Linear algebra over $\mathbb{Z}$ and $\mathsf{K}[x]$

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

Preliminaries: Setting

- Over $\mathsf{K}[x]$ we need to take into account the degree of polynomials
  - count field operations from K as before

Preliminaries: Setting

- Over $\mathsf{K}[x]$ we need to take into account the degree of polynomials
  - count field operations from K as before
- Over $\mathbb{Z}$ we need to take into account the bitlength of integers
  - count bit operations

Preliminaries: Setting

- Over $\mathsf{K}[x]$ we need to take into account the degree of polynomials
  - count field operations from K as before
- Over $\mathbb{Z}$ we need to take into account the bitlength of integers
  - count bit operations
- Note that $\mathsf{K}[x] \subset \mathsf{K}(x)$ and $\mathbb{Z} \subset \mathbb{Q}$.

Preliminaries: Setting

- Over $\mathsf{K}[x]$ we need to take into account the degree of polynomials
  - count field operations from K as before
- Over $\mathbb{Z}$ we need to take into account the bitlength of integers
  - count bit operations
- Note that $\mathsf{K}[x] \subset \mathsf{K}(x)$ and $\mathbb{Z} \subset \mathbb{Q}$.
- For problems **Rank**, **Rank Profile**, **Determinant** the solution is the same over the integral domain or the fraction field

Preliminaries: Setting

- Over $\mathsf{K}[x]$ we need to take into account the degree of polynomials
  - count field operations from K as before
- Over $\mathbb{Z}$ we need to take into account the bitlength of integers
  - count bit operations
- Note that $\mathsf{K}[x] \subset \mathsf{K}(x)$ and $\mathbb{Z} \subset \mathbb{Q}$.
- For problems **Rank**, **Rank Profile**, **Determinant** the solution is the same over the integral domain or the fraction field
- For problems **Inverse**, **Linear System Solving** and **Nullspace** the solution is over the fraction field

Preliminaries: Setting

- Over $\mathsf{K}[x]$ we need to take into account the degree of polynomials
  - count field operations from K as before
- Over $\mathbb{Z}$ we need to take into account the bitlength of integers
  - count bit operations
- Note that $\mathsf{K}[x] \subset \mathsf{K}(x)$ and $\mathbb{Z} \subset \mathbb{Q}$.
- For problems **Rank**, **Rank Profile**, **Determinant** the solution is the same over the integral domain or the fraction field
- For problems **Inverse**, **Linear System Solving** and **Nullspace** the solution is over the fraction field
- Main challenge: bitlength in output can be large...

Example: Growth of bitlength in the output

Example: Growth of bitlength in the output

- Definition: $||A|| = \max_{ij} |A_{ij}|$
  $\Rightarrow$ *length* of integers in A is bounded by $O(\log ||A||)$

Example: Growth of bitlength in the output

- Definition: $||A|| = \max_{ij} |A_{ij}|$
  $\Rightarrow$ *length* of integers in A is bounded by $O(\log ||A||)$
- Consider $n = 5$ and $\log_{10} ||A|| \approx 3$ (number of decimal digits)

$$A = \begin{bmatrix} 594 & 24 & 601 & 604 & 827 \\ 476 & 397 & 49 & 378 & 174 \\ 7 & 361 & 173 & 939 & 392 \\ 844 & 186 & 655 & 896 & 453 \\ 76 & 621 & 38 & 603 & 582 \end{bmatrix} \qquad b = \begin{bmatrix} 450 \\ 717 \\ 508 \\ 238 \\ 366 \end{bmatrix}$$

Example: Growth of bitlength in the output

- Definition: $||A|| = \max_{ij} |A_{ij}|$
  $\Rightarrow$ *length* of integers in A is bounded by $O(\log ||A||)$
- Consider $n = 5$ and $\log_{10} ||A|| \approx 3$ (number of decimal digits)

$$A = \begin{bmatrix} 594 & 24 & 601 & 604 & 827 \\ 476 & 397 & 49 & 378 & 174 \\ 7 & 361 & 173 & 939 & 392 \\ 844 & 186 & 655 & 896 & 453 \\ 76 & 621 & 38 & 603 & 582 \end{bmatrix} \qquad b = \begin{bmatrix} 450 \\ 717 \\ 508 \\ 238 \\ 366 \end{bmatrix}$$

