More on Hermite Normal Form

(focus on integer matrices)

George Labahn and Arne Storjohann

Cheriton School of Computer Science
University of Waterloo

Recent Trends in Computer Algebra, Institut Henri Poincaré,

September 2023

Computational Algebra and Geometry: A special issue in memory and honor of Agnes Szanto

Guest editors:

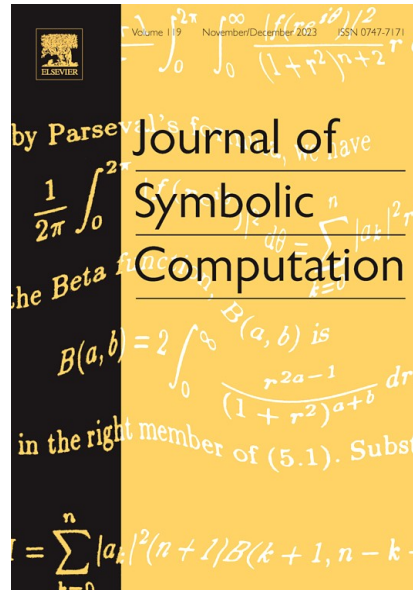
Carlos D'Andrea (Barcelona)

Hoon Hong, (North Carolina)

Evelyne Hubert (Sophia-Antipolis)

Teresa Krick (Buenos Aires, chair)

Submission Deadline: December 31, 2023



Outline

Hermite Normal Form

Recall: Hermite Normal Form

$A \in \mathbb{Z}^{m \times n}$, an integer matrix, full column rank. Hermite form of A:

$$UA = H = \begin{bmatrix} h_1 & h_{12} & \cdots & h_{1n} \\ & h_2 & \cdots & h_{2n} \\ & & \ddots & \vdots \\ & & & h_n \\ & & & \\ & & & \end{bmatrix}$$

- has all entries nonnegative
- in each column: $h_{ij} < h_j$
- A left equivalent to H, there exists U with $UA = H$
- $U \in \mathbb{Z}^{m \times m}$ unimodular, and represents the integer row operations.

Example 1:

$$\underset{U}{\begin{bmatrix} 235 & 454 & 256 & -84 & -269 & -577 \\ 194 & 374 & 209 & -70 & -221 & -473 \\ 704 & 1360 & 768 & -251 & -806 & -1730 \\ 461 & 890 & 501 & -165 & -527 & -1130 \\ 1241 & 2397 & 1352 & -443 & -1420 & -3047 \\ 1268 & 2450 & 1384 & -452 & -1452 & -3117 \end{bmatrix}} \underset{A}{\begin{bmatrix} -8 & -1 & 5 & 1 & 6 & 0 \\ 2 & -3 & -8 & -3 & 2 & -1 \\ -5 & -4 & -5 & 9 & -4 & 4 \\ 2 & -6 & -1 & -8 & 9 & -7 \\ -9 & 5 & -5 & -6 & 2 & -7 \\ 0 & -6 & -4 & 6 & 0 & -8 \end{bmatrix}} = \underset{H}{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 7657 \\ & 1 & 0 & 1 & 4 & 6283 \\ & & 1 & 0 & 1 & 22951 \\ & & & 2 & 3 & 14998 \\ & & & & 5 & 40428 \\ & & & & & 41350 \end{bmatrix}}$$

where we can easily check that $UA = H$ and that det $U = -1$.

Example 2:

$$\underset{U}{\begin{bmatrix} -25 & -160 & 109 & 128 \\ -46 & -295 & 201 & 236 \\ -25 & -156 & 107 & 125 \\ -65 & -419 & 285 & 335 \end{bmatrix}} \underset{A}{\begin{bmatrix} -13 & 27 & 0 & -21 \\ 10 & 30 & 15 & 0 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}} = \underset{H}{\begin{bmatrix} 1 & 0 & 3 & 42 \\ & 3 & 6 & 75 \\ & & 15 & 45 \\ & & & 105 \end{bmatrix}}$$

Integer matrix triangularization

- Over a field : work with bases of vector space of rows
- Over integers: work with bases of integer module of rows

  Triangular basis for the lattice generated by $\mathbb{Z}$-linear combinations of rows of an integer matrix A

Integer matrix triangularization

- Over a field : work with bases of vector space of rows
- Over integers: work with bases of integer module of rows

  Triangular basis for the lattice generated by $\mathbb{Z}$-linear combinations of rows of an integer matrix A
- Lattice is invariant under unimodular row operations
  1. Add an integer multiple of one row to a different row
  2. Multiply a row by -1
  3. Swap two rows
- Obtainable using "Gaussian elimination" with integer row operations

