1)^p \bullet_1 \bullet_2 \bullet_3$
- Cost is $\frac{1}{3}n^3 + O(n^2)$

Computing the determinant of an $A \in \mathsf{K}^{n \times n}$

- Compute the row echelon form
- Keep track of the number p of row interchanges
- If rank is $< n$ then determinant is zero...
  ... otherwise

Example is for $A \in \mathsf{K}^{3 \times 3}$

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet_1 & * & * \\ & \bullet_2 & * \\ & & \bullet_3 \end{bmatrix}$$

- $\det A = (-1)^p \bullet_1 \bullet_2 \bullet_3$
- Cost is $\frac{1}{3}n^3 + O(n^2)$
- So far: effective solutions to rank profile and determinant

Computing the determinant of an $A \in \mathsf{K}^{n \times n}$

- Compute the row echelon form
- Keep track of the number p of row interchanges
- If rank is $< n$ then determinant is zero...
  ... otherwise

Example is for $A \in \mathsf{K}^{3 \times 3}$

$$A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet_1 & * & * \\ & \bullet_2 & * \\ & & \bullet_3 \end{bmatrix}$$

- $\det A = (-1)^p \bullet_1 \bullet_2 \bullet_3$
- Cost is $\frac{1}{3}n^3 + O(n^2)$
- So far: effective solutions to rank profile and determinant
- What about linear solving?

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathrm{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathrm{K}^{1 \times n}\}$

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathrm{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathrm{K}^{1 \times n}\}$
- Method

Example is for an $A \in \mathrm{K}^{6 \times 9}$ (same as before)

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathrm{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathrm{K}^{1 \times n}\}$
- Method
  (a) Transform to an echelon form

Example is for an $A \in \mathrm{K}^{6 \times 9}$ (same as before)

$$A \xrightarrow{(a)} \begin{bmatrix} \bullet & * & * & * & * & * & * & * & * \\ & & & \bullet & * & * & * & * & * \\ & & & & & \bullet & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}$$

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathsf{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathsf{K}^{1 \times n}\}$
- Method
  (a) Transform to an echelon form
  (b) Use ops of type (2) to make pivots equal to 1

Example is for an $A \in \mathsf{K}^{6 \times 9}$ (same as before)

$$A \xrightarrow{(a)} \begin{bmatrix} \bullet & * & * & * & * & * & * & * & * \\ & & & \bullet & * & * & * & * & * \\ & & & & & \bullet & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} \xrightarrow{(b)} \begin{bmatrix} 1 & * & * & * & * & * & * & * & * \\ & & & 1 & * & * & * & * & * \\ & & & & & 1 & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}$$

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathsf{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathsf{K}^{1 \times n}\}$
- Method
  (a) Transform to an echelon form
  (b) Use ops of type (2) to make pivots equal to 1
  (c) Use ops of type (3) to zero out entries above pivots

Example is for an $A \in \mathsf{K}^{6 \times 9}$ (same as before)

$$A \xrightarrow{(a)} \begin{bmatrix} \bullet & * & * & * & * & * & * & * & * \\ & & & \bullet & * & * & * & * & * \\ & & & & & & \bullet & * & * \\ & & & & & & & & \end{bmatrix} \xrightarrow{(b)} \begin{bmatrix} 1 & * & * & * & * & * & * & * & * \\ & & & 1 & * & * & * & * & * \\ & & & & & & 1 & * & * \\ & & & & & & & & \end{bmatrix} \xrightarrow{(c)} \begin{bmatrix} 1 & * & * & & & * & * & * & * \\ & & & 1 & & * & * & * & * \\ & & & & & & 1 & * & * \\ & & & & & & & & \end{bmatrix}$$

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathsf{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathsf{K}^{1 \times n}\}$
- Method
  - (a) Transform to an echelon form
  - (b) Use ops of type (2) to make pivots equal to 1
  - (c) Use ops of type (3) to zero out entries above pivots

Example is for an $A \in \mathsf{K}^{6 \times 9}$ (same as before)

$$A \xrightarrow{(a)} \begin{bmatrix} \bullet & * & * & * & * & * & * & * & * \\ & & & \bullet & * & * & * & * & * \\ & & & & & & \bullet & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} \xrightarrow{(b)} \begin{bmatrix} 1 & * & * & * & * & * & * & * & * \\ & & & 1 & * & * & * & * & * \\ & & & & & & 1 & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} \xrightarrow{(c)} \begin{bmatrix} 1 & * & * & & & * & * & * & * \\ & & & 1 & & * & * & * & * \\ & & & & & & 1 & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}$$

- Now suppose $A \in \mathsf{K}^{n \times n}$ is nonsingular

Reduced row echelon form: Gauss-Jordan elimination

- We will write RREF for short
- The RREF of $A \in \mathsf{K}^{n \times m}$ gives canonical basis for $\{vA \mid v \in \mathsf{K}^{1 \times n}\}$
- Method
  (a) Transform to an echelon form
  (b) Use ops of type (2) to make pivots equal to 1
  (c) Use ops of type (3) to zero out entries above pivots

Example is for an $A \in \mathsf{K}^{6 \times 9}$ (same as before)

$$A \xrightarrow{(a)} \begin{bmatrix} \bullet & * & * & * & * & * & * & * & * \\ & & & \bullet & * & * & * & * & * \\ & & & & & & \bullet & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} \xrightarrow{(b)} \begin{bmatrix} 1 & * & * & * & * & * & * & * & * \\ & & & 1 & * & * & * & * & * \\ & & & & & & 1 & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix} \xrightarrow{(c)} \begin{bmatrix} 1 & * & * & & & * & * & * & * \\ & & & 1 & & * & * & * & * \\ & & & & & & 1 & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}$$

- Now suppose $A \in \mathsf{K}^{n \times n}$ is nonsingular
- The RREF of $\left[\begin{array}{c|c} A & I_n \end{array}\right] \in \mathsf{K}^{n \times 2n}$ is $\left[\begin{array}{c|c} I_n & A^{-1} \end{array}\right]$

Solution of linear system via reduced row echelon form

- Let $b \in \mathsf{K}^{n\times 1}$ be given in addition to $A \in \mathsf{K}^{n\times m}$.

Solution of linear system via reduced row echelon form

- Let $b \in \mathrm{K}^{n \times 1}$ be given in addition to $A \in \mathrm{K}^{n \times m}$.
- Goal is to find $x \in \mathrm{K}^{m \times 1}$ such that $Ax = b$...
  ... or determine that no such x exists (inconsistent)

Solution of linear system via reduced row echelon form

- Let $b \in \mathrm{K}^{n \times 1}$ be given in addition to $A \in \mathrm{K}^{n \times m}$.
- Goal is to find $x \in \mathrm{K}^{m \times 1}$ such that $Ax = b$...
  ... or determine that no such x exists (inconsistent)
- One method: Transform augmented system $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF

Solution of linear system via reduced row echelon form

- Let $b \in \mathrm{K}^{n\times 1}$ be given in addition to $A \in \mathrm{K}^{n\times m}$.
- Goal is to find $x \in \mathrm{K}^{m\times 1}$ such that $Ax = b$...
  ... or determine that no such x exists (inconsistent)
- One method: Transform augmented system $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF

Examples are for an $A \in \mathrm{K}^{3\times 4}$

Solution of linear system via reduced row echelon form

- Let $b \in \mathsf{K}^{n\times 1}$ be given in addition to $A \in \mathsf{K}^{n\times m}$.
- Goal is to find $x \in \mathsf{K}^{m\times 1}$ such that $Ax = b$...
  ... or determine that no such x exists (inconsistent)
- One method: Transform augmented system $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF

Examples are for an $A \in \mathsf{K}^{3\times 4}$

1. $[\,A\,|\,b\,] \longrightarrow \left[\begin{array}{cccc|c} 1 & * & & * & * \\ & & 1 & * & * \\ & & & & \bullet \end{array}\right] \Longrightarrow$ "inconsistent"

2. $[\,A\,|\,b\,] \longrightarrow \left[\begin{array}{cccc|c} 1 & * & & * & c_1 \\ & & 1 & * & c_2 \\ & & & & \end{array}\right] \Longrightarrow x = \begin{bmatrix} -c_1 \\ \\ -c_2 \\ \\ \end{bmatrix}$

General solution of linear system of equations

General solution of linear system of equations

- Recall previous example (consistent): $A \in \mathsf{K}^{3\times 4}$ with rank 2

$$[\, A \,|\, b \,] \longrightarrow \left[\begin{array}{cccc|c} 1 & *_{12} & & *_{14} & c_1 \\ & & 1 & *_{24} & c_2 \\ & & & & \end{array}\right] \Longrightarrow x_p = \begin{bmatrix} -c_1 \\ \\ -c_2 \\ \\ \end{bmatrix}$$

