MATHEMATICS OF COMPUTATION
VOLUME 44, NUMBER 170
APRIL 1985, PAGES 463-471

Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis

By U. Fincke and M. Pohst

Abstract. The standard methods for calculating vectors of short length in a lattice use a reduction procedure followed by enumerating all vectors of $\mathbf{Z}^m$ in a suitable box. However, it suffices to consider those $\mathbf{x} \in \mathbf{Z}^m$ which lie in a suitable ellipsoid having a much smaller volume than the box. We show in this paper that searching through that ellipsoid is in many cases much more efficient. If combined with an appropriate reduction procedure our method allows to do computations in lattices of much higher dimensions. Several randomly constructed numerical examples illustrate the superiority of our new method over the known ones.

1. Introduction. In this paper we develop a new and efficient method for determining vectors $\mathbf{b}$ of short Euclidean length $\|\mathbf{b}\|$ in a lattice Λ of rank m,

$$\Lambda = \mathbf{Z}\mathbf{b}_1 + \cdots + \mathbf{Z}\mathbf{b}_m, \tag{1.1}$$

for linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ of $\mathbf{R}^n$. The standard algorithms for solving this task (see, for example, [4] in the case $m = n$) compute all $\mathbf{x} \in \mathbf{Z}^m \setminus \{\mathbf{0}\}$ subject to

$$\mathbf{x}^{\mathrm{tr}} B^{\mathrm{tr}} B \mathbf{x} \leqslant C \tag{1.2}$$

for suitable $C \in \mathbf{R}^{>0}$, where B denotes the $n \times m$ matrix with columns $\mathbf{b}_1, \ldots, \mathbf{b}_m$. If we just want to find a vector of shortest length in Λ, we must determine

$$\min\{\mathbf{x}^{\mathrm{tr}} B^{\mathrm{tr}} B \mathbf{x} \mid \mathbf{0} \neq \mathbf{x} \in \mathbf{Z}^m\}. \tag{1.3}$$

This can also be done by the methods for solving (1.2). As initial value for C we choose the length of an arbitrary (short) vector of Λ, and each time a vector of shorter length is obtained, C is decreased suitably.

We note that $B^{\mathrm{tr}} B =: A$ is a positive-definite matrix. On the other hand, each positive-definite matrix $A \in \mathbf{R}^{m \times m}$ can be considered as the inner product matrix of the basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ of some lattice Λ of rank m. Hence, it suffices to discuss

$$\mathbf{x}^{\mathrm{tr}} A \mathbf{x} \leqslant C \tag{1.4}$$

instead of (1.2) in the sequel.

In Section 2, we describe the new algorithm for solving (1.4) which is based on reduction theory and Cholesky's method. Section 3 contains a complexity analysis

Received April 12, 1983; revised February 2, 1984 and April 10, 1984.
1980 *Mathematics Subject Classification*. Primary 10E20, 10E25, 68C25.

0025-5718/85 $1.00 + $.25 per page

proving the superiority of our algorithm over known methods, and in Section 4 we present some numerical results for randomly generated examples.

All machine computations were carried out on the CDC Cyber 76 of the Rechenzentrum der Universität zu Köln.

2. A New Algorithm. Our new algorithm uses Cholesky's method which is an efficient procedure of decomposing a positive-definite quadratic matrix $A \in \mathbf{R}^{m\times m}$ into the product of an upper triangular matrix R and its transpose. The method combines the advantage of a comparatively low number of arithmetic operations with high numerical stability.

We shall use Cholesky's method for transforming the positive-definite quadratic form

$$\mathbf{x}^{\mathrm{tr}}A\mathbf{x} \qquad (\mathbf{x} \in \mathbf{R}^{m\times 1}) \tag{2.1}$$

for $A = B^{\mathrm{tr}}B$ of (1.2) into a sum of full squares, a procedure which we call quadratic completion. Namely, $\mathbf{x}^{\mathrm{tr}}A\mathbf{x} = \sum_{i,j=1}^{m} a_{ij}x_ix_j$ becomes

$$Q(\mathbf{x}) := \sum_{i=1}^{m} q_{ii}\left(x_i + \sum_{j=i+1}^{m} q_{ij}x_j\right)^2 \tag{2.2}$$

