(Joint work with G. Havas and B. Majewski – appeared in Experimental Mathematics)

**CENTRAL PROBLEM:**

If $d_1, \ldots, d_m, \ m \geq 2$, are nonzero integers, find integers $x_1, \ldots, x_m$ such that
\[ d = \gcd(d_1, \ldots, d_m) = x_1d_1 + \cdots + x_md_m, \]
with $x_1^2 + \cdots + x_m^2$ small. We call $(x_1, \ldots, x_m)$ a multiplier vector.

Euclid’s algorithm solves the problem for $m = 2$.

Various algorithms (Jacobi 1868, Brun 1919) use integer row operations to convert
\[
\begin{bmatrix}
1 & \cdots & 0 & d_1 \\
0 & \cdots & 0 & d_2 \\
\vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 1 & d_m
\end{bmatrix}
\rightarrow
\begin{bmatrix}
b_{11} & \cdots & b_{1m} & 0 \\
b_{21} & \cdots & b_{2m} & 0 \\
\vdots & \ddots & \vdots & \vdots \\
b_{m1} & \cdots & b_{mm} & d
\end{bmatrix}
\]

Then $B = [b_{ij}]$ is unimodular and $b_m = (b_{m1}, \ldots, b_{mm})$ is a multiplier vector.

With $b_i = (b_{i1}, \ldots, b_{im})$, then
\[
\Lambda = \{ (x_1, \ldots, x_m) \in \mathbb{Z}^m | d_1x_1 + \cdots + d_mx_m = 0 \}
\]
is an $(m – 1)$-dimensional lattice in $\mathbb{Z}^m$ with basis $b_1, \ldots, b_{m-1}$. The lattice determinant $d(\Lambda)$ is given by
\[
d(\Lambda) = \frac{1}{d} \sqrt{(d_1^2 + \cdots + d_m^2)}.
\]
The general multiplier has the form
\[
b = b_m + y_1b_1 + \cdots + y_{m-1}b_{m-1},
\]
where $y_1, \ldots, y_{m-1} \in \mathbb{Z}$.

**PHILOSOPHY.** Try to find short basis vectors $b_1, \ldots, b_{m-1}$ for $\Lambda$ and integers $y_1, \ldots, y_{m-1}$ which make $||b||$ small.

(An idea which goes back to L. Babai – See Geometric algorithms and combinatorial optimization, M. Grötschel, L. Lovász, A. Schrijver, 139–150.)

---

**JACOBI’S ALGORITHM**

Iterative step ($m = 3$):
\[
(d_1, d_2, d_3) \rightarrow (d_2 \text{ mod } d_1, \ d_3 \text{ mod } d_1, d_1).
\]

**EXAMPLE:** $\gcd(4, 6, 9).

\[
\begin{bmatrix}
1 & 0 & 0 & 4 \\
0 & 1 & 0 & 6 \\
0 & 0 & 1 & 9
\end{bmatrix}
\rightarrow
\begin{bmatrix}
-1 & 1 & 0 & 2 \\
-2 & 0 & 1 & 1 \\
1 & 0 & 0 & 4
\end{bmatrix}
\rightarrow
\begin{bmatrix}
-2 & 0 & 1 & 1 \\
3 & -2 & 0 & 0 \\
-1 & 1 & 0 & 2
\end{bmatrix}
\rightarrow
\begin{bmatrix}
3 & -2 & 0 & 0 \\
3 & 1 & -2 & 0 \\
-2 & 0 & 1 & 1
\end{bmatrix}
\]

$\gcd(4, 6, 9) = 1$; multipliers $-2, 0, 1$.

---

**THE LLL REDUCED MATRIX**

Let $B$ be an $m \times n$ matrix of integers, with LI rows $b_1, \ldots, b_m$ over the rationals and Gram–Schmidt basis $b_1^*, \ldots, b_m^*$, where
\[
b_1^* = b_1, \quad b_k^* = b_k - \sum_{j=1}^{k-1} \mu_{kj}b_j^*
\]
and $\mu_{kj} = \frac{b_k^* \cdot b_j^*}{b_j^* \cdot b_j^*}$ for $1 \leq j < k \leq m$.