General solution of linear system of equations

- Recall previous example (consistent): $A \in \mathsf{K}^{3\times 4}$ with rank 2

$$[\, A \,|\, b \,] \longrightarrow \left[\begin{array}{cccc|c} 1 & *_{12} & & *_{14} & c_1 \\ & & 1 & *_{24} & c_2 \\ & & & & \end{array}\right] \Longrightarrow x_p = \begin{bmatrix} -c_1 \\ \\ -c_2 \\ \\ \end{bmatrix}$$

- Right nullspace $N \in \mathsf{K}^{4\times 2}$ of A is given for free by RREF of A

$$N = \begin{bmatrix} *_{12} & *_{14} \\ -1 & \\ & *_{24} \\ & -1 \end{bmatrix}$$

General solution of linear system of equations

- Recall previous example (consistent): $A \in \mathsf{K}^{3\times 4}$ with rank 2

$$[\,A \,|\, b\,] \longrightarrow \left[\begin{array}{cccc|c} 1 & *_{12} & & *_{14} & c_1 \\ & & 1 & *_{24} & c_2 \\ & & & & \end{array}\right] \Longrightarrow x_p = \begin{bmatrix} -c_1 \\ \\ -c_2 \\ \\ \end{bmatrix}$$

- Right nullspace $N \in \mathsf{K}^{4\times 2}$ of A is given for free by RREF of A

$$N = \begin{bmatrix} *_{12} & *_{14} \\ -1 & \\ & *_{24} \\ & -1 \end{bmatrix}$$

- General solution to $Ax = b$ is given by

$$x = \left\{ x_p + Nw \mid w \in \mathsf{K}^{2\times 1} \right\}$$

How can we be faster?

How can we be faster?

- Gaussian elimination: $O(nmr)$ solutions to **Rank**, **Rank Profile**, **Determinant**, **Inverse**, **Linear System Solving**, **Nullspace**

How can we be faster?

- Gaussian elimination: $O(nmr)$ solutions to **Rank**, **Rank Profile**, **Determinant**, **Inverse**, **Linear System Solving**, **Nullspace**
- The leading constants in the O notation are small in all cases, eg, determinant is computed in $\frac{1}{3}n^3 + O(n^2)$

How can we be faster?

- Gaussian elimination: $O(nmr)$ solutions to **Rank**, **Rank Profile**, **Determinant**, **Inverse**, **Linear System Solving**, **Nullspace**
- The leading constants in the O notation are small in all cases, eg, determinant is computed in $\frac{1}{3}n^3 + O(n^2)$
- But the approach is very iterative
- An effective way to improve is to incorporate matrix multiplication
- Our primary building block is the multiplication of two $n \times n$ matrices: $AB = C$

How can we be faster?

- Gaussian elimination: $O(nmr)$ solutions to **Rank**, **Rank Profile**, **Determinant**, **Inverse**, **Linear System Solving**, **Nullspace**
- The leading constants in the O notation are small in all cases, eg, determinant is computed in $\frac{1}{3}n^3 + O(n^2)$
- But the approach is very iterative
- An effective way to improve is to incorporate matrix multiplication
- Our primary building block is the multiplication of two $n \times n$ matrices: $AB = C$
- Incorporating matrix multiplication gives an improvement both in practice (implementation) and theory (better asymptotic complexity)
- Standard matrix multiplication uses n^2 dot products costing exactly $2n^3 - n^2 \in O(n^3)$ operations in total

Computing iteratively can be slow on a real machine

Computing iteratively can be slow on a real machine

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations

Computing iteratively can be slow on a real machine

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Let $A, B \in \mathbb{Z}/(p)^{n \times n}$ have entries chosen uniformly and randomly, with $n = 10000$ and $p = 100003$

Computing iteratively can be slow on a real machine

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Let $A, B \in \mathbb{Z}/(p)^{n \times n}$ have entries chosen uniformly and randomly, with $n = 10000$ and $p = 100003$
- Let $C \in \mathbb{Z}/(p)^{n \times n}$ be initialized to be the zero matrix

Computing iteratively can be slow on a real machine

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Let $A, B \in \mathbb{Z}/(p)^{n \times n}$ have entries chosen uniformly and randomly, with $n = 10000$ and $p = 100003$
- Let $C \in \mathbb{Z}/(p)^{n \times n}$ be initialized to be the zero matrix
- The following code does n^2 dot products to set $C = AB$

```
for i to n do
   for j to n do
      Multiply(p,A,i,B,1..n,j,C,i,j)
   od
od:
```

Computing iteratively can be slow on a real machine

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Let $A, B \in \mathbb{Z}/(p)^{n \times n}$ have entries chosen uniformly and randomly, with $n = 10000$ and $p = 100003$
- Let $C \in \mathbb{Z}/(p)^{n \times n}$ be initialized to be the zero matrix
- The following code does n^2 dot products to set $C = AB$

```
for i to n do
   for j to n do
      Multiply(p,A,i,B,1..n,j,C,i,j)
   od
od:
```

- Number of operations from $\mathbb{Z}/(p)$ is exactly $2n^3 - n^2$
- Wall clock time on my Mac Mini is 9198 seconds

Incorporating matrix multiplication can be fast

Incorporating matrix multiplication can be fast

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Suppose $A, B \in \mathbb{Z}/(p)^{n \times n}$ with $n = 10000$ and $p = 100003$
- Suppose $C \in \mathbb{Z}/(p)^{n \times n}$ is initialized to be the zero matrix

Incorporating matrix multiplication can be fast

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Suppose $A, B \in \mathbb{Z}/(p)^{n \times n}$ with $n = 10000$ and $p = 100003$
- Suppose $C \in \mathbb{Z}/(p)^{n \times n}$ is initialized to be the zero matrix
- The following code sets $C = AB$

  ```
  Multiply(p,A,B,C):
  ```

- Number of operations from $\mathbb{Z}/(p)$ is again $\Theta(n^3)$

Incorporating matrix multiplication can be fast

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Suppose $A, B \in \mathbb{Z}/(p)^{n \times n}$ with $n = 10000$ and $p = 100003$
- Suppose $C \in \mathbb{Z}/(p)^{n \times n}$ is initialized to be the zero matrix
- The following code sets $C = AB$

  ```
  Multiply(p,A,B,C):
  ```

- Number of operations from $\mathbb{Z}/(p)$ is again $\Theta(n^3)$
- But now Maple will pass off the computation to a highly optimized numerical library that takes advantage of cache effects

Incorporating matrix multiplication can be fast

- Maple's `LinearAlgebra:-Modular` package gives highly optimized implementations of matrix operations
- Suppose $A, B \in \mathbb{Z}/(p)^{n \times n}$ with $n = 10000$ and $p = 100003$
- Suppose $C \in \mathbb{Z}/(p)^{n \times n}$ is initialized to be the zero matrix
- The following code sets $C = AB$

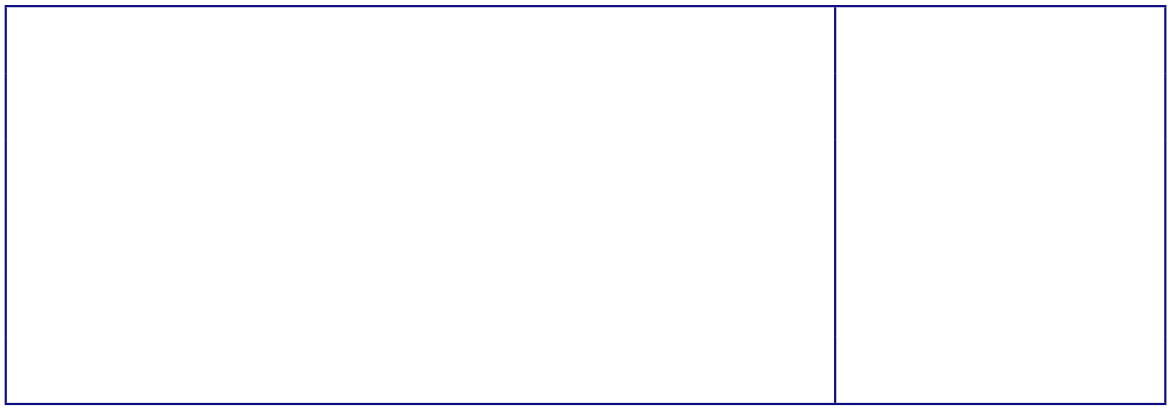
```
Multiply(p,A,B,C):
```

- Number of operations from $\mathbb{Z}/(p)$ is again $\Theta(n^3)$
- But now Maple will pass off the computation to a highly optimized numerical library that takes advantage of cache effects
- Wall clock time on my Mac Mini is 67.6 seconds, a factor of 136 times faster than the iterative method

Matrix multiplication: complexity aspects

Matrix multiplication: complexity aspects

- Henceforth, we will use ω as **feasible exponent of matrix multiplication**: two $n \times n$ matrices over K can be multiplied in time $O(n^\omega)$. We assume $2 < \omega \leq 3$.