by carrying out the computations

$$\begin{aligned}&\text{Set } q_{ij} \leftarrow a_{ij} \qquad (1 \leqslant i \leqslant j \leqslant m).\\ &\text{For } i = 1,2,\ldots,m-1, \text{ set}\\ &q_{ji} \leftarrow q_{ij}, \quad q_{ij} \leftarrow \frac{q_{ij}}{q_{ii}} \qquad (i+1 \leqslant j \leqslant m)\\ &\text{and for each } i \text{ and } k = i+1,\ldots,m, \text{ set}\\ &\qquad q_{kl} \leftarrow q_{kl} - q_{ki}q_{il} \qquad (k \leqslant l \leqslant m).\end{aligned} \tag{2.3}$$

We note that the output R of Cholesky's method which satisfies $R^{\mathrm{tr}}R = A$ slightly differs from the output q_{ij} $(1 \leqslant i \leqslant j \leqslant m)$ of (2.3). The entries r_{ij} of $R \in \mathrm{GL}(m, \mathbf{R})$ can easily be recovered by

$$\begin{aligned}r_{ij} = 0 \qquad (1 \leqslant j < i \leqslant m), \qquad r_{ii} = q_{ii}^{1/2} \qquad (1 \leqslant i \leqslant m),\\ r_{ij} = r_{ii}q_{ij} \qquad (1 \leqslant i < j \leqslant m)\end{aligned} \tag{2.4}$$

(and vice versa, of course). Generally, we are interested in the q_{ij} because of the applicability of (2.2). Namely, (2.2) makes it simple to compute all solutions $\mathbf{x} \in \mathbf{Z}^m$ of

$$\mathbf{x}^{\mathrm{tr}}A\mathbf{x} = Q(\mathbf{x}) \leqslant C \tag{2.5}$$

where C is a positive constant. This is a problem occurring in many disciplines of mathematics, for example in integer mathematical programming [3] and algebraic number theory [2], also in connection with the generation of pseudo-random numbers [4] and breaking public cryptosystems [6]. We already noted in the introduction that (2.5) is equivalent to determining all lattice vectors $\mathbf{b}$ of length $\|\mathbf{b}\|^2 \leqslant C$ in an m-dimensional lattice $\Lambda = \oplus_{i=1}^{m} \mathbf{Z}\mathbf{b}_i$ for which the inner product of two basis vectors $\mathbf{b}_i, \mathbf{b}_j$ $(1 \leqslant i \leqslant j \leqslant m)$ is given by the entry a_{ij} of A. Also, (2.5) can be interpreted as the task of computing all points of $\mathbf{Z}^m$ in an ellipsoid.

The following procedure of solving (2.5) is suggested by (2.2). Clearly, $|x_m|$ is bounded by $\lfloor (C/q_{mm})^{1/2} \rfloor$. For each possible value of x_m we obtain

$$q_{m-1,m-1}(x_{m-1} + q_{m-1,m}x_m)^2 \leqslant T_{m-1} \quad \text{for } T_{m-1} := C - q_{mm}x_m^2,$$

hence bounds

$$LB(x_{m-1}) := \left\lceil -\left(\frac{T_{m-1}}{q_{m-1,m-1}}\right)^{1/2} - q_{m-1,m}x_m \right\rceil,$$

$$UB(x_{m-1}) := \left\lfloor \left(\frac{T_{m-1}}{q_{m-1,m-1}}\right)^{1/2} - q_{m-1,m}x_m \right\rfloor,$$

such that $LB(x_{m-1}) \leqslant x_{m-1} \leqslant UB(x_{m-1})$. Proceeding to $x_{m-2}, x_{m-3}, \ldots$ we obtain for fixed $x_m, x_{m-1}, \ldots, x_{k+1}$,

$$\sum_{i=1}^{k} q_{ii}\left(x_i + \sum_{j=i+1}^{m} q_{ij}x_j\right)^2 \leqslant T_k \tag{2.6}$$

with

$$T_k = T_{k+1} - q_{k+1,k+1}\left(x_{k+1} + \sum_{j=k+2}^{m} q_{k+1,j}x_j\right)^2 \tag{2.7}$$

$$(T_m = C;\ k = m-1, m-2, \ldots, 1).$$

These considerations lead to the following algorithm.

(2.8) *Algorithm for Solving* $Q(\mathbf{x}) \leqslant C$.

