Solving problems with the LLL algorithm

Mark van Hoeij

Florida State University

hoeij@math.fsu.edu

October 17, 2015
A lattice is a discrete \( \mathbb{Z} \)-module \( \subseteq \mathbb{R}^n \)

**Example:** If \( b_1, b_2 \in \mathbb{R}^2 \) are \( \mathbb{R} \)-linearly independent then

\[
L = \text{SPAN}_{\mathbb{Z}}(b_1, b_2) = \{ n_1 b_1 + n_2 b_2 \mid n_1, n_2 \in \mathbb{Z} \}
\]

is a lattice of rank 2 and \( b_1, b_2 \) is a basis of \( L \).

**Lattice basis reduction**

**Input:** a basis of \( L \).

**Output:** a good basis of \( L \).

- For rank 2 this is easy (\( \approx \) Euclidean algorithm). For a long time it was not known how to handle rank \( n > 2 \) until:
- Has lots of applications!
Figure 1: A lattice with a bad basis $b_1, b_2$ and a good basis $v_1, v_2$.

\[ L = \{ \text{dots in Figure 1} \} = \text{SPAN}_\mathbb{Z}(b_1, b_2) = \text{SPAN}_\mathbb{Z}(v_1, v_2) \]
Gram-Schmidt process \((n = 2)\)

1. \(v_1^* = v_1\)
2. \(v_2^* = v_2 - \mu v_1^*\) \(\text{Compute } \mu \in \mathbb{R} \text{ such that } v_1^* \perp v_2^*.\)

G.S.-vectors \(v_1^*, \ldots, v_n^*\) \(\sim\) very useful information on \(L\) even though \(v_2^*, \ldots, v_n^*\) are generally not in \(L\).
$b_1, b_2$ is a bad basis because:

1. $b_1, b_2$ are almost parallel,
2. $\|b_2^*\| \ll \|b_2\|$ (good basis $\implies \|v_i^*\| \approx \|v_i\|$)
3. $\min(\|b_1^*\|, \|b_2^*\|)$ is tiny, and thus a poor bound:

Let $b_{\min} := \min(\|b_i^*\|)$

Shortest-vector-bound: $b_{\min} \leq \|\text{shortest } v \neq 0 \text{ in } L\|$
Given a bad basis $b_1, b_2$, how to find a good basis?

1. Subtract an integer-multiple of a one vector from another. (First step in the picture is: replace $b_1$ with $b_1 - b_2$).
2. Repeat as long as Step 1 can make a vector shorter.

This strategy works well for rank $n = 2$.

Efforts to extend to $n > 2$ failed until the breakthrough [LLL 1982], which uses lengths of G.S.-vectors $b_i^*$ and not the lengths of the $b_i$ themselves!
Application #1: \( p = a^2 + b^2 \)

**Theorem (Fermat)**

If \( p \) prime and \( p \equiv 1 \mod 4 \), then \( p = a^2 + b^2 \) for some \( a, b \in \mathbb{Z} \).

**Example:** \( p = 10^{400} + 69 = 100000000000000000000000000000 \ldots \ldots \ldots 00000000000069 \)

How to find \( a, b \in \mathbb{Z} \) with \( a^2 + b^2 \) equal to \( p \)?

**Observation:** \( a^2 + b^2 \equiv 0 \mod p \)

Hence \( a \equiv \alpha b \mod p \) for some solution of \( \alpha^2 + 1 \equiv 0 \mod p \).

Compute \( \alpha \) (e.g. Berlekamp’s algorithm). Then

\[
\begin{pmatrix} \pm a \\ b \end{pmatrix} \in \text{SPAN}_{\mathbb{Z}}\left( \begin{pmatrix} p \\ 0 \end{pmatrix} , \begin{pmatrix} \alpha \\ 1 \end{pmatrix} \right)
\]

(the \( \pm \) is irrelevant) \((\alpha^2 + 1 \equiv 0 \text{ has two solutions } \mod p)\)
Application #1: $p = a^2 + b^2$

$p = 10^{400} + 69 = 1000000000000000000 \ldots \ldots 000000000000000069$

Find: $a, b \in \mathbb{Z}$ with $a^2 + b^2 = p$.

