# Applying the LLL Algorithm to Solve the Shortest Vector Problem (SVP)

A Study in Lattice Reduction and Cryptographic Applications

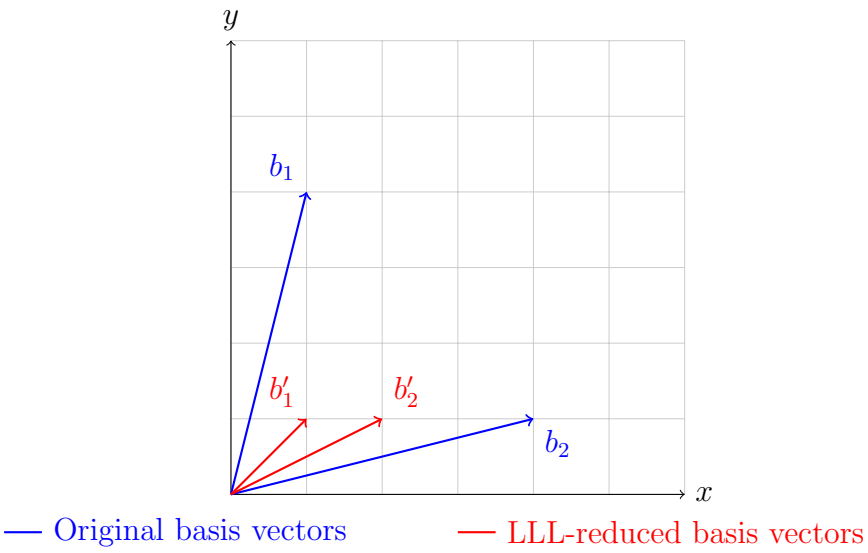
Ahadli Vugar

**Abstract**

This paper explains how the Lenstra–Lenstra–Lovász (LLL) algorithm is used for lattice basis reduction. The provided Python code implements LLL and reduces basis vectors to integer combinations relative to the original basis.

## 1 Introduction

Lattice reduction is an important technique mainly used in number theory and cryptography. The LLL algorithm generates a basis consisting of relatively short and almost orthogonal vectors. This is useful for solving problems such as the shortest vector problem (SVP) and the closest vector problem (CVP).




## 2 The Lenstra–Lenstra–Lovász (LLL) Algorithm

The *Lenstra–Lenstra–Lovász (LLL)* lattice basis reduction algorithm, introduced in 1982 by Arjen Lenstra, Hendrik Lenstra, and László Lovász [LLL82], is a polynomial-time algorithm for computing a short and nearly orthogonal basis of a lattice. Given a lattice basis

$$B = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d\} \subset \mathbb{Z}^n, \quad d \leq n,$$

the algorithm computes an *LLL-reduced* basis in time

$$\mathcal{O}\big(d^5 n \log^3 B_{\max}\big), \quad B_{\max} = \max_i \|\mathbf{b}_i\|_2.$$

## 2.1 Definition of LLL-Reduced Basis

Let $B^* = \{\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*\}$ be the Gram–Schmidt orthogonalisation of $B$, and let

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}, \quad 1 \le j < i \le n$$

be the Gram–Schmidt coefficients. A basis $B$ is *LLL-reduced* with parameter $\delta \in (0.25, 1]$ if:

1. **Size reduction:**
$$|\mu_{i,j}| \le 0.5, \quad \forall 1 \le j < i \le n.$$

2. **Lovász condition:**
$$\delta \|\mathbf{b}_{k-1}^*\|^2 \le \|\mathbf{b}_k^*\|^2 + \mu_{k,k-1}^2 \|\mathbf{b}_{k-1}^*\|^2, \quad k = 2, \ldots, n.$$

In practice, $\delta = 3/4$ is typically chosen to achieve a good reduction while retaining polynomial-time complexity.

## 2.2 Properties

For an LLL-reduced basis $B$:

- The first vector is bounded relative to the shortest lattice vector:
$$\|\mathbf{b}_1\| \le 2^{(n-1)/2} \lambda_1(L),$$
where $\lambda_1(L)$ is the length of the shortest non-zero lattice vector.

