Safe and Effective Determinant Evaluation

Kenneth L. Clarkson
AT&T Bell Laboratories
Murray Hill, New Jersey 07974
e-mail: clarkson@research.att.com

February 25, 1994

Abstract

The problem of evaluating the sign of the determinant of a small matrix arises in many geometric algorithms. Given an $n \times n$ matrix A with integer entries, whose columns are all smaller than M in Euclidean norm, the algorithm given here evaluates the sign of the determinant $\det A$ exactly. The algorithm requires an arithmetic precision of less than $1.5n + 2\lg M$ bits. The number of arithmetic operations needed is $O(n^3) + O(n^2)\log \mathcal{OD}(A)/\beta$, where $\mathcal{OD}(A)|\det A|$ is the product of the lengths of the columns of A, and β is the number of "extra" bits of precision,

$$\min\{\lg(1/\mathbf{u}) - 1.1n - 2\lg n - 2, \lg N - \lg M - 1.5n - 1\},$$

where $\mathbf{u}$ is the roundoff error in approximate arithmetic, and N is the largest representable integer. Since $\mathcal{OD}(A) \leq M^n$, the algorithm requires $O(n^3 \lg M)$ time, and $O(n^3)$ time when $\beta = \Omega(\log M)$.

1 Introduction

Many geometric algorithms require the evaluations of the determinants of small matrices, and testing the signs ($\pm$) of such determinants. Such testing is fundamental to algorithms for finding convex hulls, arrangements of lines and line segments and hyperplanes, Voronoi diagrams, and many others. By such numerical tests, a combinatorial structure is defined. If the tests are incorrect, the resulting structure may be wildly different from a "sensible" result[11, 6], and programs for computing the structures may crash.

Two basic approaches to this problem have been proposed: the use of exact arithmetic, and the design of algorithms to properly use inaccurate numerical tests. While the former solution is general, it can be quite slow: naively, n-tuple precision appears to be necessary for exact computation of the determinant of an $n \times n$ matrix with integer entries. While "adaptive" precision has been proposed and used[7], the resulting code and time complexity are still substantially larger than one might hope.

The second approach is the design (or redesign) of algorithms to use limited precision arithmetic and still guarantee sensible answers. (For example, [4, 2, 13, 6, 14, 12, 11, 9, 17].) Such an approach can be quite satisfactory, when obtainable, but seems applicable only on an *ad hoc* and limited basis: results for only a fairly restricted collection of algorithms are available so far.

This paper takes the approach of exact evaluation of the sign of the determinant, or more generally, the evaluation of the determinant with low relative error. However, the algorithm requires relatively low precision: less than $3n/2$ bits, and additionally some number more bits than were used to specify the matrix entries. The algorithm is naturally "adaptive," in the sense that the running time is proportional to the logarithm of the *orthogonality defect* $\mathcal{OD}(A)$ of the matrix A, where

$$\mathcal{OD}(A) \equiv \frac{\prod_{1 \leq i \leq n} \|a_i\|}{|\det A|}.$$

Here a_i is the ith column of A. (We let $\|a_i\| \equiv \sqrt{a_i^2}$ denote the Euclidean norm of a_i, with $a_i^2 \equiv a_i \cdot a_i$.) In geometric problems, the orthogonality defect may be small for most matrices, so such adaptivity should be a significant advantage in running time. Note that the limited precision required by the algorithm implies that "native" machine arithmetic can be used and still allow the input precision to be reasonably high. For example, the algorithm can handle 10×10 matrices with 32-bit entries in the 53 bits available in double precision on many modern machines.

The algorithm is amenable to "rank-one updates," so $\det B$ can be evaluated quickly (in about $O(n^2)$ time) after $\det A$ is evaluated, if B is the same as A in all but one column.

One limitation of the approach here is that the matrix entries are required to be integers, rather than say rational numbers. This may entail preliminary scaling and rounding of input to a geometric algorithm; this limitation can be ameliorated by allowing input expressed in homogeneous coordinates, as noted by Fortune[1]. The resulting output is plainly the output for inputs that are "near" the originals, so such an algorithm is *stable* in Fortune's sense.[2]

The new algorithm uses ideas from Lovász's basis reduction scheme [8, 10, 16], and can be viewed as a "low-rent" version of that algorithm: a result of both algorithms is a new matrix A', produced by elementary column operations, whose columns are roughly orthogonal. However, basis reduction does not allow a column of A to be replaced by an integer multiple of itself: the resulting set of matrix columns generates only a sublattice of the lattice generated by the columns of A. Since here only the determinant is needed, such a scaling of a column is acceptable since the scaling factor can be divided out of the determinant of the resulting matrix. Thus the problem solved here is easier than basis reduction, and the results are sharper with respect to running time and precision. The algorithm here yields a QR factorization of A, no matter what the condition of A, and so may be of interest in basis reduction, since Lovász's algorithm starts by finding such a factorization. In practice, that initial factorization is found as an approximation using floating-point arithmetic; if the

```
matrix MGS(matrix A)
{
    for k := 1 upto n do
        c_k := a_k;
        for j := k − 1 downto 1 do c_k −= a_j(c_k/c_j);
    return C;
}
```

Figure 1: A version of the modified Gram-Schmidt procedure.

matrix is ill-conditioned, the factorization procedure may fail.