If

$$v = \begin{pmatrix} a \\ b \end{pmatrix} \in \text{SPAN}_{\mathbb{Z}}(\begin{pmatrix} p \\ 0 \end{pmatrix}, \begin{pmatrix} \alpha \\ 1 \end{pmatrix})$$

then

$$\|v^2\| = a^2 + b^2 \equiv (\alpha b)^2 + b^2 = (\alpha^2 + 1)b^2 \equiv 0 \mod p.$$

So $\|v\|^2$ is divisible by $p$.

So $\|v\|^2$ is $p$ if $v$ is short enough: $0 < \|v\|^2 < 2p$

Such $v$ is easy to find in a good basis.

However, $\left\{ \begin{pmatrix} p \\ 0 \end{pmatrix}, \begin{pmatrix} \alpha \\ 1 \end{pmatrix} \right\}$ is a bad basis (angle $\approx 10^{-400}$ radians!)
Application #1: \( p = a^2 + b^2 \)

**Find:** \( a, b \in \mathbb{Z} \) with \( a^2 + b^2 = p \).

The simple strategy from slide 6 reduces the bad basis to a good basis.

From it we can immediately read off a solution:

\[
\begin{pmatrix}
a \\
b
\end{pmatrix} = \begin{pmatrix}
858038135984417422601 \ldots \ldots 0688928009299704710 \\
513585978387637054198 \ldots \ldots 0249251426547937937
\end{pmatrix}
\]

The computation (finding \( \alpha \) and reducing the basis) takes \(< 0.1 \) seconds.
Apply the Gram-Schmidt process to $b_1, \ldots, b_n$

- $b_1^* = b_1$
- $b_2^* = b_2 - \mu_{2,1} b_1^*$  
  (take $\mu_{ij} \in \mathbb{R}$ s.t. $b_i^* \perp b_j^*$)  
  ($j < i$)
- $b_3^* = b_3 - \mu_{3,1} b_1^* - \mu_{3,2} b_2^*$
  ...

$\det(L) = \|b_1^*\| \cdots \|b_n^*\|$  
(the determinant is basis-independent).

Replacing $b_i \leftarrow b_i - k b_j$ reduces $\mu_{ij}$ to $\mu_{ij} - k$  
($k \in \mathbb{Z}$).

LLL lattice basis reduction

1. Reduce to $|\mu_{ij}| \leq 0.51$  
   ($\leq 0.5$ if $\mu_{ij}$ known exactly).

2. If swapping $b_{i-1} \leftrightarrow b_i$ increases $\|b_i^*\|$ at least 10% for some $i$, then do so and go back to Step 1.

Output: good basis: $\|b_{i-1}^*\| \leq 1.28 \cdot \|b_i^*\|$ and $|\mu_{ij}| \leq 0.51$
Properties of LLL reduced basis

If Output(LLL) = \( b_1, \ldots, b_n \) then

\[
\|b_1^*\| \leq 1.28 \cdot \|b_2^*\| \leq 1.28^2 \cdot \|b_3^*\| \leq \cdots \leq 1.28^{n-1} \cdot b^\text{min}
\]

hence

\[
\|b_1\| \leq f_n \cdot \|\text{shortest } v \neq 0 \text{ in } L\|
\]

"fudge factor" \( f_n = 1.28^{n-1} \)

If \( L \) has a short non-zero vector then \( b_1 \) is not much longer.
If \( L \) has short independent \( S_1, \ldots, S_k \) then \( b_1, \ldots, b_k \) are not much longer.

