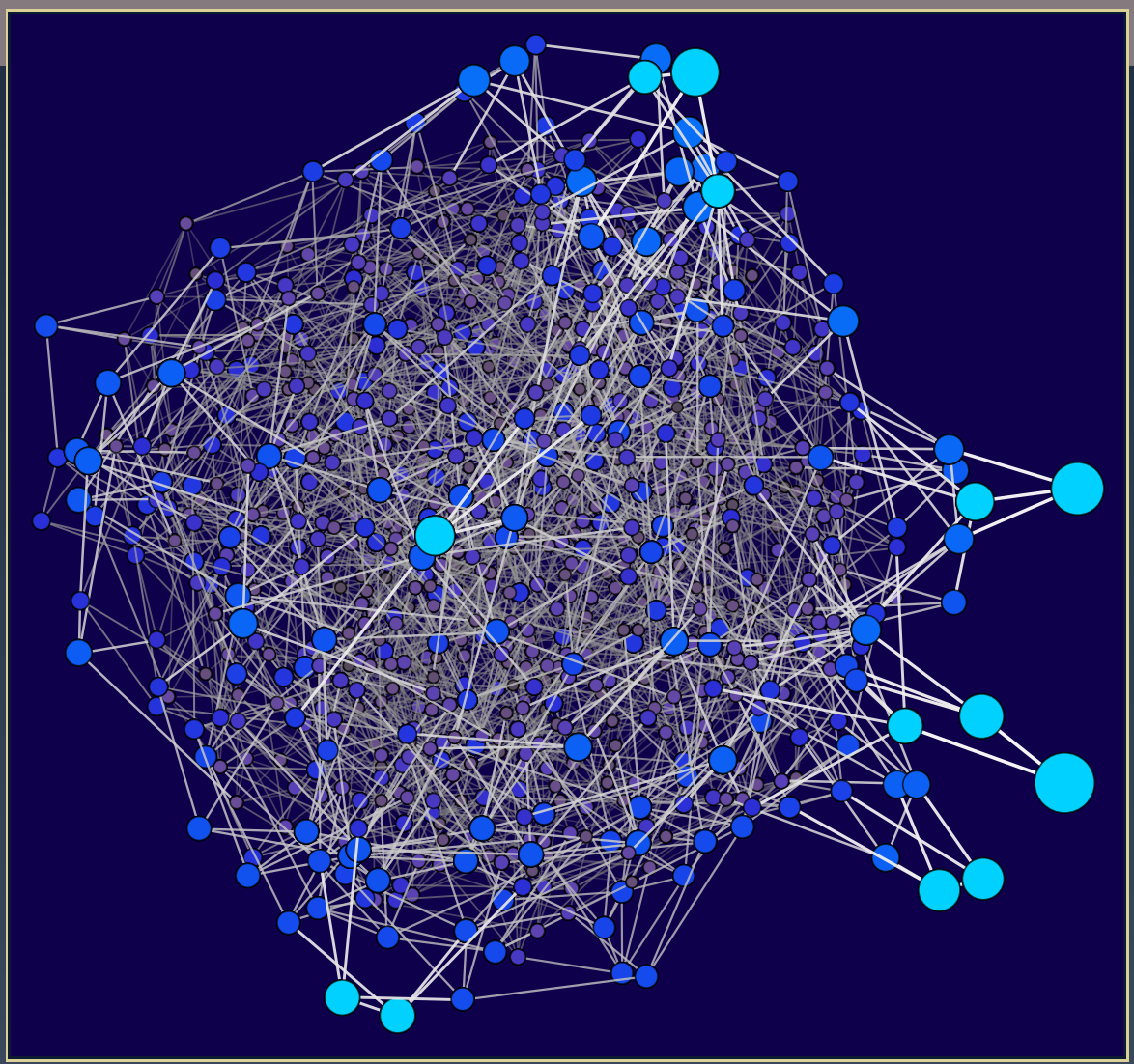
THE OPEN BOOK SERIES 4

# ANTS XIV
## Proceedings of the Fourteenth Algorithmic Number Theory Symposium

### Reductions between short vector problems and simultaneous approximation

Daniel E. Martin



msp

THE OPEN BOOK SERIES **4** (2020)
**Fourteenth Algorithmic Number Theory Symposium**
https://doi.org/10.2140/obs.2020.4.335

msp

# Reductions between short vector problems and simultaneous approximation

Daniel E. Martin

In 1982, Lagarias showed that solving the approximate shortest vector problem also solves the problem of finding good simultaneous Diophantine approximations (*SIAM J. Comput.*, **14**(1):196–209, 1985)). Here we provide a deterministic, dimension-preserving reduction in the reverse direction. It has polynomial time and space complexity, and it is gap-preserving under the appropriate norms. We also give an alternative to the Lagarias algorithm by first reducing his version of simultaneous approximation to one with no explicit range in which a solution is sought.

## 1. Introduction

Our primary result is to show that a short vector problem reduces deterministically and with polynomial complexity to a single simultaneous approximation problem as presented in the definitions below. We use $\min^{\times}$ to denote the nonzero minimum, $\{\boldsymbol{x}\} \in \left(-\frac{1}{2}, \frac{1}{2}\right]^n$ to denote the fractional part of $\boldsymbol{x} \in \mathbb{R}^n$, and $[x]$ to denote the set $\{1, \ldots, \lfloor x \rfloor\}$ for $x \in \mathbb{R}$.

**Definition 1.1.** A *short vector problem* takes input $\alpha \in [1, \infty)$ and nonsingular $M \in \mathrm{M}_n(\mathbb{Z})$. A valid output is $\boldsymbol{q}_0 \in \mathbb{Z}^n$ with $0 < \|M\boldsymbol{q}_0\| \leq \alpha \min^{\times}_{\boldsymbol{q} \in \mathbb{Z}^n} \|M\boldsymbol{q}\|$. Let SVP denote an oracle for such a problem.

**Definition 1.2.** A *good Diophantine approximation problem* takes input $\alpha, N \in [1, \infty)$ and $\boldsymbol{x} \in \mathbb{Q}^n$. A valid output is $q_0 \in [\alpha N]$ with $\|\{q_0\boldsymbol{x}\}\| \leq \alpha \min_{q \in [N]} \|\{q\boldsymbol{x}\}\|$. Let GDA denote an oracle for such a problem.

Our reduction asserts that if we can find short vectors in a very restricted family of lattices then we can find them in general, since behind a good Diophantine approximation problem is the lattice generated by $\mathbb{Z}^n$ and one additional vector, $\boldsymbol{x}$.

Literature more commonly refers to a short vector problem as a *shortest vector problem* when $\alpha = 1$ and an *approximate shortest vector problem* otherwise (often unrestricted to sublattices of $\mathbb{Z}^n$, though we have lost no generality). A brief exposition can be found in [26]. See [14] or [24] for a more

*MSC2010*: 11H06, 52C07, 68W25.
*Keywords*: lattice reduction, shortest vector problem, simultaneous Diophantine approximation, simultaneous approximation.

comprehensive overview, [27] for a focus on cryptographic applications, [19] for a summary of hardness results, and [6] for relevance and potential applications to post-quantum cryptography.

Regarding simultaneous approximation, Brentjes highlights several algorithms in [7]. For a sample of applications to attacking clique and knapsack-type problems see [13], [20], and [31]. Examples of cryptosystems built on the hardness of simultaneous approximation are [2], [4], and [16]. This version is taken from [9] and [29].

The reduction, given in Algorithm 3, preserves the gap $\alpha$ when the $\ell_\infty$-norm is used for both problems. This means the short vector problem defined by $\alpha$ and $M$ is solved by calling $\mathrm{GDA}(\alpha, \boldsymbol{x}, N)$ for some $\boldsymbol{x} \in \mathbb{Q}^n$ and $N \in \mathbb{R}$. It reverses a 1982 result of Lagarias, which reduces a good Diophantine approximation problem to SVP. (See Theorem B in [21], which refers to the problem as *good simultaneous approximation*. We borrow its name from [9] and [29].) Though there is an important contextual distinction: [21] relates simultaneous approximation under the $\ell_\infty$-norm to lattice reduction under the $\ell_2$-norm, whereas *all reductions in this paper assume a consistent norm.*

Under Lagarias' (and the most common) setup — the $\ell_\infty$-norm for GDA and the $\ell_2$-norm for SVP — we are not the first to go in this other direction. In a seminar posted online from July 1, 2019, Agrawal presented an algorithm achieving this reduction which was complete apart from some minor details [1]. Tersely stated, he takes an upper triangular basis for a sublattice of $\mathbb{Z}^n$ and transforms it inductively, using integer combinations and rigid rotations with two basis vectors at time, into a lattice (a rotated copy of the original) whose short vectors can be found via simultaneous approximation. The short vector problem defined by $\alpha$ and $M$ gets reduced to $\mathrm{GDA}(\alpha/\sqrt{2n}, \boldsymbol{x}, N)$, called multiple times in order to account for the unknown minimal vector length which is used to determine $\boldsymbol{x}$.