The lattice basis $b_1, \ldots, b_m$ is reduced if
\[
(i) \quad ||\mu_{kj}|| \leq 1/2 \text{ for } 1 \leq j < k \leq m
\]
\[
(ii) \quad ||b_k||^2 \geq (\alpha - ||b_{k-1}||^2)||b_{k-1}||^2 \quad (C2)
\]
for $1 < k \leq m$. (Here $1/4 < \alpha \leq 1$)

$b_k$ is size-reduced if $||\mu_{kj}|| \leq 1/2$ for $1 \leq j < k$. 
THE LLL LATTICE BASIS REDUCTION ALGORITHM

Start with row \( k = 2 \) of \( B \). (Row 1 has to be nonzero.) Partially size–reduce \( b_k \) by

\[
b_k \rightarrow b_k - \lfloor \mu_{k-1} \rfloor b_{k-1},
\]

where \( \lfloor \theta \rfloor \) is the nearest integer symbol, with \( \lfloor \theta \rfloor = \theta - \frac{1}{2} \), if \( \theta \) is a half–integer.

If (C2) does not hold, we swap \( b_k \) and \( b_{k-1} \) and decrement \( k \).

Otherwise size–reduce \( b_k \) completely by

\[
b_k \rightarrow b_k - \lfloor \mu_j \rfloor b_j, \quad j = k - 2, \ldots, 1,
\]

then increment \( k \).

---

PSEUDO–CODE FOR THE LLL ALGORITHM (de Weger)

**INPUT**: \( m \times n \) integer matrix \( B \);

\[
m_1 := 1; \quad n_1 := 1; \quad \alpha := m_1/n_1; \quad D_0 := 1;
\]

for \( i = 1, \ldots, m \{
\]

\[
c_j := b_i; \quad \lambda_i := b_i \cdot c_j; \quad c_i := (D_i c_i - \lambda_i c_j) / D_{i-1};
\]

\[
D_i := (c_i - c_j) / D_{i-1}; \quad \lambda_k := D_i \mu_j / D_{i-1} \ast /
\]

\[
\}
\]

while \( k \leq m \{
\]

\[
Reduce(k, k - 1); \quad \lambda_k := \lambda_k / D_k \\
\}
\]

\[
\}
\]

**OUTPUT**: \( B \), whose rows are LLL reduced;

---

Reduce \((k, i)\)

if \( 2|\lambda_k| > D_k \)

\[
q := \lfloor \lambda_k / D_k \rfloor; \quad \lambda_k := \lambda_k - qD_k,
\]

else \( q := 0 \);

if \( q \neq 0 \{
\]

\[
b_k := b_k - q b_i; \quad \lambda_k := \lambda_k - q \lambda_j;
\]

\[
\}
\]

Swap \((k)\)

\[
b_k := b_k - b_{k-1};
\]

for \( j = 1, \ldots, k - 2 \)

\[
\lambda_k := \lambda_k - 1;
\]

for \( i = k + 1, \ldots, m \{
\]

\[
t := \lambda_{k+1} D_k - \lambda_k \lambda_{k-1}; \quad \lambda_{k+1} := (\lambda_{k+1} + \lambda_k D_{k-2}) / D_{k-1};
\]

\[
\lambda_k := t / D_{k-1};
\]

\[
D_{k-1} := (D_{k-2} D_k + \lambda_{k+1}^2) / D_{k-1};
\]

}
THE LARGE N EXTENDED GCD ALGORITHM

Let $D = [d_1, \ldots, d_m]^T$. Then if $N \geq N_0(D)$, under the LLL algorithm, the steps of the algorithm become identical and

$$
\begin{bmatrix}
1 & \cdots & 0 & N d_1 \\
0 & \cdots & 0 & N d_2 \\
\vdots & \cdots & \vdots & \vdots \\
0 & \cdots & 1 & N d_m
\end{bmatrix}
\rightarrow
\begin{bmatrix}
b_{11} & \cdots & b_{1m} & 0 \\
b_{21} & \cdots & b_{2m} & 0 \\
\vdots & \cdots & \vdots & \vdots \\
b_{m1} & \cdots & b_{mm} & Ng
\end{bmatrix},
$$

where $g = \gcd(d_1, \ldots, d_m)$.