Matrix multiplication: complexity aspects

- Henceforth, we will use ω as **feasible exponent of matrix multiplication**: two $n \times n$ matrices over K can be multiplied in time $O(n^\omega)$. We assume $2 < \omega \leq 3$.
- Standard matrix has $\omega = 3$

Standard	$\omega = 3$

Matrix multiplication: complexity aspects

- Henceforth, we will use ω as **feasible exponent of matrix multiplication**: two $n \times n$ matrices over K can be multiplied in time $O(n^\omega)$. We assume $2 < \omega \leq 3$.
- Standard matrix has $\omega = 3$
- Many improvements to ω (no survey here, only highlights)

Standard	$\omega = 3$

Matrix multiplication: complexity aspects

- Henceforth, we will use ω as **feasible exponent of matrix multiplication**: two $n \times n$ matrices over K can be multiplied in time $O(n^\omega)$. We assume $2 < \omega \leq 3$.
- Standard matrix has $\omega = 3$
- Many improvements to ω (no survey here, only highlights)

Standard	$\omega = 3$
Strassen (1969)	$\omega = 2.8074$

Matrix multiplication: complexity aspects

- Henceforth, we will use ω as **feasible exponent of matrix multiplication**: two $n \times n$ matrices over K can be multiplied in time $O(n^\omega)$. We assume $2 < \omega \leq 3$.
- Standard matrix has $\omega = 3$
- Many improvements to ω (no survey here, only highlights)

Standard	$\omega = 3$
Strassen (1969)	$\omega = 2.8074$
...	
Coppersmith & Winograd (1986)	$\omega = 2.376$
Williams (2011)	$\omega = 2.373$
...	

Matrix multiplication: complexity aspects

- Henceforth, we will use ω as **feasible exponent of matrix multiplication**: two $n \times n$ matrices over K can be multiplied in time $O(n^\omega)$. We assume $2 < \omega \leq 3$.
- Standard matrix has $\omega = 3$
- Many improvements to ω (no survey here, only highlights)

Standard	$\omega = 3$
Strassen (1969)	$\omega = 2.8074$
...	
Coppersmith & Winograd (1986)	$\omega = 2.376$
Williams (2011)	$\omega = 2.373$
...	

- The algorithm lower in the table are considered to be galactic

The centrality of matrix multiplication

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication
- Eg: We can multiply $A, B \in \mathsf{K}^{n \times n}$ in the same time asymptotically as we can invert a $3n \times 3n$ unit upper triangular matrix

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication
- Eg: We can multiply $A, B \in \mathsf{K}^{n \times n}$ in the same time asymptotically as we can invert a $3n \times 3n$ unit upper triangular matrix
- Construct the following, very particular, $3n \times 3n$ upper triangular matrix U

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication
- Eg: We can multiply $A, B \in \mathsf{K}^{n \times n}$ in the same time asymptotically as we can invert a $3n \times 3n$ unit upper triangular matrix
- Construct the following, very particular, $3n \times 3n$ upper triangular matrix U

$$U = \left[\begin{array}{c|c|c} I_n & A & \\ \hline & I_n & B \\ \hline & & I_n \end{array}\right]$$

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication
- Eg: We can multiply $A, B \in \mathsf{K}^{n \times n}$ in the same time asymptotically as we can invert a $3n \times 3n$ unit upper triangular matrix
- Construct the following, very particular, $3n \times 3n$ upper triangular matrix U and invert it

$$U = \left[\begin{array}{c|c|c} I_n & A & \\ \hline & I_n & B \\ \hline & & I_n \end{array}\right] \qquad U^{-1} = \left[\begin{array}{c|c|c} I_n & -A & AB \\ \hline & I_n & -B \\ \hline & & I_n \end{array}\right]$$

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication
- Eg: We can multiply $A, B \in \mathsf{K}^{n \times n}$ in the same time asymptotically as we can invert a $3n \times 3n$ unit upper triangular matrix
- Construct the following, very particular, $3n \times 3n$ upper triangular matrix U and invert it

$$U = \left[\begin{array}{c|c|c} I_n & A & \\ \hline & I_n & B \\ \hline & & I_n \end{array}\right] \qquad U^{-1} = \left[\begin{array}{c|c|c} I_n & -A & AB \\ \hline & I_n & -B \\ \hline & & I_n \end{array}\right]$$

- What does this show?

The centrality of matrix multiplication

- Many problems are at least as difficult as matrix multiplication
- Eg: We can multiply $A, B \in \mathsf{K}^{n \times n}$ in the same time asymptotically as we can invert a $3n \times 3n$ unit upper triangular matrix
- Construct the following, very particular, $3n \times 3n$ upper triangular matrix U and invert it

$$U = \left[\begin{array}{c|c|c} I_n & A & \\ \hline & I_n & B \\ \hline & & I_n \end{array}\right] \qquad U^{-1} = \left[\begin{array}{c|c|c} I_n & -A & AB \\ \hline & I_n & -B \\ \hline & & I_n \end{array}\right]$$

- What does this show?
- It shows that triangular matrix inversion is at least as difficult as matrix multiplication

The centrality of matrix multiplication

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality
- Let $U \in \mathsf{K}^{n \times n}$ be unit upper triangular

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality
- Let $U \in \mathsf{K}^{n \times n}$ be unit upper triangular
  - Use a divide & conquer approach

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality
- Let $U \in \mathsf{K}^{n \times n}$ be unit upper triangular
  - Use a divide & conquer approach
  - Partition U into blocks of size $n/2$

$$U = \left[\begin{array}{c|c} U_1 & V \\ \hline & U_2 \end{array}\right]$$

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality
- Let $U \in \mathsf{K}^{n \times n}$ be unit upper triangular
  - Use a divide & conquer approach
  - Partition U into blocks of size $n/2$

$$U = \left[\begin{array}{c|c} U_1 & V \\ \hline & U_2 \end{array}\right] \qquad U^{-1} = \left[\begin{array}{c|c} U_1^{-1} & -U_1^{-1} V U_2^{-1} \\ \hline & U_2^{-1} \end{array}\right]$$

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality
- Let $U \in \mathsf{K}^{n \times n}$ be unit upper triangular
  - Use a divide & conquer approach
  - Partition U into blocks of size $n/2$

$$U = \left[\begin{array}{c|c} U_1 & V \\ \hline & U_2 \end{array}\right] \qquad U^{-1} = \left[\begin{array}{c|c} U_1^{-1} & -U_1^{-1} V U_2^{-1} \\ \hline & U_2^{-1} \end{array}\right]$$

  - Runtime $T(n)$ for inversion satisfies

$$T(n) = 2T(n/2) + O(n^\omega),$$

  giving $T(n) \in O(n^\omega)$.

The centrality of matrix multiplication

- The previous slide showed that inverting a triangular matrix is at least as difficult as matrix multiplication...
  ... so, if we can give an algorithm for triangular inversion in the same time as multiplication, we can claim optimality
- Let $U \in \mathsf{K}^{n \times n}$ be unit upper triangular
  - Use a divide & conquer approach
  - Partition U into blocks of size $n/2$

$$U = \left[\begin{array}{c|c} U_1 & V \\ \hline & U_2 \end{array}\right] \qquad U^{-1} = \left[\begin{array}{c|c} U_1^{-1} & -U_1^{-1} V U_2^{-1} \\ \hline & U_2^{-1} \end{array}\right]$$

  - Runtime $T(n)$ for inversion satisfies

$$T(n) = 2T(n/2) + O(n^\omega),$$

giving $T(n) \in O(n^\omega)$. Note: this holds even if $\omega = 2$.