- $\det A = -26592243059232$, with about $n \times \log_{10} ||A||$ digits

Example: Growth of bitlength in the output

- Definition: $||A|| = \max_{ij} |A_{ij}|$
  $\Rightarrow$ *length* of integers in A is bounded by $O(\log ||A||)$
- Consider $n = 5$ and $\log_{10} ||A|| \approx 3$ (number of decimal digits)

$$A = \begin{bmatrix} 594 & 24 & 601 & 604 & 827 \\ 476 & 397 & 49 & 378 & 174 \\ 7 & 361 & 173 & 939 & 392 \\ 844 & 186 & 655 & 896 & 453 \\ 76 & 621 & 38 & 603 & 582 \end{bmatrix} \qquad b = \begin{bmatrix} 450 \\ 717 \\ 508 \\ 238 \\ 366 \end{bmatrix}$$

- $\det A = -26592243059232$, with about $n \times \log_{10} ||A||$ digits
- $A^{-1}b = \begin{bmatrix} -58686180258858 \\ 70644871354626 \\ 143314986631278 \\ -49969380574326 \\ -42023211987798 \end{bmatrix} (1/\det A)$

Cost models

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$
   - Cost to compute $a = qb + r$ is $O(m(n - m))$

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$
   - Cost to compute $a = qb + r$ is $O(m(n - m))$

   $\rightarrow$ analagous for polynomials, with n and m now degrees

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$
   - Cost to compute $a = qb + r$ is $O(m(n - m))$

   $\rightarrow$ analagous for polynomials, with n and m now degrees
2. In terms of a multiplication time M
   - Two integers with length bounded by M can be multiplied in $O(\mathsf{M}(n))$ bit operations

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$
   - Cost to compute $a = qb + r$ is $O(m(n - m))$

   $\rightarrow$ analagous for polynomials, with n and m now degrees

2. In terms of a multiplication time M
   - Two integers with length bounded by M can be multiplied in $O(\mathsf{M}(n))$ bit operations
   - This model the shows dependence (or not) on the use of fast integer multiplication
     Eg, $O(n\,\mathsf{M}(n))$ vs $O(n^2 \log n)$

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$
   - Cost to compute $a = qb + r$ is $O(m(n - m))$

   $\rightarrow$ analagous for polynomials, with n and m now degrees

2. In terms of a multiplication time M
   - Two integers with length bounded by M can be multiplied in $O(\mathsf{M}(n))$ bit operations
   - This model the shows dependence (or not) on the use of fast integer multiplication
     Eg, $O(n\,\mathsf{M}(n))$ vs $O(n^2 \log n)$

   $\rightarrow$ analagous for polynomials, with n now the degree

Cost models

1. Naive / standard quadratic arithmetic / classical / school methods
   - Let $a \in \mathbb{Z}$ have length bounded by $n > 0$
   - Let $b \in \mathbb{Z}$ have length bounded by $m > 0$
   - Cost to compute ab and $sa + tb = \gcd(a, b)$ is $O(nm)$
   - Cost to compute $a = qb + r$ is $O(m(n - m))$

   $\rightarrow$ analagous for polynomials, with n and m now degrees

2. In terms of a multiplication time M
   - Two integers with length bounded by M can be multiplied in $O(\mathsf{M}(n))$ bit operations
   - This model the shows dependence (or not) on the use of fast integer multiplication
     Eg, $O(n\,\mathsf{M}(n))$ vs $O(n^2 \log n)$

   $\rightarrow$ analagous for polynomials, with n now the degree

3. Suppressing factors that are logarithmic in the input size
   - Capture additonal factors $C_1(\log n)^{C_2}(\mathrm{loglog}\,||A||)^{C_3}$ for positive real constants C_*
     Eg, $O\tilde{}(n^2 \log ||A||)$ or $(n^2 \log ||A||)^{1+o(1)}$

Basic tools

Basic tools

1. Multi-modular reduction and chinese remaindering
   - Let $P = p_1 p_2 \cdots p_k$ for pairwise relatively prime p_*

$$\mathrm{rem}(a, P) \longleftrightarrow (\mathrm{rem}(a, p_1), \mathrm{rem}(a, p_2), \ldots, \mathrm{rem}(a, p_k))$$