In contrast, the reduction here takes a completely different approach. It finds a sublattice which is nearly scaled orthonormal, so that only one additional vector is needed to generate the original lattice. This extra vector is the input for GDA. We note that when switching between norms, our reduction is also not gap-preserving. To use Algorithm 3 to solve a short vector problem with respect to the $\ell_2$-norm via GDA with respect to the $\ell_\infty$-norm, the latter must be executed with the parameter $\alpha/\sqrt{n}$ to account for the maximum ratio of nonzero norms $\|\boldsymbol{q}\|_2/\|\boldsymbol{q}\|_\infty$.

The relationship between the two problems in Definitions 1.1 and 1.2 will be studied through the following intermediary.

**Definition 1.3.** A *simultaneous approximation problem* takes input $\alpha \in [1, \infty)$ and $\boldsymbol{x} \in \mathbb{Q}^n$. A valid output is $q_0 \in \mathbb{Z}$ with $0 < \|\{q_0\boldsymbol{x}\}\| \leq \alpha \min^{\times}_{q\in\mathbb{Z}} \|\{q\boldsymbol{x}\}\|$. Let SAP denote an oracle for such a problem.

This problem prohibits only the trivial solution, the least common denominator of $\boldsymbol{x}$'s entries, while "$N$" in a good Diophantine approximation problem is generally more restrictive.

Section 2 explores the relationship between the two versions of simultaneous approximation given in Definitions 1.2 and 1.3. Among the results, only Proposition 2.1 in Section 2A is required to verify the final reduction of a short vector problem to either version of simultaneous approximation. Section 2B contains Algorithm 1. It reduces a good Diophantine approximation problem to polynomially many

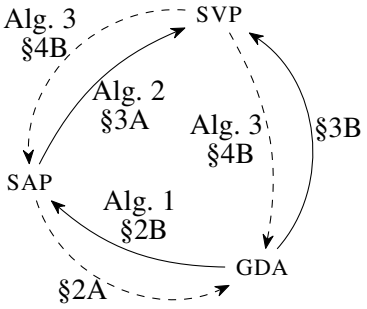



**Figure 1.** Algorithm and subsection numbers for reductions.

SAP calls, each executed with the parameter $\alpha/3.06$. So while this reduction is not gap-preserving, the inflation is independent of the input.

Section 3 reduces both versions of simultaneous approximation to SVP. It begins with Algorithm 2, which solves Definition 1.3's version. We remark at the end of Section 3A how this reduction adapts to the inhomogeneous forms of these problems, meaning the search for $q_0 \in \mathbb{Z}$ or $\boldsymbol{q}_0 \in \mathbb{Z}^n$ that makes $q_0\boldsymbol{x} - \boldsymbol{y}$ or $M\boldsymbol{q}_0 - \boldsymbol{y}$ small for some $\boldsymbol{y} \in \mathbb{Q}^n$. (In this case the latter is known as the *approximate closest vector problem*, exposited in Chapter 18 of [14], for example.) Then Section 3B combines Algorithms 1 and 2 to solve Definition 1.2's version of simultaneous approximation using SVP. This is our alternative to the Lagarias reduction.

Finally, Algorithm 3 in Section 4 reduces a short vector problem to GDA or SAP. It also adapts to the inhomogeneous versions of SVP and SAP (not GDA, as mentioned at the end of Section 4C). In Corollary 4.9 we observe that Algorithm 3 facilitates a simpler proof that GDA is NP-hard under an appropriate bound on $\alpha$, a result first obtained in [9]. Then we combine Algorithms 2 and 3 in Section 4B to solve a simultaneous approximation problem with GDA. In particular, we give all six reductions among the defined problems, as shown in Figure 1.

The two reductions in Figure 1 without algorithm numbers are achieved by following the two arrows that combine to give the same source and target. *Dashed arrows indicate a norm restriction. Each must be executed under either the $\ell_1$, $\ell_2$, or $\ell_\infty$-norm.* However, we show in Section 4C how the restriction can be alleviated to any $\ell_p$-norm provided we accept additional gap inflation by a constant arbitrarily close to 1.

The results are summarized in Table 1. It uses $m$ and $d$ to denote the maximal magnitude among input integers and the least common denominator of the input vector, respectively. The matrix or vector dimension is $n$, and $p$ defines the norm. Trivial cases that cause logarithms to equal 0 are ignored. The column descriptions are as follows:

operations: Big-$O$ bound on the number of arithmetic operations per oracle call.

integers: Big-$O$ bound on the length of integers used throughout the reduction.

inflation: Maximum gap inflation. For example, to solve a good Diophantine approximation problem with some $\alpha$ using Algorithm 1, SAP is called with $\alpha/3.06$.

calls: Upper bound on the number of required calls to the oracle.

| reduction | operations | integers | inflation | calls |
|---|---|---|---|---|
| GDA → SAP | $n \log m$ | $n \log m$ | 3.06 | $\lceil \log_2 d/\alpha N \rceil$ |
| SAP → SVP | $(n + \log m)^2$ | $n \log m$ | 1 | 1 |
| GDA → SVP | $(n + \log m)^2$ | $n \log m$ | 3.06 | $\lceil \log_2 d/\alpha N \rceil$ |
| SVP → GDA | $n^4 \log mn$ | $n^4 \log mn$ | $n^{1/p}$ | 1 |
| SVP → SAP | $n^4 \log mn$ | $n^4 \log mn$ | 1 | 1 |
| SAP → GDA | $n^5 \log m$ | $n^5 \log m$ | $n^{1/p}$ | 1 |

**Table 1.** Summary of reduction complexities and gap inflations.

## 2. Versions of simultaneous approximation

**2A. SAP *to* GDA.** Rather than give a complete reduction from a simultaneous approximation problem to GDA, which is postponed until the end of Section 4B, the purpose of this subsection is to observe a condition on the input that makes these two versions of simultaneous approximation nearly equivalent.

**Proposition 2.1.** *Suppose the $i$-th coordinate of $\boldsymbol{x}$ is of the form $x_i = 1/d$, where $d \in \mathbb{N}$ makes $d\boldsymbol{x} \in \mathbb{Z}^n$. Under an $\ell_p$-norm,* $\mathrm{GDA}(\alpha, \boldsymbol{x}, N)$ *solves the simultaneous approximation problem defined by $\alpha n^{1/p}$ and $\boldsymbol{x}$ with $N = d/2\alpha$.*

*Proof.* Let $q_{\min} \in [d/2]$ be such that $\|\{q_{\min}\boldsymbol{x}\}\|$ is the nonzero minimum. A vector's fractional part is in $\left(-\frac{1}{2}, \frac{1}{2}\right]^n$, making its length at most $n^{1/p}/2$. So we may assume that $\|\{q_{\min}\boldsymbol{x}\}\| < \frac{1}{2}\alpha$, since otherwise every integer in $[N] = [d/2\alpha]$ solves the simultaneous approximation problem defined by $\alpha n^{1/p}$ and $\boldsymbol{x}$.

Under an $\ell_p$-norm, $\|\{q_{\min}\boldsymbol{x}\}\|$ is an upper bound for its $i$-th coordinate, $q_{\min}/d$. Combined with the assumption $\|\{q_{\min}\boldsymbol{x}\}\| < \frac{1}{2}\alpha$, this gives $q_{\min} \in [d/2\alpha] = [N]$, which implies $\min_{q\in[N]}\|\{q\boldsymbol{x}\}\| \leq \min^{\times}_{q\in\mathbb{Z}}\|\{q\boldsymbol{x}\}\|$. And because $\alpha N < d$, it is guaranteed that $\mathrm{GDA}(\alpha, \boldsymbol{x}, N)$ is not a multiple of $d$. □

Note that without an assumption on $\boldsymbol{x}$ like the one used in this proposition, there is no natural choice for $N$ that makes GDA solve a simultaneous approximation problem. If we set it too small, say with $N < d/2$, then $\min_{q\in[N]}\|\{q\boldsymbol{x}\}\|$ may be unacceptably larger than $\min^{\times}_{q\in\mathbb{Z}}\|\{q\boldsymbol{x}\}\|$, potentially making GDA's approximation poor. If we set it too large, say with $N \geq d/\alpha$, then GDA may return $d$, which is not a valid output for the initial simultaneous approximation problem.

To get around this, our strategy is to first reduce a simultaneous approximation problem to SVP with Algorithm 2. Then in Algorithm 3, which reduces a short vector problem to SAP, we are careful to produce an input vector for the oracle that satisfies the hypothesis of Proposition 2.1 in order to admit GDA.

**2B. GDA *to* SAP.** Let $d$ continue to denote the least common denominator of $\boldsymbol{x}$. The problem faced in this reduction is that outputs for a good Diophantine approximation problem are bounded by $\alpha N$, which may be smaller than $d/2$. This leaves no guarantee that $\mathrm{SAP}(\alpha, \boldsymbol{x})$, call this integer $d_1 \in [d/2]$, is a solution. But knowing that $\boldsymbol{x}$ is very near a rational vector $\boldsymbol{x}_1$ with least common denominator $d_1$ allows us to call SAP again, now on $\boldsymbol{x}_1$ to get $d_2 \in [d_1/2]$. This is the least common denominator of some $\boldsymbol{x}_2$ near $\boldsymbol{x}_1$, and we continue in this fashion until the output is at most $\alpha N$. To get $d_i \in [d_{i-1}/2]$, we adopt the convention that modular reduction returns an integer with magnitude at most half the modulus.