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}}$$

- Next row operation: Swap rows 1 and 3

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & -7 & 6 \\ -6 & 3 & 3 \\ -4 & -4 & -1 \end{bmatrix}}$$

- Next row operation: Add 3 times row 1 to row 2

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 3 \\ 1 & 0 & 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & -7 & 6 \\ & -18 & 21 \\ -4 & -4 & -1 \end{bmatrix}}$$

- Next row operation: Add 2 times row 1 to row 3

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 3 \\ 1 & 0 & 2 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & -7 & 6 \\ & -18 & 21 \\ & -18 & 11 \end{bmatrix}}$$

- Next row operation: Subtract row 2 from row 3

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 3 \\ 1 & -1 & -1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & -7 & 6 \\ & -18 & 21 \\ & & -10 \end{bmatrix}}$$

- Next row operation: Multiply row 2 by -1

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & -3 \\ 1 & -1 & -1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & -7 & 6 \\ & 18 & -21 \\ & & -10 \end{bmatrix}}$$

- Next row operation: Add row 2 to row 1

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} 0 & -1 & -2 \\ 0 & -1 & -3 \\ 1 & -1 & -1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & 11 & -15 \\ & 18 & -21 \\ & & -10 \end{bmatrix}}$$

- Next row operations: Put last column in correct form

Integer matrix triangularization

Example:

- Triangularize A using integer row operations
- Record row operations in U, initially set to I_n

$$\underset{U}{\begin{bmatrix} -2 & 1 & 0 \\ -3 & 2 & 0 \\ -1 & 1 & 1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -4 & -4 & -1 \\ -6 & 3 & 3 \\ 2 & -7 & 6 \end{bmatrix}} = \underset{UA}{\begin{bmatrix} 2 & 11 & 5 \\ & 18 & 9 \\ & & 10 \end{bmatrix}}$$

- U is unimodular: $\det U = -1$
- $H := UA$ is the *Hermite form* for A

Computing a Hermite Form: A Bit of History

Elimination based methods

- Extended gcd: given $a, b \in \mathbb{Z}$ compute $s, t \in \mathbb{Z}$ such that

$$sa + tb = g = \gcd(a, b) \iff \begin{bmatrix} s & t \\ -b/g & a/g \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ \end{bmatrix}$$

- "Gaussian elimination" via unimodular transformations

$$\begin{bmatrix} s & & & t & \\ & 1 & & & \\ & & 1 & & \\ -b/g & & & a/g & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} a & * \\ & * \\ & * \\ b & * \\ * & * \end{bmatrix} = \begin{bmatrix} g & * \\ & * \\ & * \\ & * \\ * & * \end{bmatrix}$$

Example

$$A$$
$$\begin{bmatrix} -13 & 27 & 0 & -21 \\ 10 & 30 & 15 & 0 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}$$

Example

$$A = \begin{bmatrix} -13 & 27 & 0 & -21 \\ 10 & 30 & 15 & 0 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}$$

- Extended Euclidean Algorithm gives:

$$\begin{aligned} 3(-13) + 4(10) &= 1 \\ 10(-13) + 13(10) &= 0 \end{aligned}$$

Example

$$\underset{U_1}{\begin{bmatrix} 3 & 4 & 0 & 0 \\ 10 & 13 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -13 & 27 & 0 & -21 \\ 10 & 30 & 15 & 0 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}} = \underset{A_1}{\begin{bmatrix} 1 & 201 & 60 & -63 \\ & 660 & 195 & -210 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}}$$

- Extended Euclidean Algorithm gives:

$$\begin{aligned} 3(-13) + 4(10) &= 1 \\ 10(-13) + 13(10) &= 0 \end{aligned}$$

Example

$$\underset{U_4}{\begin{bmatrix} 3 & 4 & 0 & 0 \\ 10 & 13 & 0 & 0 \\ 60 & 80 & 1 & 0 \\ -81 & -108 & 0 & 1 \end{bmatrix}} \underset{A}{\begin{bmatrix} -13 & 27 & 0 & -21 \\ 10 & 30 & 15 & 0 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}} = \underset{A_4}{\begin{bmatrix} 1 & 201 & 60 & -63 \\ & 660 & 195 & -210 \\ & 4035 & 1215 & -1275 \\ & -5397 & -1614 & 1710 \end{bmatrix}}$$