Basic tools

1. Multi-modular reduction and chinese remaindering
   - Let $P = p_1 p_2 \cdots p_k$ for pairwise relatively prime p_*

$$\mathrm{rem}(a, P) \longleftrightarrow (\mathrm{rem}(a, p_1), \mathrm{rem}(a, p_2), \ldots, \mathrm{rem}(a, p_k))$$

2. Radix conversion

$$23421 = 12 + 0(17) + 13(17)^2 + 4(17)^3$$

Basic tools

1. Multi-modular reduction and chinese remaindering
   - Let $P = p_1 p_2 \cdots p_k$ for pairwise relatively prime p_*

$$\mathrm{rem}(a, P) \longleftrightarrow (\mathrm{rem}(a, p_1), \mathrm{rem}(a, p_2), \ldots, \mathrm{rem}(a, p_k))$$

2. Radix conversion

$$23421 = 12 + 0(17) + 13(17)^2 + 4(17)^3$$

3. Rational number reconstruction

$$\frac{3423432}{2323321} \equiv 28220078069280992 \bmod 10^{18}$$

Basic tools

1. Multi-modular reduction and chinese remaindering
   - Let $P = p_1 p_2 \cdots p_k$ for pairwise relatively prime p_*

$$\operatorname{rem}(a, P) \longleftrightarrow (\operatorname{rem}(a, p_1), \operatorname{rem}(a, p_2), \ldots, \operatorname{rem}(a, p_k))$$

2. Radix conversion

$$23421 = 12 + 0(17) + 13(17)^2 + 4(17)^3$$

3. Rational number reconstruction

$$\frac{3423432}{2323321} \equiv 28220078069280992 \bmod 10^{18}$$

- All of these cost $\mathsf{B}(n) \in O(\mathsf{M}(n) \log n)$

Basic tools

1. Multi-modular reduction and chinese remaindering
   - Let $P = p_1 p_2 \cdots p_k$ for pairwise relatively prime p_*

$$\operatorname{rem}(a, P) \longleftrightarrow (\operatorname{rem}(a, p_1), \operatorname{rem}(a, p_2), \ldots, \operatorname{rem}(a, p_k))$$

2. Radix conversion

$$23421 = 12 + 0(17) + 13(17)^2 + 4(17)^3$$

3. Rational number reconstruction

$$\frac{3423432}{2323321} \equiv 28220078069280992 \bmod 10^{18}$$

- All of these cost $\mathsf{B}(n) \in O(\mathsf{M}(n) \log n)$
- Reference: *Modern Computer Algebra*, von zur Gathen & Gerhard.

Gaussian elimination via cross multiplying

Consider input

$$A = \begin{bmatrix} 860 & 758 & 750 & 889 \\ 300 & 991 & 5 & 993 \\ 954 & 299 & 99 & 549 \\ 196 & 282 & 351 & 16 \end{bmatrix}$$

Gaussian elimination via cross multiplying

Consider input

$$A = \begin{bmatrix} 860 & 758 & 750 & 889 \\ 300 & 991 & 5 & 993 \\ 954 & 299 & 99 & 549 \\ 196 & 282 & 351 & 16 \end{bmatrix}$$

Gaussian elimination via cross multiplying to avoid fractions yields

$$T = \begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -496731184000 & 38741667000 \\ & & & 72667738186290413200000 \end{bmatrix}$$

Gaussian elimination via cross multiplying

Consider input

$$A = \begin{bmatrix} 860 & 758 & 750 & 889 \\ 300 & 991 & 5 & 993 \\ 954 & 299 & 99 & 549 \\ 196 & 282 & 351 & 16 \end{bmatrix}$$

Gaussian elimination via cross multiplying to avoid fractions yields

$$T = \begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -496731184000 & 38741667000 \\ & & & 72667738186290413200000 \end{bmatrix}$$

- The length of integers grows exponentially (doubling effect)

Fraction free gaussian elimination: (FFGE)

E. H. Bareiss. *Sylvester's identity and multi-step integer-preserving gaussian elimination*, 1968

Fraction free gaussian elimination: (FFGE)

E. H. Bareiss. *Sylvester's identity and multi-step integer-preserving gaussian elimination*, 1968

- After each column elimination, divide by previous pivot

Fraction free gaussian elimination: (FFGE)

E. H. Bareiss. *Sylvester's identity and multi-step integer-preserving gaussian elimination*, 1968

- After each column elimination, divide by previous pivot
- Eg, after eliminating first two columns as before we obtain

$$\begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -577594400 \times 860 & 45048450 \times 860 \\ & & 136629100 \times 860 & -180762980 \times 860 \end{bmatrix}$$