Schnorr has given an algorithm for basis reduction requiring $O(n + \lg M)$ bits; his algorithm for this harder problem is more complicated, and his constants are not as sharp.[15]

2 The algorithm

The algorithm is an adaptation of the modified Gram-Schmidt procedure for computing an orthogonal basis for the linear subspace spanned by a set of vectors. The input is the matrix $A = [a_1\ a_2 \ldots a_n]$ with the column n-vectors a_i, $i = 1 \ldots n$. One version of the modified Gram-Schmidt procedure is shown in Figure 1. We use the operation a/b for two vectors a and b, with $a/b \equiv (a \cdot b)/b^2$. Note that $(a+b)/c = a/c + b/c$, and $(a - (a/c)c) \cdot c = 0$. Implemented in exact arithmetic, this procedure yields an orthogonal matrix $C = [c_1\ c_2 \ldots c_n]$, so $c_i \cdot c_j = 0$ for $i \neq j$, such that $a_k = c_k + \sum_{1 \leq j < k} R_{jk} c_j$, for some values $R_{jk} = a_k/c_j$. (In variance with some usage, in this paper orthogonal matrices have pairwise orthogonal columns: they need not have unit length. If they do, the matrix will be termed *orthonormal.*)

The vector c_k is initially a_k, and is then *reduced* by the c_j components of a_k: after step j of the inner loop, $c_k \cdot c_j = 0$. Note that even though a_j is used in the reduction, rather than c_j, the condition $c_k \cdot c_i = 0$ for $j < i < k$ is unchanged, since a_j has no c_i components for $i > j$.

Since $A = CR$ where R is unit upper triangular, $\det A = \det C$; since C is an orthogonal matrix, $|\det C| = \prod_{1 \leq j \leq n} \|c_j\|$, and it remains only to determine the sign of the determinant of a perfectly conditioned matrix.

When the same algorithm is used with floating-point arithmetic to find a matrix B approximating C, the algorithm may fail if at some stage k a column b_k of B is very small: the vector a_k is very nearly a linear combination of the vectors $\{a_1, \ldots, a_{k-1}\}$: it is close to the *linear span* of those vectors. Here the usual algorithm might simply halt and return the answer that the determinant is nearly zero. However, since b_k is small, we can multiply a_k by some large scaling factor s, reduce sa_k by its c_j components for $j < k$, and have a small resulting vector. Indeed, that vector will be small even if we reduce using rounded coefficients for a_j. With such integer coefficients, and using the columns

```
float det_safe(matrix A)
{
    float denom := 1;
          S^b := 0;
    for k := 1 upto n do
        loop
            b_k := a_k;
            for j := k-1 ... 1  b_k -= a_j(b_k/b_j);

            if a_k^2 <= 2 b_k^2 then {S^b += b_k^2; exit_loop;}

            s := S^b/4(b_k^2 + 2 δ_k a_k^2);
            s := min{s, (N/1.58^k - √(S^b))^2 / a_k^2};
            if s < 1/2 then s := 1 else s := ⌈√s⌋;
            if s = 1 and a_k^2 <= 0.9 S^b then s := 2;

            denom *= s;
            a_k *= s;
            for j := k-1 ... 1  a_k -= a_j ⌈(a_k/b_j)⌋;
            if a_k = 0 then return 0;
        end_loop;
    return det_approx(B)/denom;
}
```

Figure 2: An algorithm for estimating $\det A$ with small relative error.

a_j and exact arithmetic for the reduction steps, $\det A$ remains unchanged, except for being multiplied by the scaling factor s.

Thus the algorithm given here proceeds in n stages as in modified Gram-Schmidt, processing a_k at stage k. First the vector b_k is found, estimating the component of a_k orthogonal to the linear span of $\{a_1, \ldots, a_{k-1}\}$. A scaling factor s inversely proportional to $\|b_k\|$ is computed, and a_k is replaced by sa_k and then reduced using exact elementary column operations that approximate Gram-Schmidt reduction. The result is a new column $a_k = s\hat{c}_k + \sum_{1 \leq j < k} \alpha_j c_j$, where $\hat{c}_k$ is the old value of that vector, and the coefficients $\alpha_j \approx 1/2$. Hence the new a_k is "more orthogonal" to earlier vectors than the old one. The processing of a_k repeats until a_k is nearly orthogonal to the previous vectors $a_1, \ldots, a_{k-1}$, as indicated by the condition $a_k^2 \leq 2b_k^2$.

The algorithm is shown in Figure 2. All arithmetic is approximate, with unit roundoff $\mathbf{u}$, except the all-integer operations $a_k \mathrel{*}= s$ and the reduction steps for a_k (although a_k/b_j is computed approximately). The value N is no more than the largest representable integer. For a real number x, the integer $\lceil x \rceil$ is the least integer no smaller than x, and $\lceil x \rfloor$ is $\lceil x - 1/2 \rceil$. The quantity δ_k is defined in Notation 3.4 below.

The precision requirements of the algorithm can be reduced somewhat by "reorthogonalization," that is, simply applying the reduction step to b_k again after computing it, or just before exiting the loop for stage k. Hoffman discusses and analyzes this technique; his analysis can be sharpened in our situation.[5]

Note that the condition $a_k^2 \leq 2b_k^2$ may hold frequently or all the time; if the latter, the algorithm is very little more than Modified Gram-Schmidt, plus the procedure for finding the determinant of the very-well-conditioned matrix B.

3 Analysis

The analysis requires some elementary facts about the error in computing the dot product and Euclidean norm.

Lemma 3.1 *For n-vectors a and b and using arithmetic with roundoff $\mathbf{u}$, $a \cdot b$ can be estimated with error at most $1.01n\mathbf{u}\|a\|\|b\|$, for $n\mathbf{u} \leq .01$. The relative error in computing a^2 is therefore $1.01n\mathbf{u}$, under these conditions. The termination condition for stage j implies $a_j^2 \leq 2b_j^2(1 + 2.03n\mathbf{u})$.*

Proof: For a proof of the first statement, see [3], p. 35. The remaining statements are simple corollaries. ∎

We'll need a lemma that will help bound the error due to reductions using b_j, rather than c_j.