- Extended Euclidean Algorithm used for elimination:

$$\begin{aligned} 3(-13) + 4(10) &= 1 \\ 10(-13) + 13(10) &= 0 \end{aligned}$$

Example

$$\underset{U}{\begin{bmatrix} -25 & -160 & 109 & 128 \\ -46 & -295 & 201 & 236 \\ -25 & -156 & 107 & 125 \\ -65 & -419 & 285 & 335 \end{bmatrix}} \quad \underset{A}{\begin{bmatrix} -13 & 27 & 0 & -21 \\ 10 & 30 & 15 & 0 \\ -20 & 15 & 15 & -15 \\ 27 & 30 & 6 & 9 \end{bmatrix}} = \underset{H}{\begin{bmatrix} 1 & 0 & 3 & 42 \\ & 3 & 6 & 75 \\ & & 15 & 45 \\ & & & 105 \end{bmatrix}}$$

- Extended Euclidean Algorithm used for elimination:

$$\begin{aligned} 3(-13) + 4(10) &= 1 \\ 10(-13) + 13(10) &= 0 \end{aligned}$$

- Issue: Exponential growth of intermediate integers

A Bit of History

Hermite form of full column rank $A \in \mathbb{Z}^{n \times m}$.

- Counting bit operations, assuming standard matrix multiplication.

- Kannan & Bachem (79)
  - polynomial time
- Chou and Collins (82)
  - $O\tilde{}(n^6 \log ||A||)$
- Domich *et al. (87), Iliopoulos (89), Hafner & McCurley (91)*
  - $O\tilde{}(nm^3 \log ||A||)$
- *Storjohann & Labahn (96), Storjohann (00)*
  - $O\tilde{}(nm^3 \log ||A||)$ *with transformation* U
- *Sims (84), Havas, Majewski & Matthews (98)*
  - *application of lattice reduction (LLL)*

Elimination based methods with potential expression swell

1. Triangularize column by column

$$\left[\begin{array}{ccccc} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right] \xrightarrow{1,2,3} \left[\begin{array}{ccc|cc} h_1 & * & * & * & * \\ & h_2 & * & * & * \\ & & h_3 & * & * \\ \hline & & & * & * \\ & & & * & * \end{array}\right] \xrightarrow{4,5} \left[\begin{array}{ccc|cc} h_1 & * & * & * & * \\ & h_2 & * & * & * \\ & & h_3 & * & * \\ \hline & & & h_4 & * \\ & & & & h_5 \end{array}\right]$$

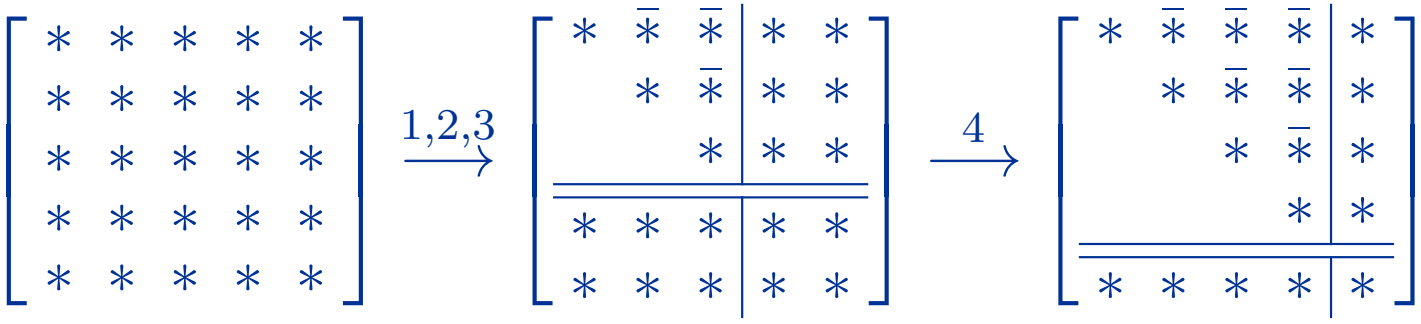
2. Reduce off diagonals column by column

$$\left[\begin{array}{ccccc} h_1 & * & * & * & * \\ & h_2 & * & * & * \\ & & h_3 & * & * \\ & & & h_4 & * \\ & & & & h_5 \end{array}\right] \xrightarrow{1,2,3} \left[\begin{array}{ccc|cc} h_1 & \bar{*} & \bar{*} & * & * \\ & h_2 & \bar{*} & * & * \\ & & h_3 & * & * \\ \hline & & & h_4 & * \\ & & & & h_5 \end{array}\right] \xrightarrow{4,5} \left[\begin{array}{ccc|cc} h_1 & \bar{*} & \bar{*} & \bar{*} & \bar{*} \\ & h_2 & \bar{*} & \bar{*} & \bar{*} \\ & & h_3 & \bar{*} & \bar{*} \\ \hline & & & h_4 & \bar{*} \\ & & & & h_5 \end{array}\right]$$