- It is also bounded in terms of the lattice determinant:
$$\|\mathbf{b}_1\| \le 2^{(n-1)/4} (\det L)^{1/n}.$$

- The product of norms satisfies
$$\prod_{i=1}^{n} \|\mathbf{b}_i\| \le 2^{n(n-1)/4} \det(L).$$

## 2.3 LLL Algorithm Pseudocode

The following description is based on Hoffstein, Pipher & Silverman (2008, Theorem 6.68), with corrections from the errata [HPS08].

**Input:**

- A lattice basis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^m$
- A parameter $\delta$ with $1/4 < \delta < 1$ (i.e., $\delta \in (0.25, 1)$)

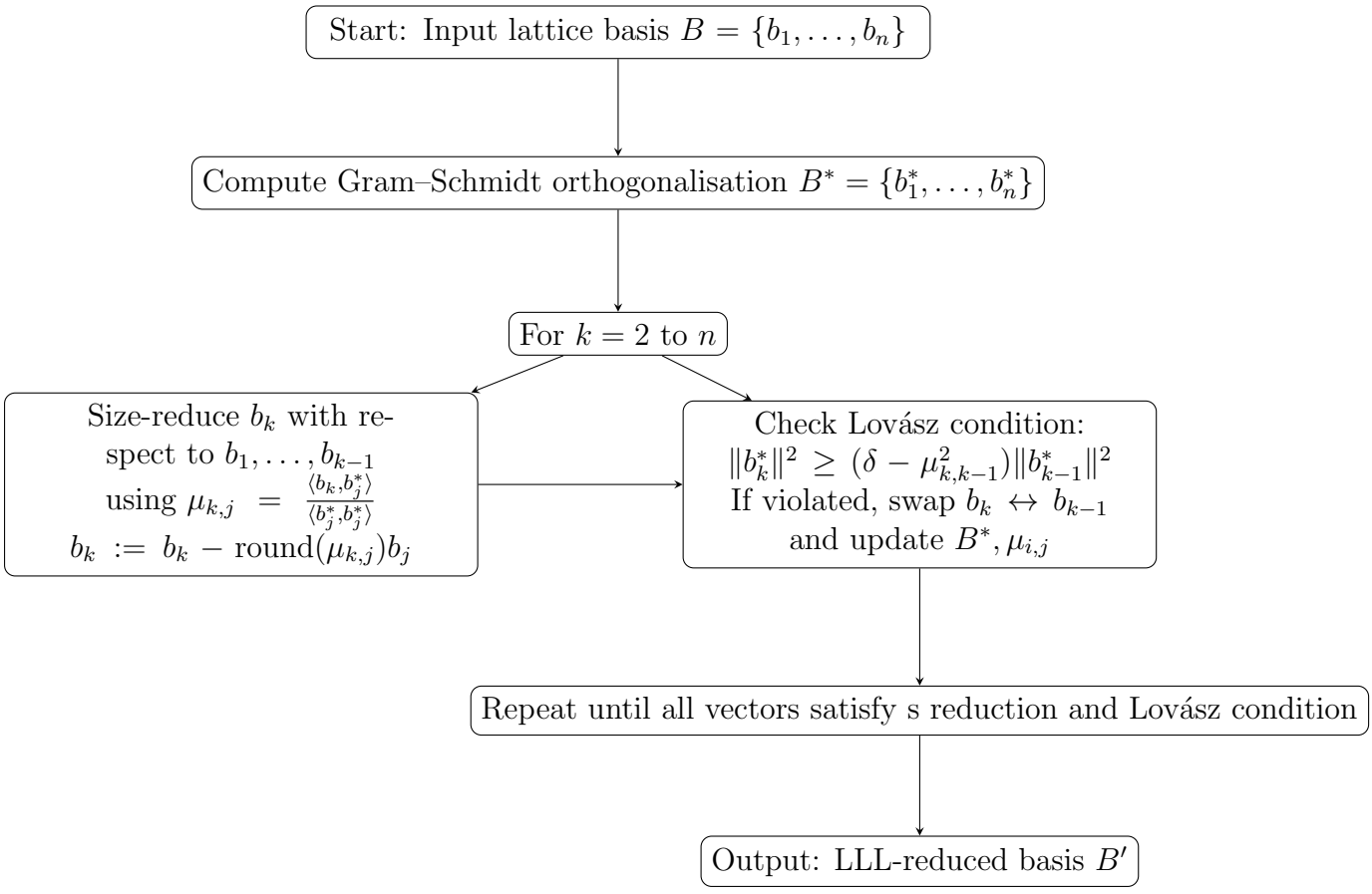
**Procedure:**

1. $B^* \leftarrow \text{GramSchmidt}(\{b_1, \ldots, b_n\}) = \{b_1^*, \ldots, b_n^*\}$ (do not normalise)
2. $\mu_{i,j} \leftarrow \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$ using current $b_i$ and $b_j^*$
3. $k \leftarrow 2$
4. **while** $k \leq n$ **do**:
   (a) **for** $j = k-1$ down to 1 **do**:
      i. **if** $|\mu_{k,j}| > 1/2$ **then** $b_k \leftarrow b_k - \lfloor \mu_{k,j} \rceil b_j$ and update $B^*$ and $\mu_{i,j}$ as needed
   (b) **if** $\langle b_k^*, b_k^* \rangle \geq (\delta - \mu_{k,k-1}^2)\langle b_{k-1}^*, b_{k-1}^* \rangle$ **then** $k \leftarrow k+1$
   (c) **else** swap $b_k$ and $b_{k-1}$, update $B^*$ and $\mu_{i,j}$, and set $k \leftarrow \max(k-1, 2)$

**Output:** The LLL-reduced basis $b_1, b_2, \ldots, b_n \in \mathbb{Z}^m$.

## 2.4 Algorithm Outline

1. Compute Gram–Schmidt orthogonalisation $B^* = \{\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*\}$.
2. For $k = 2, \ldots, n$:
   (a) *Size-reduce* $\mathbf{b}_k$ with respect to $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$.
   (b) Check Lovász condition; if violated, swap $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$ and adjust indices.
3. Repeat until all vectors satisfy size reduction and Lovász condition.