Fraction free gaussian elimination: (FFGE)

E. H. Bareiss. *Sylvester's identity and multi-step integer-preserving gaussian elimination*, 1968

- After each column elimination, divide by previous pivot
- Eg, after eliminating first two columns as before we obtain

$$\begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -577594400 \times 860 & 45048450 \times 860 \\ & & 136629100 \times 860 & -180762980 \times 860 \end{bmatrix}$$

- Final result is

$$\begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -577594400 & 45048450 \\ & & & 157239630950 \end{bmatrix}$$

Fraction free gaussian elimination: (FFGE)

E. H. Bareiss. *Sylvester's identity and multi-step integer-preserving gaussian elimination*, 1968

- After each column elimination, divide by previous pivot
- Eg, after eliminating first two columns as before we obtain

$$\begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -577594400 \times 860 & 45048450 \times 860 \\ & & 136629100 \times 860 & -180762980 \times 860 \end{bmatrix}$$

- Final result is

$$\begin{bmatrix} 860 & 758 & 750 & 889 \\ & 624860 & -220700 & 587280 \\ & & -577594400 & 45048450 \\ & & & 157239630950 \end{bmatrix}$$

- Growth in length of integers is now linear
- All integers appearing are minors of the input matrix

Incorporating matrix multiplication

Incorporating matrix multiplication

The input matrix

$$A = \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix}.$$

has leading principal minors $(e_0, e_1, e_2, e_3, e_4) = (1, 4, 7, 3, 9)$.

Incorporating matrix multiplication

The input matrix

$$A = \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix}.$$

has leading principal minors $(e_0, e_1, e_2, e_3, e_4) = (1, 4, 7, 3, 9)$.

- $\frac{E_1}{1}\begin{bmatrix} 1 & & & \\ -1 & 4 & & \\ -2 & & 4 & \\ 0 & & & 4 \end{bmatrix}\begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \end{bmatrix}$

Incorporating matrix multiplication

The input matrix

$$A = \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix}.$$

has leading principal minors $(e_0, e_1, e_2, e_3, e_4) = (1, 4, 7, 3, 9)$.

- $\frac{1}{1} \overset{E_1}{\begin{bmatrix} 1 & & & \\ -1 & 4 & & \\ -2 & & 4 & \\ 0 & & & 4 \end{bmatrix}} \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \end{bmatrix}$
- $\frac{1}{4} \overset{E_2}{\begin{bmatrix} 7 & -5 & & \\ & 4 & & \\ & -6 & 7 & \\ & -8 & & 7 \\ & 9 & & & 7 \end{bmatrix}} \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \\ & -9 & 0 & -15 \end{bmatrix} = \begin{bmatrix} 7 & & -15 & -1 \\ & 7 & 12 & 5 \\ & & 3 & 10 \\ & & -3 & 11 \\ & & 27 & -15 \end{bmatrix}$

Incorporating matrix multiplication

The input matrix

$$A = \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix}.$$

has leading principal minors $(e_0, e_1, e_2, e_3, e_4) = (1, 4, 7, 3, 9)$.

- $\frac{1}{1} \overset{E_1}{\begin{bmatrix} 1 & & & \\ -1 & 4 & & \\ -2 & & 4 & \\ 0 & & & 4 \end{bmatrix}} \begin{bmatrix} 4 & 5 & 0 & 3 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 \\ 0 & 2 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \end{bmatrix}$
- $\frac{1}{4} \overset{E_2}{\begin{bmatrix} 7 & -5 & & \\ & 4 & & \\ & -6 & 7 & \\ & -8 & & 7 \\ & 9 & & & 7 \end{bmatrix}} \begin{bmatrix} 4 & 5 & 0 & 3 \\ & 7 & 12 & 5 \\ & 6 & 12 & 10 \\ & 8 & 12 & 12 \\ & -9 & 0 & -15 \end{bmatrix} = \begin{bmatrix} 7 & & -15 & -1 \\ & 7 & 12 & 5 \\ & & 3 & 10 \\ & & -3 & 11 \\ & & 27 & -15 \end{bmatrix}$
- We obtain $\frac{1}{e_3} E_4 \left(\frac{1}{e_2} E_3 \left(\frac{1}{e_1} E_2 \left(\frac{1}{e_0} E_1 \right) \right) \right) A = (\det A) I_4$