Input. Entries q_{ij} $(1 \leqslant i \leqslant j \leqslant m)$ of $Q(\mathbf{x})$ of (2.2) and a positive constant C.

Output. All $\mathbf{x} \in \mathbf{Z}^m$ subject to $\mathbf{x} \neq \mathbf{0}$ and $Q(\mathbf{x}) \leqslant C$ as well as $Q(\mathbf{x})$.

Step 1. (Initialization) Set $i \leftarrow m$, $T_i \leftarrow C$, $U_i \leftarrow 0$.

Step 2. (Bounds for x_i) Set $Z \leftarrow (T_i/q_{ii})^{1/2}$, $UB(x_i) \leftarrow \lfloor Z - U_i \rfloor$, $x_i \leftarrow \lceil -Z - U_i \rceil - 1$.

Step 3. (Increase x_i) Set $x_i \leftarrow x_i + 1$. For $x_i \leqslant UB(x_i)$ go to 5, else to 4.

Step 4. (Increase i) Set $i \leftarrow i + 1$ and go to 3.

Step 5. (Decrease i) For $i = 1$ go to 6. Else set $i \leftarrow i - 1$, $U_i \leftarrow \sum_{j=i+1}^{m} q_{ij}x_j$, $T_i \leftarrow T_{i+1} - q_{i+1,i+1}(x_{i+1} + U_{i+1})^2$ and go to 2.

Step 6. (Solution found) For $\mathbf{x} = \mathbf{0}$ terminate, else print $\mathbf{x}$, $-\mathbf{x}$, $Q(\mathbf{x}) = C - T_1 + q_{11}(x_1 + U_1)^2$ and go to 3.

Remark. We note that the highest nonvanishing coordinate of $\mathbf{x}$ is restricted to be negative, and we terminate in case $\mathbf{x} = \mathbf{0}$ is obtained. By then we know all solutions because of $Q(\mathbf{x}) = Q(-\mathbf{x})$. If we modify the task and want to obtain only those solutions $\mathbf{x}$ of (2.5) which additionally satisfy $Q(\mathbf{x}) \geqslant C' > 0$, we just need to change the bounds for the coordinate x_1 adequately. Unfortunately, this does not have a considerable effect on the computation time.

A preliminary version of (2.8) was already given in [7]. A comparison with the methods of U. Dieter [1] and D. Knuth [4], however, lead to a further improvement of (2.8). Namely, denoting the ith columns of R, $(R^{-1})^{\mathrm{tr}}$ by $\mathbf{r}_i$, $\mathbf{r}_i'$, respectively, we obtain for the ith coordinate x_i of a solution $\mathbf{x} \in \mathbf{Z}^{m\times 1}$ of (1.2)

$$x_i^2 = \left(\mathbf{r}_i'^{\mathrm{tr}}\left(\sum_{k=1}^{m} x_k\mathbf{r}_k\right)\right)^2 \leqslant \mathbf{r}_i'^{\mathrm{tr}}\mathbf{r}_i'(\mathbf{x}^{\mathrm{tr}}R^{\mathrm{tr}}R\mathbf{x}) \leqslant \|\mathbf{r}_i'\|^2 C \qquad (1 \leqslant i \leqslant m). \tag{2.9}$$

Hence, it is clear that reducing the rows of the matrix R^{-1} with respect to their length usually diminishes the range for the coordinates of possible candidates drastically. Common reduction methods are described in [1], [4], [5], [7]. We note that the reduction of Knuth [4] essentially coincides with the pair reduction algorithm of [7]. In any case the reduced version is obtained from R^{-1} by multiplying R^{-1} with a suitable unimodular matrix U^{-1} from the left to obtain $S^{-1} := U^{-1}R^{-1}$. Then, instead of solving $\mathbf{x}^{\mathrm{tr}}R^{\mathrm{tr}}R\mathbf{x} \leqslant C$, we solve

$$\mathbf{y}^{\mathrm{tr}}S^{\mathrm{tr}}S\mathbf{y} \leqslant C \tag{2.10}$$

by (2.8) and recover $\mathbf{x}$ from $\mathbf{y}$ via

$$\mathbf{x} = U\mathbf{y}. \tag{2.11}$$

A further improvement is obtained by reordering the columns of S, i.e., the order of the elimination process (2.3), adequately. Namely, in (2.8) it should be avoided that some segment $(x_m,\ldots,x_{k+1})$ cannot be extended to a solution $(x_m,\ldots,x_1)$. And the probability for this phenomenon decreases, if the range for $x_k,\ldots,x_1$ derived from (2.9) increases.

