J. Symbolic Computation (2000) **11**, 1–000

Complexity of the Havas, Majewski, Matthews LLL Hermite Normal Form algorithm

WILBERD VAN DER KALLEN

Mathematisch Instituut, Universiteit Utrecht

P.O. Box 80.010

NL-3508 TA Utrecht, The Netherlands

(Received Januari 1999; revised July 1999; accepted April 2000)

We consider the complexity of the LLL HNF algorithm (Havas *et al.* 1998, Algorithm 4). This algorithm takes as input an m by n matrix G of integers and produces as output a matrix $b \in GL_m(\mathbb{Z})$ so that $A = bG$ is in Hermite normal form (upside down). The analysis is similar to that of an extended LLL algorithm as given in (van der Kallen 1998).

1. The result

Let $B \geq 2$ be such that the rows of the input matrix G have squared length at most B. Our main result is

Theorem 1.1. *All through the algorithm all entries have bit length $\mathcal{O}(m \log(mB))$.*

We do not care about the constants in this estimate. We leave to the reader the easy task of estimating the number of operations on the entries in the manner of (Lenstra *et al.* 1982). One finds that $\mathcal{O}((m+n)^4 \log(mB))$ such operations will do.

Our result should be compared with the estimate $\mathcal{O}(m \log(B))$ of (Lenstra *et al.* 1982). The reason our estimate is a little worse is the presence of the transformation matrix b.

The theorem should also be compared with the result of (Kannan and Bachem 1979), where again it is the transformation matrix which gives the worst estimate, namely basically $\mathcal{O}(m^4 \log(mB))$. (For the matrix A they need only $\mathcal{O}(m^3 \log(mB))$.)

2. Notations

We have tried to use notations that are consistent with the literature, but the B of (Havas *et al.* 1998) clashes with the B of (Lenstra *et al.* 1982), which is why we have rebaptized it b. For a proper understanding the reader should keep (Lenstra *et al.* 1982) and (Havas *et al.* 1998) at hand. See also (Cohen 1993). Here are our main notations.
$\langle v, w\rangle = (vG, wG)$.
$(v, w)_{\text{mix}} = (\text{pr}_{\text{iso}} v, \text{pr}_{\text{iso}} w) + \langle v, w\rangle$, see section 6.

© 2000 Academic Press Limited

A	$A = bG$ is eventually in upside down Hermite normal form, see abstract.
B	integer so that $B \geq 2$ and $(e_iG, e_iG) \leq B$ for all i.
b	The transformation matrix, see abstract.
b_i	i-th row of b.
b_i^*	i-th Gram-Schmidt vector of the rows of b, with respect to $(\ ,\ )_{\mathrm{mix}}$, see section 4.
C	integer so that $\lvert\mu_{ij}\rvert^2 \leq C$.
d_i	$d_i = \prod_{j=1}^{i} \langle b_{j+m_{\mathrm{iso}}}^*, b_{j+m_{\mathrm{iso}}}^* \rangle$.
d_i^{iso}	$d_i^{\mathrm{iso}} = \prod_{j=1}^{i} (b_j^*, b_j^*)$ for $i \leq m_{\mathrm{iso}}$.
e_i	element of standard basis of $\mathbb{R}^m$.
gram	Gram matrix of $\langle\ ,\ \rangle$: $gram_{ij} = \langle e_i, e_j \rangle$.
k	index pointing at the row b_k.
k_{max}	maximum value that k has attained.
G	the submatrix of the input matrix consisting of the columns that correspond with columns of A in which a pivot has already appeared, see section 6.
m	the number of rows of the input matrix.
m_{iso}	the number of zero rows with which A starts presently.
μ_{ij}	$b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$.
n	the number of columns of the input matrix.
$\mathrm{pr}_{\mathrm{iso}}$	orthogonal projection of $\mathbb{R}^m$ onto the computed part of the isotropic subspace, see section 6.
rank	the rank of G.
r_k	the index of $\mathbb{Z}$ in $\mathbb{Z} + \mathbb{Z}\mu_{k,k-1}$ in a `trickledown` step, see section 9.
z	a nonzero vector in `trickledown` with $(z, z)_{\mathrm{mix}} = 0$.

3. Introduction

The proof is a rather technical modification of the proof in (Lenstra *et al.* 1982). We will describe the situation in gradually increasing detail.

