

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

CENTRAL PROBLEM:

If d_1, \dots, d_m , $m \geq 2$, are nonzero integers,
find integers x_1, \dots, x_m such that

$$d = \gcd(d_1, \dots, d_m) = x_1 d_1 + \dots + x_m d_m,$$

with $x_1^2 + \dots + x_m^2$ small. We call (x_1, \dots, 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

$$\begin{bmatrix} 1 & \cdots & 0 & d_1 \\ 0 & \cdots & 0 & d_2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & d_m \end{bmatrix} \text{ to } \begin{bmatrix} b_{11} & \cdots & b_{1m} & 0 \\ b_{21} & \cdots & b_{2m} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & \cdots & b_{mm} & d \end{bmatrix}.$$

Then $B = [b_{ij}]$ is unimodular and
 $\mathbf{b}_m = (b_{m1}, \dots, b_{mm})$ is a multiplier vector.

With $\mathbf{b}_i = (b_{i1}, \dots, b_{im})$, then

$$\Lambda = \{(x_1, \dots, x_m) \in \mathbb{Z}^m \mid d_1 x_1 + \dots + d_m x_m = 0\}$$

is an $(m - 1)$ -dimensional lattice in \mathbb{Z}^m with basis $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$. The lattice determinant $d(\Lambda)$ is given by

$$d(\Lambda) = \|D\|/d = \frac{1}{d} \sqrt{(d_1^2 + \dots + d_m^2)}.$$

The general multiplier has the form

$$\mathbf{b} = \mathbf{b}_m + y_1 \mathbf{b}_1 + \dots + y_{m-1} \mathbf{b}_{m-1},$$

where $y_1, \dots, y_{m-1} \in \mathbb{Z}$.

PHILOSOPHY. Try to find short basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$ for Λ and integers y_1, \dots, y_{m-1} which make $\|\mathbf{b}\|$ 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 ($m = 3$):

$$(d_1, d_2, d_3) \rightarrow (d_2 \bmod d_1, d_3 \bmod d_1, d_1).$$

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 $\mathbf{b}_1, \dots, \mathbf{b}_m$ over the rationals and Gram–Schmidt basis $\mathbf{b}_1^*, \dots, \mathbf{b}_m^*$, where

$$\mathbf{b}_1^* = \mathbf{b}_1, \quad \mathbf{b}_k^* = \mathbf{b}_k - \sum_{j=1}^{k-1} \mu_{kj} \mathbf{b}_j^*$$

and $\mu_{kj} = \frac{\mathbf{b}_k \cdot \mathbf{b}_j^*}{\mathbf{b}_j^* \cdot \mathbf{b}_j^*}$ for $1 \leq j < k \leq m$.

The lattice basis $\mathbf{b}_1, \dots, \mathbf{b}_m$ is *reduced* if

$$(i) \quad |\mu_{kj}| \leq 1/2 \text{ for } 1 \leq j < k \leq m$$

$$(ii) \quad \|\mathbf{b}_k^*\|^2 \geq (\alpha - \mu_{k, k-1}^2) \|\mathbf{b}_{k-1}^*\|^2 \quad (C2)$$

for $1 < k \leq m$. (Here $1/4 < \alpha \leq 1$.)

\mathbf{b}_k is *size-reduced* if $|\mu_{kj}| \leq 1/2$ for $1 \leq j < k$.

THE LLL LATTICE BASIS REDUCTION ALGORITHM (1982)

Start with row $k = 2$ of B . (Row 1 has to be nonzero.) Partially size-reduce \mathbf{b}_k by

$$\mathbf{b}_k \rightarrow \mathbf{b}_k - \lceil \mu_{k k-1} \rceil \mathbf{b}_{k-1},$$

where $\lceil \theta \rceil$ is the nearest integer symbol, with $\lceil \theta \rceil = \theta - \frac{1}{2}$, if θ is a half-integer.