After these observations the following algorithm is immediate.

(2.12) *Improved Algorithm for Solving* $\mathbf{x}^{\mathrm{tr}}A\mathbf{x} \leqslant C$.

Input. $A \in \mathbf{R}^{m\times m}$ positive-definite, $C \in \mathbf{R}^{>0}$.

Output. All $\mathbf{x} \in \mathbf{Z}^m$ subject to $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{x}^{\mathrm{tr}}A\mathbf{x} \leqslant C$.

Step 1. (Cholesky decomposition of A) Compute the upper triangular matrices R, R^{-1} from A by (2.3), (2.4).

Step 2. (Reduction) Compute a row-reduced version S^{-1} of R^{-1} as well as $U^{-1} \in \mathrm{GL}(m,\mathbf{Z})$ subject to $S^{-1} = U^{-1}R^{-1}$. Compute $S = RU$.

Step 3. (Reorder columns of S) Determine a permutation $\pi \in \gamma_m$ such that $\|\mathbf{s}'_{\pi(1)}\| \geqslant \|\mathbf{s}'_{\pi(2)}\| \geqslant \cdots \geqslant \|\mathbf{s}'_{\pi(m)}\|$. Let S be the matrix with columns $\mathbf{s}_{\pi^{-1}(i)}$ $(1 \leqslant i \leqslant m)$.

Step 4. (Cholesky decomposition of $S^{\mathrm{tr}}S$) Compute the upper triangular matrix $Q = (q_{ij})$ from $S^{\mathrm{tr}}S$ by (2.3).

Step 5. (Application of (2.8)) Compute all solutions $\mathbf{y} \in \mathbf{Z}^m$, $\mathbf{y} \neq \mathbf{0}$, of $Q(\mathbf{y}) \leqslant C$ by (2.8) and print $\mathbf{x} = U(y_{\pi^{-1}(1)},\ldots,y_{\pi^{-1}(m)})^{\mathrm{tr}}$ for each $\mathbf{y}$.

The algorithm can easily be modified such that it computes all solutions $\mathbf{x}$ of $C' \leqslant \mathbf{x}^{\mathrm{tr}}A\mathbf{x} \leqslant C$ or a solution of (1.3).

Finally, we note that a similar algorithm for the solution of (1.2) can be developed starting from the computation of orthogonal vectors $\mathbf{b}_1^*,\ldots,\mathbf{b}_m^*$ from $\mathbf{b}_1,\ldots,\mathbf{b}_m$ by the Gram-Schmidt-orthogonalization procedure. This was pointed out to us by the referee whom we thank for useful hints.

3. Complexity Analysis. In this section we compare the enumeration techniques of [1], [4] with our algorithm (2.12) by estimating the number of arithmetic operations in both cases. We count each addition, multiplication, and extraction of a square-root as one operation.

Algorithm (2.12) produces at most as many lattice points as the enumeration techniques. This is an immediate consequence of (2.9). Whereas (2.12), respectively (2.8), needs at most $O(m^2)$ arithmetic operations to proceed from one vector $\mathbf{x}$ (or

part of it) to the next (by decreasing or increasing i and because of Step 5 of (2.8)), the enumeration method requires $O(m^2)$ arithmetic operations for computing $Q(\mathbf{x})$ for each $\mathbf{x}$. Hence, the complexity of both techniques is at most

$$O\left(m^2 \prod_{i=1}^{m} \left(2\left\lfloor \|\mathbf{r}_i'\| \sqrt{C} \right\rfloor + 1\right)\right). \tag{3.1}$$

(For the enumeration method this can eventually be improved by a factor m^{-1} using refined storing techniques.) However, (3.1) suggests that both algorithms are exponential in the input data, a disadvantage which is inherent in the problem itself. Namely, in case $A = I_m$ (the m-dimensional unit matrix) the solutions of (2.5) are the lattice points of $\mathbf{Z}^m$ in the m-dimensional ball centered at the origin of radius $\sqrt{C}$. It is well-known that the number of those lattice points is proportional to the volume of the ball and therefore increases with $\sqrt{C}^m$.