Lemma 3.2 *For vectors a, b, and c, let $d \equiv b - c$, and let $\delta \equiv \|d\|/\|b\|$. Then*

$$a/b - a/c \leq \|a\|\|d\|/\|c\|\|b\| = \delta\|a\|/\|c\|.$$

Note also $a/b \leq \|a\|/\|b\|$.

Proof: The Cauchy-Schwartz inequality implies the second statement, and also implies

$$a/b - a/c \leq \|a\|\|b/b^2 - c/c^2\|.$$

Using elementary manipulations

$$\frac{b}{b^2} - \frac{c}{c^2} = \frac{(dc^2 - cd \cdot c) - c(d \cdot c + d^2)}{c^2 b^2}.$$

The two terms in the numerator are orthogonal, so the norm of this vector is

$$\frac{\sqrt{(dc^2 - cd \cdot c)^2 + c^2(d \cdot c + d^2)^2}}{c^2 b^2} = \frac{\sqrt{c^2 d^2 b^2}}{c^2 b^2} = \delta/\|c\|,$$

as desired. ∎

Lemma 3.3 *For vectors a, b and c, and d, and δ as in the previous lemma, suppose $a^2 \leq 2b^2(1 + \alpha)$, $a/c = 1$, and $\alpha < 1$. Then $\|a - b\|/\|b\| \leq 1 + 4\delta + \alpha$.*

Proof: We have

$$\frac{(a-b)^2}{b^2} = \frac{a^2 - 2a \cdot b + b^2}{b^2} \leq 3 + 2\alpha - 2a \cdot b/b^2,$$

using $a^2 \leq 2b^2(1+\alpha)$. By the assumption $a/c = 1$,

$$\frac{a \cdot b}{b^2} = \frac{a \cdot (c+d)}{b^2} = \frac{c^2 + a \cdot d}{b^2} \geq (1-\delta)^2 - \sqrt{2(1+\alpha)},$$

where the last inequality follows using the triangle and Cauchy-Schwartz inequalities and the assumption $a^2 \leq 2b^2(1+\alpha)$. Hence

$$\frac{(a-b)^2}{b^2} \leq 3 + 2\alpha - 2((1-\delta)^2 - \sqrt{2(1+\alpha)}\delta) \leq 1 + 2\alpha + 8\delta,$$

giving $\|a-b\|/\|b\| \leq 1 + 4\delta + \alpha$, as claimed. ∎

Notation 3.4 *Let*

$$\eta \equiv 3(n+2)\mathbf{u},$$

and for $j = 1 \ldots n$, *let*

$$\delta_j \equiv L_j \eta - \eta,$$

where

$$L_j^2 \equiv j\mu^2(\mu^2 + 2.04)^{j-1},$$

and

$$\mu \equiv 1.58.$$

Let

$$\phi \equiv 1 + 2.04/\mu^2.$$

Let

$$d_k \equiv b_k - c_k$$

for $k = 1 \ldots n$.

Theorem 3.5 *Suppose the reductions for* b_k *are performed using rounded arithmetic with unit roundoff* $\mathbf{u}$. *Assume that* $(n+2)\mathbf{u} < .01$ *and* $\delta_n < 1/32$. *(For the latter,* $\lg(1/\mathbf{u}) \geq 1.1n + 2\lg n + 7$ *suffices.) Then:*

(i) $\|b_k - a_j(b_k/b_j)\| \leq 1.54\|b_k\|$.

(ii) *The computed value of a single reduction step for* b_k, *or* $b_k - (b_k/b_j)a_j$, *differs from the exact value by a vector of norm no more than* $\eta\|b_k\|$. *Before a reduction of* b_k *by* a_j, $\|b_k\| \leq \|a_k\|\mu^{k-1-j}$.

(ii) *After the reduction loop for* b_k, d_k *satisfies*

$$\|d_k\| \leq \delta_k \|a_k\|/\sqrt{2},$$

and after stage k, $\|d_k\| \leq \delta_k\|b_k\|$.

Proof: First, part (i): let

$$a'_j \equiv (a_j - b_j) - ((a_j - b_j)/b_j)b_j,$$

which is orthogonal to b_j, and express $b_k - a_j(b_k/b_j)$ as

$$\begin{aligned} b_k &- (b_k/b_j)b_j \\ &- (b_k/b_j)a'_j \\ &- (b_k/b_j)((a_j - b_j)/b_j)b_j. \end{aligned}$$

Since $b_k^2 = \alpha^2 + \beta^2 b_j^2 + \gamma^2 (a'_j)^2$, for some α, $\beta = b_k/b_j$, and $\gamma = b_k/a'_j$, we have

$$(b_k - (b_k/b_j)(b_j + a'_j))^2 = \alpha^2 + (\gamma - \beta)^2 (a'_j)^2,$$

and so

$$(b_k - (b_k/b_j)(b_j + a'_j))^2/b_k^2 \leq (\gamma - \beta)^2/(\beta^2 b_j^2/(a'_j)^2 + \gamma^2),$$

which has a maximum value

$$1 + (a'_j)^2/b_j^2 \leq 2 + 4\delta_j + 2.03n\mathbf{u},$$

and so

$$\|b_k - (b_k/b_j)(b_j + a'_j)\| \leq 1.47\|b_k\|. \tag{1}$$

Using part (ii) inductively and Lemmas 3.3 and 3.1, and the assumption $\delta_j \leq 1/32$,

$$\begin{aligned} &\|(b_k/b_j)((a_j - b_j)/b_j)b_j\| \\ &\quad \leq \frac{\|b_k\|}{b_j^2}(a_j - c_j - d_j) \cdot (c_j + d_j) \\ &\quad = \frac{\|b_k\|}{b_j^2}(-c_j \cdot d_j + d_j \cdot (a_j - b_j)) \\ &\quad \leq \|b_k\|(\delta_j(1 + \delta_j) + \delta_j(1 + 4\delta_j + 2.03n\mathbf{u})) \\ &\quad \leq \|b_k\|\delta_j(2.15) \end{aligned}$$

Thus with (1),

$$\|b_k - a_j(b_k/b_j)\| \leq (1.47 + 2.15/32)\|b_k\| \leq 1.54\|b_k\|.$$

This completes part (i).