If (C2) does not hold, we swap \mathbf{b}_k and \mathbf{b}_{k-1} and decrement k .

Otherwise size-reduce \mathbf{b}_k completely by

$$\mathbf{b}_k \rightarrow \mathbf{b}_k - \lceil \mu_{kj} \rceil \mathbf{b}_j, \quad j = k - 2, \dots, 1,$$

then increment k .

PSEUDO-CODE FOR THE LLL ALGORITHM (de Weger)

```

INPUT:  $m \times n$  integer matrix  $B$ ;
 $m_1 := 1$ ;  $n_1 := 1$ ;                                /*  $\alpha = m_1/n_1$  */
 $D_0 := 1$ ;
for  $i = 1, \dots, m$  {
     $\mathbf{c}_i := \mathbf{b}_i$ ;                                /*  $\mathbf{c}_i = D_{i-1} \mathbf{b}_i^*$  */
    for  $j = 1, \dots, i - 1$  {
         $\lambda_{ij} := \mathbf{b}_i \cdot \mathbf{c}_j$ ;            /*  $\lambda_{ij} = D_j \mu_{ij}$  */
         $\mathbf{c}_i := (D_j \mathbf{c}_i - \lambda_{ij} \mathbf{c}_j) / D_{j-1}$ ;
    }
     $D_i := (\mathbf{c}_i \cdot \mathbf{c}_i) / D_{i-1}$                 /*  $\|\mathbf{b}_i^*\|^2 = D_i / D_{i-1}$  */
}
 $k := 2$ ;
while  $k \leq m$  {
    Reduce( $k, k - 1$ );
    if  $\{n_1(D_{k-2}D_k + \lambda_{kk-1}^2) < m_1D_{k-1}^2\}$  {
        Swap( $k$ );
        if  $k > 2$ 
             $k := k - 1$ ;
    }
    else {
        Reduce( $k, i$ ),     $i = k - 2, \dots, 1$ ;
         $k := k + 1$ ;
    }
}
OUTPUT:  $B$ , whose rows are LLL reduced;

```

```

Reduce( $k, i$ )
  if  $2|\lambda_{ki}| > D_i$ 
     $q := \lceil \lambda_{ki}/D_i \rceil$ ;
    else  $q := 0$ ;

  if  $q \neq 0$  {
     $\mathbf{b}_k := \mathbf{b}_k - q\mathbf{b}_i$ ;
     $\lambda_{ki} := \lambda_{ki} - qD_i$ ;

    for  $j = 1, \dots, i - 1$ 
       $\lambda_{kj} := \lambda_{kj} - q\lambda_{ij}$ ;
  }

```

Swap (k)

$\mathbf{b}_k \leftrightarrow \mathbf{b}_{k-1};$

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

$\lambda_{kj} \leftrightarrow \lambda_{k-1j};$

for $i = k + 1, \dots, m$ {

$t := \lambda_{ik-1}D_k - \lambda_{ik}\lambda_{kk-1};$

$\lambda_{ik-1} := (\lambda_{ik-1}\lambda_{kk-1} + \lambda_{ik}D_{k-2})/D_{k-1};$

$\lambda_{ik} := t/D_{k-1};$

}

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

THE LARGE N EXTENDED GCD ALGORITHM

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

$$\begin{bmatrix} 1 & \cdots & 0 & Nd_1 \\ 0 & \cdots & 0 & Nd_2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & Nd_m \end{bmatrix} \rightarrow \begin{bmatrix} b_{11} & \cdots & b_{1m} & 0 \\ b_{21} & \cdots & b_{2m} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & \cdots & b_{mm} & Ng \end{bmatrix},$$

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

The resulting multiplier vector (b_{m1}, \dots, 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 $[I_m | \mathbf{D}]$, so as to perform essentially the same sequence of operations.