**Algorithm 1:** A reduction from a good Diophantine approximation problem to multiple calls to SAP under a consistent norm.

**input:** $\alpha, N \in [1, \infty)$, $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Q}^n$

**output:** $q_0 \in [\alpha N]$ with $\|\{q_0\boldsymbol{x}\}\| \le \alpha \min_{q\in[N]}\|\{q\boldsymbol{x}\}\|$

$d \leftarrow \operatorname{lcd}(x_1, \ldots, x_n) > 0$

**while** $d > \alpha N$ **do**

  $d \leftarrow |\mathrm{SAP}(\alpha/3.06, \boldsymbol{x}) \bmod d|$ ▷ good, but large denominator

  $\boldsymbol{x} \leftarrow \boldsymbol{x} - \{d\boldsymbol{x}\}/d$ ▷ now $\operatorname{lcd}(\boldsymbol{x}) = d$, at most half of the previous iteration's lcd

**return** $d$

**Proposition 2.2.** *The output of Algorithm 1 solves the initial good Diophantine approximation problem.*

*Proof.* Let $d_i$ and $\boldsymbol{x}_i$ denote the values of $d$ and $\boldsymbol{x}$ after $i$ **while** loop iterations have been completed. In particular, $d_0$ and $\boldsymbol{x}_0$ are defined by the input. Also let $I + 1$ be the total number of iterations executed, so the output is $d_{I+1}$.

The triangle inequality gives

$$\|\{d_{I+1}\boldsymbol{x}\}\| \le \|\{d_{I+1}\boldsymbol{x}_I\}\| + d_{I+1}\sum_{i=1}^{I}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|. \tag{2-1}$$

With $\lambda_i = \min_{q\in[N]}\|\{q\boldsymbol{x}_i\}\|$, the choice of $d_{I+1}$ bounds the first summand by $\alpha\lambda_I/c$, where $c = 3.06$ in the algorithm but is left undetermined for now. Similarly, the choice of $d_i = \mathrm{SAP}(\alpha/c, \boldsymbol{x}_{i-1})$ and the fact that $d_{i-1} > \alpha N \ge N$ give

$$\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| = \frac{\|\{d_i\boldsymbol{x}_{i-1}\}\|}{d_i} \le \frac{\alpha \min^{\times}_{q\in\mathbb{Z}}\|\{q\boldsymbol{x}_{i-1}\}\|}{cd_i} \le \frac{\alpha\lambda_{i-1}}{cd_i}. \tag{2-2}$$

So to bound (2-1) it must be checked that the $\lambda_i$'s are not too large. To this end, fix some $i \le I$ and let $q_{\min} \in [N]$ satisfy $\|\{q_{\min}\boldsymbol{x}_{i-1}\}\| = \lambda_{i-1}$. Then we have the following upper bound on $\lambda_i$, where the three inequalities are due to the triangle inequality, inequality (2-2), and $q_{\min} \le N < d_I/\alpha \le d_i/2^{I-i}\alpha$, respectively:

$$\|\{q_{\min}\boldsymbol{x}_i\}\| \le \lambda_{i-1} + q_{\min}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| \le \lambda_{i-1}\left(1 + \frac{\alpha q_{\min}}{cd_i}\right) < \lambda_{i-1}\left(1 + \frac{1}{2^{I-i}c}\right).$$

Inductively, this gives

$$\lambda_i < \lambda_0 \prod_{j=1}^{i}\left(1 + \frac{1}{2^{I-j}c}\right). \tag{2-3}$$

Now (2-1), (2-2), and (2-3) can be combined to get

$$\|\{d_{I+1}\boldsymbol{x}\}\| \le \frac{\alpha d_{I+1}}{c}\sum_{i=0}^{I}\frac{\lambda_i}{d_{i+1}} \le \frac{\alpha}{c}\sum_{i=0}^{I}\frac{\lambda_i}{2^{I-i}} \le \frac{\alpha\lambda_0}{c}\sum_{i=0}^{I}\frac{1}{2^{I-i}}\prod_{j=1}^{i}\left(1 + \frac{1}{2^{I-j}c}\right).$$

Thus the output approximation quality, $\|\{d_{I+1}\boldsymbol{x}\}\|$, is at most $\alpha \min_{q\in[N]}\|\{q\boldsymbol{x}\}\| = \alpha\lambda_0$ provided $c$ satisfies

$$1 \geq \frac{1}{c}\sum_{i=0}^{\infty}\frac{1}{2^i}\prod_{j=i}^{\infty}\left(1+\frac{1}{2^j c}\right).$$

This justifies our choice of $c = 3.06$ in line 3. □

**Proposition 2.3.** *Let $m > 1$ be the maximum magnitude among integers defining $\boldsymbol{x}$, and let $d > 1$ be its least common denominator. The reduction in Algorithm 1 requires an initial $O(n \log m)$ operations plus $O(n)$ operations for each call to* SAP, *of which there are at most $\lceil \log_2(d/\alpha N)\rceil$, on integers of length $O(n \log m)$.*

*Proof.* Repeatedly applying the Euclidean algorithm computes $d$ with $O(n \log m)$ operations on integers of length $O(n \log m)$. Modular reduction in line 3 decreases each successive least common denominator by at least a factor of $\frac{1}{2}$. This bounds the number of **while** loop iterations by $\lceil \log_2(d/\alpha N)\rceil$. □

## 3. Reducing to SVP

First we restrict attention to Definition 1.3's version of simultaneous approximation (SAP) in Algorithm 2. Then we will compare the combination with Algorithm 1 to Lagarias' reduction in [21] from Definition 1.2's version (GDA).

**3A. SAP *to* SVP.** Here we replace the $n+1$ vectors associated to simultaneous approximation, namely $\boldsymbol{x}$ and a basis for $\mathbb{Z}^n$, with $n$ vectors generating the same lattice. There are algorithms for which this is a byproduct, like Pohst's modified (to account for linearly dependent vector inputs) LLL algorithm [23] or Kannan and Bachem's Hermite normal form algorithm [18]. But as a consequence of achieving additional basis properties, they are more complicated and require more operations than necessary. We briefly present an alternative because the improved time complexity is relevant to the next subsection.

**Algorithm 2:** A gap-preserving reduction from a simultaneous approximation problem to one call to SVP under a consistent norm.

**input:** $\alpha \in [1, \infty)$, $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Q}^n$
**output:** $q_0 \in \mathbb{Z}$ with $0 < \|\{q_0\boldsymbol{x}\}\| \leq \alpha \min^{\times}_{q\in\mathbb{Z}}\|\{q\boldsymbol{x}\}\|$
1 $d \leftarrow \operatorname{lcd}(x_1, \ldots, x_n)$
2 $x_n \leftarrow x_n + a$ with $a$ an integer that makes
$\gcd(dx_1, \ldots, dx_{n-1}, d(x_n + a)) = 1$ ▷ make sure $d\boldsymbol{x}$ extends to a basis for $\mathbb{Z}^n$
3 $M \leftarrow$ matrix in $\mathrm{SL}_n(\mathbb{Z})$ with first column $d\boldsymbol{x}$
4 $M \leftarrow M$ with last $n-1$ columns scaled by $d$ ▷ generates scaled original lattice
5 **return** $\mathrm{SVP}(\alpha, M)_1$ ▷ first coordinate is a solution

**Proposition 3.1.** *The output of Algorithm* 2 *solves the initial simultaneous approximation problem.*

*Proof.* First note that $a$ in line 2 exists. As $d$ is the *least* common denominator, $\gcd(dx_1, \ldots, dx_n)$ and $d$ are coprime. So take $a$ to be divisible by those primes which divide $\gcd(dx_1, \ldots, dx_{n-1})$ but not $dx_n$. Also, since $a$ is an integer, the new value of $\boldsymbol{x}$ defines the same simultaneous approximation problem as the input.

Coprime entries means $d\boldsymbol{x}$ extends to some $M \in \mathrm{SL}_n(\mathbb{Z})$. (One method is mentioned in the next proof.) The columns of $dM$ generate $d\mathbb{Z}^n$, so the same is true if we only scale the last $n-1$ columns by $d$. In particular, the columns of the new $M$ in line 4 generate $d\boldsymbol{x}$ and $d\mathbb{Z}^n$, which in turn generate each column. Thus $M$ defines a basis for the original simultaneous approximation lattice scaled by $d$.

Finally, the last $n-1$ columns of $M$ are vectors in $d\mathbb{Z}^n$, so that $M\,\mathrm{SVP}(\alpha, M) \equiv \mathrm{SVP}(\alpha, M)_1 d\boldsymbol{x} \bmod d\mathbb{Z}^n$. This verifies that $\mathrm{SVP}(\alpha, M)_1$ is the integer we seek. □

**Proposition 3.2.** *Let $m > 1$ be the maximum magnitude among integers defining $\boldsymbol{x}$. The reduction in Algorithm* 2 *requires $O((n + \log m)^2)$ operations on integers of length $O(n \log m)$.*

*Proof.* As with Algorithm 1, line 1 requires $O(n \log m)$ operations on integers of length $O(n \log m)$. The integer outputs of these operations also have length $O(n \log m)$.