Now for part (ii). From Lemma 3.1 and Lemma 3.2, the error in computing $b_k/b_j = b_k \cdot b_j/b_j^2$ is at most

$$((1 + \mathbf{u})(1 + 2.03n\mathbf{u}) - 1)\|b_k\|/\|b_j\|,$$

and $\|a_j\| \leq \sqrt{2}(1 + 1.02n\mathbf{u})\|b_j\|$, and so the difference between $a_j(b_k \cdot b_j/b_j^2)$ and its computed value is a vector ϵ with norm no more than $\|b_k\|$ times

$$\begin{aligned} &\sqrt{2}(1 + 1.02n\mathbf{u})((1 + \mathbf{u})^2(1 + 2.03n\mathbf{u}) - 1) && (2) \\ &\qquad < 2.9002(n + 1)\mathbf{u} && (3) \end{aligned}$$

A reduction step computes the nearest floating-point vector to $b_k - a_j(b_k/b_j) + \epsilon$. Hence the error is a vector $\epsilon + \epsilon'$, where

$$\begin{aligned} \|\epsilon'\| &\leq \mathbf{u}\|b_k - a_j(b_k/b_j) + \epsilon\| \\ &\leq \mathbf{u}(\|b_k - a_j(b_k/b_j)\| + \|\epsilon\|). \end{aligned}$$

With this and (2), the computed value of b_k after the reduction step, differs from its real value by a vector that has norm no more than $\|b_k\|$ times

$$\begin{aligned} &2.9002(n+1)\mathbf{u}(1+\mathbf{u}) + 1.54\mathbf{u} \\ &\quad < 3(n+2)\mathbf{u} \\ &\quad = \eta. \end{aligned}$$

Using part (i), the norm of b_k after the reduction step is no more than $(1.54+\eta)\|b_k\| < \mu\|b_k\|$.

We turn to part (iii), using (ii) and inductively (iii) for $j < k$. (We have $d_1 = 0$ for the inductive basis.)

The vector b_k is initially a_k, which is c_k plus a vector in the linear span of $a_1 \ldots a_{k-1}$, which is the linear span of $c_1 \ldots c_{k-1}$. When b_k is reduced by a_j, it is replaced by a vector

$$b_k - (b_k/b_j)a_j + \epsilon_j, \tag{4}$$

where ϵ_j is the roundoff. Except for ϵ_j, the vector $b_k - c_k$ continues to be in the linear span of $\{c_1 \ldots c_{k-1}\}$. Hence $d_k \equiv b_k - c_k$ can be expressed as $e_k + \Upsilon$, where $e_k = \sum_{1\leq j<k} \gamma_j c_j$, for some values γ_j, and Υ is no longer than the sum of the roundoff vectors, so

$$\|\Upsilon\| \leq \eta\|a_k\|\mu^{k-1}/(\mu - 1), \tag{5}$$

using part (ii). From the triangle inequality,

$$d_k^2 \leq (\|e_k\| + \Upsilon)^2. \tag{6}$$

The quantity γ_j is

$$\begin{aligned} &\frac{1}{c_j} \cdot [b_k - (b_k/c_j)a_j + (b_k/c_j)a_j - (b_k/b_j)a_j] \\ &\qquad = b_k/c_j - b_k/b_j, \end{aligned}$$

where b_k is the value of that vector for the a_j reduction. (Note that except for roundoff, b_k/c_j does not change after the reduction by a_j.) By Lemma 3.2 and this part of Theorem 3.5 as applied to d_j,

$$|\gamma_j| \leq \|b_k\|\delta_j/\|c_j\|, \tag{7}$$

Using part (ii) and (7),

$$\begin{aligned} e_k^2 &= \sum_{1\leq j<k} \gamma_j^2 c_j^2 \\ &\leq \sum_{1\leq j<k} \delta_j^2 \mu^{2(k-1-j)} a_k^2. \end{aligned}$$

Using (5),(6), and $\delta_j < L_j\eta$,

$$\frac{d_k^2}{a_k^2} \le \mu^{2k}\frac{\eta^2}{\mu^2}\left(\frac{1}{\mu-1} + \sqrt{\sum_{1\le j<k} L_j^2/\mu^{2j}}\right)^2. \tag{8}$$

It suffices for part (iii) that

$$\delta_k \ge \mu^k\sqrt{2(1+\eta)}\frac{\eta}{\mu}\left(\frac{1}{\mu-1} + \sqrt{\sum_{1\le j<k} L_j^2/\mu^{2j}}\right),$$

where the $(1+\eta)$ term allows the statement

$$d_k^2 \le \delta_k^2 a_k^2/2(1+2.03n\mathbf{u}). \tag{9}$$

Letting $M_k = L_k/\mu^k$, we need

$$M_k \ge \frac{1}{\mu^k} + \frac{\sqrt{2(1+\eta)}}{\mu}\left(\frac{1}{\mu-1} + \sqrt{\sum_{1\le j<k} M_j^2}\right),$$