Multiplying rectangular matrices

Multiplying rectangular matrices

- Need to use blocking to multiply rectangular matrices

Multiplying rectangular matrices

- Need to use blocking to multiply rectangular matrices
- Fact
  - Let A be $a \times b$
  - Let B be $b \times c$
  - Then computing AB has cost $O(abck^{\omega-3})$ where $k = \min(a, b, c)$

Multiplying rectangular matrices

- Need to use blocking to multiply rectangular matrices
- Fact
  - Let A be $a \times b$
  - Let B be $b \times c$
  - Then computing AB has cost $O(abck^{\omega-3})$ where $k = \min(a, b, c)$
- Example:
  - $a = 4k$, $b = 3k$, $c = k$
  - $A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \in \mathrm{K}^{4k \times 3k}$, $B = \begin{bmatrix} * \\ * \\ * \end{bmatrix} \in \mathrm{K}^{3k \times k}$

Multiplying rectangular matrices

- Need to use blocking to multiply rectangular matrices
- Fact
  - Let A be $a \times b$
  - Let B be $b \times c$
  - Then computing AB has cost $O(abck^{\omega-3})$ where $k = \min(a, b, c)$
- Example:
  - $a = 4k$, $b = 3k$, $c = k$
  - $A = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \in \mathsf{K}^{4k \times 3k}$, $B = \begin{bmatrix} * \\ * \\ * \end{bmatrix} \in \mathsf{K}^{3k \times k}$
  - Computing AB costs 12 multiplications of $k \times k$ matrices (plus some additions which do not dominate)

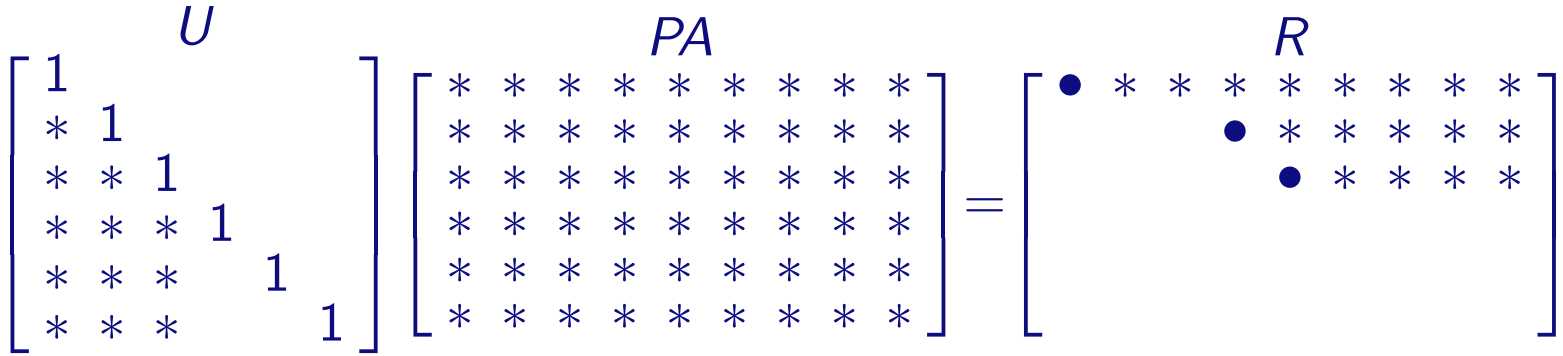
Incorporating matrix multiplication into Gaussian elimination

Incorporating matrix multiplication into Gaussian elimination

- Idea: Use a divide and conquer approach

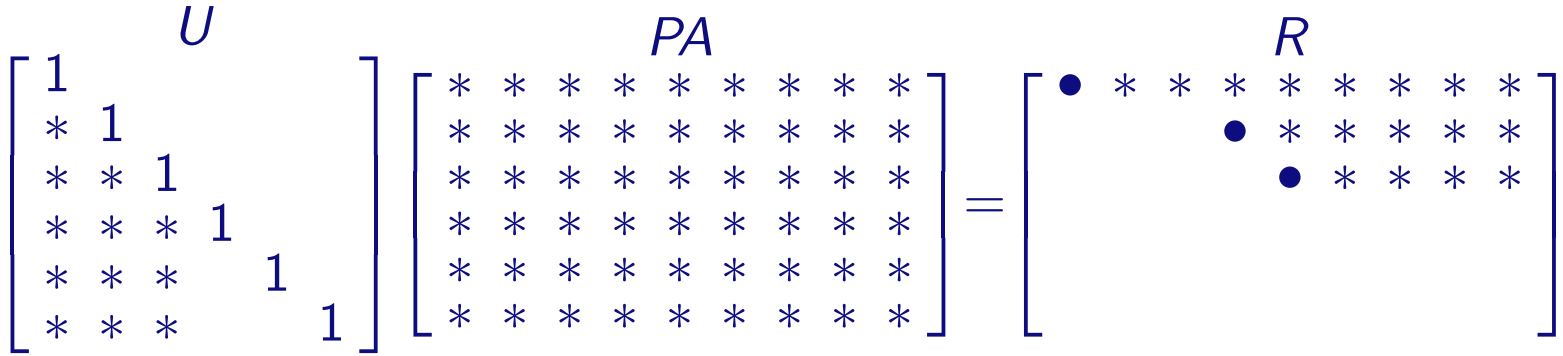
Incorporating matrix multiplication into Gaussian elimination

- Idea: Use a divide and conquer approach
- Need to keep track of pivoting operations in permutation matrix P and row transformation in transform matrix U



Incorporating matrix multiplication into Gaussian elimination

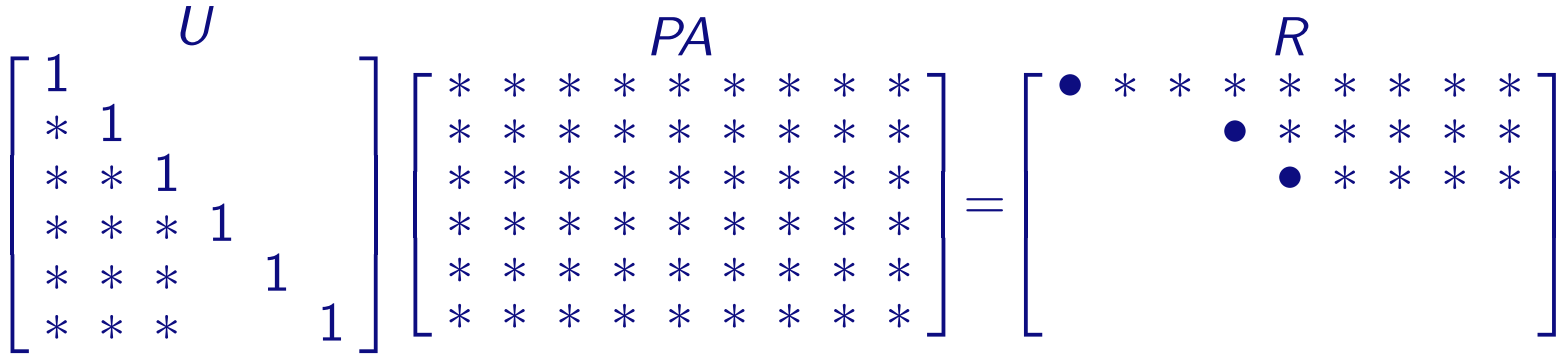
- Idea: Use a divide and conquer approach
- Need to keep track of pivoting operations in permutation matrix P and row transformation in transform matrix U



- Call the pair (U, P) a Gauss-transform for A

Incorporating matrix multiplication into Gaussian elimination

- Idea: Use a divide and conquer approach
- Need to keep track of pivoting operations in permutation matrix P and row transformation in transform matrix U



- Call the pair (U, P) a Gauss-transform for A
- To divide and conquer we split the matrix into half according to columns

Recursive echelon form: first subproblem

Recursive echelon form: first subproblem

- Split the input matrix in half according to columns.

Recursive echelon form: first subproblem

- Split the input matrix in half according to columns.
  Let A_1 be the first 4 columns

$$A = \overset{A_1}{\left[\begin{array}{cccc|ccccc} * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{array}\right]}$$

Recursive echelon form: first subproblem

- Split the input matrix in half according to columns.
  Let A_1 be the first 4 columns

$$A = \overset{A_1}{\left[\begin{array}{cccc|ccccc} * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{array}\right]}$$

- Compute a Gauss-transform (U_1, P_1) for A_1

Recursive echelon form: first subproblem

- Split the input matrix in half according to columns.
  Let A_1 be the first 4 columns

$$A = \overset{A_1}{\left[\begin{array}{cccc|ccccc} * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{array}\right]}$$

- Compute a Gauss-transform (U_1, P_1) for A_1

$$\overset{U_1}{\begin{bmatrix} 1 & & & & & \\ * & 1 & & & & \\ * & * & 1 & & & \\ * & * & & 1 & & \\ * & * & & & 1 & \\ * & * & & & & 1 \end{bmatrix}} \overset{P_1 A_1}{\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}} = \overset{R_1}{\begin{bmatrix} \bullet & * & * & * \\ & & & \bullet \\ & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}}$$