### 2.5 Applications

LLL has been applied in:

- Polynomial factorisation with rational coefficients [LLL82].
- Simultaneous rational approximation of real numbers.
- Integer linear programming in fixed dimensions.
- Cryptanalysis, including knapsack cryptosystems, NTRUEncrypt, and certain RSA variants [Gal12, Reg19].
- Integer relation algorithms for discovering integer-coefficient polynomials from approximate roots.

## 3 Python Implementation of LLL

The following Python code implements the LLL algorithm and tracks the transformation matrix to express reduced vectors in terms of the original basis:

```
import math

def dot(a,b):
    return sum(x*y for x,y in zip(a,b))

def sub(a,b):
    return [x-y for x,y in zip(a,b)]

def mul(k,v):
    return [k*x for x in v]

def lll_track(B, delta=0.75):
    n = len(B)
    B = [v[:] for v in B]
    Z = [[1 if i==j else 0 for j in range(n)] for i in range(n)]

    k = 1
    while k < n:
        for j in range(k-1,-1,-1):
            mu = dot(B[k], B[j])/dot(B[j], B[j])
            q = round(mu)
            B[k] = sub(B[k], mul(q,B[j]))
            Z[k] = sub(Z[k], mul(q,Z[j]))
        mu_prev = dot(B[k],B[k-1])/dot(B[k-1],B[k-1])
        if dot(B[k],B[k]) >= (delta - mu_prev**2) * dot(B[k-1],B[
           k-1]):
            k += 1
        else:
            B[k], B[k-1] = B[k-1], B[k]
            Z[k], Z[k-1] = Z[k-1], Z[k]
            k = max(k-1,1)
```

```
    return B, Z

def print_vectors(B, Z):
    print("LLL-reduced basis:")
    for i,(v,z) in enumerate(zip(B,Z),1):
        vec = ", ".join(f"{int(round(x)):>3}" for x in v)
        terms = []
        for j, coef in enumerate(z):
            coef = int(round(coef))
            if coef != 0:
                terms.append(f"{coef}*b{j+1}")
        print(f"b{i}: V{{{vec}}}  =  {' + '.join(terms)}")

B2 = [[5,6],[4,3]]
reduced, Z = lll_track(B2)
print_vectors(reduced, Z)
```