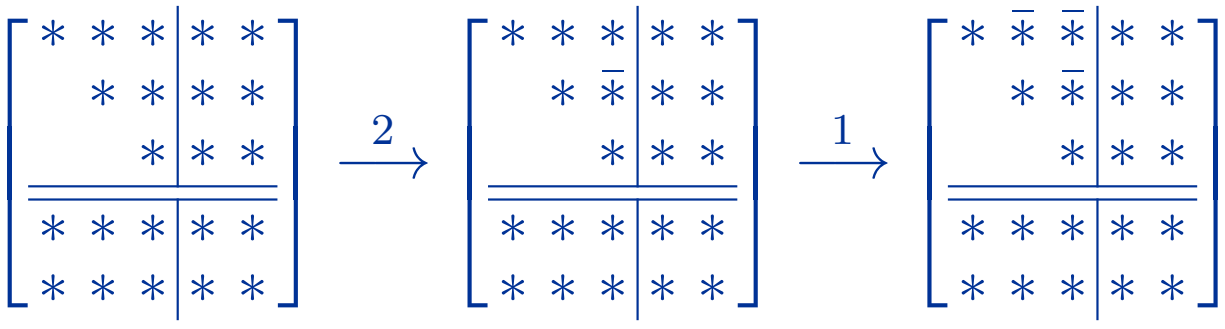
- Fang & Havas (ISSAC 97): Exponential growth in bit length

Elimination based methods with controlled growth

1. Triangularize row by row



2. During triangularization reduce off diagonals row by row



- Chou & Collins (86): $O\tilde{}(n^6 \log ||A||)$ bit operations

Modulo determinant methods

Let $A \in \mathbb{Z}^{n \times n}$ be nonsingular.

1. Compute $d = \det A$.
2. Triangularize working modulo d
3. "Fix up" diagonal entries in using extended gcd computations
4. Reduce off diagonal entries working modulo d

Cost is $O(n^3)$ operations modulo d: $O\tilde{}(n^4 \log ||A||)$ bit operations

why would this work?

Suppose $d = \det A$

- If $UA = H$ with HNF $A = H$ then

  Easy to see that $\text{HNF} \begin{bmatrix} A & 0 \\ dI_n & I_n \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & I_n \end{bmatrix}$.

why would this work?

Suppose $d = \det A$

- If $UA = H$ with HNF $A = H$ then

  Easy to see that $\mathrm{HNF} \begin{bmatrix} A & 0 \\ dI_n & I_n \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & I_n \end{bmatrix}$.

  since $\begin{bmatrix} U & 0 \\ -\mathrm{adj}(A) & I_n \end{bmatrix} \begin{bmatrix} A & 0 \\ dI_n & I_n \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & I_n \end{bmatrix}$

why would this work?

Suppose $d = \det A$

- If $UA = H$ with HNF $A = H$ then

  Easy to see that $\mathrm{HNF} \begin{bmatrix} A & 0 \\ dI_n & I_n \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & I_n \end{bmatrix}$.

  since $\begin{bmatrix} U & 0 \\ -\mathrm{adj}(A) & I_n \end{bmatrix} \begin{bmatrix} A & 0 \\ dI_n & I_n \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & I_n \end{bmatrix}$

  and so $\begin{bmatrix} U & 0 \\ -\mathrm{adj}(A) & I_n \end{bmatrix} \begin{bmatrix} A \\ dI_n \end{bmatrix} = \begin{bmatrix} H \\ 0 \end{bmatrix}$

- Implies we can compute HNF of $\begin{bmatrix} A \\ dI_n \end{bmatrix}$ working modulo d

Example: Domich et al algorithm

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

Example: Domich et al algorithm

$$
\begin{bmatrix}
h_{11} & b_{12} & \cdots & \cdots & b_{1n} \\
 & b_{22} & \cdots & \cdots & b_{2n} \\
 & \vdots & & & \vdots \\
 & b_{n2} & \cdots & \cdots & b_{nn} \\
d & * & * & * & * \\
 & d & & & \\
 & & \ddots & & \\
 & & & \ddots & \\
 & & & & d
\end{bmatrix}
$$