The resulting multiplier vector $(b_{m1}, \ldots, b_{mm})$ is small in practice.

The large N extended gcd algorithm has the disadvantage that LLL has to be performed on matrices with large entries.

On studying what the sequence of operations for large $N$, one sees how to modify the LLL algorithm, starting instead with the matrix $[I_m | D]$, so as to perform essentially the same sequence of operations.

AN EXAMPLE OF THE LARGE N EXTENDED GCD ALGORITHM

Take $m = 2$ and $(d_1, d_2) = (2, 5), \alpha = 1$.

Let $B = \begin{bmatrix} 1 & 0 & 2N \\ 0 & 1 & 5N \end{bmatrix}$.

Then

$$
\mu_{21} = \frac{b_2 \cdot b_1}{b_1 \cdot b_1} = \frac{10N^2}{1 + 4N^2}
$$

Hence $2 < \mu_{21} < 5/2$ and $[\mu_{21}] = 2$.

We thus perform $R_2 \rightarrow R_2 - 2R_1$:

$$
B \rightarrow \begin{bmatrix} 1 & 0 & 2N \\ -2 & 1 & N \end{bmatrix}.
$$

Here $\|b_1\|^2 = 1 + 4N^2$, $\|b_2\|^2 = 5 + N^2$.

So if $N = 1$, $\|b_1\|^2 \leq \|b_2\|^2$ and $B$ is LLL reduced.

If $N > 1$, $\|b_1\|^2 > \|b_2\|^2$ and we swap rows:

$$
B \rightarrow \begin{bmatrix} -2 & 1 & N \\ 1 & 0 & 2N \end{bmatrix}.
$$

$$
\mu_{21} = -\frac{2 + 2N^2}{5 + N^2} = 2 - \frac{12}{5 + N^2},
$$

so $\frac{3}{2} < \mu_{21} < 2 \Rightarrow N \geq 5$.

$N = 2, 3, 4$: Here

$$
\mu_{21} = 2/3, 8/7, 10/7, \text{ so } [\mu_{21}] = 1
$$

and we perform $R_2 \rightarrow R_2 - R_1$:

$$
B \rightarrow \begin{bmatrix} -2 & 1 & N \\ 3 & -1 & N \end{bmatrix}
$$

and $B$ is LLL reduced.
\[ N \geq 5: \text{ Here} \]
\[
3/2 < \mu_{21} < 2, \text{ so } [\mu_{21}] = 2
\]
and we perform \( R_2 \rightarrow R_2 - 2R_1 \):
\[
B \rightarrow \begin{bmatrix}
-2 & 1 & | & N \\
5 & -2 & | & 0
\end{bmatrix}.
\]
Then \( \|b_2\|^2 = 29, \|b_1\|^2 = 5 + N^2 \geq 30 \), so we swap rows:
\[
B \rightarrow \begin{bmatrix}
5 & -2 & | & 0 \\
-2 & 1 & | & N
\end{bmatrix}.
\]
Finally \(-1/2 < \mu_{21} = -12/29 < 0 \) and \( B \) is LLL reduced.