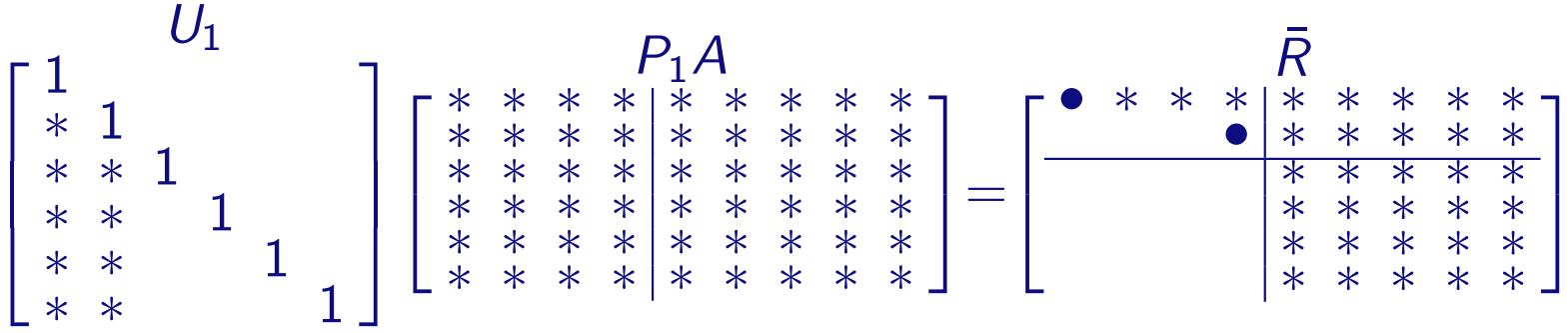
Recursive echelon form: construct second subproblem

Recursive echelon form: construct second subproblem

- Multiply through to get second subproblem

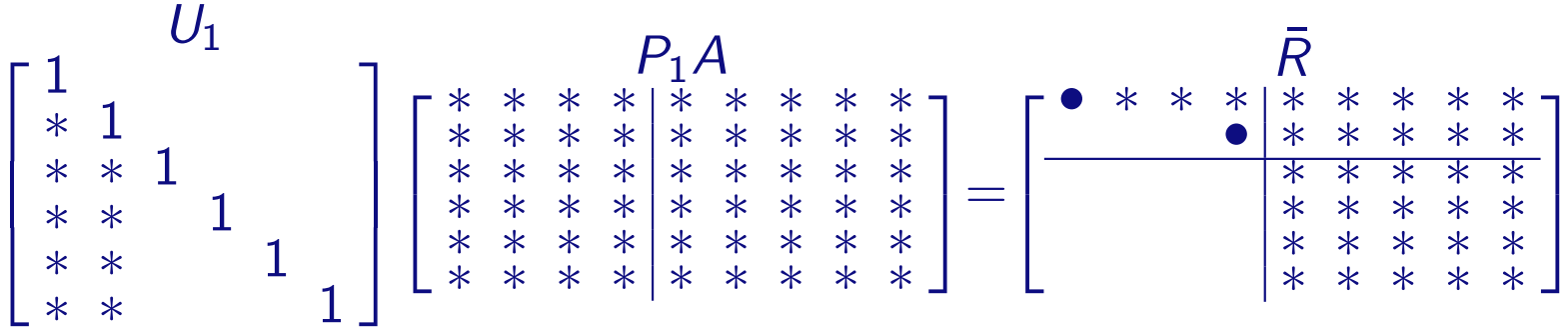
Recursive echelon form: construct second subproblem

- Multiply through to get second subproblem



Recursive echelon form: construct second subproblem

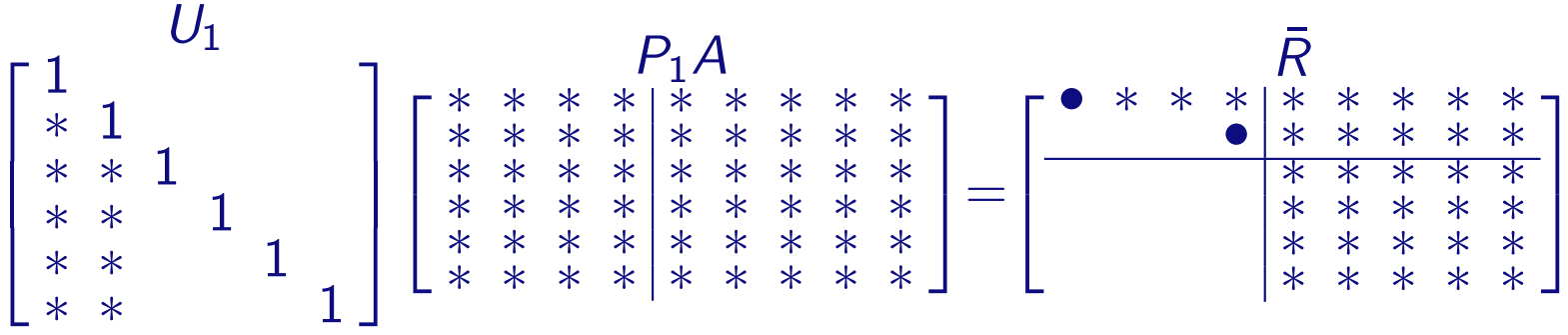
- Multiply through to get second subproblem



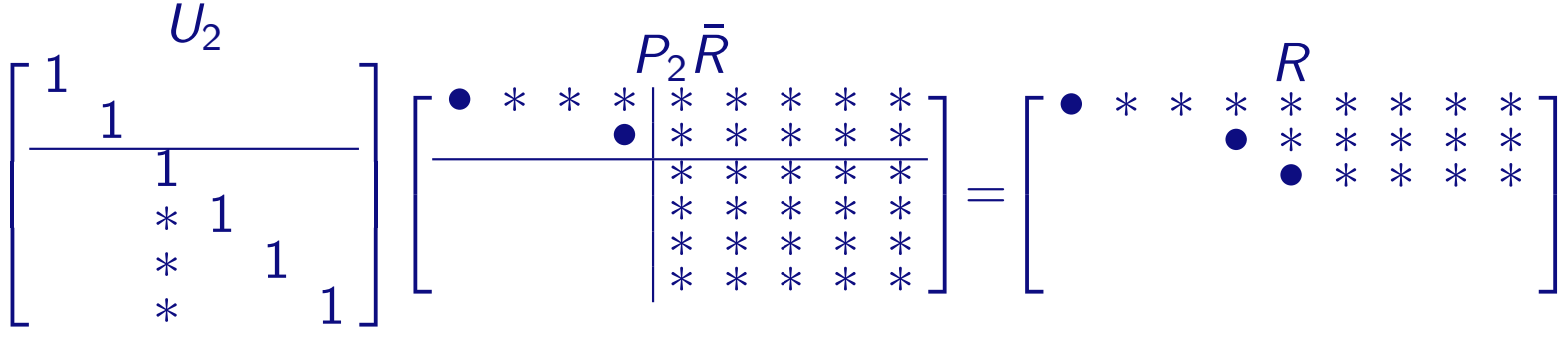
- Compute a Gauss-transform for last five columns of transformed matrix $\bar{R}$ (below horizontal line only)

Recursive echelon form: construct second subproblem

- Multiply through to get second subproblem

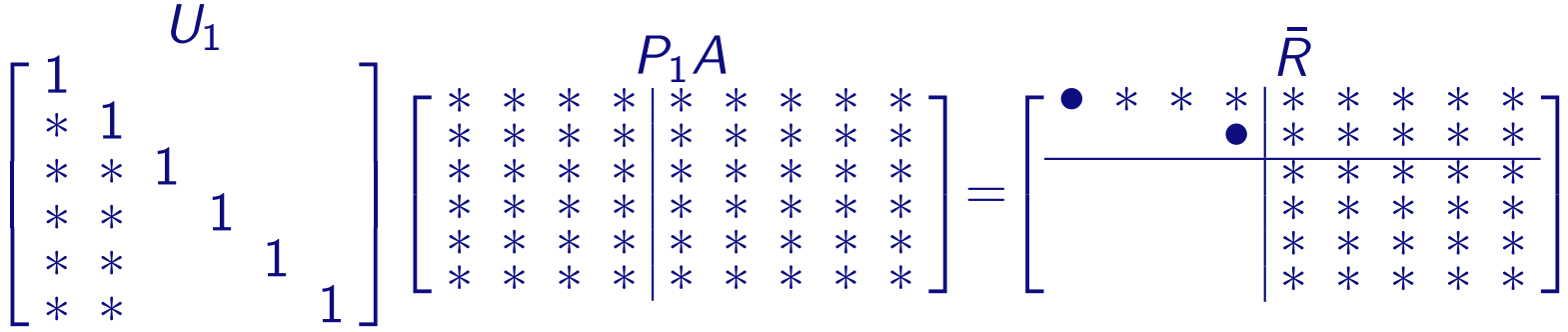


- Compute a Gauss-transform for last five columns of transformed matrix $\bar{R}$ (below horizontal line only)