Incorporating matrix multiplication

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal
  - $\frac{1}{e_3} E_4 \left(\frac{1}{e_2} E_3 \left(\frac{1}{e_1} E_2 \left(\frac{1}{e_0} E_1 \right) \right) \right)$

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal
  - $\frac{1}{e_3} E_4 \left(\frac{1}{e_2} E_3 \left(\frac{1}{e_1} E_2 \left(\frac{1}{e_0} E_1 \right) \right) \right)$
  - $\frac{1}{e_3} E_4 \frac{1}{e_1} (\frac{1}{e_2} E_3 E_2) E_1)$

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal
  - $\frac{1}{e_3} E_4 \left(\frac{1}{e_2} E_3 \left(\frac{1}{e_1} E_2 \left(\frac{1}{e_0} E_1 \right) \right) \right)$
  - $\frac{1}{e_3} E_4 \frac{1}{e_1} (\frac{1}{e_2} E_3 E_2) E_1)$
  - $\frac{1}{e_2} (\frac{1}{e_3} E_4 E_3) (\frac{1}{e_1} E_2 E_1))$

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal
  - $\frac{1}{e_3}E_4\left(\frac{1}{e_2}E_3\left(\frac{1}{e_1}E_2\left(\frac{1}{e_0}E_1\right)\right)\right)$
  - $\frac{1}{e_3}E_4\frac{1}{e_1}(\frac{1}{e_2}E_3E_2)E_1)$
  - $\frac{1}{e_2}(\frac{1}{e_3}E_4E_3)(\frac{1}{e_1}E_2E_1))$
- Last formula would occur in the recursive Gauss-Jordan transform algorithm

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal
  - $\frac{1}{e_3}E_4\left(\frac{1}{e_2}E_3\left(\frac{1}{e_1}E_2\left(\frac{1}{e_0}E_1\right)\right)\right)$
  - $\frac{1}{e_3}E_4\frac{1}{e_1}(\frac{1}{e_2}E_3E_2)E_1)$
  - $\frac{1}{e_2}(\frac{1}{e_3}E_4E_3)(\frac{1}{e_1}E_2E_1))$
- Last formula would occur in the recursive Gauss-Jordan transform algorithm
- For an $A \in \mathbb{Z}^{n\times m}$ of rank r, FFGE costs

$$O(nmr^{\omega-2}\,\mathsf{M}(r(\log r + \log ||A||))$$

Incorporating matrix multiplication

- Slices can be combined in various orders; following all equal
  - $\frac{1}{e_3}E_4\left(\frac{1}{e_2}E_3\left(\frac{1}{e_1}E_2\left(\frac{1}{e_0}E_1\right)\right)\right)$
  - $\frac{1}{e_3}E_4\frac{1}{e_1}(\frac{1}{e_2}E_3E_2)E_1)$
  - $\frac{1}{e_2}(\frac{1}{e_3}E_4E_3)(\frac{1}{e_1}E_2E_1))$
- Last formula would occur in the recursive Gauss-Jordan transform algorithm
- For an $A \in \mathbb{Z}^{n\times m}$ of rank r, FFGE costs

$$O(nmr^{\omega-2}\,\mathsf{M}(r(\log r + \log ||A||))$$

- Gives solutions to all problems: **Rank**, **Rank Profile**, **Determinant**, **Inverse**, **Linear System Solving**, **Nullspace**

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF $\left[\begin{array}{c|c} (\det A)I_n & (\det A)A^{-1}b \end{array}\right]$

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF $\left[\begin{array}{c|c} (\det A)I_n & (\det A)A^{-1}b \end{array}\right]$
2. Chinese remaindering
   - Choose a bunch of primes p_*
     $\rightarrow$ in theory, for analysis, can choose $2, 3, 5, 7, \ldots$
     $\rightarrow$ in practice we choose word-size primes
   - Compute $\mathrm{Rem}(\mathsf{adj}(A)b, p_*)$ and $\mathrm{Rem}(\det A, p_*)$
   - Reconstruct $\det A$ and $(\det A)A^{-1}b$ using Chinese remaindering