But what happens, if we keep C fixed and just increase m? Then the enumeration method is still exponential whereas—somewhat surprisingly—(2.12) is polynomial time, if we additionally require that the lengths of the rows of R^{-1} for the matrix R of the Cholesky decomposition $A = R^{\mathrm{tr}}R$ stay bounded.

To prove this, we first derive an upper bound for the number of tuples $(x_i, \ldots, x_m)$ generated by Algorithm (2.8) under the assumption that the input data $q_{\mu\mu}$ satisfy $q_{\mu\mu} \geqslant 1$ $(1 \leqslant \mu \leqslant m)$. We define

$$Q_i(x_i, \ldots, x_m) := \sum_{\mu=i}^{m} q_{\mu\mu} \left(x_\mu + \sum_{\nu=\mu+1}^{m} q_{\mu\nu} x_\nu \right)^2 \qquad (1 \leqslant i \leqslant m). \tag{3.2}$$

We know from (2.8) that for fixed $x_{i+1}, \ldots, x_m \in \mathbf{Z}$ subject to $Q_{i+1}(x_{i+1}, \ldots, x_m) \leqslant C$ there are at most

$$\left\lfloor 2(T_i/q_{ii})^{1/2} \right\rfloor + 1 \tag{3.3}$$

possibilities for $x_i \in \mathbf{Z}$, say $x_{i_1}, \ldots, x_{i_k}$, such that $Q_i(x_i, \ldots, x_m) \leqslant C$. We order the x_{i_j} according to

$$|x_{i_1} + U_i| \leqslant |x_{i_2} + U_i| \leqslant \cdots \leqslant |x_{i_k} + U_i|, \tag{3.4}$$

where $U_i = \sum_{\nu=i+1}^{m} q_{i\nu} x_\nu$ as in Step 5 of (2.8). Then it is easily seen that

$$(j-1)/2 \leqslant |x_{i_j} + U_i| \leqslant j/2 \qquad (1 \leqslant j \leqslant k). \tag{3.5}$$

This leads to an important recursive estimate for the numbers

$$P_i(r) := \#\left\{(x_i, \ldots, x_m) \in \mathbf{Z}^{m+1-i} \mid Q_i(x_i, \ldots, x_m) \leqslant r\right\} \tag{3.6}$$

for arbitrary $r \in \mathbf{R}^{\geqslant 0}$.

Indeed, (3.5) and $q_{\mu\mu} \geqslant 1$ $(1 \leqslant \mu \leqslant m)$ yield

$$P_i(r) \leqslant \sum_{j=0}^{\lfloor 2\sqrt{r} \rfloor} P_{i+1}\left(r - \frac{j^2}{4}\right) \leqslant \sum_{j=0}^{\lfloor 4r \rfloor} P_{i+1}(r - j/4). \tag{3.7}$$

We point out that (2.8) can produce segments $(x_{i+1}, \ldots, x_m) \in \mathbf{Z}^{m-i}$ subject to $Q_{i+1}(x_{i+1}, \ldots, x_m) \leqslant C$ for which no $x_i \in \mathbf{Z}$ with $Q_i(x_i, \ldots, x_m) \leqslant C$ exists. However, the total number of tuples generated by the algorithm is bounded by $\overline{P}_1(C)$,

where the numbers $\overline{P}_i(r)$ are defined by

$$\overline{P}_i(r) := \sum_{j=i}^{m} P_j(r) \qquad (1 \leqslant i \leqslant m). \tag{3.8}$$

Obviously, inequality (3.7) also holds for the $\overline{P}_i(r)$. This enables us to obtain an upper bound for $\overline{P}_1$ in terms of $\overline{P}_m$ by computing coefficients β_{ij} such that

$$\overline{P}_1(C) \leqslant \sum_{j=0}^{\lfloor 4C \rfloor} \beta_{ij} \overline{P}_i(C - j/4) \qquad (1 \leqslant i \leqslant m). \tag{3.9}$$

As initial values we get

$$\beta_{10} = 1, \qquad \beta_{1j} = 0 \quad \text{for } j > 0, \tag{3.10}$$

and if the β_{ij} $(0 \leqslant j \leqslant \lfloor 4C \rfloor)$ are known for fixed i, then (3.7) applied to $\overline{P}_i(r)$ yields the recursive formula

$$\beta_{i+1,j} = \sum_{k=0}^{j} \beta_{ik} \qquad (0 \leqslant j \leqslant \lfloor 4C \rfloor). \tag{3.11}$$

Hence, we can compute the β_{ij} row by row.