The main issue is whether we can estimate the entries of b in terms of B, m, n during the algorithm. The entries of A can then be estimated through $A = bG$. As they do not affect b, we may remove from G all columns that do not contribute a pivot to the Hermite normal form. Once that is done, A has as many columns as its rank, and at the end of the algorithm the product of its pivots is the covolume of the lattice spanned by its rows. This covolume can be estimated in terms of any non-vanishing maximal minor of G, which by Hadamard is of size at most $B^{rank/2}$. This is analogous to the estimate $d_i \leq B^i$ of (Lenstra *et al.* 1982).

As in (van der Kallen 1998) we use the ordinary Euclidean inner product $(\ ,\ )$ for the rows of b, but also an inner product $\langle v, w \rangle = (vG, wG)$. (For the new G which has a rank equal to its number of columns.) The vectors v with $\langle v, v \rangle = 0$ are called *isotropic*. We let $\mathrm{pr}_{\mathrm{iso}}$ be the orthogonal projection according to $(\ ,\ )$ of $\mathbb{R}^m$ onto the isotropic subspace and put

$$(v, w)_{\mathrm{mix}} = (\mathrm{pr}_{\mathrm{iso}} v, \mathrm{pr}_{\mathrm{iso}} w) + \langle v, w \rangle.$$

(Compare (Pohst 1987).) One can estimate the ratio between (v, v) and $(v, v)_{\mathrm{mix}}$. The problem then becomes to estimate $(b_i, b_i)_{\mathrm{mix}}$ for any row b_i of b.

The algorithm computes a Hermite normal form first for the top k_{max} rows of G, starting with $k_{\mathrm{max}} = 1$, and increasing k_{max} in steps of one. Each time just before one wants to increase k_{max} the situation looks like the one at the end, but now only for the first $k_{\max}$ rows. Right after one wants to increase $k_{\max}$ we enter a stage which we will emulate with a procedure called `trickledown`, which we analyze as in (van der Kallen 1998). This is where we have to deviate most from (Lenstra *et al.* 1982). The `trickledown` stage is followed by an ordinary LLL stage and then we get back to increasing $k_{\max}$. For all these stages and the transitions between them we have to give estimates.

4. The analogy with an extended LLL algorithm

The LLL HNF algorithm (Havas *et al.* 1998, Algorithm 4) is based on lattice basis reduction. For us this will be much more important than the fact that it computes a Hermite normal form. Our task is to give estimates as long as the while loop of (Havas *et al.* 1998) runs. To emphasize the LLL nature of their algorithm, we now describe properties of its output in terms familiar from (Lenstra *et al.* 1982), (van der Kallen 1998). It is only because of the close similarity with the extended LLL algorithm of (van der Kallen 1998) that we can prove the present theorem.

Let $e_1,\ldots,e_m$ be the standard basis of $\mathbb{R}^m$. The Gram matrix $gram = (\langle e_i, e_j\rangle)_{i,j=1}^m$ belongs to a positive semidefinite inner product $\langle\ ,\ \rangle$ on $\mathbb{R}^m$. Note that *gram* has integer entries. Let *rank* be the rank of G and assume that we have removed from G the columns that do not contribute a pivot. (In this paper a pivot is an entry of A that is the first nonzero entry in its row and also in its column.) Put $m_{\text{iso}} = m - rank$ and let b_i^* denote the i-th Gram-Schmidt vector with respect to $(\ ,\ )_{\text{mix}}$. We may characterize the b_i^* as follows. Firstly, b_i^* lies in $(b_i + \sum_{j=1}^{i-1}\mathbb{R}b_i)$. Secondly, if $1 \le j < i \le m$ and $j \le m_{\text{iso}}$ then $(b_i^*, b_j) = 0$, but if $1 \le j < i \le m$ and $j > m_{\text{iso}}$ then $\langle b_i^*, b_j\rangle = 0$.

With those notations the output satisfies:

1 The first m_{iso} rows b_i of b are isotropic.
2 With respect to $(\ ,\ )$ the first m_{iso} rows of b form an LLL reduced basis of $\sum_{j=1}^{m_{\text{iso}}}\mathbb{Z}b_i$.
3 The last *rank* rows of b form a basis of the lattice they span, and this lattice contains no nonzero isotropic vector.
4 If $m_{\text{iso}} + 1 \le i < j \le m$ then $|\langle b_i^*, b_j\rangle| \le \langle b_i^*, b_i\rangle$.
5 If $1 \le i \le m_{\text{iso}}$ and $i < j \le m$ then we have $|(b_i^*, b_j)| \le 1/2(b_i^*, b_i)$.