---

**A LLL BASED EXTENDED GCD ALGORITHM**

**INPUT:** Positive integers \( d_1, \ldots, d_m \);
\( B := I_m \);

**for** \( r = 2, \ldots, m \)

**for** \( s = 1, \ldots, r - 1 \)

\( \lambda_{si} := 0; \)

\( D_i := 1; \)

\( a_i := d_s; \)

\( a_1 := 1; \)

\( m_1 := 1; \)

\( n_1 := 1; \)

\( \alpha := m_1/n_1 \star \)

\( k := 2; \)

**while** \( k \leq m \)

\( \text{Reduce1}(k, k - 1); \)

**if** \( a_{s-1} \neq 0 \) **or** \( \{ a_{s-1} = 0 \text{ and } a_k = 0 \text{ and } a_1(\lambda_{k-1}D_1 + \lambda_{s-1}^2D_{s-1}) < m_1D_{s-1} \} \)

\( \text{Swap1}(k); \)

**if** \( k > 2 \)

\( k := k - 1; \)

**else**

\( \text{Reduce1}(k, i); \)

\( i := k - 2, \ldots, 1; \)

\( k := k + 1; \)

**if** \( a_m < 0 \)

\( a_m := -a_m; \)

\( b_m := -b_m; \)

**OUTPUT:** \( a_0 = \gcd(d_1, \ldots, d_m); \)

small multipliers \( b_{1l}, \ldots, b_{1m}; \)

small null space basis \( b_1, \ldots, b_{m-1}; \)

---

We perform LLL on the rows of \([B|A] = [I_m|D]\), except that when processing rows \( k \) and \( k - 1 \), if we encounter
\( a_1 = 0, \ldots, a_{k-2} = 0, a_{k-1} \neq 0 \), instead of the usual partial size-reduction, we perform
\[
R_k \rightarrow R_k - \frac{a_k}{a_{k-1}} R_{k-1}.
\]

We then interchange rows \( k - 1 \) and \( k \) and perform the LLL algorithm on the first \( k \) rows with no interchange of row \( k \).

The effect is to successively produce for \( k = 2, \ldots, m \), a multiplier vector \( (b_{k1}, \ldots, b_{kk}) \) for \( d_1, \ldots, d_k \) which is size-reduced with respect to a LLL reduced lattice basis \( (b_{11}, \ldots, b_{1k}), \ldots, (b_{k-1}, \ldots, b_{kk}) \) for the lattice defined by \( x_1d_1 + \cdots + x_kd_k = 0 \).
Swap $k$,

\[ a_k \leftarrow a_{k-1}; \]
\[ b_k \leftarrow b_{k-1}; \]

for $j = 1, \ldots, k-2$

\[ \lambda_{k-j} \leftarrow \lambda_{k-1-j}; \]

for $i = k + 1, \ldots, m$

\[ t := \lambda_{i, k-1} D_i - \lambda_{i, k-1}; \]
\[ \lambda_{i, k-1} := (\lambda_{i, k-1} \lambda_{k-1} + \lambda_{i, k-1} D_i) / D_{k-1}; \]
\[ \lambda_{i, k-1} := t / D_{k-1}; \]

\[ D_{k-1} := (D_{k-2} D_i + \lambda_{i, k-1}^2) / D_{k-1}; \]

---

**Example 1: LLL BASED EXTENDED GCD ALGORITHM:** \( \text{gcd}(4, 6, 9), a = 1 \)

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$0$</td>
</tr>
<tr>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

---

Example 2: Fibonacci numbers

\( F_n, \ldots, F_{n+m-1} \)

For $n \geq 2$ there is exactly one shortest multiplier vector and it can be described explicitly. (See K.R. Matthews, *Minimal multipliers for consecutive Fibonacci numbers*, Acta Arith. 75 (1996) 205-218.)

\[ a = (1, 1, -1) \]

and performing our algorithm on \( \text{gcd}(9, 6, 4) \) yields this multiplier.

Hence \( \text{gcd}(4, 6, 9) = 1 \); multipliers \((-2, 0, 1)\).