AN EXAMPLE OF THE LARGE N EXTENDED GCD ALGORITHM

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

$$\text{Let } B = \left[\begin{array}{cc|c} 1 & 0 & 2N \\ 0 & 1 & 5N \end{array} \right].$$

Then

$$\mu_{21} = \frac{\mathbf{b}_2 \cdot \mathbf{b}_1}{\mathbf{b}_1 \cdot \mathbf{b}_1} = \frac{10N^2}{1 + 4N^2}$$

Hence $2 < \mu_{21} < 5/2$ and $\lceil \mu_{21} \rceil = 2$.

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

$$B \rightarrow \left[\begin{array}{cc|c} 1 & 0 & 2N \\ -2 & 1 & N \end{array} \right].$$

Here $\|\mathbf{b}_1\|^2 = 1 + 4N^2$, $\|\mathbf{b}_2\|^2 = 5 + N^2$.

So if $N = 1$, $\|\mathbf{b}_1\|^2 \leq \|\mathbf{b}_2\|^2$ and B is LLL reduced.

If $N > 1$, $\|\mathbf{b}_1\|^2 > \|\mathbf{b}_2\|^2$ and we swap rows:

$$B \rightarrow \left[\begin{array}{cc|c} -2 & 1 & N \\ 1 & 0 & 2N \end{array} \right].$$

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

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

$N = 2, 3, 4$: Here

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

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

$$B \rightarrow \left[\begin{array}{cc|c} -2 & 1 & N \\ 3 & -1 & N \end{array} \right]$$

and B is LLL reduced.

$N \geq 5$: Here

$$3/2 < \mu_{21} < 2, \text{ so } \lceil \mu_{21} \rceil = 2$$

and we perform $R_2 \rightarrow R_2 - 2R_1$:

$$B \rightarrow \left[\begin{array}{cc|c} -2 & 1 & N \\ 5 & -2 & 0 \end{array} \right].$$

Then $\|\mathbf{b}_2\|^2 = 29$, $\|\mathbf{b}_1\|^2 = 5 + N^2 \geq 30$, so we swap rows:

$$B \rightarrow \left[\begin{array}{cc|c} 5 & -2 & 0 \\ -2 & 1 & N \end{array} \right].$$

Finally $-1/2 < \mu_{21} = -12/29 < 0$ and B is LLL reduced.

We perform LLL on the rows of $[B|A] = [I_m|D]$, except that when processing rows k and $k - 1$, if we encounter $a_1 = 0, \dots, a_{k-2} = 0, a_{k-1} \neq 0$, instead of the usual partial size-reduction, we perform

$$R_k \rightarrow R_k - \left\lfloor \frac{a_k}{a_{k-1}} \right\rfloor R_{k-1}.$$

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 $k = 2, \dots, m$, a multiplier vector (b_{k1}, \dots, b_{kk}) for d_1, \dots, d_k which is size-reduced with respect to a LLL reduced lattice basis $(b_{11}, \dots, b_{1k}), \dots, (b_{k-11}, \dots, b_{k-1k})$ for the lattice defined by $x_1 d_1 + \dots + x_k d_k = 0$.