(3.12) LEMMA. *The numbers* $\beta_{ij} \in \mathbf{Z}^{\geqslant 0}$ *defined by* (3.10) *and* (3.11) *satisfy*

$$\beta_{i+1,j} = \prod_{k=1}^{i-1} \frac{j+k}{k} \qquad (i \in \mathbf{N}, j \in \mathbf{Z}^{\geqslant 0}).$$

Proof. We show the lemma by induction on i. For $i = 1$ we have

$$\prod_{k=1}^{i-1} \frac{j+k}{k} = 1 = \sum_{k=0}^{j} \beta_{1k} = \beta_{2j}.$$

For fixed $i \geqslant 1$, we obtain

$$\beta_{i+2,j} = \sum_{k=0}^{j} \beta_{i+1,k} = \sum_{k=0}^{j} \prod_{l=1}^{i-1} \frac{k+l}{l} = \prod_{k=1}^{i} \frac{j+k}{k}$$

because of

$$\sum_{k=0}^{j} \prod_{l=1}^{i-1} (k+l) = \frac{1}{i} \prod_{l=1}^{i} (j+l).$$

But the last equation is valid for arbitrary $j \in \mathbf{Z}^{\geqslant 0}$, $i \in \mathbf{N}$ and is itself proved by induction on j. □

Using Lemma (3.12) and (3.3) we easily obtain the following upper bound for $\overline{P}_1(C)$ from (3.8):

$$\begin{aligned} \overline{P}_1(C) &\leqslant \sum_{j=0}^{\lfloor 4C \rfloor} \beta_{mj} \overline{P}_m(C - j/4) = \sum_{j=0}^{\lfloor 4C \rfloor} \left(\prod_{k=1}^{m-2} \frac{j+k}{k} \right) \left(2\lfloor (C - j/4)^{1/2} \rfloor + 1 \right) \\ &\leqslant \left(2\lfloor C^{1/2} \rfloor + 1 \right) \sum_{j=0}^{\lfloor 4C \rfloor} \prod_{k=1}^{m-2} \frac{j+k}{k}. \end{aligned}$$

By the usual relations for binomial coefficients, we get

$$\overline{P}_1(C) \leqslant \left(2\lfloor C^{1/2} \rfloor + 1 \right) \binom{\lfloor 4C \rfloor + m - 1}{\lfloor 4C \rfloor}, \tag{3.13}$$

and, for large m, Stirling's formula yields that $\overline{P}_1(C)$ increases at most like

$$O\left(\left(1 + \frac{m-1}{\lfloor 4C \rfloor}\right)^{\lfloor 4C \rfloor}\right). \tag{3.14}$$

Now it is easy to give an upper bound for the total number of arithmetic operations used by (2.12).

(3.15) THEOREM. *Let $C \in \mathbf{R}^{>0}$ and $A \in \mathbf{R}^{m\times m}$ be positive-definite. Let d^{-1} be a lower bound for the entries $q_{\mu\mu}$ $(1 \leqslant \mu \leqslant m)$ computed from A by* (2.3). *Then Algorithm* (2.12) (*without Steps* 1, 2, 3) *uses at most*

$$\frac{1}{6}(2m^3 + 3m^2 - 5m) + \frac{1}{2}(m^2 + 12m - 7)\left((2\lfloor\sqrt{Cd}\rfloor + 1)\binom{\lfloor 4Cd \rfloor + m - 1}{\lfloor 4Cd \rfloor} + 1\right) \tag{3.16}$$

arithmetic operations for computing all $\mathbf{x} \in \mathbf{Z}^m$ subject to $\mathbf{x}^{\mathrm{tr}}A\mathbf{x} \leqslant C$.