Skipping line 2 for now, the $i$-th column (for $i \geq 2$) of $M$ in line 3 can be set to

$$\left( \frac{b_1 dx_1}{\gcd(dx_1, \ldots, dx_{i-1})}, \ldots, \frac{b_1 dx_{i-1}}{\gcd(dx_1, \ldots, dx_{i-1})}, b_2, 0, \ldots, 0 \right)$$

(transposed), where $b_2 \gcd(dx_1, \ldots, dx_{i-1}) - b_1 dx_i = \gcd(dx_1, \ldots, dx_i)$. The determinant of the top-left $i \times i$ minor is then $\gcd(dx_1, \ldots, dx_i)$ by induction. To find $b_1$ and $b_2$ we execute the Euclidean algorithm on $\gcd(d_i x_1, \ldots, d_i x_{i-1})$ and $d_i x_i$, where $d_i = \mathrm{lcd}(x_1, \ldots, x_i)$. But $\gcd(d_i x_1, \ldots, d_i x_{i-1})$ is at most $m$ times $\gcd(d_{i-1} x_1, \ldots, d_{i-1} x_{i-1})$, which divides the greatest common divisor of the numerators of $x_1, \ldots, x_{i-1}$. So for each $i$ the Euclidean algorithm needs $O(\log m)$ operations.

Before computing the last column of $M$, we find $a$ in line 2 to ensure a determinant of 1. As discussed in the last proof, we can start with $a = \gcd(dx_1, \ldots, dx_{n-1})$ and replace it with $a/\gcd(a, dx_n)$ until nothing changes. This requires $O(\log a) = O(\log m)$ executions of the Euclidean algorithm, each taking $O(\log m)$ operations.

Scaling all but the first column by $d$ in line 4 takes $O(n^2)$ operations. □

We remark that this algorithm adapts to inhomogeneous forms of these problems. To find $q_0 \in \mathbb{Z}$ with $\|\{q_0 \boldsymbol{x} - \boldsymbol{y}\}\| \leq \alpha \min_{q \in \mathbb{Z}}^{\times} \|\{q\boldsymbol{x} - \boldsymbol{y}\}\|$ when $q\boldsymbol{x} - \boldsymbol{y} \in \mathbb{Z}^n$ has no solution, we can perform the same reduction and finish by calling an oracle which solves the approximate *closest* vector problem defined by $\alpha$, $M$, and $d\boldsymbol{y}$.

**3B. GDA *to* SVP.** Combining Algorithms 1 and 2 gives an alternative to the Lagarias reduction from good Diophantine approximation to SVP in [21]. We execute Algorithm 1, but use Algorithm 2 to compute $\mathrm{SAP}(\alpha/3.06, \boldsymbol{x})$ in line 3. By Proposition 2.3, this requires at most $\lceil \log_2(d/\alpha N) \rceil$ calls to SVP. And Proposition 3.2 states that each call requires $O((n + \log m)^2)$ operations on integers of length $O(n \log m)$.

Recall that switching from $\ell_2$ to $\ell_\infty$ decreases a nonzero norm by at most a factor of $1/\sqrt{n}$. In particular, by executing this combination of Algorithms 1 and 2 with respect to the $\ell_2$-norm, we get an $\ell_\infty$ solution to the initial good Diophantine approximation problem provided we use $\alpha/3.06\sqrt{n}$ for SVP.

Lagarias achieves this reduction with the now well-known trick from [22] of reducing the lattice generated by $\mathbb{Z}^n$ and $\boldsymbol{x}$, bumped up a dimension by putting 0 in every $(n+1)$-th coordinate but $\boldsymbol{x}$'s. The ideal value for the last coordinate of $\boldsymbol{x}$, which is guessed at using $\lfloor n + \log_2 dN \rfloor$ calls of the form $\text{SVP}(\alpha/\sqrt{5n}, M)$ for varying $M$, is $\min_{q\in[N]} \|\{q\boldsymbol{x}\}\|/N$. (The gap inflation approaches $\sqrt{n}$ as our guesses get better.) The Lagarias reduction requires an initial $O(n \log m)$ arithmetic operations to compute the least common denominator, then only one additional operation per call. The integers involved have input length $O(\log m^n N)$.

Whether the benefit of fewer calls to SVP outweighs the increased operations per call depends on the complexity of the oracle. Ours is an asymptotic improvement when the number of operations performed by SVP exceeds $O((n + \log m)^2)$.

## 4. Reducing to GDA or SAP

We focus first on the reduction to SAP.

**4A.** ***Intuition.*** Consider an input matrix $M \in \mathrm{M}_n(\mathbb{Z})$ for a short vector problem. Let $d = \det M$, and let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ denote the standard basis vectors for $\mathbb{Z}^n$. If there were one vector, call it $\boldsymbol{b} \in \mathbb{Z}^n$, for which the set $\{M\boldsymbol{b}, d\boldsymbol{e}_1, \ldots, d\boldsymbol{e}_n\}$ generated the columns of $M$, our reduction would just amount to finding it. This is exactly the setup for simultaneous approximation: $n+1$ vectors, $n$ of which are scaled orthonormal. A solution could be obtained by doing simultaneous approximation on $M\boldsymbol{b}/d$, scaling the resulting short vector by $d$, and applying $M^{-1}$ (to comply with Definition 1.1). Unfortunately, unless $n \le 2$ or $d = \pm 1$, such a $\boldsymbol{b}$ does not exist. Indeed, the adjugate matrix, $\operatorname{adj} M = dM^{-1}$, has at most rank 1 over $\mathbb{Z}/p\mathbb{Z}$ for a prime $p$ dividing $d$. So at least $n-1$ additional vectors are required to have full rank modulo $p$, a prerequisite to having full rank over $\mathbb{Q}$. But asking that $M\boldsymbol{b}$ generate the columns of $M$ alongside $d\boldsymbol{e}_1, \ldots, d\boldsymbol{e}_n$ is equivalent to asking that $\boldsymbol{b}$ generate $\mathbb{Z}^n$ alongside the columns of $\operatorname{adj} M$.

What matters is the matrix with columns $d\boldsymbol{e}_1, \ldots, d\boldsymbol{e}_n$ being scaled orthonormal. As such, multiplying by it or its inverse has no effect on a vector's relative length. So we plan to find a different set of $n$ column vectors—a set for which just one additional $M\boldsymbol{b}$ is needed to generate the original lattice—which is nearly scaled orthonormal, making the effect of its corresponding matrix multiplication on $\alpha$ negligible. The initial short vector problem becomes a search for an integer combination of $M\boldsymbol{b}$ and these columns, say $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$. We can then solve the simultaneous approximation problem defined by $\alpha$ and $[\boldsymbol{c}_1 \cdots \boldsymbol{c}_n]^{-1} M\boldsymbol{b}$. This works as long as multiplying by $[\boldsymbol{c}_1 \cdots \boldsymbol{c}_n]$ changes the ratio between the lengths of the shortest vector and our output by less than whatever is afforded by the fact that lattice norms form a discrete set.

An arbitrary lattice may have all of its scaled orthonormal sublattices contained in $d\mathbb{Z}^n$. So as candidates for the matrix $[\boldsymbol{c}_1 \cdots \boldsymbol{c}_n]$, we look for something of the form $cd\,\mathrm{Id} + MA = M(c \operatorname{adj} M + A)$ for

some $c \in \mathbb{Z}$ and $A \in \mathrm{M}_n(\mathbb{Z})$. If the entries of $A$ are sufficiently small, then multiplication by $cd\,\mathrm{Id} + MA$ has a similar effect on relative vector norms as multiplying by $cd\,\mathrm{Id}$, which has no effect.

We will tailor our choice of $c$ and $A$ so that a coordinate of the simultaneous approximation vector, $(c\,\mathrm{adj}\,M + A)^{-1}\boldsymbol{b}$, is $1/\det(c\,\mathrm{adj}\,M + A)$. This admits Proposition 2.1 and hence GDA.

**4B. SVP *to* GDA *or* SAP.** Algorithm 3 uses the following.

**Notation 4.1.** For polynomials $f_1 = \sum_i f_{1,i}x^i$ and $f_2 = \sum_i f_{2,i}x^i$ with maximum degree $d$, let $C(f_1, f_2)$ denote the matrix of their coefficients,

$$\begin{bmatrix} f_{1,d} & & 0 & f_{2,d} & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ f_{1,1} & \cdots & f_{1,d} & f_{2,1} & \cdots & f_{2,d} \\ f_{1,0} & \cdots & f_{1,d-1} & f_{2,0} & \cdots & f_{2,d-1} \\ & \ddots & \vdots & & \ddots & \vdots \\ 0 & & f_{1,0} & 0 & & f_{2,0} \end{bmatrix}.$$

The matrix above can determine when $f_1$ and $f_2$ are coprime over $\mathbb{Q}(x)$ in lieu of polynomial long division, where coefficient growth is exponential without complicated mitigations as in [8]. We demonstrate this now to give some clarity to the meaning behind lines 5 and 6 of Algorithm 3.