Since $L_1 = 1$ allows

$$\|d_1\| = 0 = \delta_1 = \eta - \eta,$$

$M_k = \sqrt{j(1+2(1+\eta)/\mu^2)^{k-1}}$, or $L_k^2 = k\mu^2(\mu^2+2.04)^{k-1}$ for $k>1$, gives sufficiently large δ_k, using the facts

$$\sum_{1\le j<k} jx^{j-1} = \frac{kx^{k-1}}{x-1} - \frac{x^k-1}{(x-1)^2}$$

and

$$\sqrt{x-y} \le \sqrt{x} - y/2\sqrt{x}.$$

The last statement of part (iii) follows from the termination condition for stage k, which from Lemma 3.1 implies $a_k^2 \le 2b_k^2(1+2.03n\mathbf{u}) < 2b_k^2(1+\eta)$. ∎

Lemma 3.6 *With Notation 3.4 and the assumptions of Theorem 3.5, let* $S_k^c \equiv \sum_{1\le j<k} c_j^2$ *and* S_k^b *the computed value of* $\sum_{1\le j<k} b_j^2$. *Then*

$$|S_k^c - S_k^b|/S_c^k < 1/10.$$

Proof: With Theorem 3.5(iii), we have

$$b_j^2/c_j^2 \le 1/(1-\delta_j)^2,$$

implying

$$\sum_{1\le j<k} b_j^2/S_k^c \le 1/(1-\delta_j)^2.$$

Lemma 3.1 implies $S_k^b / \sum_{1 \le j < k} b_j^2 \le 1 + 1.03(k+n)\mathbf{u}$, and so

$$\frac{S_k^b}{S_k^c} \le \frac{1 + 1.03(k+n)\mathbf{u}}{(1-\delta_j)^2},$$

implying $S_k^b - S_k^c < S_k^c/10$. A similar argument shows that $S_k^c - S_k^b < S_k^c/10$. ■

For the remaining discussion we'll need even more notation:

Notation 3.7 *Let*

$$f_k \equiv a_k \% c_k \equiv a_k - (a_k/c_k)c_k,$$

so that $a_k = c_k + f_k$*,* c_k *and* f_k *are perpendicular, and* $a_k^2 = c_k^2 + f_k^2$*. Suppose a reduction loop for* a_k *is done. Let* $\hat{a}_k$ *denote* a_k *before scaling it by* s*, and let* $\hat{c}_k$ *and* $\hat{f}_k$ *denote the components* c_k *and* f_k *at that time. Let* $a_k^{(j)}$ *denote the vector* a_k *when reducing by* a_j*, so* $a_k^{(k-1)}$ *is* a_k *before the reduction loop, and* $a_k^{(0)}$ *is* a_k *after the reduction loop. We'll use* z_{jk} *to refer to a computed value of* a_k/b_j*.*

Lemma 3.8 *With assumptions as in the previous theorem, before a reduction of* a_k *by* a_j*,*

$$\|a_k\| \le s\|\hat{a}_k\|\mu^{k-1-j} + (3/4)\sum_{j<i<k} \|c_i\|\mu^{i-1-j}.$$

Proof: When reducing by a_j, $a_k = a_k^{(j)}$ is replaced by

$$a_k^{(j-1)} = a_k - z_{jk}a_j + \alpha a_j,$$

for some $|\alpha| \le 1/2$. As in Theorem 3.5(ii), the norm of $a_k - z_{jk}a_j$ is no more than $\mu\|a_k\|$, so

$$\|a_k^{(j-1)}\| \le \mu\|a_k^{(j)}\| + \|a_j\|/2.$$

Thus

$$\begin{aligned}
\|a_k^{(j)}\| &\le s\|\hat{a}_k\|\mu^{k-1-j} + \sum_{j<i<k} \|a_i\|\mu^{i-1-j}/2 \\
&\le s\|\hat{a}_k\|\mu^{k-1-j} + (3/4)\sum_{j<i<k} \|c_i\|\mu^{i-1-j}, \qquad (10)
\end{aligned}$$

where the last inequality follows from Lemma 3.1 and Theorem 3.5(ii). ■

Lemma 3.9 *With the assumptions of Theorem 3.5, in a reduction loop for* a_k*,*

$$\hat{c}_k^2 \le 0.55\hat{a}_k^2$$

and

$$\hat{f}_k^2 \ge 0.45\hat{a}_k^2.$$

Proof: Since the reduction loop is done, the conditional "$a_k^2 \leq 2b_k^2$" returned **false**. Therefore, using Theorem 3.5(iii) and reasoning as in Lemma 3.1,

$$\begin{aligned} \|\hat{a}_k\| &> \sqrt{2}\|b_k\|(1 - 2.03n\mathbf{u}) \\ &\geq \sqrt{2}(\|\hat{c}_k\| - \delta_k\|\hat{a}_k\|/\sqrt{2})(1 - 2.03n\mathbf{u}), \end{aligned}$$

which implies

$$\hat{c}_k^2 \leq 0.55\hat{a}_k^2, \tag{11}$$

using the assumption $\delta_k < 1/32$. The last statement of the lemma follows from $\hat{a}_k^2 = \hat{c}_k^2 + \hat{f}_k^2$. ■

Theorem 3.10 *With assumptions as Theorem 3.5, after the reduction loop for a_k it satisfies*

$$a_k^2 \leq \max\{\hat{a}_k^2, 2.6S_k^c\}.$$

Also

$$S_k^c \leq M^2(3.3)^{k-2}.$$

The condition

$$\lg N \geq \lg M + 1.5n + 1$$

allows the algorithm to execute.