Many problems \( P \) can be solved this way:

1. Construct a lattice \( L = \text{SPAN}_\mathbb{Z}(b_1, \ldots, b_n) \) for which Solution(\( P \)) can be read some solution-vectors \( S_1, \ldots, S_k \in L \).
2. Construct \( L \) in such a way that vectors in \( L - \text{SPAN}_\mathbb{Z}(S_1, \ldots, S_k) \) are \( \geq f_n \) times longer than \( S_1, \ldots, S_k \).
3. Replace \( b_1, \ldots, b_n \) by an LLL-reduced basis, then:
4. \( S_1, \ldots, S_k \in \text{SPAN}_\mathbb{Z}(b_1, \ldots, b_k) \). Separates \( S_1, \ldots, S_k \) from rest of \( L \).
Application #2: Reconstruct algebraic number from an approximation

Suppose $\beta$ is an algebraic number, a root of an irreducible $P \in \mathbb{Z}[x]$. Suppose $P = \sum_{i=0}^{n-1} c_i x^i$ with $|c_i| \leq 10^b$.

Suppose we have an approximation $\alpha \in \mathbb{R}$ with error $< 10^{-a}$. We need $a \geq bn + \epsilon n^2$ because $P$ has $\approx bn$ digits of data. (fudge factor $f_n \sim \epsilon n^2$)

Problem: Compute exact $\beta$ (compute $P$) from the approximation $\alpha$.

Can read $P$ from solution-vector $S := (c_0, \ldots, c_{n-1}) \in \mathbb{Z}^n$.

Problem: $\mathbb{Z}^n$ contains unwanted vectors as well.

$S = \text{Sculpture} \subseteq \text{rock}$. Use chisel to separate unwanted rock.

Idea:

Add one (or more) entries $\mathbb{Z}^n \to \mathbb{Z}^{n+1}$ that make unwanted vectors at least $f_n$ times longer than $S$. Use LLL to separate them.
Application #2:
Reconstruct algebraic number from an approximation

Define $E : \mathbb{Z}^n \to \mathbb{Z}^{n+1}$

$$(c_0, \ldots, c_{n-1}) \mapsto (c_0, \ldots, c_{n-1}, \sum c_i [10^a \alpha^i])$$

$b_1, \ldots, b_n := E(\text{standard basis of } \mathbb{Z}^n)$

$b_1, \ldots, b_n$ spans a lattice $L \subseteq \mathbb{Z}^{n+1}$ of rank $n$.

$b_1, \ldots, b_n$ is a bad basis. **Typical example:** degree($P$) < 40 and $|\text{coefficients}| \leq 10^{100}$. Angles will be $\approx 10^{-4000}$ radians!

LLL quickly turns this into a good basis.

With suitable precision $a$, this either leads to the minpoly $P = \sum c_i x^i$ or a proof that no $P$ exists within the chosen bounds.
Theorem (LLL 1982)

Factoring in $\mathbb{Q}[x]$ can be done in polynomial time.

Proof sketch: Compute a root of $f$ to precision $a$. Use the previous slide to compute its minpoly. Choose $a$ in such a way that this produces either a non-trivial factor, or an irreducibility proof.

Remarks:

2. Neither was used in computer algebra systems; [Zassenhaus 1969] (not polynomial time!) is usually much faster.
3. Faster algorithm [vH 2002]: apply LLL to a much smaller lattice.
Integer solutions of approximate and/or modular equations.

Find: \( x_1, \ldots, x_n \in \mathbb{Z} \) when given:

1. Approximate linear equations: \[ |a_{i,1}x_1 + \cdots + a_{i,n}x_n| < \epsilon_i \quad (a_{ij} \in \mathbb{R}) \]
2. or modular linear equations \( b_{i,1}x_1 + \cdots + b_{i,n}x_n \equiv 0 \pmod{m_i} \)
3. or a mixture of the above, and other variations

then use LLL.