**Lemma 4.2.** *Let $f_1, f_2 \in \mathbb{Z}[x]$, not both constant. As an ideal in $\mathbb{Z}[x]$, $(f_1, f_2)$ contains* $\det C(f_1, f_2)$, *which is nonzero if and only if $f_1$ and $f_2$ have no common root in the algebraic closure of $\mathbb{Q}$.*

*Proof.* Let $d = \max(\deg f_1, \deg f_2)$. Consider the vector in $\mathbb{Z}^{2d}$ whose only (perhaps) nonzero entry is $\det C(f_1, f_2)$ in the last coordinate. This is the image under $C(f_1, f_2)$ of some nonzero integer vector. We can split the entries of this vector down the middle to get coefficients for $g_1, g_2 \in \mathbb{Z}[x]$ that have degree at most $d-1$ and satisfy $\det C(f_1, f_2) = f_1g_1 + f_2g_2 \in (f_1, f_2)$.

Plugging a common root of $f_1$ and $f_2$ into this last equation, should one exist, shows $\det C(f_1, f_2) = 0$. Conversely, suppose $f_1g_1 + f_2g_2 = 0$ and that $\deg f_1 = d \geq 1$. Then $g_2$ must be nonzero to avoid the same being true of $g_1$, contradicting our choice of nonzero coefficient vector. But $g_2$ has degree at most $d-1$. So $f_1g_1 = -f_2g_2$ implies that at least one of $f_1$'s $d$ roots must be shared by $f_2$. □

**Notation 4.3.** For a matrix $M$, let $M_{i,j}$ denote the entry in its $i$-th row and $j$-th column, and let $\check{M}_i$ denote its top-left $i \times i$ minor.

Line 1 of Algorithm 3 requires knowing the position of a nonzero entry in the input matrix, and line 8 requires knowing the maximum magnitude among entries. For notational convenience, we assume that $M_{n,1}$ is the nonzero maximum.

Let us turn to the **for** loop, which builds the matrix Section 4A called $A$.

**Lemma 4.4.** *For $i = 2, \ldots, n$, there is some $j \leq 2i-2$ satisfying the criterion of line* 5 *in the* **for** *loop iteration corresponding to $i$.*

**Algorithm 3:** A reduction from a short vector problem with $n \geq 2$ to one call to SAP (gap-preserving) or GDA under a consistent $\ell_p$-norm with $p \in \{1, 2, \infty\}$.

**input:** $a \geq b \in \mathbb{N}$ ($\alpha = a/b$), $M \in \mathrm{M}_n(\mathbb{Z})$ with $0 \neq \det M$ and $M_{n,1} = \max_{i,j} |M_{i,j}|$
**output:** $\boldsymbol{q}_0 \in \mathbb{Z}^n$ with $0 < \|M\boldsymbol{q}_0\| \leq \alpha \min^{\times}_{\boldsymbol{q} \in \mathbb{Z}^n} \|M\boldsymbol{q}\|$

1 $p \leftarrow$ least prime not dividing $M_{n,1} \det M$
2 $M \leftarrow x \operatorname{adj} M + p \operatorname{Id}$ ▷ $M = M(x)$ has linear polynomial entries
3 **for** $i \leftarrow 2$ **to** $n$ **do**
4 $\quad M_{i,1} \leftarrow M_{i,1} + p$
5 $\quad M_{i,i-1} \leftarrow M_{i,i-1} + p^j$ with $j > 0$ minimal so $\det C((\operatorname{adj} \check{M}_i)_{i,1}, (\operatorname{adj} \check{M}_i)_{i,2}) \neq 0$ ▷ need not compute determinant to test each $j$; see Theorem 4.8
6 $c \leftarrow \det C((\operatorname{adj} M)_{n,1}, (\operatorname{adj} M)_{n,2})$
7 $c \leftarrow c/p^j$ with $j$ maximal or $p+1$ if $|c| = p^j$ ▷ make $c$ coprime to $p$
8 $M \leftarrow M(c^j)$ with $j = \lceil \log_{|c|} a^2 (2M_{n,1} n)^{3n} \rceil$ ▷ substitute for $x$ so $M \in \mathrm{M}_n(\mathbb{Z})$
9 $b_1, b_2 \leftarrow$ integers with $|b_1|$ minimal so $1 = b_1 (\operatorname{adj} M)_{n,1} + b_2 (\operatorname{adj} M)_{n,2}$ ▷ that these exist guarantees $M\boldsymbol{x}$ (line 10) and $M$ generate $\mathbb{Z}^n$
10 $\boldsymbol{x} \leftarrow M^{-1}(b_1, b_2, 0, \ldots, 0)$
11 $q_0 \leftarrow \mathrm{SAP}(\alpha, \boldsymbol{x})$ or $\mathrm{GDA}(\alpha/n^{1/p}, \boldsymbol{x}, N)$ with $N = n^{1/p} \det M / 2\alpha$ ▷ GDA works since $x_n = 1/\det M$; recall Proposition 2.1
12 **return** $M\{q_0 \boldsymbol{x}\}$

*Proof.* When $i = 2$ we are asked to find $j$ for which the linear polynomials $M_{1,1}$ and $M_{2,1} + p^j$ do not share a root (by Lemma 4.2). The constant term of $M_{1,1}$ is $p$ by line 2, meaning it has at most one root. So asking that $j \leq 2i - 2 = 2$ gives enough space to avoid the at-most-one value of $j$ that fails. Now suppose $i \geq 3$ and that the claim holds for $i - 1$. Let $M$ be its value after line 4 in the **for** loop iteration corresponding to $i$, and let

$$f_1 = (\operatorname{adj} \check{M}_{i-1})_{i-1,1} \quad \text{and} \quad f_2 = (\operatorname{adj} \check{M}_{i-1})_{i-1,2}.$$

By assumption there are $g_1, g_2 \in \mathbb{Z}[x]$ with $g_1 f_1 + g_2 f_2 = \det C(f_1, f_2) \neq 0$. Fix an integer $j$, and let $h_1 = (\operatorname{adj} \check{M}_i)_{i,1} - p^j f_1$ and $h_2 = (\operatorname{adj} \check{M}_i)_{i,2} - p^j f_2$, the polynomials we hope to make coprime with the appropriate choice of $j$. We have

$$\begin{bmatrix} f_2 & -f_1 \\ g_1 & g_2 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} f_2 (\operatorname{adj} \check{M}_i)_{i,1} - f_1 (\operatorname{adj} \check{M}_i)_{i,2} \\ g_1 (\operatorname{adj} \check{M}_i)_{i,1} + g_2 (\operatorname{adj} \check{M}_i)_{i,2} - p^j \det C(f_1, f_2) \end{bmatrix}.$$

In the column on the right, where we now focus our attention, $p^j$ has been isolated.

For each root of the top polynomial, there is at most one value of $j$ that makes it a root of the bottom. Thus it suffices to show that $f_2 (\operatorname{adj} \check{M}_i)_{i,1} - f_1 (\operatorname{adj} \check{M}_i)_{i,2}$ is not the zero polynomial. Then its degree, which is at most $2i - 3$, bounds how many values of $j$ can make the right-side polynomials share a root. As this occurs whenever $h_1$ and $h_2$ share a root, Lemma 4.2 would complete the proof.

To show that $f_2(\operatorname{adj}\check{M}_i)_{i,1} - f_1(\operatorname{adj}\check{M}_i)_{i,2}$ is nonzero, we compute its constant term from the matrix

$$\begin{bmatrix} p & 0 & \cdots & 0 & 0 & 0 \\ p+p^{j_2} & p & & & 0 & 0 \\ p & p^{j_3} & & & & 0 \\ \vdots & & \ddots & & & \vdots \\ p & 0 & & p^{j_{i-1}} & p & 0 \\ p & 0 & \cdots & 0 & p^{j} & p \end{bmatrix}. \tag{4-1}$$

These are the constants in $\check{M}_i$ after adding $p^j$ in the $i, i-1$ position — the main diagonal comes from line 2, the first column comes from line 4, and the second diagonal comes from line 5. To compute $h_1$ or $h_2$, we use cofactor expansion along the bottom row after deleting the last column and the first or second row. The $(i-2)\times(i-2)$ minor determinants that are multiplied by the bottom row constant $p^j$ are exactly $f_1$ and $f_2$ up to a sign. What remains sums to $(\operatorname{adj}\check{M}_i)_{i,1}$ or $(\operatorname{adj}\check{M}_i)_{i,2}$. So the constant terms of $(\operatorname{adj}\check{M}_i)_{i,1}$, $(\operatorname{adj}\check{M}_i)_{i,2}$, and $f_2$ are $p^{i-1}$, 0, and $p$ to the power $1+j_3+\cdots j_{i-1}$, respectively. This makes $p$ to the power $i+j_3+\cdots+j_{i-1}$ the constant term of $f_2(\operatorname{adj}\check{M}_i)_{i,1} - f_1(\operatorname{adj}\check{M}_i)_{i,2}$. □

We remark that by using a large integer instead of $x$ in line 2, the **for** loop could successively make pairs of integers coprime rather than polynomials. Then the Euclidean algorithm could test $j$ in line 5; determinants involving polynomial entries need not be computed. We might expect such an algorithm to require $O(n^3 \log M_{n,1} n)$ operations (this uses that the average ratio with Euler's phi function, $\varphi(n)/n$, is a positive constant), but the provable worst case is bad. The best current asymptotic upper bound on the size of the interval that must be sieved or otherwise searched to find $j$ is due to Iwaniec [17]. It only limits the algorithm to $O(n^7 \log M_{n,1} n)$ operations. We favored the polynomial approach because of an easier bound on $j$ (Lemma 4.4) and a better provable worst case (Theorem 4.8).