Proof. The computation of the q_{ij} $(1 \leqslant i \leqslant j \leqslant m)$ in (2.3) requires

$$\frac{1}{6}(2m^3 + 3m^2 - 5m)$$

arithmetic operations. The number of tuples obtained by Algorithm (2.8) was estimated under the premise $q_{\mu\mu} \geqslant 1$ $(1 \leqslant \mu \leqslant m)$. However, the case $\min\{q_{\mu\mu} \mid 1 \leqslant \mu \leqslant m\} = 1/d < 1$ is tantamount to this situation if we only replace the constant C by Cd. During the execution of Step 5 of (2.12), i.e., of Algorithm (2.8), we obtain at most $\frac{1}{2}(\overline{P}_1(Cd) + 1)$ vectors $(x_1, \ldots, x_m) \in \mathbf{Z}^m$ (or segments of such vectors), since the highest nonvanishing coefficient is restricted to be negative. The transition from one vector (or segment) to the next requires at most $m^2 + 12m - 12$ arithmetic operations (in Steps 2 to 5 of (2.8)). For admissible vectors $\mathbf{x} \in \mathbf{Z}^m$, the computation of $Q(\mathbf{x})$ consists of 5 arithmetic operations (though the computation of $Q(\mathbf{x})$ is actually not necessary since $Q(\mathbf{x}) \leqslant C$ is guaranteed by the algorithm). □

(3.17) COROLLARY. *Let $C \in \mathbf{R}^{>0}$ and $A \in \mathbf{R}^{m\times m}$ be positive-definite. Let $A = R^{\mathrm{tr}}R$ be the Cholesky decomposition of A and $d > 0$ be an upper bound for the square of the norms of the rows of R^{-1}. Then* (3.16) *is an upper bound for the number of arithmetic operations for computing all $\mathbf{x} \in \mathbf{Z}^m$ subject to $\mathbf{x}^{\mathrm{tr}}A\mathbf{x} \leqslant C$ by* (2.12) (*without Steps* 1 *to* 3).

Proof. We denote the rows of R^{-1} by $\mathbf{r}_i'^{\mathrm{tr}}$ as in the preceding section. Multiplying all entries of R by $d^{1/2}$ then implies $\|\mathbf{r}_i'\| \leqslant 1$ $(1 \leqslant i \leqslant m)$ for the row-vectors $\mathbf{r}_i'^{\mathrm{tr}}$ of the scaled matrix R^{-1}. If r_{ii} is the ith diagonal entry of R, then $1/r_{ii}$ is the ith diagonal entry of R^{-1}. Hence, $q_{ii} \geqslant 1$ is equivalent to $1/r_{ii} \leqslant 1$ because of (2.5). But the latter is certainly correct because of $1/r_{ii} \leqslant \|\mathbf{r}_i'\| \leqslant 1$. □

We note that the corollary remains valid if we replace the Cholesky decomposition $A = R^{\mathrm{tr}}R$ by any decomposition $A = B^{\mathrm{tr}}B$ $(B \in \mathbf{R}^{m\times m})$ and $\sqrt{d}$ by an upper bound for the norms of the rows of B^{-1}.

The upper bound (3.16) for the number of arithmetic operations does not contain those operations carried out in Steps 1 to 3 of Algorithm (2.12). We note that the

algorithm works also without those steps (and is then essentially (2.8)). The operations of Steps 1 and 3 are comparable to those of computing the q_{ij} from A in (2.3) and are therefore negligible. The number of operations required by the reduction Step 2 of course depends on the method applied. If we use the reduction algorithm of Lenstra, Lenstra and Lovasz, for example, it can be estimated without difficulties. That algorithm also turned out to be the best reduction method for our numerical examples in the next section.

4. Numerical Investigations. The two lists presented in this section show that our method of combining a reduction algorithm with a (2.8)-type procedure is indeed very efficient. We introduce several suggestive abbreviations:

RLLL: Reduction algorithm of Lenstra, Lenstra and Lovasz [5].

RDIE: Reduction algorithm of U. Dieter [1].

RKNU: Reduction algorithm of D. E. Knuth [4].

We consider numerical examples for problems of type (1.2). The input consists of a randomly generated matrix $B \in \mathbf{R}^{m \times m}$ and a positive constant C. In a first step B^{-1} is computed. Optionally, one of the reduction procedures RLLL, RDIE, RKNU is then applied to the rows of B^{-1}. The routine RENU "enumerates" all lexicographically negative vectors of $\mathbf{Z}^m$ subject to (2.9) (with B in place of R) and tests whether they solve (1.2). The routine RCHO carries out Steps 3 to 5 of (2.12) with B in place of S.