Remarks:

- Linear equations over \( \mathbb{R} \): Ordinary linear algebra gives a basis solutions over \( \mathbb{R} \), but this does not help to find solutions over \( \mathbb{Z} \).
- Equations (approximate and/or modular etc.) are inserted in a lattice by adding entries (like \( E : \mathbb{Z}^n \to \mathbb{Z}^{n+1} \) on p. 13).
- [vH, Novocin 2010]: Efficient method for:
  "amount(data in equations)" \( \gg \) "amount(data in solution)"
Application: Integer relation finding

Given \(a_1, \ldots, a_n \in \mathbb{R}\), find \(x_1, \ldots, x_n \in \mathbb{Z}\) (say \(|x_i| \leq 10^{100}\)) with
\[
a_1x_1 + \cdots + a_nx_n = 0.
\]

Notable algorithms:

- [LLL 1982]

Beautiful applications e.g. PSLQ \(\leadsto\) Bailey-Borwein-Plouffe formula for \(\pi\)

Remarks:

- [LLL 1982] is a more complete solution because [PSLQ 1992] gave no bound(precision(\(a_i\))) \(\leadsto\) provable result.
- PSLQ won SIAM Top 10 Algorithms of the Century award.
- The fastest implementations I have seen can handle \(n = 500\) (using modern versions of LLL).
Recent paper: [Derickx, vH, Zeng]
Computing Galois representations and equations for modular curves $X_H(\ell)$.

This paper uses LLL for:

1. **Finding low-degree functions:**
   To a function, associate a vector containing root/pole orders. Low degree functions have short vectors, use **LLL** to find them.

2. **The paper computes an algebraic number $\alpha$ mod primes $p_1, \ldots, p_n$.**
   Use **LLL** to reconstruct the minpoly $P$ of $\alpha$.

3. **$P$ has huge coefficients ($> 10^{1000}$).**
   Use **LLL** to find a smaller $Q \in \mathbb{Z}[x]$ with $\mathbb{Q}[x]/(P) \cong \mathbb{Q}[x]/(Q)$.

4. **One of the tests for $Q$ is to compute its Galois group.**
   Galois group computations use **LLL directly and indirectly** (factoring resolvent polynomials uses [vH 2002], which uses LLL).
Other applications

[Imamoglu, vH 2015]: solve linear differential equations in terms of hypergeometric functions $2F_1(a, b; c | f)$ where $f$ is a rational function.

Problem:
We can recover $f$ if the first term $cx^d$ of the Taylor series of $f$ is known. We have no direct way to compute $c$, but given $c$, we can check if $c$ is OK.

Strategy for finding $c$:
Work modulo a prime $p$, then try all $p$ cases! Then work mod higher $p$-powers (or other primes) until one can recover $c \in \overline{\mathbb{Q}}$ with LLL.

This strategy has other applications. It may help if a system of polynomial equations is too complicated to be solved directly with Gröbner basis.
Polynomial factorization until 2000.

\( f \in \mathbb{Z}[x] \), degree \( N \), square-free and primitive.

### Step 1:

Factor \( f \equiv f_1 \cdots f_r \mod p \)

and Hensel lift:

\[
f \equiv f_1 \cdots f_r \mod p^a
\]

### Step 2 in [Zassenhaus 1969]:

- Try \( S \subseteq \{f_1, \ldots, f_r\} \) with \( 1, 2, \ldots \lfloor r/2 \rfloor \) elements, and check if the product (lifted to \( \mathbb{Z}[x] \)) is a factor of \( f \) in \( \mathbb{Z}[x] \).
- Up to \( 2^{r-1} \) cases \( S \subseteq \{f_1, \ldots, f_r\} \) (Combinatorial Problem)

### [LLL 1982] Bypasses Combinatorial Problem:

- \( L := \{(c_0, \ldots, c_{N-1}) \mid \sum c_i x^i \equiv 0 \mod (p^a, f_1)\} \) (rank = \( N \))
- LLL-reduce, take first vector, and compute \( \gcd(f, \sum c_i x^i) \).
Factor \( f \) in \( \mathbb{Q}[x] \), degree \( N = 1000 \)

[LLL 1982] reduces a lattice of rank \( N \)

- Algorithm runs in polynomial time.
- However, lattice reduction for rank 500 is very time consuming.
- rank \( N = 1000 \) is a problem!