Example: Domich et al algorithm

$$
\begin{bmatrix}
h_{11} & b_{12} & \cdots & \cdots & b_{1n} \\
 & b_{22} & \cdots & \cdots & b_{2n} \\
 & \vdots & & & \vdots \\
 & b_{n2} & \cdots & \cdots & b_{nn} \\
 & * & * & * & * \\
 & d & & & \\
 & & \ddots & & \\
 & & & \ddots & \\
 & & & & d
\end{bmatrix}
$$

Example: Domich et al algorithm

$$
\begin{bmatrix}
h_{11} & h_{12} & c_{13} & \cdots & c_{1n} \\
 & h_{22} & c_{23} & \cdots & c_{2n} \\
 & & \vdots & & \vdots \\
 & & c_{n3} & \cdots & c_{nn} \\
 & & * & * & * \\
 & d & * & * & * \\
 & & d & & \\
 & & & \ddots & \\
 & & & & d
\end{bmatrix}
$$

Example: Domich et al algorithm

$$\begin{bmatrix} h_{11} & h_{12} & c_{13} & \cdots & c_{1n} \\ & h_{22} & c_{23} & \cdots & c_{2n} \\ & & \vdots & & \vdots \\ & & c_{n3} & \cdots & c_{nn} \\ & & * & * & * \\ & & * & * & * \\ & & d & & \\ & & & \ddots & \\ & & & & d \end{bmatrix}$$

Example: Domich et al algorithm

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1n} \\ & h_{22} & h_{23} & \cdots & h_{2n} \\ & & & \ddots & \vdots \\ & & & & h_{nn} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

Quick aside: Smith normal form

Given $A \in \mathbb{Z}^{n \times n}$ a nonsingular integer matrix.

The Smith normal form

- $S = \operatorname{diag}(s_1, s_2, \ldots, s_n) \in \mathbb{Z}^{n \times n}$.
- $s_1 \,|s_2\, | \ldots |\, s_n$. (invariant factors)

Quick aside: Smith normal form

Given $A \in \mathbb{Z}^{n \times n}$ a nonsingular integer matrix.

The Smith normal form

- $S = \operatorname{diag}(s_1, s_2, \ldots, s_n) \in \mathbb{Z}^{n \times n}$.
- $s_1 \,|s_2\, | \ldots |\, s_n$. (invariant factors)

$$\underset{A}{\begin{bmatrix} -8 & -1 & 5 & 1 & 6 & 0 \\ 2 & -3 & -8 & -3 & 2 & -1 \\ -5 & -4 & -5 & 9 & -4 & 4 \\ 2 & -6 & -1 & -8 & 9 & -7 \\ -9 & 5 & -5 & -6 & 2 & -7 \\ 0 & -6 & -4 & 6 & 0 & -8 \end{bmatrix}} \rightarrow \underset{S}{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 2 & \\ & & & & & 206750 \end{bmatrix}}$$

Quick aside: Smith normal form

Given $A \in \mathbb{Z}^{n \times n}$ a nonsingular integer matrix.

The Smith normal form

- $S = \mathrm{diag}(s_1, s_2, \ldots, s_n) \in \mathbb{Z}^{n \times n}$.
- $s_1 \,|s_2\, | \ldots |\, s_n$. (invariant factors)

$$\underset{A}{\begin{bmatrix} -8 & -1 & 5 & 1 & 6 & 0 \\ 2 & -3 & -8 & -3 & 2 & -1 \\ -5 & -4 & -5 & 9 & -4 & 4 \\ 2 & -6 & -1 & -8 & 9 & -7 \\ -9 & 5 & -5 & -6 & 2 & -7 \\ 0 & -6 & -4 & 6 & 0 & -8 \end{bmatrix}} \rightarrow \underset{S}{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 2 & \\ & & & & & 206750 \end{bmatrix}}$$

- S obtained using unimodular row and column operations.
- typically $UAV = S$ or $AV = WS$

Some Oddities

Oddity 1: Faster algorithms for polynomial HNF:

Labahn-Neiger-Zhou (2017)	$O^{\sim}(n^{\omega}s)$	Deterministic

- s minimum of the average of column/row degrees

Oddity 2: Faster algorithms for SNF (with multipliers)

Birmpilis, Labahn, Storjohann (2023)	$O^{\sim}(n^{\omega}\log\|\|A\|\|)$	LV

- Complexity is given without the extra $\log n$ and $\log\log||A||$ factors.