The proof is clear.

5. Stages of the algorithm

Now that we have described a way to look at the final result, let us discuss how we view things along the way. We need to cut the algorithm into many stages. This despite the fact that one keeps running one and the same while loop. Our estimates are simply different for the different stages. The stages are separated by certain key events. One such key event is when k wants to go beyond $k_{\max}$. (As in (Cohen 1993) we use $k_{\max}$ to denote the maximum value that k has attained.) The event is followed by a stage which we emulate by the procedure `trickledown`, which ends when a new pivot appears in A or a new zero row appears in A (in the actual Havas, Majewski, Matthews LLL Hermite Normal Form algorithm). During this stage we estimate b in the same manner as in (van der Kallen 1998).

Thereafter one turns back to a stage which we call an ordinary LLL stage. One basically runs an ordinary LLL algorithm for the inner product $(\ ,\ )_{\text{mix}}$ until k wants to go beyond

$k_{\max}$ again. Then one turns to `trickledown` again, and so on. So the ordinary LLL stages alternate with `trickledown` stages.

What makes it all rather technical is that $(\ ,\ )_{\mathrm{mix}}$ depends on the stage. For instance, if at the end of `trickledown` a new zero row appears in A, then we have to change $\mathrm{pr}_{\mathrm{iso}}$ to take the newly found isotropic vector into account. But that means that $(\ ,\ )_{\mathrm{mix}}$ changes meaning.

During `trickledown` another technical difficulty is that one is dealing not just with an MLLL in the sense of (Pohst 1987), but even with an *extended* MLLL algorithm, *i.e.* one also requires the transformation matrix b. It is the latter which makes that one can not refer to (Pohst 1987) for the analysis.

6. An ordinary LLL stage

We now describe the situation during an ordinary LLL stage, after any execution of the body of the while loop. We leave the checks to the reader. One should assume for now that all claims hold when entering the present ordinary LLL stage. This should be checked after the discussion of `trickledown` below.

In the definition of the inner product $\langle\ ,\ \rangle$ we work with a G from which all columns have been removed where no pivot has been found yet in A. We have

1 An integer matrix b of determinant ± 1,
2 Integers k, $k_{\max}$, $1 \leq k \leq k_{\max} \leq m$,
3 An integer $m_{\mathrm{iso}} \geq 0$, so that the first m_{iso} rows b_i of b span the isotropic subspace of $\sum_{j=1}^{k_{\max}} \mathbb{R} b_j$.

For $i > k_{\max}$ we have $b_i = e_i$, the i-th row of the identity matrix.

Let $\mathrm{pr}_{\mathrm{iso}}$ be the orthogonal projection according to $(\ ,\ )$ of $\mathbb{R}^m$ onto $\sum_{j=1}^{m_{\mathrm{iso}}} \mathbb{R} b_j$ and put

$$(v, w)_{\mathrm{mix}} = (\mathrm{pr}_{\mathrm{iso}} v, \mathrm{pr}_{\mathrm{iso}} w) + \langle v, w\rangle.$$

Let b_i^* denote the i-th Gram-Schmidt vector with respect to $(\ ,\ )_{\mathrm{mix}}$, as in section 4. Let $\mu_{i,j}$ be defined for $i > j$ so that

$$b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*.$$

The first standard fact is then that, with respect to $(\ ,\ )_{\mathrm{mix}}$, the first $k-1$ rows of b form an LLL reduced basis of $\sum_{j=1}^{k-1} \mathbb{Z} b_j$, except that one does *not* require

$$|b_i^* + \mu_{i,i-1} b_{i-1}^*|_{\mathrm{mix}}^2 \geq 3/4 |b_{i-1}^*|_{\mathrm{mix}}^2$$

when $i > m_{\mathrm{iso}}$, and that the usual condition $|\mu_{i,j}| \leq 1/2$ is weakened to $|\mu_{i,j}| \leq 1$ for $j > m_{\mathrm{iso}}$. And the second standard fact is that, as in (Cohen 1993), the first $k_{\max}$ rows of b form a basis of $\sum_{j=1}^{k_{\max}} \mathbb{Z} e_i$.