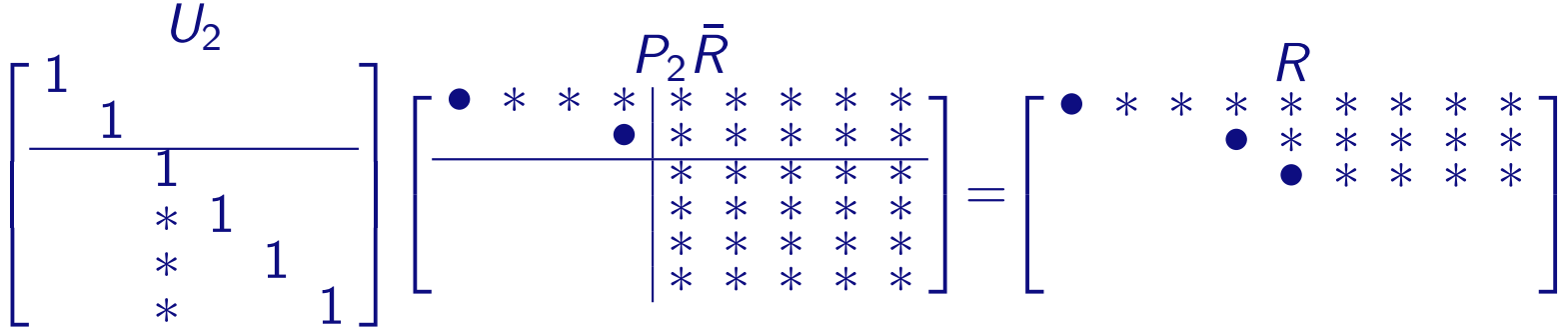


Recursive echelon form: construct second subproblem

- Multiply through to get second subproblem



- Compute a Gauss-transform for last five columns of transformed matrix $\bar{R}$ (below horizontal line only)



- Combine: Set $U = U_2 P_2 U_1 P_2^{-1}$ and $P = P_2 P_1$.

Analysis

Analysis

- Choose first problem to be on the first $m_1 = \lfloor m/2 \rfloor$ columns, second subproblem to be on last $\lceil m/2 \rceil$ columns

Analysis

- Choose first problem to be on the first $m_1 = \lfloor m/2 \rfloor$ columns, second subproblem to be on last $\lceil m/2 \rceil$ columns
- Suppose first subproblem has rank r_1, second subproblem has rank r_2

Analysis

- Choose first problem to be on the first $m_1 = \lfloor m/2 \rfloor$ columns, second subproblem to be on last $\lceil m/2 \rceil$ columns
- Suppose first subproblem has rank r_1, second subproblem has rank r_2
- Model cost as binary tree: label at root is cost of combining subproblems. For some constant c we have:

Analysis

- Choose first problem to be on the first $m_1 = \lfloor m/2 \rfloor$ columns, second subproblem to be on last $\lceil m/2 \rceil$ columns
- Suppose first subproblem has rank r_1, second subproblem has rank r_2
- Model cost as binary tree: label at root is cost of combining subproblems. For some constant c we have:
  - Root node: $cnmr^{\omega-2}$

Analysis

- Choose first problem to be on the first $m_1 = \lfloor m/2 \rfloor$ columns, second subproblem to be on last $\lceil m/2 \rceil$ columns
- Suppose first subproblem has rank r_1, second subproblem has rank r_2
- Model cost as binary tree: label at root is cost of combining subproblems. For some constant c we have:
  - Root node: $cnmr^{\omega-2}$
  - Left child: $cn(m/2)r_1^{\omega-2}$
  - Right child: $cn(m/2)r_2^{\omega-2}$

Analysis

- Choose first problem to be on the first $m_1 = \lfloor m/2 \rfloor$ columns, second subproblem to be on last $\lceil m/2 \rceil$ columns
- Suppose first subproblem has rank r_1, second subproblem has rank r_2
- Model cost as binary tree: label at root is cost of combining subproblems. For some constant c we have:
  - Root node: $cnmr^{\omega-2}$
  - Left child: $cn(m/2)r_1^{\omega-2}$
  - Right child: $cn(m/2)r_2^{\omega-2}$
- Summing over nodes and using $\omega > 2$ gives overall cost $O(nmr^{\omega-2})$

Summary so far and comments

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:
  - recursive divide & conquer approach

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:
  - recursive divide & conquer approach
  - reduce to matrix multiplication via blocking

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:
  - recursive divide & conquer approach
  - reduce to matrix multiplication via blocking
  - tri-parameter analysis in terms of n, m, and r

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:
  - recursive divide & conquer approach
  - reduce to matrix multiplication via blocking
  - tri-parameter analysis in terms of n, m, and r
- Instead of Gauss-tranform, better to compute a decomposition, eg, Turing, CUP, LSP, LQUP, QLUP

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:
  - recursive divide & conquer approach
  - reduce to matrix multiplication via blocking
  - tri-parameter analysis in terms of n, m, and r
- Instead of Gauss-tranform, better to compute a decomposition, eg, Turing, CUP, LSP, LQUP, QLUP
- Lot of work has focused on improving the leading constants in the bigO; using no extra memory; doing the transformation in-place

Summary so far and comments

- Can compute rank, rank profile, determinant, system solution, nullspace in time $O(nmr^{\omega-2})$
- Key ideas are:
  - recursive divide & conquer approach
  - reduce to matrix multiplication via blocking
  - tri-parameter analysis in terms of n, m, and r
- Instead of Gauss-tranform, better to compute a decomposition, eg, Turing, CUP, LSP, LQUP, QLUP
- Lot of work has focused on improving the leading constants in the bigO; using no extra memory; doing the transformation in-place
- Example: CUP decomposition of $A \in \mathrm{K}^{7\times 5}$ with rank 3

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{AP^T} = \underbrace{\begin{bmatrix} \overline{*}_1 & & \\ * & & \\ * & \overline{*}_2 & \\ * & * & \overline{*}_3 \\ * & * & * \\ * & * & * \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} 1 & * & * & * & * \\ & 1 & * & * & * \\ & & 1 & * & * \\ & & & 1 & \\ & & & & 1 \end{bmatrix}}_{U}$$

Selected references

Selected references

- C.-P. Jeannerod, C. Pernet and A. Storjohann, *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*. J. Symb. Comput., 2013.
  - Surveys all decompositions mentioned on previous slide
  - Compares with Gauss transform and gives pseudo code
  - Works out leading constants

Selected references

- C.-P. Jeannerod, C. Pernet and A. Storjohann, *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*. J. Symb. Comput., 2013.
  - Surveys all decompositions mentioned on previous slide
  - Compares with Gauss transform and gives pseudo code
  - Works out leading constants
- J.-G. Dumas, C. Pernet, Z. Sultan, *Fast computation of the rank profile matrix and the generalized Bruhat decomposition.* J. Symb. Comput., 2017
  - Analyses pivoting strategies
  - Rank profile matrix: computing rank profile of **all** leading submatrices with one decomposition

Selected references

- C.-P. Jeannerod, C. Pernet and A. Storjohann, *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*. J. Symb. Comput., 2013.
  - Surveys all decompositions mentioned on previous slide
  - Compares with Gauss transform and gives pseudo code
  - Works out leading constants
- J.-G. Dumas, C. Pernet, Z. Sultan, *Fast computation of the rank profile matrix and the generalized Bruhat decomposition.* J. Symb. Comput., 2017
  - Analyses pivoting strategies
  - Rank profile matrix: computing rank profile of **all** leading submatrices with one decomposition
- J.-G. Dumas, P. Giorgi, C. Pernet, *Dense Linear Algebra over Word-Size Prime Fields: the FFLAS and FFPACK Packages. ACM Trans. Mathematical. Software*, 2008
  - Highly-optimized implementations over small finite fields

Characteristic polynomial

Characteristic polynomial

- Characteristic polynomial of (square) $A \in \mathrm{K}^{n \times n}$ is defined as

$$\det(xI_n - A) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \in \mathrm{K}[x]$$

Characteristic polynomial

- Characteristic polynomial of (square) $A \in \mathrm{K}^{n \times n}$ is defined as

$$\det(xI_n - A) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \in \mathrm{K}[x]$$

- If A is *simple* then there exists a $v \in \mathrm{K}^{n \times 1}$ such that

Characteristic polynomial

- Characteristic polynomial of (square) $A \in \mathrm{K}^{n \times n}$ is defined as

$$\det(xI_n - A) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \in \mathrm{K}[x]$$