Hermite form with arbitrary shape and rank

The Hermite form: arbitrary shape and rank input

- Example: 6×9 input matrix with rank 3

$$A \xrightarrow{HNF} \begin{bmatrix} h_1 & * & * & \bar{*} & \bar{*} & * & * & * & * \\ & & & h_2 & \bar{*} & * & * & * & * \\ & & & & h_3 & * & * & * & * \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}$$

- Column rank profile of A is $[1, 4, 5]$
- Define Hermite basis $\bar{H} = \begin{bmatrix} h_1 & \bar{*} & \bar{*} \\ & h_2 & \bar{*} \\ & & h_3 \end{bmatrix}$

The Hermite form: Full row rank

- Example: 3×9 input matrix with rank 3

$$A \xrightarrow{HNF} \begin{bmatrix} h_1 & * & * & \bar{*} & \bar{*} & * & * & * & * \\ & & & h_2 & \bar{*} & * & * & * & * \\ & & & & h_3 & * & * & * & * \end{bmatrix}$$

- Let $\bar{A} = \left[\begin{array}{c|c} A_1 & A_2 \end{array}\right] = AC$, perm. C based on rank profile

$$\bar{A} \xrightarrow{HNF} \left[\begin{array}{ccc||cccccc} h_1 & \bar{*} & \bar{*} & * & * & * & * & * & * \\ & h_2 & \bar{*} & * & * & * & * & * & * \\ & & h_3 & * & * & * & * & * & * \end{array}\right]$$

The Hermite form: Full row rank

- Example: 3×9 input matrix with rank 3

$$A \longrightarrow \begin{bmatrix} h_1 & * & * & \bar{*} & \bar{*} & * & * & * & * \\ & & & h_2 & \bar{*} & * & * & * & * \\ & & & & h_3 & * & * & * & * \end{bmatrix}$$

- Let $\bar{A} = \left[\begin{array}{c|c} A_1 & A_2 \end{array}\right] = AC$, perm. C based on rank profile

$$\bar{A} \quad \stackrel{HNF}{\longrightarrow} \quad \left[\begin{array}{c|c} \bar{H} & \bar{H}A_1^{-1}A_2 \end{array}\right]$$

The Hermite form: Arbitrary rank

- Input $A \in \mathbb{Z}^{n \times m}$ of arbitrary rank r

Let

$$\left[\begin{array}{c||c} A_{11} & A_{12} \\ \hline\hline A_{21} & A_{22} \end{array}\right] = RAC$$

where

- C a permutation based on column rank profile
- R *any* row permutation such that $A_{11} \in \mathbb{Z}^{r \times r}$ is nonsingular

Then

$$\left[\begin{array}{c||c} A_{11} & A_{12} \\ \hline\hline A_{21} & A_{22} \end{array}\right] \quad \xrightarrow{HNF} \quad \left[\begin{array}{c||c} \bar{H} & \bar{H}A_{11}^{-1}A_{12} \\ \hline\hline & \end{array}\right]$$

Computing the transforming matrix

What about the transforming matrix? $UA = H$

Case $A \in \mathbb{Z}^{n \times n}$ nonsingular

- U is unique and can be found using linear solving: $U = HA^{-1}$
- also note that $\left[\begin{array}{c|c} A & I_n \end{array}\right] \overset{HNF}{\longrightarrow} \left[\begin{array}{c|c} H & U \end{array}\right]$

Case $A \in \mathbb{Z}^{n \times m}$ of full column rank

- use rank computation to reduce to nonsingular case

$$A \xrightarrow{(1)} \left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right] \xrightarrow{(2)} \left[\begin{array}{c|c} A_1 & \\ \hline A_2 & I_{n-m} \end{array}\right] \overset{HNF}{\longrightarrow} \left[\begin{array}{c|c} \bar{H} & U_1 \\ \hline & U_2 \end{array}\right]$$

Then

$$U = \left[\begin{array}{c|c} \bar{H} - U_1 A_2 & U_1 \\ \hline -U_2 A_2 & U_2 \end{array}\right] \left[\begin{array}{c|c} A_1^{-1} & \\ \hline & I_{n-m} \end{array}\right]$$

Computing a transforming matrix

- recursive kernel basis computation of Hafner & McCurley (91)
- example of $8m \times m$ input matrix $(n = 8m)$
- kernel produced has at most $(n/m)\log_2(n/m)$ nonzero blocks

$$\begin{bmatrix} I & & & & & & & \\ & I & & & & & & \\ & & I & & & & & \\ & & & I & & & & \\ & & & & I & & & \\ & & & & & I & & \\ & & & & & & I & \\ & & & & & & & I \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}$$