[Zassenhaus 1969] tries \( \leq 2^{r-1} \) cases

- \( r = 12 \) → 😊 (Finishes in seconds)
- \( r = 80 \) → 😞 (Millions of years, even with \( 10^9 \) cases per second)

If: \( f \) has degree \( N = 1000 \), few factors in \( \mathbb{Q}[x] \) but \( r = 80 \) factors in \( \mathbb{F}_p[x] \)

However, **80 bits of data** reduces CPU time from **eons** to **seconds**!

[vH 2002]: Use lattice reduction to compute **only those bits!** \( (\text{rank } \approx r) \)
Factor $f$ in $\mathbb{Q}[x]$, degree $N$, with $f \equiv f_1 \cdots f_r \mod p^a$

[LLL 1982]: (polynomial time)
Reduce a lattice of rank $N$ (and large entries)

[Zassenhaus 1969]: (not poly time, usually faster than [LLL 1982])
Try (exponentially many) subsets $S \subseteq \{f_1, \ldots, f_r\}$ (Combinatorial Problem)

[vH 2002]: (fastest)
- $S \iff (v_1, \ldots, v_r) \in \{0, 1\}^r$
- Insert data: $\{0, 1\}^r \subseteq \mathbb{Z}^r \to \mathbb{Z}^{r+\epsilon}$ to construct lattice of rank $r + \epsilon$
- Sequence of lattice reductions leads to $v_1, \ldots, v_r$, and hence $S$.
- Test (as in Zassenhaus) if $\prod S \mod p^a \sim$ a factor in $\mathbb{Q}[x]$.
- [vH 2002]: correctness and termination proof, no complexity bound.
- Complexity bound: [vH, Novocin 2010] and [vH 2013].
\[ S \subseteq \{f_1, \ldots, f_r\} \iff v \in \{0, 1\}^r \subseteq \mathbb{Z}^r \rightarrow \mathbb{Z}^{r+\epsilon} \]

**If:** we have: approximate/modular linear equations for \( v = (v_1, \ldots, v_r) \)

**then:** lattice reduction \( \leadsto v \).

**However,** the factor \( \prod S = \prod f_i^{v_i} \) of \( f \) depends non-linearly on \( v \).

**Idea:** coefficients \( (f \cdot f'_i / f_i) \leadsto \) equations for \( v \)

(\( f'_i / f_i \) is the logarithmic derivative; turns products into sums)

Remarks:

- [vH 2002] runs fast; lattice reduction is only used to construct \( r \) bits.
- Lots of data in coefficients \( (f \cdot f'_i / f_i) \): \( N \cdot \log_2(p^a) \) bits \( \leadsto r \) bits.
- How to select from this data? (select all \( \leadsto \) no speedup)
- Arbitrary choice \( \leadsto \) fast in practice but no complexity bound.
- [vH Novocin 2010] and [vH 2013] solve this \( \leadsto \) best complexity bound and practical performance, in the same algorithm.
References, polynomial factorization over $\mathbb{Q}$

H. Zassenhaus (1969)
On Hensel Factorization I.
*J. Number Theory*, 1, 291-311

Lenstra, Lenstra, Lovász (1982)
Factoring polynomials with rational coefficients
*Math Ann.*, 261, 515-534

M. van Hoeij (2002)
Factoring polynomials and the knapsack problem
*J. of Number Theory*, 95, 167-189

M. van Hoeij, A. Novocin (2010)
Gradual sub-lattice reduction and a new complexity for factoring polynomials
*LATIN*, 539-553

M. van Hoeij (2013)
The complexity of factoring univariate polynomials over the rationals
*ISSAC’2013 tutorial*, slides at www.math.fsu.edu/~hoeij