- If A is *simple* then there exists a $v \in \mathrm{K}^{n \times 1}$ such that

$$U = [\, v \,|\, Av \,|\, \cdots \,|\, A^{n-1}v \,]$$

Characteristic polynomial

- Characteristic polynomial of (square) $A \in \mathrm{K}^{n \times n}$ is defined as

$$\det(xI_n - A) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \in \mathrm{K}[x]$$

- If A is *simple* then there exists a $v \in \mathrm{K}^{n \times 1}$ such that

$$U = [\, v \,|\, Av \,|\, \cdots \,|\, A^{n-1}v \,] \quad \text{and} \quad U^{-1}AU = \begin{bmatrix} 0 & \cdots & 0 & -c_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -c_{n-2} \\ & & 1 & -c_{n-1} \end{bmatrix}$$

Characteristic polynomial

- Characteristic polynomial of (square) $A \in \mathsf{K}^{n \times n}$ is defined as

$$\det(xI_n - A) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \in \mathsf{K}[x]$$

- If A is *simple* then there exists a $v \in \mathsf{K}^{n \times 1}$ such that

$$U = [\, v \,|\, Av \,|\, \cdots \,|\, A^{n-1}v \,] \quad \text{and} \quad U^{-1}AU = \begin{bmatrix} 0 & \cdots & 0 & -c_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -c_{n-2} \\ & & 1 & -c_{n-1} \end{bmatrix}$$

- So, A is *similar* to a single companion matrix in this case.

Warmup: simple case

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av
  - Compute $A^2 \left[\, v \, \middle| \, Av \, \right]$ to get $\left[\, A^2 v \quad A^3 v \, \right]$

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av
  - Compute $A^2 \left[\, v \,\middle|\, Av \, \right]$ to get $\left[\, A^2 v \quad A^3 v \, \right]$
  - Compute $A^4 \left[\, v \,\middle|\, Av \,\middle|\, A^2 v \,\middle|\, A^3 v \, \right]$ to get $\left[\, A^4 v \,\middle|\, A^5 v \,\middle|\, A^6 v \,\middle|\, A^7 v \, \right]$

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av
  - Compute $A^2 \left[\, v \,\middle|\, Av \, \right]$ to get $\left[\, A^2 v \quad A^3 v \, \right]$
  - Compute $A^4 \left[\, v \,\middle|\, Av \,\middle|\, A^2 v \,\middle|\, A^3 v \, \right]$ to get $\left[\, A^4 v \,\middle|\, A^5 v \,\middle|\, A^6 v \,\middle|\, A^7 v \, \right]$
  - etc

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av
  - Compute $A^2 \left[\, v \,\middle|\, Av \,\right]$ to get $\left[\, A^2v \quad A^3v \,\right]$
  - Compute $A^4 \left[\, v \,\middle|\, Av \,\middle|\, A^2v \,\middle|\, A^3v \,\right]$ to get $\left[\, A^4v \,\middle|\, A^5v \,\middle|\, A^6v \,\middle|\, A^7v \,\right]$
  - etc
- Let $U = \left[\, v \mid Av \mid \cdots \mid A^{n-1}v \,\right]$

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av
  - Compute $A^2 \left[\, v \,\middle|\, Av \,\right]$ to get $\left[\, A^2v \quad A^3v \,\right]$
  - Compute $A^4 \left[\, v \,\middle|\, Av \,\middle|\, A^2v \,\middle|\, A^3v \,\right]$ to get $\left[\, A^4v \,\middle|\, A^5v \,\middle|\, A^6v \,\middle|\, A^7v \,\right]$
  - etc
- Let $U = \left[\, v \mid Av \mid \cdots \mid A^{n-1}v \,\right]$
- Compute $U^{-1}AU$

Warmup: simple case

- Suppose $A \in \mathsf{K}^{n \times n}$ is simple and we know a good $v \in \mathsf{K}^{n \times 1}$
- Start by computing $A^2, A^4, A^8, \ldots, A^{2^{\lceil \log_2 n \rceil}}$
  - Compute Av
  - Compute $A^2 \left[\, v \,\middle|\, Av \,\right]$ to get $\left[\, A^2v \quad A^3v \,\right]$
  - Compute $A^4 \left[\, v \,\middle|\, Av \,\middle|\, A^2v \,\middle|\, A^3v \,\right]$ to get $\left[\, A^4v \,\middle|\, A^5v \,\middle|\, A^6v \,\middle|\, A^7v \,\right]$
  - etc
- Let $U = \left[\, v \mid Av \mid \cdots \mid A^{n-1}v \,\right]$
- Compute $U^{-1}AU$
- Total cost: $O(n^\omega \log n)$ operations

General case

General case

- Let $\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = I_n$

General case

- Let $\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = I_n$
  Note: any set of n v_* that are linearly independent work

General case

- Let $\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = I_n$
  Note: any set of n v_* that are linearly independent work
- If $(d_1, d_2, \dots, d_n) \in \mathbb{Z}^n$ is lexicographically maximal such that

  $$U = \left[\begin{array}{c|c|c|c||c|c|c|c|c} v_1 & Av_1 & \cdots & A^{d_1-1}v_1 & \cdots & v_n & Av_n & \cdots & A^{d_n-1}v_n \end{array}\right]$$

  is nonsingular, then

  $$U^{-1}AU = \begin{bmatrix} C_1 & B_{12} & \cdots & B_{1n} \\ & C_2 & & B_{2n} \\ & & \ddots & \vdots \\ & & & C_n \end{bmatrix}$$

  where each C_* is a companion matrix.

General case

- Let $\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = I_n$
  Note: any set of n v_* that are linearly independent work
- If $(d_1, d_2, \dots, d_n) \in \mathbb{Z}^n$ is lexicographically maximal such that

$$U = \left[\begin{array}{c|c|c|c|c|c|c|c} v_1 & Av_1 & \cdots & A^{d_1-1}v_1 & \cdots & v_n & Av_n & \cdots & A^{d_n-1}v_n \end{array}\right]$$

  is nonsingular, then

$$U^{-1}AU = \begin{bmatrix} C_1 & B_{12} & \cdots & B_{1n} \\ & C_2 & & B_{2n} \\ & & \ddots & \vdots \\ & & & C_n \end{bmatrix}$$

  where each C_* is a companion matrix.
  Note: many of C_* may be 0×0

General case

- Let $\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = I_n$
  Note: any set of n v_* that are linearly independent work
- If $(d_1, d_2, \dots, d_n) \in \mathbb{Z}^n$ is lexicographically maximal such that

  $$U = \left[\begin{array}{c|c|c|c||c|c|c|c|c} v_1 & Av_1 & \cdots & A^{d_1-1}v_1 & \cdots & v_n & Av_n & \cdots & A^{d_n-1}v_n \end{array}\right]$$

  is nonsingular, then

  $$U^{-1}AU = \begin{bmatrix} C_1 & B_{12} & \cdots & B_{1n} \\ & C_2 & & B_{2n} \\ & & \ddots & \vdots \\ & & & C_n \end{bmatrix}$$

  where each C_* is a companion matrix.
  Note: many of C_* may be 0×0
- Characteristic polynomial of A is then the product of the polynomials corresponding to the C_*

General case: continued

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before
- Initialize $U := \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = I_n$

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before
- Initialize $U := \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = I_n$
- First step: $AU = \begin{bmatrix} Av_1 & Av_2 & \cdots & Av_n \end{bmatrix}$ and add column vectors to U to obtain

$$U = \left[\begin{array}{c|c|c|c} v_1 & Av_1 & v_2 & Av_2 \end{array} \middle| \cdots \middle| \begin{array}{c|c} v_n & Av_n \end{array}\right] \in \mathrm{K}^{n \times 2n}$$

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before
- Initialize $U := \left[\begin{array}{cccc} v_1 & v_2 & \cdots & v_n \end{array} \right] = I_n$
- First step: $AU = \left[\begin{array}{cccc} Av_1 & Av_2 & \cdots & Av_n \end{array} \right]$ and add column vectors to U to obtain

$$U = \left[\begin{array}{c|c|c|c} v_1 & Av_1 & v_2 & Av_2 \end{array} \middle| \cdots \middle| \begin{array}{cc} v_n & Av_n \end{array} \right] \in \mathrm{K}^{n \times 2n}$$

- Compute column rank profile of U and remove all linearly dependent vectors (those not corresponding to the rank profile)

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before
- Initialize $U := \left[\begin{array}{cccc} v_1 & v_2 & \cdots & v_n \end{array} \right] = I_n$
- First step: $AU = \left[\begin{array}{cccc} Av_1 & Av_2 & \cdots & Av_n \end{array} \right]$ and add column vectors to U to obtain

$$U = \left[\begin{array}{c|c|c|c} v_1 & Av_1 & v_2 & Av_2 \end{array} \right| \left. \begin{array}{c|c} \cdots & v_n \end{array} \right. \left. \begin{array}{c} Av_n \end{array} \right] \in \mathrm{K}^{n \times 2n}$$