The shortest multiplier vector is in fact

The Fibonacci and Lucas numbers are defined by
\[ F_1 = F_2 = 1, \quad F_{n+2} = F_{n+1} + F_n, \quad n \geq 1; \]
\[ L_1 = 1, L_2 = 3, \quad L_{n+2} = L_{n+1} + L_n, \quad n \geq 1. \]
\[
\begin{pmatrix}
 1 & 1 \\
 0 & 1
\end{pmatrix}
\ldots
\begin{pmatrix}
 1 & 1 \\
 1 & 0
\end{pmatrix}
= \begin{pmatrix}
 F_{n+1} & F_n \\
 F_n & F_{n-1}
\end{pmatrix}.
\]

Taking determinants of both sides gives
\[ F_{n+1}F_{n-1} - F_n^2 = (-1)^n, \]
which in turn gives
\[ F_{n-1}F_n - F_{n-2}F_{n+1} = (-1)^n. \]
Hence
\[ M_n = ((-1)^nF_{n-1}, (-1)^{n+1}F_{n-2}, 0, \ldots, 0). \]
is a multiplier vector.

Λ is the lattice of \((x_1, \ldots, x_m) \in \mathbb{Z}^m\) satisfying
\[ x_1F_n + x_2F_{n+1} + \cdots + x_mF_{n+m-1} = 0. \]
Λ has a lattice basis consisting of the vectors:
\[ \mathcal{L}_1, \ldots, \mathcal{L}_{m-2}, \mathcal{M}_{n+2}, \]
where
\[ \mathcal{L}_1 = (1, 1, -1, 0, \ldots, 0), \]
\[ \mathcal{L}_2 = (0, 1, 1, -1, 0, \ldots, 0), \]
\[ \vdots \]
\[ \mathcal{L}_{m-2} = (0, \ldots, 0, 1, 1, -1). \]

The general multiplier vector has the form
\[ M_n + y_1\mathcal{L}_1 + \cdots + y_{m-2}\mathcal{L}_{m-2} + y_{m-1}M_{n+2}, \]
where \(y_1, \ldots, y_{m-1}\) are integers.

If \(n \geq 2\), there is a unique multiplier vector \(W_{n,m}\) of least length and defined as follows:
\[ W_{n,m} = M_n + (-1)^n \sum_{j=1}^{m-2} (-1)^j G_{n,j,m} \mathcal{L}_j, \]
where:

If \(m\) is even,
\[ G_{n,r,m} = \begin{cases} H_{n,r,m} & r \text{ even,} \\ H_{n-1,r+1,m} & r \text{ odd.} \end{cases} \]
If \(m\) is odd,
\[ G_{n,r,m} = \begin{cases} H_{n,r,m-1} & r \text{ even,} \\ H_{n-1,r+1,m+1} & r \text{ odd.} \end{cases} \]
\[ H_{n,r,m} = \left[ \frac{F_{n-r}(F_{n-2} + F_r)}{F_m} \right]. \]

Alternatively
\[ W_{n,m} = (-1)^n(W_{n,1,m}, -W_{n,2,m}, \ldots, -W_{n,m,m}), \]
\[ W_{n,r,m} = G_{n,r-2,m} + G_{n,r-1,m} - G_{n,r,m}. \]
The vectors
\[ L_1 \ldots, L_{m-2}, W_{a+2,m} \]
form a \( \mathbb{Z} \)-basis for \( \Lambda \) and (apart from sign) this is the one always found by our LLL based algorithm.

**THEOREM.** If \( B \) is a unimodular \( 3 \times 3 \) integer matrix such that the first 2 rows \( b_1, b_2 \) form a LLL-reduced basis for the lattice \( \Lambda \) with \( 3/8 \leq \alpha \leq 1 \), while \( b_3 \) is size-reduced and is a multiplier vector for \( d_1, d_2, d_3 \), then the smallest multiplier is one of the vectors \( b_3 + \epsilon_1 b_1 + \epsilon_2 b_2 \), where \( \epsilon_i = -1, 0, 1 \) for \( i = 1, 2 \), \( (\epsilon_1, \epsilon_2) \neq (\pm 1, 0) \).