Proof: We begin by bounding the c_j components of a_k for $j < k$, and use these to bound $\|f_k\|$, after the reduction loop. Note that the c_j component of a_k remains fixed after the reduction of a_k by a_j. That is, using Notation 3.7,

$$a_k^{(0)}/c_j = a_k^{(j-1)}/c_j = (a_k^{(j)} - \lceil z_{jk} \rfloor a_j)/c_j.$$

Using $a_j/c_j = 1$, this implies

$$\begin{aligned} &|a_k^{(0)}/c_j| \\ &\quad \leq |a_k^{(j)}/c_j - a_k^{(j)}/b_j| + |a_k^{(j)}/b_j - \lceil z_{jk} \rfloor| \\ &\quad \leq \|a_k^{(j)}\|\delta_j/\|c_j\| + \left(1/2 + 2.2(n+1)\mathbf{u}\|a_k^{(j)}\|/\|c_j\|\right) \\ &\quad < 1/2 + \|a_k^{(j)}\|(\delta_j + \eta)/\|c_j\|, \end{aligned} \tag{12}$$

where the next-to-last inequality follows from Theorem 3.5(iii) and from reasoning as for (2). Since

$$f_k^2 = \|a_k^{(0)}\|^2 - s^2\hat{c}_k^2 = \sum_{1 \leq j < k} (a_k^{(0)}/c_j)^2 c_j^2,$$

and the triangle inequality implies $\|x + y\|^2 \leq (\|x\| + \|y\|)^2$, we have f_k^2 no more than

$$\begin{aligned} &\sum_{1 \leq j < k} (\|c_j\|/2 + \|a_k^{(j)}\|(\delta_j + \eta))^2 \\ &\quad \leq \left(\sqrt{S_k^c}/2 + \sqrt{\sum_{1 \leq j < k} (\|a_k^{(j)}\|(\delta_j + \eta))^2}\right)^2. \end{aligned} \tag{13}$$

From the definition of δ_j and (10),

$$\begin{aligned}
&\sum_{1\le j<k} \|a_k^{(j)}\|^2(\delta_j+\eta)^2 \\
&\le \eta^2 k \sum_{1\le j<k} \left(s\|\hat{a}_k\|\mu^{k-j} + \frac{3}{4}\sum_{j<i<k}\|c_i\|\mu^{i-j}\right)^2 (\mu^2\phi)^j \\
&\le \eta^2 k \left(\sqrt{\sum_{1\le j<k} s^2\hat{a}_k^2\mu^{2k}\phi^j}\right. \\
&\qquad \left. + \frac{3}{4}\sqrt{\sum_{1\le j<k}\Big(\sum_{j<i<k}\|c_i\|\mu^i\Big)^2\phi^j}\right)^2 \\
&\le \delta_k^2(s\|\hat{a}_k\|\mu/\sqrt{2} + 3\sqrt{S_k^c}/4)^2
\end{aligned}$$

With (13), f_k^2 is no more than

$$\begin{aligned}
&\left(\sqrt{S_k^c}/2 + \delta_k(s\|\hat{a}_k\|\mu/\sqrt{2} + 3\sqrt{S_k^c}/4)\right)^2 \\
&\qquad \le \left(\sqrt{S_k^c}(1/2+\delta_k) + 1.2\delta_k s\|\hat{a}_k\|\right)^2. \qquad (14)
\end{aligned}$$

If $s=1$, then using Lemma 3.9 and $\delta_k \le 1/32$,

$$\begin{aligned}
a_k^2 = c_k^2 + f_k^2 &\le 0.55\hat{a}_k^2 + (\sqrt{S_k^c}(1/2+\delta_k) + 1.2\delta_k\|\hat{a}_k\|)^2 \\
&\le 0.55\hat{a}_k^2 + (0.54\sqrt{S_k^c} + 1.2\|\hat{a}_k/32\|)^2 \\
&\le \max\{\hat{a}_k^2, S_k^c\}. \qquad (15)
\end{aligned}$$

If $s=2$ because the conditional "$s=1$ **and** $a_k^2 \le 0.9S^b$" returned **true**, then $\|\hat{a}_k\| \le 1.01\sqrt{S_k^c}$, and

$$\begin{aligned}
a_k^2 &\le 4(0.55)\hat{a}_k^2 + (\sqrt{S_k^c}(1/2+\delta_k) + 2(1.2)\delta_k\|\hat{a}_k\|)^2 \qquad (16) \\
&\le 2.6S_k^c. \qquad (17)
\end{aligned}$$

Now suppose $s>1$ and the conditional "$s=1$ **and** $a_k^2 \le 0.9S^b$" returned **false**. By Theorem 3.5(iii), $\hat{c}_k^2 \le b_k^2 + 2\delta_k\hat{a}_k^2$, so that when $\lceil\sqrt{s}\rfloor$ is evaluated it is no more than $\sqrt{S^b/4c_k^2}(1.03)+1/2$, where the 1.03 factor bounds the error in computing s. Thus

$$\begin{aligned}
c_k^2 &\le s^2\hat{c}_k^2 \le (\sqrt{S^b/4}(1.03) + \|\hat{c}_k\|/2)^2 \\
&\le (0.55\sqrt{S_k^c} + \|\hat{c}_k\|/2)^2 \\
&\le (0.55\sqrt{S_k^c} + 0.375\|\hat{a}_k\|)^2, \qquad (18)
\end{aligned}$$

using Lemma 3.9. When $\lceil\sqrt{s}\rfloor$ is evaluated, it is smaller than

$$1.03\sqrt{S^b/4\delta_k\hat{a}_k^2} + 1/2,$$

and so

$$s\delta_k\|\hat{a}_k\| \leq 0.55\sqrt{\delta_k S_k^c} + \delta_k\|\hat{a}_k\|/2.$$