The numbers in the two lists below are the CPU-time for ten examples of dimension m each. The symbol "–" means that no computations were carried out since no results could be expected in a reasonable amount of time. "$> t$" means that in t seconds not all ten examples could be computed. "$\gg t$" finally means that in t seconds no solution vector of the first example was obtained. (All examples considered had nontrivial solutions because of the choice of C.)

(4.1) *List.* The entries b_{ij} $(1 \leqslant i, j \leqslant m)$ of the matrix B of (1.2) are independent, uniformly distributed variables in the interval $[0, 1]$; $C := 0.2 + 0.07(m - 5)$; CPU-time in seconds.

	$m = 5$	$m = 10$	$m = 15$	$m = 20$	$m = 25$
RENU	> 4000.	–	–	–	–
RKNU + RENU	0.056	5.736	–	–	–
RDIE + RENU	0.059	7.515	–	–	–
RLLL + RENU	0.057	4.873	> 16000.	–	–
RCHO	0.046	0.133	1.048	2.071	70.025
RKNU + RCHO	0.044	0.144	0.646	1.768	64.562
RDIE + RCHO	0.047	0.158	0.589	1.432	56.112
RLLL + RCHO	0.043	0.129	0.464	1.086	16.544

(4.2) *List.* The entries b_{ij} $(1 \leqslant i, j \leqslant m)$ of the matrix B of (1.2) are independent, randomly generated variables, where the b_{ij} $(2 \leqslant i \leqslant m - 1)$ are uniformly distributed in $[0, 1]$ and the b_{1j}, b_{mj} are uniformly distributed in $[0, 1/m]$ $(1 \leqslant j \leqslant m)$; $C := 0.05 + 0.04(m - 5)$; CPU-time in seconds.

	$m = 5$	$m = 10$	$m = 15$	$m = 20$
RENU	$\gg 254.$	–	–	–
RKNU + RENU	0.040	8.580	$\gg 254.$	–
RDIE + RENU	0.045	14.563	$\gg 254.$	–
RLLL + RENU	0.043	6.558	$\gg 254.$	–
RCHO	0.043	0.404	5.063	13.602
RKNU + RCHO	0.040	0.205	1.134	2.626
RDIE + RCHO	0.045	0.218	0.735	1.548
RLLL + RCHO	0.039	0.188	0.573	0.909

Our computational investigations show that the enumeration strategy RENU favored by U. Dieter and—implicitly—by D. E. Knuth yields acceptable results in comparison with algorithm RCHO only if a preceding reduction procedure succeeds in reducing the initial box into a box of very small volume. But this is not possible in general. It turned out that the combination of the Lenstra, Lenstra and Lovasz algorithm with routine RCHO was the fastest of the algorithms considered and also produced the smallest boxes. This was a little surprising since the theoretical results on RLLL are not very strong except for the polynomial time behavior. Clearly, we recommend to use RLLL + RCHO for solving problems of type (1.2). The amount of computation time, however, hardly depends on the choice of the reduction subroutine but mainly on the use of RCHO instead of RENU. RCHO is obviously the only method among the ones considered which is suited for handling problems in higher dimensions.

Mathematisches Institut
Universität Düsseldorf
Universitätsstr. 1
4 Düsseldorf 12, Federal Republic of Germany

1. U. DIETER, "How to calculate shortest vectors in a lattice," *Math. Comp.*, v. 29, 1975, pp. 827–833.

2. U. FINCKE & M. POHST, "Some applications of a Cholesky-type method in algebraic number theory." (To appear.)

3. U. FINCKE & M. POHST, *On Reduction Algorithms in Non Linear Integer Mathematical Programming*, DGOR-Operations Research Proceedings 83, Springer-Verlag, Berlin and New York, 1983, pp. 289–295.

4. D. E. KNUTH, *The Art of Computer Programming*, Vol. II, 2nd ed., Addison-Wesley, Reading, Mass., 1981, pp. 95–97.

5. A. K. LENSTRA, H. W. LENSTRA, JR. & L. LOVASZ, "Factoring polynomials with rational coefficients," *Math. Ann.*, v. 261, 1982, pp. 515–534.

6. A. ODLYZKO, "Cryptoanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature system." Preprint, 1983.

7. M. POHST, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bull.*, v. 15, 1981, pp. 37–44.