**PROOF.** We look for smaller multipliers than \( b_3 \). These satisfy
\[
\|b_3 + z b_1 + y b_2\|^2 < \|b_3\|^2, \text{ or equivalently}
\]
\[
\|b_3\|^2 + (x + \mu_{21} y + \mu_{31}) \|b_1\|^2 + (y + \mu_{32}) \|b_2\|^2 < \|b_3\|^2 + \mu_{31} \|b_1\|^2 + \mu_{32} \|b_2\|^2.
\]
Hence
\[
(y + \mu_{32})^2 < \mu_{31} \frac{\|b_1\|^2}{\|b_2\|^2} + \mu_{32}
\]
\[
(y + \mu_{32})^2 < \frac{1}{4} \cdot \frac{8}{4} = \frac{9}{4}, \text{ if } \alpha \geq \frac{3}{8}
\]
\[
|y + \mu_{32}| < \frac{3}{2} \Rightarrow |y| < 2 \Rightarrow |y| \leq 1.
\]
Then as \( y(y + 2\mu_{32}) \geq 0 \) if \( y \in \mathbb{Z} \),
\[
(x + \mu_{21} y + \mu_{31})^2 |b_1|^2 + y(y + 2\mu_{32}) |b_2|^2
\leq \mu_{31}^2 |b_1|^2
\Rightarrow (x + \mu_{21} y + \mu_{31})^2 |b_1|^2 < \mu_{31}^2 |b_1|^2
\Rightarrow |x| < |\mu_{21} y + \mu_{31}| + |\mu_{31}|
\Rightarrow |x| < |\mu_{21}| + 2|\mu_{31}| \leq \frac{1}{2} + 1 = \frac{3}{2}
\Rightarrow |x| \leq 1.
\]
Also from inequality (1), \( y = 0 \) implies \( x = 0 \).

The argument above goes through with a slight twist for \( m = 4 \), as was pointed out to me by vacation scholar Sean Byrnes. One only needs to assume \( \alpha > (5 + \sqrt{33})/16 \).

For \( m = 5 \), the example \((d_1, d_2, d_3, d_4, d_5) = (2, 5, 14, 23, 29)\) has shortest multiplier \( b_5 - 2b_1 + b_2 + b_3 + b_4 \) with \( \alpha = 1 \) and this was the only example with an \( |\epsilon_i| > 1 \) in the range \( 2 \leq d_i \leq 30 \).

A LLL BASED UPSIDE–DOWN ROW ECHELON FORM ALGORITHM

An \( m \times n \) integer matrix \( B \) is said to be in Hermite normal form if

(i) the first \( r \) rows of \( B \) are nonzero and the remaining rows are zero;

(ii) for \( 1 \leq i \leq r \), if \( b_{ij} \) is the first nonzero entry in row \( i \) of \( B \), then \( j_1 < j_2 < \cdots < j_r \);

(iii) \( b_{ij} > 0 \) for \( 1 \leq i \leq r \);

(iv) if \( 1 \leq k < i \leq r \), then \( 0 \leq b_{kj} < b_{ij} \).

Let \( G \in M_{m \times n}(\mathbb{Z}) \). Then there are various algorithms for finding a unimodular matrix \( P \) such that \( PG = B \) is in Hermite normal form and which attempt to reduce coefficient explosion during their execution, eg. Kannan–Bachem (1979).

Let \( G = [G_1| \cdots |G_n] \in M_{m \times n}(\mathbb{Z}) \).