During the ordinary LLL stage we run the LLL algorithm with respect to $(\ ,\ )_{\mathrm{mix}}$, except that one leaves out many swaps. (From now on we suppress mentioning the annoying weakening of the condition on the $\mu_{i,j}$.) Leaving out swaps will be harmless for our estimates, as the size estimates in (Lenstra *et al.* 1982) for the μ_{ij} *etcetera* do not require that one executes a swap whenever such is recommended by the LLL test.

Running LLL with respect to $(\ ,\ )_{\text{mix}}$ roughly amounts to running two LLL algorithms, one for $(\ ,\ )$ and one for $\langle\ ,\ \rangle$. That is why the pseudo-code in (Havas *et al.* 1998) makes the distinction between $col1 = n+1$ and $col1 \le n$.

One runs LLL until k tries to go to $k_{\max}+1$. If $k_{\max} = m$ we are through. If $k_{\max} < m$ then one should realize that because of the removal of columns from G the row $e_{k_{\max}+1}G$ will be dependent on the earlier rows. That makes that $(\ ,\ )_{\text{mix}}$ is degenerate on $\sum_{j=1}^{k_{\max}+1}\mathbb{R}b_j$, so that we enter an MLLL situation if we do not adapt $(\ ,\ )_{\text{mix}}$. But we cannot adapt $(\ ,\ )_{\text{mix}}$ yet, as this would destroy the link with what the algorithm of (Havas *et al.* 1998) actually does. That is why we will switch to `trickledown` at this point. One may see `trickledown` as the search for the missing isotropic vector.

7. Estimates

We keep the notations of section 6. We want to give estimates by changing (van der Kallen 1998) minimally. Recall that $B \ge 2$ is such that the entries of $gram = (\langle e_i, e_j\rangle)$ are at most B. We start with investigating the connection between $(\ ,\ )$ and $(\ ,\ )_{\text{mix}}$.

7.1. DETERMINANTS

Let $gram_{\text{mix}}$ be the Gram matrix $((e_i, e_j)_{\text{mix}})$ with respect to $e_1, \ldots, e_{k_{\max}}$. Its entries are at most $B+1$. With Hadamard this gives

$$|\det(gram_{\text{mix}})| \le (\sqrt{m}(B+1))^m$$

and the same estimate holds for its subdeterminants. We claim that the determinant of $gram_{\text{mix}}$ is an integer, so that we also get this upper bound for the entries of $gram_{\text{mix}}^{-1}$. To see the claim, consider as in (Pohst 1987) the inner product $(\ ,\ )_\epsilon$ given by $(v,w)_\epsilon = \epsilon(v,w) + \langle v, w\rangle$. Its Gram matrix has a determinant which is a polynomial $\det_\epsilon$ of ϵ with integer coefficients. One may also compute $\det_\epsilon$ with respect to a basis which is obtained from $e_1, \ldots, e_{k_{\max}}$ through an orthogonal transformation matrix. By diagonalizing the Gram matrix of $\langle\ ,\ \rangle$ we see that $\det(gram_{\text{mix}})$ is the coefficient of $\epsilon^{m_{\text{iso}}}$ in $\det_\epsilon$. □

7.2. LENGTHS OF VECTORS

LEMMA 7.1. *For $v \in \mathbb{R}^m$ one has*

$$(v,v)_{\text{mix}} \le m(B+1)(v,v)$$

and for $v \in \sum_{j=1}^{k_{\max}}\mathbb{R}e_j$ one has

$$(v,v) \le m(\sqrt{m}(B+1))^m(v,v)_{\text{mix}}.$$

PROOF. The supremum of $\{(v,v)_{\text{mix}} \mid (v,v) = 1\}$ is the largest eigenvalue of the gram matrix of $(\ ,\ )_{\text{mix}}$ with respect to $e_1, \ldots, e_m$. The largest eigenvalue is no larger than the trace of this matrix. So it is at most $m(B+1)$. Similarly the largest eigenvalue of $gram_{\text{mix}}^{-1}$ it is at most $m(\sqrt{m}(B+1))^m$ by subsection 7.1. □