A LLL BASED EXTENDED GCD ALGORITHM

INPUT: Positive integers d_1, \dots, d_m ;
 $B := I_m$;
for $r = 2, \dots, m$
 for $s = 1, \dots, r - 1$
 $\lambda_{rs} := 0$;
 $D_i := 1, \quad i = 0, \dots, m$;
 $a_i := d_i, \quad i = 1, \dots, m$;
 $m_1 := 1; n_1 := 1; /* \alpha = m_1/n_1 */$
 $k := 2$;
while $k \leq m$ {
 $Reduce1(k, k - 1)$;
 if $a_{k-1} \neq 0$ **or** $\{a_{k-1} = 0$ **and** $a_k = 0$
 and $n_1(D_{k-2}D_k + \lambda_{kk-1}^2) < m_1D_{k-1}^2\}$ {
 $Swap1(k)$;
 if $k > 2$
 $k := k - 1$;
 }
 }
 else {
 $Reduce1(k, i), \quad i = k - 2, \dots, 1$;
 $k := k + 1$;
 }
}
if $a_m < 0$ { $a_m := -a_m; \mathbf{b}_m := -\mathbf{b}_m;$ }
OUTPUT: $a_m = \gcd(d_1, \dots, d_m)$;
 small multipliers $b_{m1}, \dots, b_{m,m}$;
 small null space basis $\mathbf{b}_1, \dots, \mathbf{b}_{m-1}$;

```

Reduce1 (k, i)
  if  $a_i \neq 0$ 
     $q := \left\lceil \frac{a_k}{a_i} \right\rceil$ ;
  else {
    if  $2|\lambda_{ki}| > D_i$ 
       $q := \lceil \lambda_{ki}/D_i \rceil$ ;
    else  $q := 0$ ;
  }

  if  $q \neq 0$  {
     $a_k := a_k - qa_i$ ;
     $\mathbf{b}_k := \mathbf{b}_k - q\mathbf{b}_i$ ;
     $\lambda_{ki} := \lambda_{ki} - qD_i$ ;

    for  $j = 1, \dots, i - 1$ 
       $\lambda_{kj} := \lambda_{kj} - q\lambda_{ij}$ ;
  }

```

Swap1 (k)

$$a_k \leftrightarrow a_{k-1};$$

$$\mathbf{b}_k \leftrightarrow \mathbf{b}_{k-1};$$

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

$$\lambda_{kj} \leftrightarrow \lambda_{k-1j};$$

for $i = k + 1, \dots, m$ {

$$t := \lambda_{ik-1}D_k - \lambda_{ik}\lambda_{kk-1};$$

$$\lambda_{ik-1} := (\lambda_{ik-1}\lambda_{kk-1} + \lambda_{ik}D_{k-2})/D_{k-1};$$

$$\lambda_{ik} := t/D_{k-1};$$

}

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

**Example 1: LLL BASED EXTENDED
GCD ALGORITHM: $\gcd(4, 6, 9), \alpha = 1$**

$$\begin{array}{cccc} 1 & 0 & 0 & 4 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 1 & 9 \end{array} \rightarrow \begin{array}{cccc} 1 & 0 & 0 & 4 \\ -1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 9 \end{array} \rightarrow \begin{array}{cccc} -1 & 1 & 0 & 2 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 9 \end{array}$$

$$\begin{array}{cccc} -1 & 1 & 0 & 2 \\ 3 & -2 & 0 & 0 \\ 0 & 0 & 1 & 9 \end{array} \rightarrow \begin{array}{cccc} 3 & -2 & 0 & 0 \\ -1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 9 \end{array} \rightarrow \begin{array}{cccc} 3 & -2 & 0 & 0 \\ -1 & 1 & 0 & 2 \\ 4 & -4 & 1 & 1 \end{array}$$

$$\begin{array}{cccc} 3 & -2 & 0 & 0 \\ 4 & -4 & 1 & 1 \\ -1 & 1 & 0 & 2 \end{array} \rightarrow \begin{array}{cccc} 3 & -2 & 0 & 0 \\ -2 & 0 & 1 & 1 \\ -1 & 1 & 0 & 2 \end{array} \rightarrow \begin{array}{cccc} 3 & -2 & 0 & 0 \\ -2 & 0 & 1 & 1 \\ 3 & 1 & -2 & 0 \end{array}$$

$$\begin{array}{cccc} 3 & -2 & 0 & 0 \\ 3 & 1 & -2 & 0 \\ -2 & 0 & 1 & 1 \end{array} \rightarrow \begin{array}{cccc} 3 & -2 & 0 & 0 \\ 0 & 3 & -2 & 0 \\ -2 & 0 & 1 & 1 \end{array}$$

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

The shortest multiplier vector is in fact

$$\mathbf{b}[3] + \mathbf{b}[1] + \mathbf{b}[2] = (1, 1, -1)$$

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

Example 2: Fibonacci numbers

$$F_n, \dots, F_{n+m-1}$$

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.)