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF $\left[\begin{array}{c|c} (\det A)I_n & (\det A)A^{-1}b \end{array}\right]$
2. Chinese remaindering
   - Choose a bunch of primes p_*
     $\rightarrow$ in theory, for analysis, can choose $2, 3, 5, 7, \ldots$
     $\rightarrow$ in practice we choose word-size primes
   - Compute $\mathrm{Rem}(\mathsf{adj}(A)b, p_*)$ and $\mathrm{Rem}(\det A, p_*)$
   - Reconstruct $\det A$ and $(\det A)A^{-1}b$ using Chinese remaindering

- Both of these methods cost $(n^{\omega+1} \log ||A||)^{1+o(1)}$

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF $\left[\begin{array}{c|c} (\det A)I_n & (\det A)A^{-1}b \end{array}\right]$
2. Chinese remaindering
   - Choose a bunch of primes p_*
     $\rightarrow$ in theory, for analysis, can choose $2, 3, 5, 7, \ldots$
     $\rightarrow$ in practice we choose word-size primes
   - Compute $\mathrm{Rem}(\mathsf{adj}(A)b, p_*)$ and $\mathrm{Rem}(\det A, p_*)$
   - Reconstruct $\det A$ and $(\det A)A^{-1}b$ using Chinese remaindering

- Both of these methods cost $(n^{\omega+1} \log ||A||)^{1+o(1)}$

But, we knew already over 30 years ago that we can do better:

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\ A \mid b\ \right]$ to RREF $\left[\ (\det A)I_n \mid (\det A)A^{-1}b\ \right]$
2. Chinese remaindering
   - Choose a bunch of primes p_*
     $\rightarrow$ in theory, for analysis, can choose $2, 3, 5, 7, \ldots$
     $\rightarrow$ in practice we choose word-size primes
   - Compute $\mathrm{Rem}(\mathsf{adj}(A)b, p_*)$ and $\mathrm{Rem}(\det A, p_*)$
   - Reconstruct $\det A$ and $(\det A)A^{-1}b$ using Chinese remaindering

- Both of these methods cost $(n^{\omega+1} \log ||A||)^{1+o(1)}$

But, we knew already over 30 years ago that we can do better:

- Dixon (1982): $O(n^3(\log n + \log ||A||)^2)$ LV for $A^{-1}b$

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF $\left[\begin{array}{c|c} (\det A) I_n & (\det A) A^{-1} b \end{array}\right]$
2. Chinese remaindering
   - Choose a bunch of primes p_*
     $\rightarrow$ in theory, for analysis, can choose $2, 3, 5, 7, \ldots$
     $\rightarrow$ in practice we choose word-size primes
   - Compute $\mathrm{Rem}(\mathsf{adj}(A)b, p_*)$ and $\mathrm{Rem}(\det A, p_*)$
   - Reconstruct $\det A$ and $(\det A)A^{-1}b$ using Chinese remaindering

- Both of these methods cost $(n^{\omega+1} \log ||A||)^{1+o(1)}$

But, we knew already over 30 years ago that we can do better:

- Dixon (1982): $O(n^3(\log n + \log ||A||)^2)$ LV for $A^{-1}b$
- Kaltofen (1992): $(n^{3.5} \log ||A||)^{1+o(1))}$ LV for $\det A$

Summary: Classical methods for linear solving

1. Fraction free gaussian elimination
   - Transform $\left[\begin{array}{c|c} A & b \end{array}\right]$ to RREF $\left[\begin{array}{c|c} (\det A)I_n & (\det A)A^{-1}b \end{array}\right]$
2. Chinese remaindering
   - Choose a bunch of primes p_*
     $\rightarrow$ in theory, for analysis, can choose $2, 3, 5, 7, \ldots$
     $\rightarrow$ in practice we choose word-size primes
   - Compute $\mathrm{Rem}(\mathsf{adj}(A)b, p_*)$ and $\mathrm{Rem}(\det A, p_*)$
   - Reconstruct $\det A$ and $(\det A)A^{-1}b$ using Chinese remaindering

- Both of these methods cost $(n^{\omega+1} \log ||A||)^{1+o(1)}$

But, we knew already over 30 years ago that we can do better:

- Dixon (1982): $O(n^3(\log n + \log ||A||)^2)$ LV for $A^{-1}b$
- Kaltofen (1992): $(n^{3.5} \log ||A||)^{1+o(1))}$ LV for $\det A$

Our goal: Reduce cost to $(n^{\omega} \log ||A||)^{1+o(1)}$