7.3. DISCRIMINANTS

Now put

$$d_i^{\mathrm{iso}} = \prod_{j=1}^{i} (b_j^*, b_j^*)$$

for $i \leq m_{\mathrm{iso}}$ and

$$d_i = \prod_{j=1}^{i} \langle b_{j+m_{\mathrm{iso}}}^*, b_{j+m_{\mathrm{iso}}}^* \rangle$$

for $i \leq \mathit{rank}$. As far as d_i is concerned we may compute modulo isotropic vectors, or also with $(\ ,\ )_{\mathrm{mix}}$. Indeed

$$\langle b_{j+m_{\mathrm{iso}}}^*, b_{j+m_{\mathrm{iso}}}^* \rangle = (b_{j+m_{\mathrm{iso}}}^*, b_{j+m_{\mathrm{iso}}}^*)_{\mathrm{mix}}$$

for $1 \leq j \leq \mathit{rank}$. Both d_i^{iso} and d_j are integers and they descend when applying LLL. (Throughout we must assume familiarity with the arguments in (Lenstra *et al.* 1982).) In fact the $\langle b_{j+m_{\mathrm{iso}}}^*, b_{j+m_{\mathrm{iso}}}^* \rangle$ are themselves squares of integers. (Squares of the pivots of the moment.) And they do not change during an ordinary LLL stage, because the swaps that would make them descend have been deleted.

One may also compute $\det(\mathit{gram}_{\mathrm{mix}})$ with the b_i^* basis, as the transition matrix from the e_i basis to the b_i^* basis has determinant ± 1. From that one sees that it is just $d_{m_{\mathrm{iso}}}^{\mathrm{iso}} d_{\mathit{rank}}$. So we get from subsection 7.1 that $d_{m_{\mathrm{iso}}}^{\mathrm{iso}} \leq (\sqrt{m}(B+1))^m$. In fact, for $i \leq m_{\mathrm{iso}}$ one has the same estimate

$$d_i^{\mathrm{iso}} \leq (\sqrt{m}(B+1))^m$$

because i was equal to m_{iso} earlier in the algorithm and LLL only makes d_i^{iso} go down. (`trickledown` will not touch it.)

Recall from section 3 that the pivots are integers whose product is at most $B^{\mathit{rank}/2}$. It follows that

$$d_i \leq B^{\mathit{rank}}$$

for $i \leq \mathit{rank}$.

LEMMA 7.2. *Let $1 \leq i \leq k_{\mathrm{max}}$. Then*

$$(\sqrt{m}(B+1))^{-m} \leq (b_i^*, b_i^*)_{\mathrm{mix}} \leq (\sqrt{m}(B+1))^m$$

and if $C \geq 1$ is such that $|\mu_{ij}|^2 \leq C$ for $1 \leq j < i$ then

$$(b_i, b_i)_{\mathrm{mix}} \leq mC(\sqrt{m}(B+1))^m$$

PROOF. Use the estimates of d_i^{iso}, d_i. □

7.4. PRESERVED ESTIMATES

Put $C = (4mB)^{5m}$. We will see in subsection 9.1 that

$$|\mu_{i,j}|^2 \leq C \quad \text{for} \quad 1 \leq \mathrm{j} < \mathrm{i} \leq \mathrm{k}_{\mathrm{max}}$$

at the start of an ordinary LLL stage.

LEMMA 7.3. *The following estimates hold after each execution of the body of the while loop in an ordinary LLL stage.*

1 $d_i^{\mathrm{iso}} \leq (\sqrt{m}(B+1))^m$ *for* $i \leq m_{\mathrm{iso}}$,
2 $d_i \leq B^{rank}$ *for* $i \leq rank$,
3 $(b_i, b_i)_{\mathrm{mix}} \leq mC(\sqrt{m}(B+1))^m$ *if* $i \neq k$ *and* $i \leq k_{\max}$,
4 $(b_k, b_k)_{\mathrm{mix}} \leq m^2 9^m C(\sqrt{m}(B+1))^{3m}$,
5 $|\mu_{i,j}| \leq 1$ *for* $1 \leq j < i < k$,
6 $|\mu_{k,j}| \leq 3^{m-k}\sqrt{mC}(\sqrt{m}(B+1))^m$ *for* $1 \leq j < k$,
7 $|\mu_{i,j}| \leq \sqrt{mC}(\sqrt{m}(B+1))^m$ *if* $1 \leq j < i$ *and* $k < i \leq k_{\max}$.