Then the LLL algorithm applied to the matrix
\[
G(N) = [In | N^aG_1 | N^{a-1}G_2 | \cdots | NG_n]
\]
(where \( G_i \) is the \( i \)th column of \( G \)) will perform the same sequence of steps for \( N \geq N_0(G) \)

Also the last \( n \) columns of the LLL reduced form of \( G(N) \) will be a matrix whose rows, starting from the bottom, are in row echelon form, corresponding to the indices \( j_1, \ldots, j_r \).
EXAMPLE If $G = \begin{pmatrix} 8 & 44 & 43 \\ 4 & 10 & 43 \\ 56 & -550 & -328 \\ 76 & 10 & 42 \end{pmatrix}$, then for $N \geq 2595$ and $a = 1$, the LLL algorithm applied to $B = [I_3 | N^3 G_1 | N^2 G_2 | NG_3]$ will almost certainly perform the same operations, reducing $B$ to

$$
\begin{pmatrix}
12245 & -3855 & 878 & -1733 & 0 & 0 & 0 \\
530 & -167 & 38 & -75 & 0 & 0 & -5N \\
2134 & -672 & 153 & -302 & 0 & 6N^2 & -2N \\
502 & -158 & 36 & -71 & 4N^3 & -2N^2 & 2N
\end{pmatrix}.
$$

We can imitate the limiting form that LLL takes and perform the sequence instead on $[I_m | G]$ to get an algorithm for the upside-down HNF. If rank $G < m$, we expect the unimodular transformation matrix to have entries of moderate size.

LLL HNF ALGORITHM

INPUT: An $m \times n$ integer matrix $G$;

1. $B = I_m$
2. for $r = 2, \ldots, m$
   1. for $s = 1, \ldots, r - 1$
      1. $\lambda_{rs} := 0$;
      2. $A := \phi$;
      3. $D_i := 1$, $i = 0, \ldots, m$;
      4. $m_1 := 1$; $n_1 := 1$; $/ * \alpha = m_1/n_1 */$
      5. if ($\exists$ $a_{ij} \neq 0$ in first nonzero column of $G$) and $a_{ij} < 0$ [$a_i := -a_i, b_j := -b_j$];
      6. $k := 2$;
   2. while $k \leq m$
      1. Reduce2($k$, $k - 1$);
      2. if $\{\text{coll1 } \leq \text{col2} \text{ and } \text{col1 } \leq n\}$
         1. $n_1(D_{k-2} - D_{k-1} + \lambda_{k-1}^2) < m_1 D_{k-1}$
            1. Swap1($k$);
            2. if $k > 2$
               1. $k := k - 1$;
      3. else
         1. Reduce2($k$, $i$), $i = k - 2, \ldots, 1$
         2. $k := k + 1$;
   3. end

OUTPUT: $A$, the (upside-down) $HNF(G)$; $B$ the corresponding transformation matrix;

We perform LLL on the rows of $[B | A] = [I_m | G_1 | \cdots | G_n]$, the difference when processing rows $k - 1$ and $k$ being that if $A$ has the form

$$
A = \begin{pmatrix}
0 & \ldots & 0 & 0 & \ldots \\
\vdots & \ddots & \vdots & \vdots & \ddots \\
0 & \ldots & 0 & a_{k-1, \text{col1}} & \ldots \\
a_{k1} & \ldots & a_{k, \text{col1}} & \ldots \\
\vdots & \ddots & \vdots & \vdots & \ddots \\
\end{pmatrix},
$$

where $a_{k-1, \text{col1}} \neq 0$, instead of the partial size-reduction, we perform

$$
R_k \rightarrow R_k - \frac{a_{k, \text{col1}}}{a_{k-1, \text{col1}}} R_{k-1}.
$$

Then we interchange rows $k - 1$ and $k$ and perform the LLL algorithm on the first $k$ rows with no interchange of row $k$.

Similarly with $k - 1$ replaced by $i < k - 1$, but with no swapping.
Minus \( j \)

\[
\begin{align*}
  \text{for } r &= 2, \ldots, m \\
  \text{for } s &= 1, \ldots, r - 1 \\
  \text{if } r &= j \text{ or } s = j \\
  \lambda_{rs} &:= -\lambda_{rs};
\end{align*}
\]