The next lemma allows the vector in line 10 to pass as $\boldsymbol{b}$ from Section 4A.

**Lemma 4.5.** *With $M$ denoting its value in line* 9, $\gcd((\operatorname{adj} M)_{n,1}, (\operatorname{adj} M)_{n,2}) = 1$.

*Proof.* By Lemma 4.2, it suffices to prove $\gcd((\operatorname{adj} M)_{n,1}, (\operatorname{adj} M)_{n,2}, c) = 1$ with $c$ as in line 6. Now let $c'$ be $c/p^j$ or $p+1$ as in line 7. Recall the constant terms displayed in (4-1), which show that $(\operatorname{adj} M)_{n,2}$ is a power of $p$ modulo $c'$. This implies $\gcd((\operatorname{adj} M)_{n,1}, (\operatorname{adj} M)_{n,2}, c)$ is a power of $p$ since $p \nmid c'$. But the constants added throughout the **for** loop are multiples of $p$. So before substituting for $x$, only the leading coefficient of $(\operatorname{adj} M)_{n,1}$ might have been nonzero modulo $p$. With $M$ now the original input matrix, the leading term is $M_{n,1} \det M^{n-2} x^{n-1}$. By line 1 this is coprime to $p$ whenever the same is true of the integer substituted for $x$. □

**Lemma 4.6.** *Let $M$ be the input matrix, let $c^j$ be as in line* 8, *and let $A$ be such that $c^j \operatorname{adj} M + A$ is Algorithm 3's value of $M$ in line* 9. *Then $\|MA\|_{\mathrm{op}} < (2nM_{n,1})^{3n}/5n$ under any $\ell_p$-norm.*

*Proof.* The operator norm is $\max_{\|\boldsymbol{u}\|=1} \|MA\boldsymbol{u}\|$. Using $\|\boldsymbol{u}\|_\infty \le 1$ gives

$$\|MA\boldsymbol{u}\| \le n\|MA\boldsymbol{u}\|_\infty \le n^2 \max_{i,j\in[n]} |(MA)_{i,j}|. \tag{4-2}$$

We refer back to (4-1), which displays the entries of $A$ when $i = n$. Lemma 4.4 says $j_i \le 2i - 2$, so the entries of $MA$ are bounded in magnitude by

$$\max_{i,j\in[n]} |M_{i,j}| \max(np + p^2,\, p + p^{2n-2}) \le 2M_{n,1}p^{2n-2} \le 2M_{n,1}^3 p^{2n-2}. \tag{4-3}$$

(Recall that $n \ge 2$ for this inequality.) Here $np + p^2$ comes from the first column of $A$, and $p + p^{2n-2}$ comes from the $(n-1)$-th column.

Now we turn to the size of $p$. If $x \in \mathbb{R}$ is such that $x\#$, the product of primes not exceeding $x$, is larger than $M_{n,1}|\det M|$, then it must be that $p < x$. Rosser and Schoenfeld's lower bound on Chebyshev's theta function, $\vartheta(x) = \sum_{p\le x} \log p$, gives $\vartheta(x) > 0.231x$ when $x \ge 2$ [28]. For the determinant we use Hadamard's bound: $|\det M| \le (M_{n,1}\sqrt{n})^n$ [15]. So take $x = (\log M_{n,1}^{3n} n^n)/0.462$ (note that $x \ge 2$ even when $n = 2$ and $M_{n,1} = 1$, allowing for the Rosser–Schoenfeld bound) to get

$$\log x\# = \vartheta(x) > 0.231x = \tfrac{1}{2}\log M_{n,1}^{3n} n^n \ge \log M_{n,1}^{n+1} n^{n/2} \ge \log M_{n,1}|\det M|.$$

Combining $p < x$ with (4-2) and (4-3) gives $\|MA\|_{\mathrm{op}} < 2M_{n,1}^3 n^2 x^{2n-2}$. We must show that this is less than the stated bound of $(2nM_{n,1})^{3n}/5n$. To do this, raise both expressions to the power $1/(n-1)$ and use $\left(\frac{5}{4}\right)^{1/(n-1)} \le \frac{5}{4}$. This simplifies the desired inequality to $(\log M_{n,1}^3 n)^2 < 1.366 M_{n,1}^3 n$, which is true. □

**Theorem 4.7.** *Under the $\ell_1$, $\ell_2$, or $\ell_\infty$-norm, the output of Algorithm* 3 *solves the initial short vector problem.*

*Proof.* There are two parts to the proof: (1) showing that the algorithm replaces the columns of $M$ with $n+1$ vectors that define the same lattice, $n$ of them being nearly scaled orthonormal, and (2) showing that nearly scaled orthonormal is as good as being scaled orthonormal. Throughout the proof, let $M$ be the input matrix, let $c^j$ be as in line 8, let $M'$ be Algorithm 3's value of $M$ in line 9, and let $A = M' - c^j \operatorname{adj} M$ be the matrix of constants added throughout the **for** loop (as used in Lemma 4.6 and as shown in (4-1) when $i = n$).

For part (1), with $\boldsymbol{b} = (b_1, b_2, 0, \ldots, 0)$ from line 10, Lemma 4.5 gives

$$\boldsymbol{x} = M'^{-1}\boldsymbol{b} = \frac{(x_1, x_2, \ldots, 1)}{\det M'}. \tag{4-4}$$

By Cramer's rule [10], the 1 in the last coordinate is the determinant after replacing the last column of $M'$ by $\boldsymbol{b}$, so that these $n$ columns generate $\mathbb{Z}^n$. This in turn shows that the columns of $MM'$ and $M\boldsymbol{b}$ generate the input lattice. Also note by Proposition 2.1, that a coordinate of $\det M'\boldsymbol{x}$ being 1 allows for GDA in place of SAP with $N$ set to $n^{1/p}\det M'/2\alpha$ and $\alpha$ scaled by $1/n^{1/p}$.

Instead of finding a short integer combination of $M\boldsymbol{b}$ and the columns of

$$MM' = c^j \det M \,\mathrm{Id} + MA, \tag{4-5}$$

Algorithm 3 uses $(MM')^{-1}(M\boldsymbol{b}) = \boldsymbol{x}$ and the columns of $(MM')^{-1}(MM') = \mathrm{Id}$. Then $MM'\{q_0\boldsymbol{x}\}$ is proposed as a short vector. It is indeed an element of the original lattice since the coordinates

of $M'\{q_0\boldsymbol{x}\} \equiv q_0\boldsymbol{b} \bmod \mathbb{Z}^n$ are all integers. But it must be checked is that $MM'\{q_0\boldsymbol{x}\}$ is short whenever $\{q_0\boldsymbol{x}\}$ is. Part (2) of the proof is to make precise the insignificance of the second matrix summand, $MA$, in (4-5). We begin by computing how much multiplication by the full matrix in (4-5) is allowed to inflate the gap without invalidating the output of GDA or SAP.

By Minkowski's theorem [25], the magnitude of the shortest vector in the original lattice with respect to the $\ell_\infty$-norm is not more than $|\det M|^{1/n}$. So under an $\ell_p$-norm with $p \in \mathbb{N}$, the shortest vector has some magnitude, say $\lambda$, with $(n^{1/p}|\det M|^{1/n})^p \geq \lambda^p \in \mathbb{Z}$. In particular, $n|\det M|^{2/n} \geq \lambda^2 \in \mathbb{Z}$ when $p \in \{1, 2, \infty\}$. Now, if $\boldsymbol{q} \in \mathbb{Z}^n$ is such that $\|M\boldsymbol{q}\|^2 < (a^2\lambda^2 + 1)/b^2$, then it must be that $\|M\boldsymbol{q}\| \leq a\lambda/b$ since there are no integers strictly between $(a\lambda/b)^2$ and $(a^2\lambda^2 + 1)/b^2$. Thus multiplication by $MM'$ may harmlessly inflate the gap between the norms of our output vector and shortest vector by anything less than

$$\frac{\sqrt{a^2\lambda^2 + 1}}{b\alpha\lambda} = \frac{\sqrt{a^2\lambda^2 + 1}}{a\lambda} \geq \frac{\sqrt{a^2 n|\det M|^{2/n} + 1}}{a\sqrt{n}|\det M|^{1/n}}. \tag{4-6}$$