The Fibonacci and Lucas numbers are defined by

$$F_1 = F_2 = 1, \quad F_{n+2} = F_{n+1} + F_n, \quad n \geq 1;$$

$$L_1 = 1, L_2 = 3, \quad L_{n+2} = L_{n+1} + L_n, \quad n \geq 1.$$

$$\underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}}_n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}.$$

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

$$\mathcal{M}_n = ((-1)^n F_{n-1}, (-1)^{n+1} F_{n-2}, 0, \dots, 0).$$

is a multiplier vector.

Λ is the lattice of $(x_1, \dots, x_m) \in \mathbb{Z}^m$ satisfying

$$x_1 F_n + x_2 F_{n+1} + \dots + x_m F_{n+m-1} = 0.$$

Λ has a lattice basis consisting of the vectors:

$$\mathcal{L}_1, \dots, \mathcal{L}_{m-2}, \mathcal{M}_{n+2},$$

where

$$\begin{aligned} \mathcal{L}_1 &= (1, 1, -1, 0, \dots, 0), \\ \mathcal{L}_2 &= (0, 1, 1, -1, 0, \dots, 0), \\ &\vdots \\ \mathcal{L}_{m-2} &= (0, \dots, 0, 1, 1, -1). \end{aligned}$$

The general multiplier vector has the form

$$\mathcal{M}_n + y_1 \mathcal{L}_1 + \cdots + y_{m-2} \mathcal{L}_{m-2} + y_{m-1} \mathcal{M}_{n+2},$$

where y_1, \dots, y_{m-1} are integers.

If $n \geq 2$, there is a unique multiplier vector $\mathcal{W}_{n,m}$ of least length and defined as follows:

$$\mathcal{W}_{n,m} = \mathcal{M}_n + (-1)^n \sum_{j=1}^{m-2} (-1)^j G_{n,j,m} \mathcal{L}_j,$$

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,

$$G_{n,r,m} = \begin{cases} H_{n,r,m-1} & r \text{ even,} \\ H_{n-1,r+1,m+1} & r \text{ odd.} \end{cases}$$

$$H_{n,r,m} = \left\lfloor \frac{F_{m-r}(F_{n-2} + F_r)}{F_m} \right\rfloor.$$

Alternatively

$$W_{n,m} = (-1)^n (W_{n,1,m}, -W_{n,2,m}, \dots, -W_{n,m,m}),$$

$$W_{n,r,m} = G_{n,r-2,m} + G_{n,r-1,m} - G_{n,r,m}.$$

The vectors

$$\mathcal{L}_1 \dots, \mathcal{L}_{m-2}, \mathcal{W}_{n+2,m}$$

form a \mathbb{Z} -basis for Λ and (apart from sign) this is the one always found by our LLL based algorithm.

EXAMPLES

$W_{n,3} = \mathcal{M}_n + G_n\mathcal{L}_1$, where

$$G_n = (-1)^{n+1} \lfloor \frac{F_{n-3} + 1}{3} \rfloor.$$

$W_{n,4} = \mathcal{M}_n + G_n\mathcal{L}_1 + H_n\mathcal{L}_2$, where

$$G_n = (-1)^{n+1} \lfloor \frac{F_{n-3} + 1}{3} \rfloor,$$

$$H_n = (-1)^n \lfloor \frac{F_{n-2} + 1}{3} \rfloor.$$

$W_{n,5} = \mathcal{M}_n + G_n\mathcal{L}_1 + H_n\mathcal{L}_2 + I_n\mathcal{L}_3$, where

$$G_n = (-1)^{n+1} \lfloor \frac{3F_{n-3} + 3}{8} \rfloor,$$

$$H_n = (-1)^n \lfloor \frac{F_{n-2} + 1}{3} \rfloor,$$

$$I_n = (-1)^{n+1} \lfloor \frac{F_{n-3} + 3}{8} \rfloor.$$

FINDING THE SHORTEST MULTIPLIER VECTORS FOR $m = 3$.