With (14) and the assumption $\delta_k \leq 1/32$,

$$\begin{aligned} f_k^2 &\leq (\sqrt{S_k^c}(1/2+\delta_k) + 0.55\sqrt{\delta_k S_k^c} + \delta_k\|\hat{a}_k\|/2)^2 \\ &\leq (0.63\sqrt{S_k^c} + \delta_k\|\hat{a}_k\|/2)^2, \end{aligned} \tag{19}$$

and so with (18),

$$\begin{aligned} a_k^2 &= c_k^2 + f_k^2 \\ &\leq (0.55\sqrt{S_k^c} + 0.375\|\hat{a}_k\|)^2 + (0.63\sqrt{S_k^c} + \|\hat{a}_k\|/64)^2 \\ &\leq \max\{\hat{a}_k^2, 1.4S_k^c\}. \end{aligned} \tag{20}$$

With (15), (16), and (20), we have $a_k^2 \leq \max\{\hat{a}_k^2, 2.6S_k^c\}$. First we show that $c_k^2 \leq \max\{\hat{c}_k^2, 2.2S_k^c\}$. It remains to bound S_k^c inductively. If $s=1$ then $c_k^2 = \hat{c}_k^2$. If $s=2$ because the conditional "$s=1$ **and** $a_k^2 \leq 0.9S^b$" returned **true**, then $c_k^2 \leq 2.2S_k^c$. If $s>1$ and the conditional "$s=1$ **and** $a_k^2 \leq 0.9S^b$" returned **false**, then with (18),

$$c_k^2 \leq (0.55\sqrt{S_k^c} + \|\hat{c}_k\|/2)^2 \leq \max\{\hat{c}_k^2, 2.2S_k^c\},$$

Let σ_k^c be an upper bound for S_k^c, so $\sigma_2^c = M^2$ suffices, and when stage k is finished, $\sigma_{k+1}^c = \sigma_k^c + 2.2\sigma_k^c$ is large enough, so $\sigma_k^c = (3.2)^{k-2}M^2$ is an upper bound for S_k^c. ∎

Theorem 3.11 *Let β be the minimum of*

$$\lg(1/\mathbf{u}) - 1.1n - 2\lg n - 2$$

and

$$\lg N - \lg M - 1.5n - 1.$$

(The conditions of Theorems 3.5 and 3.10 are implied by $\beta > 4$.) The algorithm requires $O(n^3) + O(n^2)\log \mathcal{OD}(A)/\beta$ time.

Proof: Clearly each iteration of the main loop requires $O(n^2)$ time. The idea is to show that for each stage k, the loop body is executed $O(1)$ times except for executions that reduce $\mathcal{OD}(A)$ by a large factor; such reductions may occur when a_k is multiplied by a large scale factor s, increasing $\det A$, or by reducing $\|a_k\|$ substantially when s is small.

We consider some cases, remembering that c_k^2 never decreases during stage k. The discussion below assumes that $S^b/4(b_k^2 + 2\delta_k a_k^2)$ evaluates to no more than $(N/1.58^k - \sqrt{S^b})^2/a_k^2$; if not, s is limited by the latter, and β is bounded to the second term in its definition.

$\mathbf{s > 2, b_k^2 \geq \delta_k \hat{a}_k^2}$.

After two iterations with this condition, either $s \leq 2$ or c_k^2 will be sufficiently large that $2b_k^2 \geq a_k^2$.

$\mathbf{s > 2, b_k^2 \leq \delta_k \hat{a}_k^2}$.
Here (19) and Lemma 3.9 show that

$$\|f_k\| \leq 0.62\sqrt{S_k^c} + \delta_k\|\hat{a}_k\|/2,$$

and $s > 2$ implies

$$\delta_k\|\hat{a}_k\|/2 \leq \delta_k\sqrt{S^b/\delta_k}/\sqrt{4(2.5)2}/2 < 0.04\sqrt{S_k^c},$$

so $\|\hat{a}_k\| \leq 0.66\sqrt{S_k^c}/\sqrt{.45} < \sqrt{S_k^c}$ during the second iteration under these conditions. Thus $s \geq 0.37/\sqrt{\delta_k}$ here for all but possibly the first iteration, while a_k^2 is bounded. Hence $\lg \mathcal{OD}(A)$ decreases by $0.5\lg(1/\delta_k) - 2$ during these steps.

$\mathbf{s \leq 2}$.
From (14) and Lemma 3.9,

$$\begin{aligned} \|f_k\| &\leq (1/2 + \delta_k)\sqrt{S_k^c} + 2.4\delta_k\|\hat{f}_k\|/\sqrt{.45} \\ &\leq 0.54\sqrt{S_k^c} + 4delta_k\|\hat{f}_k\|, \end{aligned}$$

so $y_k \equiv \|f_k\|/0.54\sqrt{S_k^c}$ converges to $1/(1 - 4\delta_k)$.

Suppose $y_k \geq 1/(4\delta_k)^2/(1 - 4\delta_k)$; here $s = 1$, and y_k decreases by a factor of $4\delta_k/(1 - 4\delta_k) \leq 4.6\delta_k$ at each reduction loop. Hence $\|a_k\|$ is reduced by a factor of $4.6\delta_k$, while $\det A$ remains the same. Hence $\lg \mathcal{OD}(A)$ decreases by at least $\lg(1/\delta_k) - 2.3$.