Scaling does not affect the ratio of vector norms, so to determine the effect of multiplication by (4-5) it suffices to consider the matrix

$$\mathrm{Id} + MA/c^j \det M \tag{4-7}$$

instead. If $\boldsymbol{q}_{\min}$ is a shortest nonzero vector in the simultaneous approximation lattice generated by $\mathbb{Z}^n$ and $\boldsymbol{x}$, a shortest vector after applying (4-7) to this lattice has norm at least $(1 - \|MA\|_{\mathrm{op}}/|c^j \det M|)\|\boldsymbol{q}_{\min}\|$. Similarly, the vector $\{q_0\boldsymbol{x}\}$ obtained using $q_0$ from line 11 increases in norm by at most a factor of $(1 + \|MA\|_{\mathrm{op}}/|c^j \det M|)$. Combining this with our conclusion regarding (4-6) shows that it suffices to verify the following inequality holds:

$$\frac{1 + \|MA\|_{\mathrm{op}}/|c^j \det M|}{1 - \|MA\|_{\mathrm{op}}/|c^j \det M|} \leq \frac{\sqrt{a^2 n|\det M|^{2/n} + 1}}{a\sqrt{n}|\det M|^{1/n}}. \tag{4-8}$$

Now solve for $|c^j|$ to get a lower bound of

$$\frac{\sqrt{a^2 n|\det M|^{2/n} + 1} + a\sqrt{n}|\det M|^{1/n}}{\sqrt{a^2 n|\det M|^{2/n} + 1} - a\sqrt{n}|\det M|^{1/n}} \cdot \frac{\|MA\|_{\mathrm{op}}}{|\det M|} < \frac{(5a^2 n|\det M|^{2/n})\|MA\|_{\mathrm{op}}}{|\det M|}.$$

Ignoring the powers of $|\det M|$ on the right-hand side since $2/n \leq 1$, we see that $j$ in line 8 is chosen to make the bound above agree exactly with Lemma 4.6. □

**Theorem 4.8.** *Let $m = \max(a^{1/n^3}, M_{n,1})$. The reduction in Algorithm 3 requires $O(n^4 \log mn)$ operations on integers of length $O(n^4 \log mn)$.*

*Proof.* We will use that finding determinants, adjugates, inverses, or characteristic polynomials of $n \times n$ matrices with entry magnitudes bounded by $m$ requires $O(n^3)$ operations on integers of length $O(n \log mn)$. For example, see Danilevsky's method for the characteristic polynomial [11] and the Bareiss algorithm for the others [5]. Note that we may then compute determinants of matrices with linear

polynomial entries in $O(n^3)$ operations provided the matrix of linear terms or the matrix of constant terms is invertible.

In the proof of Lemma 4.6 we showed that the prime $p$ from line 1 is less than $(\log M_{n,1}^{3n} n^n)/0.462$. So finding it does not contribute to asymptotic complexity.

Now consider the **for** loop, where we must avoid recomputing the determinant in line 5 for each value of $j$ in order to meet the prescribed bound on operations.

Let $i \geq 3$ and fix some notation: $M$ is its value after line 4, $f_1 = (\operatorname{adj} \check{M}_{i-1})_{i-1,1}$ and $f_2 = (\operatorname{adj} \check{M}_{i-1})_{i-1,2}$, $g_1$ and $g_2$ have degree at most $i-3$ and $f_1 g_1 + f_2 g_2 = \det C(f_1, f_2) \neq 0$, and for some $j$, $h_1 = (\operatorname{adj} \check{M}_i)_{i,1} - p^j f_1$ and $h_2 = (\operatorname{adj} \check{M}_i)_{i,2} - p^j f_2$. Note for computing $(\operatorname{adj} \check{M}_i)_{i,2}$ that the constant term matrix is not invertible (see (4-1)), which may also be true of the linear term matrix. Because this complicates combining the Bareiss and Danilevsky algorithms, we could find $(\operatorname{adj} \check{M}_i)_{i,2}$ indirectly by computing $h_2$ for two values of $j$ that produce an invertible constant term matrix (recall from (4-1) that $f_2$ has nonzero constant term), and then solving for it.

Call the polynomials in the resulting column vector below $h_1'$ and $h_2'$:

$$\begin{bmatrix} f_2 + p^j g_1 x^{2i-3} & -f_1 + p^j g_2 x^{2i-3} \\ g_1 & g_2 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} f_2 (\operatorname{adj} \check{M}_i)_{i,1} - f_1 (\operatorname{adj} \check{M}_i)_{i,2} - p^j \det C(f_1, f_2) x^{2i-3} \\ g_1 (\operatorname{adj} \check{M}_i)_{i,1} + g_2 (\operatorname{adj} \check{M}_i)_{i,2} - p^j \det C(f_1, f_2) \end{bmatrix}. \quad (4\text{-}9)$$

Remark that if $j$ makes $h_1'$ and $h_2'$ avoid a common root, it does so for $h_1$ and $h_2$.

View $C(h_1', h_2')$ as a matrix with linear polynomial entries where $p^j$ is the variable. This variable only appears in the leading term of $h_1'$ and the constant term of $h_2'$. So $p^j$ only occurs on the main diagonal of $C(h_1', h_2')$, where its coefficient is nonzero. In particular, the polynomial $\det C(h_1', h_2')$ can be found in $O(n^3)$ operations. Substituting different values of $p^j$ into this polynomial until one is nonzero avoids repeatedly finding determinants. And note that we still need only test up to $j = 2i-2$ as stated in Lemma 4.4 because the determinant of the matrix in (4-9) is a constant (a unit in $\mathbb{Q}(x)$). Thus each **for** loop iteration requires $O(n^3)$ operations.

The integers composing the linear polynomial matrix entries that begin each **for** loop iteration are small powers of $p = O(n \log M_{n,1} n)$ and entries in the adjugate of the input matrix, $M$. By Hadamard's bound they are thus $O(n \log M_{n,1} n)$ in length. Hadamard's bound also applies to the coefficients of $(\operatorname{adj} \check{M}_i)_{i,1}$ and $(\operatorname{adj} \check{M}_i)_{i,2}$, making their lengths $O(n^2 \log M_{n,1} n)$. And it then applies again to make $\det C((\operatorname{adj} \check{M}_i)_{i,1}, (\operatorname{adj} \check{M}_i)_{i,2}))$ have length $O(n^3 \log M_{n,1} n)$. This is our bound on the length of $c$ in line 6 and hence the length of $c$ in line 7. The length of $c^j$ in line 8 is then $O(\max(\log a^2 (2M_{n,1} n)^{3n}, \log |c|)) = O(n^3 \log mn)$, with the maximum accommodating the ceiling function. Then a final application of Hadamard's bound for lines 9 and 10 makes integer lengths $O(n^4 \log mn)$. This is therefore a bound on the number of operations required by the Euclidean algorithm in line 9. □

In [12], Dinur proves the NP-hardness of short vector problems under the $\ell_\infty$-norm when $\alpha = n^{c/\log\log n}$ for some $c > 0$ by giving a direct reduction from the Boolean satisfiability problem (SAT).

As a consequence, Theorems 4.7 and 4.8 prove the same for both good Diophantine approximation and simultaneous approximation problems. (There is no gap inflation for GDA in line 11 under the $\ell_\infty$-norm.)

**Corollary 4.9.** *Good Diophantine approximation and simultaneous approximation problems are* NP-*hard under the $\ell_\infty$-norm with $\alpha = n^{c/\log\log n}$ for some $c > 0$.* □

This result is known for good Diophantine approximation [9], though the reduction SAT → SVP → GDA completed here is simpler. Chen and Meng adapt the work of Dinur as well as Rössner and Seifert [30] to reduce SAT to finding short integer vectors that solve a homogeneous system of linear equations (HLS) via an algorithm from [3], which changes the problem to finding pseudo-labels for a regular bipartite graph (PSL). The number of equations in the HLS system is then decreased to one (now called SIR), wherefrom a reduction to GDA is known [29]. Each link, SAT → PSL → HLS → SIR → GDA, is gap-preserving under the $\ell_\infty$-norm.

Short vector problems are only known to be NP-hard under the $\ell_\infty$-norm. But there are other hardness results under a general $\ell_p$-norm for which Theorems 4.7 and 4.8 can be considered complementary. See [19] for an exposition.

Another corollary is the reduction from a simultaneous approximation problem to GDA, giving the final row of Table 1. By Proposition 3.2, Algorithm 2 results in one call to SVP with integers of length $O(n \log m)$, where we can take $m$ to be the maximum magnitude among $a^{1/n^4}$ (still $\alpha = a/b$) and the integers defining $\boldsymbol{x}$. Then Theorem 4.8 implies the reduction to GDA requires $O(n^4 \log m^n n) = O(n^5 \log m)$ (absorbing the operations required by Algorithm 2) on integers of length $O(n^5 \log m)$.

**4C.** ***Further discussion.*** The last algorithm was restricted to an $\ell_p$-norm for $p \in \{1, 2, \infty\}$. So we will discuss what happens with a more general approach.