PROOF. That these are preserved under LLL follows as in (Lenstra *et al.* 1982), so one has to check that they hold right after `trickledown`. Given the above this will be straightforward. Note that one could insert reduction steps in the algorithm to get $C = 1$ instead of the outrageously pessimistic $C = (4mB)^{5m}$. □

8. Description of `trickledown`

Before we can do estimates concerning `trickledown` we must describe it. One starts with having $k = k_{\max}+1 \leq m$. (So we look at the moment that $k_{\max}$ should be increased, but we do not increase it yet.) Consider the lattice generated by $b_1, \ldots, b_{k_{\max}+1}$ where $b_{k_{\max}+1} = e_{k_{\max}+1}$. As $e_{k_{\max}+1}G$ is dependent on the earlier rows of G now, this lattice contains a nonzero vector z with $(z, z)_{\mathrm{mix}} = 0$. Modulo $\mathbb{R}z$ the vector b_k is linearly dependent on the b_i with $i < k$. Changing the basis of $\mathbb{Z}b_{k-1} + \mathbb{Z}b_k$ we can achieve that modulo $\mathbb{R}z$ the vector b_{k-1} is linearly dependent on the b_i with $i < k-1$. Then lower k by one and repeat until $k = m_{\mathrm{iso}} + 1$, where m_{iso} is the one from before the present `trickledown`. Or stop when the Havas, Majewski, Matthews LLL Hermite Normal Form algorithm produces a new pivot (in a column of A corresponding with one that we have removed from G). If a new pivot has been created we add back the relevant column to G and pass to a new $(\ ,\ )_{\mathrm{mix}}$. If $b_k = b_{m_{\mathrm{iso}}+1}$ is itself isotropic we increase m_{iso} by one and again pass to a new $(\ ,\ )_{\mathrm{mix}}$. This describes `trickledown`.

One may worry about the fact that `trickledown` does not trace the Havas, Majewski, Matthews LLL Hermite Normal Form algorithm faithfully. We are close enough though. (And our replacement has worse estimates than the original.) We are just leaving out some size reductions and we are taking together some swaps and reductions that make up the required change of basis of $\mathbb{Z}b_{k-1} + \mathbb{Z}b_k$. The change of basis is the one coming from an extended euclidean algorithm. Thus we will further ignore that `trickledown`, which we took from (van der Kallen 1998), does not quite trace this stage of the Havas, Majewski, Matthews algorithm. We simply blame their algorithm.

9. Estimates during `trickledown`

We look in more detail. Upon entering `trickledown` we change notation and freeze the old m_{iso}, $k_{\max}$ and the b_i^*, even though the b_i will change. We also do not change $(\ ,\ )_{\mathrm{mix}}$. Let $\mu_{i,0}$ stand for $(e_{k_{\max}+1}, b_i)$ and let $\mu_{i,j}$ stand for $(b_j^*, b_i)_{\mathrm{mix}}/(b_j^*, b_j^*)_{\mathrm{mix}}$ if $j > 0$. Note that initially $|\mu_{i,j}| \leq 1$ for $0 \leq j \leq i \leq k_{\max}$. We will estimate $|\mu_{i,j}|$ as k descends. The key point is that we can also estimate $\mu_{i,0}$. This compensates for the fact that $(\ ,\ )_{\mathrm{mix}}$ is degenerate on $\sum_{i=1}^{k_{\max}+1}\mathbb{R}e_i$. By combining $\mu_{i,0}$ with $(\ ,\ )_{\mathrm{mix}}$ we will be able to estimate (b_i,b_i). It is to explain the estimate of $\mu_{i,0}$ that we prefer to work with `trickledown`.