# 4 Example Output

Consider the lattice basis:

$$B_2 = \begin{bmatrix} 5 & 6 \\ 4 & 3 \end{bmatrix}.$$

Applying the LLL algorithm to this basis produces the LLL-reduced basis:

$$b_1' = V\{2,1\} = -1 \cdot b_1 + 1 \cdot b_2,$$
$$b_2' = V\{4,3\} = 0 \cdot b_1 + 1 \cdot b_2.$$

Each $b_i'$ is expressed as a linear combination of the original basis vectors $b_1$ and $b_2$.

## 4.1 Interpretation

- The vectors $b_1'$ and $b_2'$ form a new basis of the same lattice generated by $B_2$.
- The integer coefficients in $V\{\cdot\}$ indicate how each reduced vector can be expressed as an integer linear combination of the original basis vectors. For example, $b_1' = -1 \cdot b_1 + 1 \cdot b_2$ shows that $b_1'$ is obtained by subtracting $b_1$ from $b_2$.
- The reduced basis vectors are shorter and closer to orthogonal compared to the original vectors. In particular, the norm of $b_1'$ is smaller than the norm of $b_1$, which demonstrates the "shortening" property of LLL.
- $b_2'$ is aligned more closely with the coordinate axes, reflecting the LLL algorithm's effort to produce nearly orthogonal vectors.

# 5 Conclusion

The LLL algorithm significantly reduces lattice bases to shorter, more orthogonal vectors. By keeping track of these integer combinations, any vector in the reduced lattice basis can be represented in terms of the original lattice basis, permitting analyses to be performed on lattice structures. Among cryptographic applications, being able to represent any vector in the reduced lattice basis in terms of the original lattice basis is especially vital, as LLL algorithms are used in attacks on lattice-based cryptographic systems such as knapsack cryptosystems, some variants of RSA, NTRU encryption, and in cryptographic primitives such as secure lattice-based cryptosystems, encryption algorithms, digital signatures, and homomorphic encryption.

# References


[Bos10] Wieb Bosma. 4. lll. Lecture notes, 2010. Retrieved 28 February 2010.

[Div18] Jose Divasón. A formalization of the lll basis reduction algorithm. In *Interactive Theorem Proving, ITP 2018*, volume 10895, pages 160–177, 2018.

[Gal12] Steven Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[HPS08] Jeffrey Hoffstein, Jill Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2008.

[LLL82] A. K. Lenstra, H. W. Jr. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[NS09a] Phong Q. Nguyen and Damien Stehlé. An lll algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.

[NS09b] Phong Q. Nguyen and Damien Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Transactions on Algorithms*, 5(4):1–48, 2009.

[OtR85] Andrew Odlyzko and Herman J. J. te Riele. Disproving mertens conjecture. *Journal für die reine und angewandte Mathematik*, 357:138–160, 1985.

[Reg19] Oded Regev. Lattices in computer science: Lll algorithm. New York University, 2019. Retrieved 1 February 2019.

[Sil15] Joseph Silverman. Introduction to mathematical cryptography errata. Brown University Mathematics Dept., 2015.

[Sim07] D. Simon. Selected applications of lll in number theory. LLL+25 Conference, Caen, France, 2007.

[W+11] D. Wübben et al. Lattice reduction. *IEEE Signal Processing Magazine*, 28(3):70–91, 2011.