Even when $m = 3$, our LLL based gcd algorithm does not always produce the shortest multiplier: in the example 4, 6, 9, LLL always produces the multiplier $\mathbf{b}_3 = (-2, 0, 1)$, whereas $\mathbf{b}_3 + \mathbf{b}_2 + \mathbf{b}_1 = (1, 1, -1)$ is the shortest.

THEOREM. If B is a unimodular 3×3 integer matrix such that the first 2 rows $\mathbf{b}_1, \mathbf{b}_2$ form a LLL-reduced basis for the lattice Λ with $3/8 \leq \alpha \leq 1$, while \mathbf{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 7 vectors $\mathbf{b}_3 + \epsilon_1 \mathbf{b}_1 + \epsilon_2 \mathbf{b}_2$, 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 \mathbf{b}_3 . These satisfy

$$\|\mathbf{b}_3 + x\mathbf{b}_1 + y\mathbf{b}_2\|^2 < \|\mathbf{b}_3\|^2, \text{ or equivalently}$$

$$\begin{aligned} \|\mathbf{b}_3^*\|^2 + (x + \mu_{21}y + \mu_{31})^2\|\mathbf{b}_1^*\|^2 + (y + \mu_{32})^2\|\mathbf{b}_2^*\|^2 \\ < \|\mathbf{b}_3^*\|^2 + \mu_{31}^2\|\mathbf{b}_1^*\|^2 + \mu_{32}^2\|\mathbf{b}_2^*\|^2. \end{aligned}$$

Hence

$$(y + \mu_{32})^2 < \mu_{31}^2 \frac{\|\mathbf{b}_1^*\|^2}{\|\mathbf{b}_2^*\|^2} + \mu_{32}^2$$

$$(y + \mu_{32})^2 < \frac{1}{4} \cdot 8 + \frac{1}{4} = \frac{9}{4}, \text{ if } \alpha \geq \frac{3}{8}$$

$$|y + \mu_{32}| < \frac{3}{2} \Rightarrow |y| < 2 \Rightarrow |y| \leq 1.$$

Then as $y(y + 2\mu_{32}) \geq 0$ if $y \in \mathbb{Z}$,

$$\begin{aligned} & (x + \mu_{21}y + \mu_{31})^2 \|\mathbf{b}_1\|^2 + y(y + 2\mu_{32}) \|\mathbf{b}_2^*\|^2 \\ & < \mu_{31}^2 \|\mathbf{b}_1^*\|^2 \\ \Rightarrow & (x + \mu_{21}y + \mu_{31})^2 \|\mathbf{b}_1\|^2 < \mu_{31}^2 \|\mathbf{b}_1^*\|^2 \\ \Rightarrow & |x + \mu_{21}y + \mu_{31}| < |\mu_{31}| \\ \Rightarrow & |x| < |\mu_{21}y + \mu_{31}| + |\mu_{31}| \quad (1) \\ \Rightarrow & |x| < |\mu_{21}| + 2|\mu_{31}| \leq \frac{1}{2} + 1 = \frac{3}{2} \\ \Rightarrow & |x| \leq 1. \end{aligned}$$

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_i} is the first nonzero entry in row i of B , then $j_1 < j_2 < \cdots < j_r$;
- (iii) $b_{ij_i} > 0$ for $1 \leq i \leq r$;
- (iv) if $1 \leq k < i \leq r$, then $0 \leq b_{kj_i} < b_{ij_i}$.

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

$$G(N) = [I_m | N^n G_1 | N^{n-1} G_2 | \cdots | N G_n]$$

(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, \dots, j_r .