Say $k > m_{\mathrm{iso}}+1$ and modulo $\mathbb{R}z$ the vector b_k is linearly dependent on the b_i with $i<k$. Let us compute with b_k, b_{k-1} modulo $V=\mathbb{R}z+\sum_{i=1}^{k-2}\mathbb{R}b_i$. We have $b_k \equiv \mu_{k,k-1}b^*_{k-1}$ and $b_{k-1}\equiv b^*_{k-1}$ modulo V. With the extended euclidean algorithm of (Cohen 1993) we find an integer matrix $\begin{pmatrix}\alpha & \beta\\ \gamma & \delta\end{pmatrix}$ of determinant one so that $\begin{pmatrix}\alpha & \beta\\ \gamma & \delta\end{pmatrix}\begin{pmatrix}1\\ \mu_{k,k-1}\end{pmatrix}=\begin{pmatrix}0\\ -1/r_k\end{pmatrix}$ where r_k is the index of $\mathbb{Z}$ in $\mathbb{Z}+\mathbb{Z}\mu_{k,k-1}$. More specifically, one has $\begin{pmatrix}\delta & -\beta\\ -\gamma & \alpha\end{pmatrix}\begin{pmatrix}0\\ -1/r_k\end{pmatrix}=\begin{pmatrix}1\\ \mu_{k,k-1}\end{pmatrix}$ so $\beta=r_k$ and $\alpha=-r_k\mu_{k,k-1}$. By (Cohen 1993) we have $|\gamma|\le|\mu_{k,k-1}r_k|$ and $|\delta|\le r_k$. (Actually this is wrong. Indeed (Cohen 1993) only claims it when $\mu_{k,k-1}$ is nonzero. We leave the modifications for the case $\mu_{k,k-1}=0$ as an exercise.)

Now put $c_{k-1}=\alpha b_{k-1}+\beta b_k$ and $c_k=\gamma b_{k-1}+\delta b_k$. The algorithm `trickledown` tells us to replace b_k with c_k and b_{k-1} with c_{k-1}. We want to estimate the resulting new $\mu_{i,j}$, which we call $\nu_{i,j}$. For i different from k, $k-1$, nothing changes. (By convention the b^*_j are frozen.) Further $|\nu_{k-1,j}|=|\alpha\mu_{k-1,j}+\beta\mu_{k,j}|\le r_k|\mu_{k,k-1}\mu_{k-1,j}|+r_k|\mu_{k,j}|$ and $|\nu_{k,j}|=|\gamma\mu_{k-1,j}+\delta\mu_{k,j}|\le r_k|\mu_{k,k-1}\mu_{k-1,j}|+r_k|\mu_{k,j}|$, which is the same bound.

Lemma 9.1. *As k descends we have*

1 $|\mu_{k,j}|\le\sqrt{B}\prod_{i=k+1}^{k_{\max}+1}(2r_i)$ *for* $k>j\ge 0$,
2 $|\mu_{i,j}|\le 1$ *when* $k>i\ge j\ge 0$,
3 $|\mu_{i,j}|\le 2^m(\sqrt{B})^{rank+1}$ *if* $k\le i\le k_{\max}+1$ *and* $k_{\max}\ge j\ge 0$.

Proof. Initially we have $k=k_{\max}+1$ and $|\mu_{k,j}|^2\le B$. Now assume the estimates are true for the present k. If $j<k-1$, we get $|\nu_{k-1,j}|\le r_k|\mu_{k,k-1}\mu_{k-1,j}|+r_k|\mu_{k,j}|\le 2r_k\max_i|\mu_{k,i}|$ which takes care of $|\nu_{k-1,j}|$. This will be the new $\mu_{k,j}$ after the lowering of k. Recall $|\nu_{k,j}|$ satisfies the same bound. Let us forget to interrupt `trickledown` in case a new pivot is produced. Then $\prod_{k=m_{\mathrm{iso}}+2}^{k_{\max}+1}r_k$ is the ratio by which the covolume drops when adding $e_{k_{\max}+1}G$ to the lattice spanned by $e_1G,\ \dots,\ e_{k_{\max}}G$. So it is at most $(\sqrt{B})^{rank}$. Thus $|\nu_{k,j}|\le 2^m(\sqrt{B})^{rank+1}$ and $|\nu_{k-1,j}|\le 2^m(\sqrt{B})^{rank+1}$. Finally $|\nu_{k,k-1}|=1/r_k$ and $\nu_{k-1,k-1}=\nu_{k,j}=0$ for $k_{\max}\ge j>k$. $\square$