Suppose $y_k < 1/(4\delta_k)^2/(1 - 4\delta_k)$; then in $O(1)$ (less than 7) reductions, $y_k \leq \sqrt{1.01}/(1 - 4\delta_k)$, so that

$$\begin{aligned} a_k^2 &\leq f_k^2/0.45 \\ &\leq 1.01(0.54)^2 S_k^c/(1 - 4\delta_k)^2/0.45 \\ &\leq 0.75 S_k^c, \end{aligned}$$

and so the conditional "$a_k^2 \leq 0.9S^b$" will return **true**. Thereafter $O(1)$ executions suffice before stage k completes, as c_k doubles in magnitude at each step.
∎

Here is one way to get the estimate *det_approx*(B): estimate the norms $\|b_i\|$, normalize each column of B to obtain a matrix $\bar{B}$ with columns $\bar{b}_j \equiv b_j/\|b_j\|$, for $j = 1, \ldots, n$, and then return $\det \bar{B} \approx \pm 1$ times $\prod_{1 \leq i \leq n}\|b_i\|$. To suggest that $\det \bar{B}$ can be estimated using low-precision arithmetic, we show that indeed $|\det \bar{B}| \approx 1$ and that the condition number of $\bar{B}$ is small. In the proof, we use the matrix $\bar{C}$ whose columns are $\bar{c}_j \equiv c_j/\|c_j\|$, for $j = 1, \ldots, n$. (Note that the truly orthonormal matrix $\bar{C}$ has $\det \bar{C} = 1$ and Euclidean condition number $\kappa_2(\bar{C}) = 1$.)

Theorem 3.12 *With assumptions as in Theorem 3.5, for any $x \in R^n$,*

$$|\|\bar{B}^T x\|/\|x\| - 1| \leq 1.68\delta_n,$$

and so $|\det \bar{B}| \geq 1 - 2n\delta_n$, and $\bar{B}$ has condition number $\kappa_2(\bar{B}) \leq 1 + 3\delta_n$.

Proof: It's easy to show that for $x \in R^n$,

$$|\bar{b}_j \cdot x - \bar{c}_j \cdot x| \le (\delta_j + \eta)\|x\|,$$

where the η on the right accounts for roundoff in computing $\bar{b}_j$ from b_j. Thus

$$\begin{aligned}(\bar{B}^T x - \bar{C}^T x)^2 &\le x^2 \sum_{1\le j\le n} (\delta_j + \eta)^2 \\ &\le x^2 \eta^2 n \sum_{1\le j\le n} (\mu^2 + 2.04)^{j-1} \\ &\le 2\delta_n^2 x^2.\end{aligned}$$

Since $\|\bar{C}^T x\| = \|x\|$,

$$|\|\bar{B}^T x\|/\|x\| - 1| \le \sqrt{2}\delta_n.$$

Recall that $\bar{B}^T$ and $\bar{B}$ have the same singular values.[3] Since the bound above holds for any x, the singular values σ_j of $\bar{B}$ satisfy $|\sigma_j - 1| \le \delta_n$, for $j = 1, \ldots, n$, and so

$$\kappa_2(\bar{B}) = \sigma_1/\sigma_n \le \frac{1 + \sqrt{2}\delta_n}{1 - \sqrt{2}\delta_n} \le 1 + 3\delta_n,$$

using $(1+x)/(1-x) = 1 + 2x/(1-x)$ and $1/(1-x)$ increasing in x. Since $|\det \bar{B}| = \sigma_1\sigma_2\cdots\sigma_n$, we have

$$|\det \bar{B}| \ge (1 - \sqrt{2}\delta_n)^n \ge \exp(-1.5n\delta_n) \ge 1 - 1.5n\delta_n,$$

where we use $1 - x \ge \exp(-x\frac{\log(1-\alpha)}{-\alpha})$ for $0 < x \le \alpha < 1$. ■

With these results, it is easy to show that Gaussian elimination with partial pivoting can be used to find $\det \bar{B}$, using for example the backwards error bound that the computed factors L and U have $LU = \bar{B} + \Delta$, where Δ is a matrix with norm no more than $n^2\sqrt{n}2^{n-1}\|\bar{B}\|\mathbf{u}$; this implies, as above, that the singular values, and hence the determinant, of LU are close to B.

Acknowledgements

It's a pleasure to thank Steve Fortune, John Hobby, Andrew Odlyzko, and Margaret Wright for many helpful discussions.

References

[1] S. Fortune. personal communication.

[2] S. Fortune. Stable maintenance of point-set triangulations in two dimensions. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 494–499, 1989.

[3] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore and London, 1989.

[4] D. Greene and F. Yao. Finite-resolution computational geometry. In *Proc. 27th IEEE Symp. on Foundations of Computer Science*, pages 143–152, 1986.

[5] W. Hoffman. Iterative algorithms for Gram-Schmidt orthogonalization. *Computing*, 41:335–348, 1989.

[6] C. Hoffmann, J. Hopcroft, and M. Karasick. Robust set operations on polyhedral solids. *IEEE Comp. Graph. Appl.*, 9:50–59, 1989.

[7] M. Karasick, D. Lieber, and L. Nackman. Efficient delaunay triangulation using rational arithmetic. *ACM Transactions on Graphics*, 10:71–91, 1990.

[8] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[9] Z. Li and V. Milenkovic. Constructing strongly convex hulls using exact or rounded arithmetic. In *Proc. Sixth ACM Symp. on Comp. Geometry*, pages 235–243, 1990.

[10] L. Lovász. *An algorithmic theory of numbers, graphs, and complexity.* SIAM, Philadelphia, 1986.

[11] V. Milenkovic. Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artificial Intelligence*, 37:377–401, 1988.

[12] V. Milenkovic. *Verifiable Implementations of Geometric Algorithms using Finite Precision Arithmetic.* PhD thesis, Carnegie Mellon U., 1988.

[13] V. Milenkovic. Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 500–505, 1989.

[14] D. Salesin, J. Stolfi, and L. Guibas. Epsilon geometry: building robust algorithms from imprecise calculations. In *Proc. Seventh ACM Symp. on Comp. Geometry*, pages 208–217, 1989.

[15] C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *J. Algorithms*, 9:47–62, 1988.

[16] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[17] K. Sugihara and M. Iri. Geometric algorithms in finite-precision arithmetic. Technical Report RMI 88-10, U. of Tokyo, 1988.