EXAMPLE If $G = \begin{bmatrix} 8 & 44 & 43 \\ 4 & 10 & 43 \\ 56 & -550 & -328 \\ 76 & 10 & 42 \end{bmatrix}$, then

for $N \geq 2595$ and $\alpha = 1$, the LLL algorithm applied to

$$B = [I_3 | N^3 G_1 | N^2 G_2 | N G_3]$$

will almost certainly perform the same operations, reducing B to

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

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 $\text{rank } G < m$, we expect the unimodular transformation matrix to have entries of moderate size.

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

$$A = \begin{bmatrix} 0 & \cdots & 0 & 0 & \cdots \\ \vdots & & \vdots & \vdots & \\ 0 & \cdots & 0 & 0 & \cdots \\ 0 & \cdots & 0 & a_{k-1,col1} & \cdots \\ a_{k1} & \cdots & & a_{k,col1} & \cdots \\ & \cdots & & \cdots & \end{bmatrix},$$

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

$$R_k \rightarrow R_k - \left\lfloor \frac{a_{k,col1}}{a_{k-1,col1}} \right\rfloor R_{k-1}.$$

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.

LLL HNF ALGORITHM

INPUT: An $m \times n$ integer matrix G ;
 $B := I_m$;
for $r = 2, \dots, m$
 for $s = 1, \dots, r - 1$
 $\lambda_{rs} := 0$;
 $A := G$;
 $D_i := 1, \quad i = 0, \dots, m$;
 $m_1 := 1; n_1 := 1; /* \alpha = m_1/n_1 */$
if ($\exists! a_{ij} \neq 0$ in first nonzero column of G)
and ($i = m$ and $a_{mj} < 0$) $\{\mathbf{a}_m \leftarrow -\mathbf{a}_m; b_{m,m} = -1\}$;
 $k := 2$;
while $k \leq m$ {
 Reduce2($k, k - 1$);
 if $\{col1 \leq col2 \text{ and } col1 \leq n\}$
 or $\{col1 = col2 = n + 1 \text{ and}$
 $n_1(D_{k-2}D_k + \lambda_{kk-1}^2) < m_1D_{k-1}^2\}$ {
 Swap1(k);
 if $k > 2$
 $k := k - 1$;
 }
 else {
 Reduce2(k, i), $i = k - 2, \dots, 1$;
 $k := k + 1$;
 }
}

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

Reduce2(k, i)

```
if  $\exists j$  such that  $a_{i,j} \neq 0$  {  
    col1 := least  $j$  such that  $a_{i,j} \neq 0$ ;  
    if  $a_{i,col1} < 0$  {Minus( $i$ );  $\mathbf{a}_i = -\mathbf{a}_i$ ;  $\mathbf{b}_i := -\mathbf{b}_i$ ;}  
}  
else  
    col1 :=  $n + 1$ ;  
if  $\exists j$  such that  $a_{k,j} \neq 0$  {  
    col2 := least  $j$  such that  $a_{k,j} \neq 0$ ;  
}  
else  
    col2 :=  $n + 1$ ;  
if  $col1 \leq n$   
     $q := \left\lfloor \frac{a_{k,col1}}{a_{i,col1}} \right\rfloor$ ;  
else {  
    if  $2|\lambda_{ki}| > D_i$   
         $q := \lceil \lambda_{ki}/D_i \rceil$ ;  
    else  $q := 0$ ;  
}  
if  $q \neq 0$  {  
     $\mathbf{a}_k := \mathbf{a}_k - q\mathbf{a}_i$ ;  $\mathbf{b}_k := \mathbf{b}_k - q\mathbf{b}_i$ ;  
     $\lambda_{ki} := \lambda_{ki} - qD_i$ ;  
    for  $j = 1, \dots, i - 1$   
         $\lambda_{kj} := \lambda_{kj} - q\lambda_{ij}$ ;  
}
```

Minus(j)

for $r = 2, \dots, m$

for $s = 1, \dots, r - 1$

if $r = j$ **or** $s = j$

$\lambda_{rs} := -\lambda_{rs};$