Multiplication by $MM'$, shown in (4-7), may change the gap between the length of the shortest vector in the simultaneous approximation lattice and that of the vector output by GDA or SAP. That this potential inflation does not invalidate our output relies on the set of vector norms being discrete and $\alpha$ being rational — facts that were exploited to produce the expression in (4-6). The idea behind the paragraph preceding (4-6) is to find a nonempty interval $(\alpha\lambda, \alpha'\lambda)$, where $\lambda = \min^{\times}_{\boldsymbol{q}\in\mathbb{Z}^n} \|M\boldsymbol{q}\|$, that contains no norms from the lattice defined by $M$ (or even $\mathbb{Z}^n$ for the interval tacitly given in the proof). This creates admissible inflation, $\alpha'/\alpha$, which equals (4-6).

The purpose of restricting to $\ell_1$, $\ell_2$, or $\ell_\infty$ is to facilitate finding this interval. Knowing that $(b\alpha\lambda)^2 \in \mathbb{Z}$ for some $b \in \mathbb{Z}$ simplifies the search for $\alpha'$. The same is true for any $\ell_p$-norm with $p \in \mathbb{N}$. But the immediate analogs of (4-6), (4-7), and (4-8) lead to a replacement for the very last bound used in the proof of the form

$$\frac{(5pa^p n|\det M|^{p/n})\|MA\|_{\mathrm{op}}}{2|\det M|}.$$

This makes the number of operations needed to execute line 9 depend exponentially on the input length $\log p$ (though it is still polynomial for any *fixed* $p$). We have not taken into account, however, the possibility of a nontrivial lower bound for the difference between large consecutive integers which are

sums of $n$ perfect $p$-th powers. Such a bound would allow for a longer interval, $(\alpha\lambda, \alpha'\lambda)$, that provably contains no lattice norms.

These arguments are all in effort to perfectly preserve the gap when reducing to SAP or, when $p = \infty$, GDA. The situation clarifies if a small amount of inflation is allowed. To solve a short vector problem with gap $\alpha$ using SAP with gap $\alpha' < \alpha$, inequality (4-8) becomes

$$\frac{1 + \|MA\|_{\mathrm{op}}/|c^j \det M|}{1 - \|MA\|_{\mathrm{op}}/|c^j \det M|} \leq \frac{\alpha}{\alpha'}.$$

We still need to substitute a power of $c$ for $x$ in line 8 for the purpose of Lemma 4.5. Given these two constraints, it is sufficient to take $M \leftarrow M(c^j)$ for

$$j = \left\lceil \log_{|c|} \frac{(\alpha + \alpha')\|MA\|_{\mathrm{op}}}{(\alpha - \alpha')|\det M|} \right\rceil,$$

which can be made more explicit with Lemma 4.6. There is no need to insist that $\alpha$ is rational or impose a restriction on $p \in [1, \infty]$ defining the norm.

As a final note, the reduction to SAP again adapts to inhomogeneous forms of these problems while the reduction to GDA does not. If $\boldsymbol{y} \in \mathbb{Q}^n$, then the algorithm (which now reduces the *closest* vector problem) can end by solving the simultaneous approximation problem of finding $q_0 \in \mathbb{Z}$ with $\|\{q_0\boldsymbol{x} - (MM')^{-1}\boldsymbol{y}\}\| \leq \alpha \min_{q\in\mathbb{Z}}^{\times} \|\{q\boldsymbol{x} - (MM')^{-1}\boldsymbol{y}\}\|$, using the matrix from (4-7). But unless we know that the last coordinate (where the 1 is located in (4-4)) of $(MM')^{-1}\boldsymbol{y}$ is an integer, there is no clear modification to Proposition 2.1 that permits the use of GDA.

## References


[1] M. Agrawal. Simultaneous Diophantine approximation and short lattice vectors, 2019. Accessed: 2019-12-01.

[2] F. Armknecht, C. Elsner, and M. Schmidt. Using the inhomogeneous simultaneous approximation problem for cryptographic design. In *Africacrypt*, pages 242–259. Springer, 2011.

[3] S. Arora, L. Babai, J. Stern, and E. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. System Sci.*, 54(2):317–331, 1997.

[4] W. Baocang and H. Yupu. Public key cryptosystem based on two cryptographic assumptions. *IEE Proc. Comms.*, 152(6):861–865, 2005.

[5] E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp.*, 22(103):565–578, 1968.

[6] D. J. Bernstein and T. Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[7] A. J. Brentjes. Multi-dimensional continued fraction algorithms. *MC Tracts*, 1981.

[8] W. S. Brown. On Euclid's algorithm and the computation of polynomial greatest common divisors. *J. ACM*, 18(4):478–504, 1971.

[9] W. Chen and J. Meng. An improved lower bound for approximating Shortest Integer Relation in $\ell_\infty$-norm ($\mathrm{SIR}_\infty$). *Inform. Process. Lett.*, 101(4):174–179, 2007.

[10] G. Cramer. *Introduction à l'analyse des lignes courbes algébriques*. Cramer & Cl. Philibert, 1750.

[11] A. Danilevsky. On the numerical solution of the secular equation. *Mat. Sb.*, 44(2):169–172, 1937.

[12] I. Dinur. Approximating $\mathrm{SVP}_\infty$ to within almost-polynomial factors is NP-hard. *Theor. Comput. Sci.*, 285(1):55–71, 2002.

[13] A. Frank and É. Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[14] S. D. Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[15] J. S. Hadamard. Résolution d'une question relative aux determinants. *B. Sc. Math.*, 2:240–246, 1893.

[16] H. Inoue, S. Kamada, and K. Naito. Simultaneous approximation problems of $p$-adic numbers and $p$-adic knapsack cryptosystems—Alice in $p$-adic numberland. *p-Adic Numbers Ultrametric Anal. Appl.*, 8(4):312–324, 2016.

[17] H. Iwaniec. On the problem of Jacobsthal. *Demonstr. Math.*, 11(1):225–232, 1978.

[18] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8(4):499–507, 1979.

[19] R. Kumar and D. Sivakumar. Complexity of SVP. *SIGACT News*, 32(3):40–52, 2001.

[20] J. C. Lagarias. Knapsack public key cryptosystems and Diophantine approximation. In *Eurocrypt*, pages 3–23. Springer, 1984.

[21] J. C. Lagarias. The computational complexity of simultaneous Diophantine approximation problems. *SIAM J. Comput.*, 14(1):196–209, 1985.

[22] H. W. Lenstra, A. K. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 264(4):515–534, 1982.

[23] M. Pohst. A modification of the LLL reduction algorithm. *J. Sym. Comp.*, 4(1):123–127, 1987.

[24] D. Micciancio and S. Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer Science & Business Media, 2012.

[25] H. Minkowski. *Geometrie der zahlen*, volume 40. R. G. Teubner: Leipzig/Berlin, 1910.

[26] P. Nguyen. Lattice reduction algorithms: Theory and practice. In *Eurocrypt*, pages 2–6. Springer, 2011.

[27] C. Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016.

[28] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6(1):64–94, 1962.

[29] C. Rössner and J.-P. Seifert. Approximating good simultaneous Diophantine approximations is almost NP-hard. In *MFCS*, pages 494–505. Springer, 1996.

[30] C. Rössner and J.-P. Seifert. On the hardness of approximating shortest integer relations among rational numbers. *Theor. Comput. Sci.*, 209(1-2):287–297, 1998.

[31] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *FOCS*, pages 145–152. IEEE, 1982.

Received 28 Feb 2020. Revised 1 Aug 2020.

DANIEL E. MARTIN: daniel.e.martin@colorado.edu
*Department of Mathematics, University of Colardo, Boulder, CO, United States*

VOLUME EDITORS

Stephen D. Galbraith
Mathematics Department
University of Auckland
New Zealand
https://orcid.org/0000-0001-7114-8377

---

The cover image is based on an illustration from the article "Supersingular curves with small noninteger endomorphisms", by Jonathan Love and Dan Boneh (see p. 9).



Electronic copies can be obtained free of charge from http://msp.org/obs/4 and printed copies can be ordered from MSP (contact@msp.org).

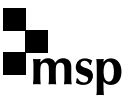
The Open Book Series is a trademark of Mathematical Sciences Publishers.

ISSN: 2329-9061 (print), 2329-907X (electronic)

ISBN: 978-1-935107-07-1 (print), 978-1-935107-08-8 (electronic)

First published 2020.

---



**MATHEMATICAL SCIENCES PUBLISHERS**
798 Evans Hall #3840, c/o University of California, Berkeley CA 94720-3840
contact@msp.org **http://msp.org**

THE OPEN BOOK SERIES 4

# Fourteenth Algorithmic Number Theory Symposium

The Algorithmic Number Theory Symposium (ANTS), held biennially since 1994, is the premier international forum for research in computational and algorithmic number theory. ANTS is devoted to algorithmic aspects of number theory, including elementary, algebraic, and analytic number theory, the geometry of numbers, arithmetic algebraic geometry, the theory of finite fields, and cryptography.

This volume is the proceedings of the fourteenth ANTS meeting, which took place 29 June to 4 July 2020 via video conference, the plans for holding it at the University of Auckland, New Zealand, having been disrupted by the COVID-19 pandemic. The volume contains revised and edited versions of 24 refereed papers and one invited paper presented at the conference.

## TABLE OF CONTENTS