Computing a transforming matrix

- recursive kernel basis computation of Hafner & McCurley (91)
- example of $8m \times m$ input matrix $(n = 8m)$
- kernel produced has at most $(n/m)\log_2(n/m)$ nonzero blocks

$$\begin{bmatrix} * & * & & & & & & \\ * & * & & & & & & \\ & & * & * & & & & \\ & & * & * & & & & \\ & & & & * & * & & \\ & & & & * & * & & \\ & & & & & & * & * \\ & & & & & & * & * \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} = \begin{bmatrix} b_1 \\ \\ b_2 \\ \\ b_3 \\ \\ b_4 \\ \end{bmatrix}$$

Computing a transforming matrix

- recursive kernel basis computation of Hafner & McCurley (91)
- example of $8m \times m$ input matrix $(n = 8m)$
- kernel produced has at most $(n/m)\log_2(n/m)$ nonzero blocks

$$\begin{bmatrix} * & * & * & * & & & & \\ * & * & & & & & & \\ * & * & * & * & & & & \\ & & * & * & & & & \\ & & & & * & * & * & * \\ & & & & * & * & & \\ & & & & * & * & * & * \\ & & & & & & * & * \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} = \begin{bmatrix} c_1 \\ \\ \\ \\ c_2 \\ \\ \\ \end{bmatrix}$$

Computing a transforming matrix

- recursive kernel basis computation of Hafner & McCurley (91)
- example of $8m \times m$ input matrix $(n = 8m)$
- kernel produced has at most $(n/m)\log_2(n/m)$ nonzero blocks

$$\left[\begin{array}{cccccccc} * & * & * & * & * & * & * & * \\ \hline * & * & & & & & & \\ * & * & * & * & & & & \\ & & * & * & & & & \\ * & * & * & * & * & * & * & * \\ & & & & * & * & & \\ & & & & * & * & * & * \\ & & & & & & * & * \end{array}\right] \left[\begin{array}{c} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{array}\right] = \left[\begin{array}{c} d_1 \\ \hline \\ \\ \\ \\ \\ \\ \\ \end{array}\right]$$

The transforming matrix: a closer view

- Let $A \in \mathbb{Z}^{n \times m}$ have full column rank
- Let $U \in \mathbb{Z}^{n \times n}$ be unimodular such that $UA = H$

$$\left[\begin{array}{c} U \\ \hline E \\ \hline \\ K \\ \\ \end{array}\right] \left[\begin{array}{c} A \\ \hline * \\ \hline \\ * \\ \\ \end{array}\right] = \left[\begin{array}{c} \bar{H} \\ \hline \\ \\ \\ \end{array}\right]$$

with $E \in \mathbb{Z}^{m \times n}$ and $K \in \mathbb{Z}^{(n-m) \times n}$ satisfying

1. $EA = \bar{H}$
2. $KA = 0_{(n-m) \times m}$

Other choices

- Let $A \in \mathbb{Z}^{n \times m}$ have full column rank
- Let $U \in \mathbb{Z}^{n \times n}$ be unimodular such that $UA = H$

$$\begin{array}{c} U \\ \left[\begin{array}{c} E \\ \hline \\ K_1 \\ \\ \end{array}\right] \end{array} \begin{array}{c} A \\ \left[\begin{array}{c} * \\ \hline \\ * \end{array}\right] \end{array} = \begin{array}{c} \\ \left[\begin{array}{c} \bar{H} \\ \hline \\ \end{array}\right] \end{array}$$

with $E_1 \in \mathbb{Z}^{m \times n}$ and $K_1 \in \mathbb{Z}^{(n-m) \times n}$ satisfying

1. $K_1 = VK$ with V unimodular

Other choices

- Let $A \in \mathbb{Z}^{n \times m}$ have full column rank
- Let $U \in \mathbb{Z}^{n \times n}$ be unimodular such that $UA = H$

$$\overset{U}{\left[\begin{array}{c} E_1 \\ \hline \\ K_1 \\ \\ \end{array}\right]} \overset{A}{\left[\begin{array}{c} * \\ \hline \\ * \\ \\ \end{array}\right]} = \left[\begin{array}{c} \bar{H} \\ \hline \\ \\ \\ \end{array}\right]$$

with $E_1 \in \mathbb{Z}^{m \times n}$ and $K_1 \in \mathbb{Z}^{(n-m) \times n}$ satisfying

1. $K_1 = VK$ with V unimodular
2. $E_1 = E - WK$, with $W \in \mathbb{Z}^{(n-m) \times (n-m)}$

Summary: From Arne's Thesis

Input $A \in \mathbb{Z}^{n \times m}$ of rank r.