9.1. bailing out of `trickledown`

When k has reached $m_{\mathrm{iso}}+1$ or a new pivot has been created, it is time to forget the old $(\ ,\ )_{\mathrm{mix}}$. But first use lemma 7.2 and the estimates of the $\mu_{i,j}$ to estimate, for $i\le k_{\max}+1$,

$$(b_i,b_i)_{\mathrm{mix}}\le m4^mB^{rank+1}(\sqrt{m}(B+1))^m.$$

Similarly we have

$$(\mu_{i,0}e_{k_{\max}+1},\mu_{i,0}e_{k_{\max}+1})_{\mathrm{mix}}\le(B+1)4^mB^{rank+1},$$

and thus

$$(b_i-\mu_{i,0}e_{k_{\max}+1},b_i-\mu_{i,0}e_{k_{\max}+1})\le m^2(\sqrt{m}(B+1))^{2m}4^{m+1}B^{rank+1}$$

by means of Lemma 7.1, and finally

$$(b_i, b_i) \leq (4mB)^{4m}$$

say. These estimates hold all through `trickledown` and thus in particular at its end.

Now update G, m_{iso}, $k_{\max}$, $(\ ,\ )_{\text{mix}}$, b_i^* and so on. We have to estimate the new $\mu_{j,i}$. This is easy, as we have an estimate for $(b_j, b_j)_{\text{mix}}$ by lemma 7.1 and also one for $(b_i^*, b_i^*)_{\text{mix}}^{-1}$ by lemma 7.2. We get the estimate $|\mu_{j,i}|^2 \leq (4mB)^{5m}$, which was needed in subsection 7.4. Now go check that the description in section 6 is satisfied at the beginning of the subsequent ordinary LLL stage.

10. The actual entries

Now that we can estimate the b_i and d_i^{iso} all through the algorithm, it is time to consider the entries in the pseudo-code of Algorithm 4 in (Havas *et al.* 1998). It is clear how to estimate the entries of A, and their B is our b. The D_r require more care. They may be defined as in (Lenstra *et al.* 1982, (1.24)) by

$$D_r = \det((b_i, b_j)_{1 \leq i,j \leq r}).$$

It will suffice to estimate them during the ordinary LLL stages, as during `trickledown` D_r changes just once, from the value at the end of the previous stage to the value at the start of the subsequent stage. (Of course `trickledown` is only an emulation, but by means of $D_r \leq D_{r-1}(b_r, b_r)$ one can easily deal with the difference.)

Now for $r \leq m_{\text{iso}}$ we have $D_r = d_r^{\text{iso}}$, whence

$$D_r \leq (\sqrt{m}(B+1))^m$$

by subsection 7.3. But the pseudo-code also uses D_r for larger r. (We suspect that this may be costly.) For $r \geq k_{\max}$ we have $D_r = 1$. For $m_{\text{iso}} < r < k_{\max}$ we may use the following trick. Remove from G the columns corresponding with the pivots in $b_i G$ for $m_{\text{iso}} < i \leq r$. That does not affect D_r, but it makes m_{iso} become equal to r. So then the above estimate miraculously applies.

Finally we have the λ_{ij} of the pseudo-code to deal with. One estimates them through

$$\lambda_{ij}^2 \leq (b_i, b_i)(b_j, b_j)D_{j-1}^2.$$

11. Conclusion

We have proved the theorem by exploiting the analogy with the extended LLL algorithm of (van der Kallen 1998) for which a similar result holds with a similar proof. Indeed the main difference with the analysis of that algorithm is that there one does not remove any columns from G before defining $\langle v, w\rangle$. And one executes all swaps that make d_i go down. The output then also satisfies the properties of section 4 and on top of that the last rank rows of bG are LLL reduced. We refer to (van der Kallen 1998) for further details of analysis and implementation.

References

Cohen H. (1993). *A course in computational Algebraic Number Theory*, Graduate Texts in Mathematics **138**, Berlin: Springer.

Havas G., Majewski B.S., Matthews K.R. (1998). Extended gcd and Hermite normal form algorithms via lattice basis reduction, *Experimental Mathematics*, **7**, 125–136.

Kannan R., Bachem A. (1979). Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. Comput.* **8**, 499-507.

Lenstra A.K., Lenstra H.W. Jr., Lovász L. (1982). Factoring polynomials with rational coefficients, *Math. Ann.* **261**, 515–534.

Pohst M. (1987). A modification of the LLL-algorithm, *J. Symb. Comp.* **4**, 123–128.

Van der Kallen W. (1998). Complexity of an extended lattice reduction algorithm, electronic note `http://www.math.uu.nl/people/vdkallen/`