- Compute column rank profile of U and remove all linearly dependent vectors (those not corresponding to the rank profile)
- Repeat process with $A^2, A^4, \ldots$ until U stabilizes

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before
- Initialize $U := \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = I_n$
- First step: $AU = \begin{bmatrix} Av_1 & Av_2 & \cdots & Av_n \end{bmatrix}$ and add column vectors to U to obtain

$$U = \left[\begin{array}{c|c|c|c} v_1 & Av_1 & v_2 & Av_2 \end{array} \middle| \cdots \middle| \begin{array}{c|c} v_n & Av_n \end{array}\right] \in \mathrm{K}^{n \times 2n}$$

- Compute column rank profile of U and remove all linearly dependent vectors (those not corresponding to the rank profile)
- Repeat process with $A^2, A^4, \ldots$ until U stabilizes
- $U^{-1}AU$ now has desired block upper triangular shape

General case: continued

- Compute $A^2, A^4, \ldots, A^{2^{\lceil \log_2 n \rceil}}$ as before
- Initialize $U := \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = I_n$
- First step: $AU = \begin{bmatrix} Av_1 & Av_2 & \cdots & Av_n \end{bmatrix}$ and add column vectors to U to obtain

$$U = \left[\begin{array}{cc|cc|c|cc} v_1 & Av_1 & v_2 & Av_2 & \cdots & v_n & Av_n \end{array}\right] \in \mathsf{K}^{n \times 2n}$$

- Compute column rank profile of U and remove all linearly dependent vectors (those not corresponding to the rank profile)
- Repeat process with $A^2, A^4, \ldots$ until U stabilizes
- $U^{-1}AU$ now has desired block upper triangular shape
- Total cost: $O(n^\omega \log n)$

Selected references: Characteristic polynomial

Selected references: Characteristic polynomial

- W. Keller-Gehrig. *Fast algorithms for the characteristic polynomial*, Theor. Comp. Sci., 1985.
  - $O(n^\omega \log n)$
  - Includes algorithm just described

Selected references: Characteristic polynomial

- W. Keller-Gehrig. *Fast algorithms for the characteristic polynomial*, Theor. Comp. Sci., 1985.
  - $O(n^\omega \log n)$
  - Includes algorithm just described

- E. Kaltofen and G. Villard. *On the complexity of computing determinants*, Comp. Compl., 2005.
  - Gives fast algorithms for **division free** computation

Selected references: Characteristic polynomial

- W. Keller-Gehrig. *Fast algorithms for the characteristic polynomial*, Theor. Comp. Sci., 1985.
  - $O(n^\omega \log n)$
  - Includes algorithm just described

- E. Kaltofen and G. Villard. *On the complexity of computing determinants*, Comp. Compl., 2005.
  - Gives fast algorithms for **division free** computation

- V. Neiger and C. Pernet. *Deterministic computation of the characteristic polynomial in the time of matrix multiplication*, J. Complex., 2021
  - $O(n^\omega)$
  - Winner of 2021 J. Complex. best paper award

Frobenius form

Frobenius form

- As before, $A \in \mathsf{K}^{n \times n}$

Frobenius form

- As before, $A \in \mathsf{K}^{n \times n}$
- There exist very special (rare) vectors $v_1, v_2, \ldots, v_k \in \mathsf{K}^{n \times 1}$ such that

$$F := U^{-1}AU = \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_k \end{bmatrix},$$

with the polynomial corresponding to C_i a multiple of that corresponding to C_{i+1}, $i = 1, 2, \ldots, k-1$

Frobenius form

- As before, $A \in \mathsf{K}^{n \times n}$
- There exist very special (rare) vectors $v_1, v_2, \ldots, v_k \in \mathsf{K}^{n \times 1}$ such that

$$F := U^{-1}AU = \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_k \end{bmatrix},$$

  with the polynomial corresponding to C_i a multiple of that corresponding to C_{i+1}, $i = 1, 2, \ldots, k-1$
- F is the Frobenius canonical form of A

Frobenius form

- As before, $A \in \mathsf{K}^{n \times n}$
- There exist very special (rare) vectors $v_1, v_2, \ldots, v_k \in \mathsf{K}^{n \times 1}$ such that

$$F := U^{-1}AU = \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_k \end{bmatrix},$$

  with the polynomial corresponding to C_i a multiple of that corresponding to C_{i+1}, $i = 1, 2, \ldots, k-1$
- F is the Frobenius canonical form of A
- Many applications, eg, compute $A^s = UF^sU^{-1}$ quickly

Giesbrecht's randomized reduction to matrix multiplication

Giesbrecht's randomized reduction to matrix multiplication

- Requires a subset $S \subseteq K$ with $\#S \geq n^2$

Giesbrecht's randomized reduction to matrix multiplication

- Requires a subset $S \subseteq \mathrm{K}$ with $\#S \geq n^2$
- Choose vectors $v_1, v_2, \ldots, v_n \in S^{n \times 1}$ uniformly and randomly

Giesbrecht's randomized reduction to matrix multiplication

- Requires a subset $S \subseteq \mathsf{K}$ with $\#S \geq n^2$
- Choose vectors $v_1, v_2, \ldots, v_n \in S^{n \times 1}$ uniformly and randomly
- Run Keller-Gehrig's algorithm for the charpoly to get $\bar{U} \in \mathsf{K}^{n \times n}$ such that $\bar{F} = \bar{U}^{-1} A \bar{U}$ is block upper triangular

Giesbrecht's randomized reduction to matrix multiplication

- Requires a subset $S \subseteq \mathsf{K}$ with $\#S \geq n^2$
- Choose vectors $v_1, v_2, \ldots, v_n \in S^{n \times 1}$ uniformly and randomly
- Run Keller-Gehrig's algorithm for the charpoly to get $\bar{U} \in \mathsf{K}^{n \times n}$ such that $\bar{F} = \bar{U}^{-1} A \bar{U}$ is block upper triangular
- With high probability, the diagonal blocks of $\bar{F}$ will be correct, ie, those of the Frobenius form (this is easy to check)

Giesbrecht's randomized reduction to matrix multiplication

- Requires a subset $S \subseteq \mathsf{K}$ with $\#S \geq n^2$
- Choose vectors $v_1, v_2, \ldots, v_n \in S^{n \times 1}$ uniformly and randomly
- Run Keller-Gehrig's algorithm for the charpoly to get $\bar{U} \in \mathsf{K}^{n \times n}$ such that $\bar{F} = \bar{U}^{-1} A \bar{U}$ is block upper triangular
- With high probability, the diagonal blocks of $\bar{F}$ will be correct, ie, those of the Frobenius form (this is easy to check)
- "Purifying" the vectors in $\bar{U}$ to get U with $U^{-1}AU$ in Frobenius form is easily done with one more run of Keller-Gehrig's algorithm

Giesbrecht's randomized reduction to matrix multiplication

- Requires a subset $S \subseteq \mathsf{K}$ with $\#S \geq n^2$
- Choose vectors $v_1, v_2, \ldots, v_n \in S^{n \times 1}$ uniformly and randomly
- Run Keller-Gehrig's algorithm for the charpoly to get $\bar{U} \in \mathsf{K}^{n \times n}$ such that $\bar{F} = \bar{U}^{-1} A \bar{U}$ is block upper triangular
- With high probability, the diagonal blocks of $\bar{F}$ will be correct, ie, those of the Frobenius form (this is easy to check)
- "Purifying" the vectors in $\bar{U}$ to get U with $U^{-1}AU$ in Frobenius form is easily done with one more run of Keller-Gehrig's algorithm
- This is the first "nearly-optimal" algorithm for Frobenius form. But it is randomized and requires the field be large enough

Selected references: Frobenius form with transform

- M. Giesbrecht (1993). M. Giesbrect, *Nearly optimal algorithms for canonical matrix forms*, Siam J. Comp., 1995.
  - Las Vegas $O(n^\omega \log n)$ when $\#\mathsf{K} \geq n^2$
- W. Eberly, *Asymptotically efficient algorithms for the Frobenius form*, 2000, Dept. of Comp. Sci., University of Calgary
  - Las Vegas $O(n^\omega \log n)$ also over small fields
- A. Storjohann, PhD thesis, 2000
  - Deterministic $O(n^\omega \log n \operatorname{loglog} n)$
- C. Pernet and A. Storjohann, *Faster algorithms for the characteristic polynomial*, ISSAC' 07.
  - Las Vegas $O(n^\omega \operatorname{loglog} n)$ when $\#K \geq n^2$