Summary: From Arne's Thesis

Input $A \in \mathbb{Z}^{n \times m}$ of rank r. Let $\beta = (\sqrt{r}||A||)^r$

$$\begin{bmatrix} U \\ E \\ \hline \\ K \\ \\ \end{bmatrix} \begin{bmatrix} A \\ * \\ \hline \\ * \\ \\ \end{bmatrix} = \begin{bmatrix} H \\ * \\ \hline \\ \\ \\ \end{bmatrix}$$

We can compute U, H in $O\tilde{}(nmr^2 \log ||A||)$ bit operations with

- $||H|| \leq \beta$
- $||E|| \leq \beta$
- $||K|| \leq r\beta^2$

Example: Using lattice reduction

- Sims (84): Use LLL after the fact to reduce size of U
- Example for random 8×4 input

```
>RandomMatrix(8,4);
```

$$A = \begin{bmatrix} 50 & -38 & -93 & 8 \\ 10 & -18 & -76 & 69 \\ -16 & 87 & -72 & 99 \\ -9 & 33 & -2 & 29 \\ -50 & -98 & -32 & 44 \\ -22 & -77 & -74 & 92 \\ 45 & 57 & -4 & -31 \\ -81 & 27 & 27 & 67 \end{bmatrix}$$

Example: Using lattice reduction

- Sims (84): Use LLL after the fact to reduce size of U
- Example for random 8×4 input

```
>HermiteForm(A,output=['U']);
```

$$U = \left[\begin{array}{c} E \\ \hline K \end{array}\right] = \left[\begin{array}{rrrrr} 9086206 & -8629726 & -5060749 & 29822997 & 361156 \\ 8363956 & -7943761 & -4658477 & 27452408 & 332448 \\ 2467635 & -2343664 & -1374400 & 8099340 & 980829 \\ 3307464 & -3141301 & -1842160 & 10855850 & 131464 \\ \hline 9507196 & -9029566 & -5295228 & 31204782 & 377889 \\ 9199746 & -8737563 & -5123987 & 30195660 & 365669 \\ 6367234 & -6047352 & -3546362 & 20898713 & 253083 \\ 3264133 & -3100146 & -1818026 & 10713625 & 129741 \end{array}\right]$$

Example: Using lattice reduction

- Sims (84): Use LLL after the fact to reduce size of U
- Example for random 8×4 input

```
>HermiteForm(A,output=['U']);
```

$$U = \left[\frac{E}{\mathrm{LLL}(K)}\right] = \left[\begin{array}{rrrrr} 9086206 & -8629726 & -5060749 & 29822997 & 361156 \\ 8363956 & -7943761 & -4658477 & 27452408 & 332448 \\ 2467635 & -2343664 & -1374400 & 8099340 & 980829 \\ 3307464 & -3141301 & -1842160 & 10855850 & 131464 \\ \hline 11 & -80 & -2 & 50 & 18 \\ 2 & 26 & -26 & 69 & 43 \\ -41 & 36 & 16 & 22 & 41 \\ 12 & 1 & 15 & 47 & -20 \end{array}\right]$$

Example: Using lattice reduction

- Sims (84): Use LLL after the fact to reduce size of U
- Example for random 8×4 input

```
>HermiteForm(A,output=['U'],method='integer[reduced]')
```

$$U = \left[\frac{E + M\,\mathsf{LLL}(K)}{\mathsf{LLL}(K)}\right] = \left[\begin{array}{cccccccc} 0 & 4 & -3 & 59 & 21 & -18 & -8 & -18 \\ -18 & 24 & 12 & 1 & 8 & -25 & -31 & -26 \\ 3 & 5 & -3 & -16 & -7 & 1 & 2 & 10 \\ -21 & -57 & 24 & -18 & 9 & 48 & 33 & -23 \\ \hline 11 & -80 & -2 & 50 & 18 & 58 & 65 & 1 \\ 2 & 26 & -26 & 69 & 43 & -24 & 47 & 8 \\ -41 & 36 & 16 & 22 & 41 & -36 & -9 & -47 \\ 12 & 1 & 15 & 47 & -20 & -31 & -101 & -36 \end{array}\right]$$