

Babai's Nearest Plane Algorithm

Eric Ren, Sage Binder, and Katherine Kosaian

February 6, 2026

Abstract

γ -CVP is the problem of finding a vector in L that is within γ times the closest possible to t , where L is a lattice and t is a target vector. If the basis for L is LLL-reduced, Babai's Closest Hyperplane algorithm solves γ -CVP for $\gamma = 2^{n/2}$, where n is the dimension of the lattice L , in time polynomial in n . This session formalizes said algorithm, using the AFP formalization of LLL [2, 1] and adapting a proof of correctness from the lecture notes of Stephens-Davidowitz [4].

Contents

1	Introduction	1
2	Locale setup for Babai	4
3	Coordinates	5
4	Lattice Lemmas	8
5	Lemmas on closest distance	9
6	More linear algebra lemmas	9
7	Coord-Invariance	11
8	Bound on distance to target, with ϵ factor.	12

1 Introduction

The (exact) *closest vector problem* (CVP) is the problem of finding the closest vector within a lattice L to a target vector t . This is equivalent to finding the shortest vector in the *lattice coset* $L - t := \{l - t : l \in L\}$. There is a corresponding family of weaker problems, γ -CVP (where γ is some real parameter), where one needs only find a vector in $L - t$ whose length is at most γ times the shortest possible. Through a reduction to the *shortest*

vector problem [4], solutions to these problems may be used to factor rational polynomials. This problem is therefore of cryptographic interest.

Although exact CVP (or 1-CVP) is NP-Complete [3], Babai's Nearest Plane Algorithm solves $2^{n/2}$ -CVP, where n is the dimension of L , in polynomial time, provided that L is presented using an LLL-reduced basis with parameter $\alpha = 4/3$. The proof in this document is mostly a straightforward algebraicization of the proof in Stephens-Davidowitz' lecture notes. It makes use of the coordinate systems defined by the original basis (denoted β) and the Gram-Schmidt orthogonalization of that basis (denoted $\tilde{\beta}$). Let $[u]_\beta$ denote the representation of a vector u under β , with coordinates $[u]_\beta^j$; $j = 1, \dots, n$ (likewise for $\tilde{\beta}$). Also, let s_i denote the output of the algorithm after step i and let d be the shortest lattice coset vector, as witnessed by the vector v . The proof works by analysing the coordinates of $[s_n]_{\tilde{\beta}}$, showing that all are at most $1/2$ and that some later coordinates are exactly those of $[v]_{\tilde{\beta}}$.

The algorithm modifies coordinate $n - i$ in both bases for the last time in step i (formalized in lemma `coord_invariance`), during which both coordinates are decreased below $1/2$ (formalized in lemma `small_coord`). Combined, these facts imply that the output s_n has $\left|[s_n]_{\tilde{\beta}}^j\right| \leq 1/2$ for all indices j .

Since $\tilde{\beta}$ is orthogonal, we have

$$\|s_n\|^2 = \sum_{i=1}^n \left([s_n]_{\tilde{\beta}}^i \|\tilde{\beta}_i\| \right)^2, \quad (1)$$

so the preceding coordinate bounds $\|s_n\|^2$ by $\frac{1}{4} \sum_{i=1}^n \|\tilde{\beta}_i\|^2$. If the $\tilde{\beta}_i$ are all short compared to d , this bound suffices. In fact, if there is any short vector $\tilde{\beta}_I$ in $\tilde{\beta}$ then because β is LLL-reduced, any vector preceding $\tilde{\beta}_I$ in $\tilde{\beta}$ will not be much longer. This bounds the first I terms in Equation 1. By selecting I maximal, we may assume that $\tilde{\beta}$ ends in a series of $n - I$ long vectors. In this case it can be shown $[v]_{\tilde{\beta}}^j$ and $[s_n]_{\tilde{\beta}}^j$ differ by an integral amount for $j = I + 1, \dots, n$. Therefore, if $[v]_{\tilde{\beta}}^j$ and $[s_n]_{\tilde{\beta}}^j$ differ at all, they differ by at least 1, which would mean $\left|[v]_{\tilde{\beta}}^j\right| \geq 1/2$, since $\left|[s_n]_{\tilde{\beta}}^j\right| \leq 1/2$. This would force v to be longer than d , a contradiction. So $[v]_{\tilde{\beta}}^j = [s_n]_{\tilde{\beta}}^j$ for $j = I + 1, \dots, n$, which gives a tighter bound on the last $n - I$ terms in equation 1.

Precisely, let I denote $\max\{i : \|\tilde{\beta}_i\| \leq 2d\}$, meaning for all indices $j > I$, $\|\tilde{\beta}_j\| > 2d$. Now, for all $j > I$, $d^2 = \|v\|^2 \geq ([v]_{\tilde{\beta}}^j)^2 \|\tilde{\beta}_j\|^2 > ([v]_{\tilde{\beta}}^j)^2 \cdot 4d^2$, meaning $1/4 > (\tilde{\beta}^j)^2$, or $1/2 > \left|[v]_{\tilde{\beta}}^j\right|$. Since $\left|[s_j]_{\tilde{\beta}}^j\right| \leq 1/2$ from the previous section, $\left|[v]_{\tilde{\beta}}^j - [s_j]_{\tilde{\beta}}^j\right| < 1$. Using properties of the change-of-basis between $\beta, \tilde{\beta}$ formalized in the LLL AFP session, we show that $[v]_{\tilde{\beta}}^j - [s_j]_{\tilde{\beta}}^j =$

$[v]_\beta^j - [s_j]_\beta^j = [v - s_j]_\beta^j$, so that $\left| [v - s_j]_\beta^j \right| < 1$. But since $v - s_j$ lies in the lattice, $[v - s_j]_\beta^j$ is integral, so $\left| [v - s_j]_\beta^j \right| = 0$, meaning $[v]_\beta^j = [s_j]_\beta^j$. Lemma `coord_invariance` gives that $[v]_\beta^j = [s_j]_\beta^j = [s_n]_\beta^j$. This is formalized by lemma `correct_coord`.

Now $\|s_n\|^2 = \sum_{i=1}^n ([s_n]_{\tilde{\beta}}^i \|\tilde{\beta}_i\|)^2$, since $\tilde{\beta}$ is orthogonal. Splitting the sum around I equates this to $\sum_{i=1}^I ([s_n]_{\tilde{\beta}}^i)^2 + \sum_{i=I+1}^n ([s_n]_{\tilde{\beta}}^i)^2$. Lemma `small_coord` bounds the terms in the first sum by $\|\tilde{\beta}_i\|^2/4$, while lemma `correct_coord` bounds the terms in the second sum by d^2 , giving $\|s_n\|^2 \leq (n - I)d^2 + \sum_{i=1}^I \|\tilde{\beta}_i\|^2/4$. If β is LLL-reduced with parameter α , $\|\tilde{\beta}_i\|^2 \leq \alpha^I \|\tilde{\beta}_I\|^2$ for all $i \leq I$, which, by the definition of I , is at most $4d^2$. So $\|s_n\|^2 \leq ((n - I) + I\alpha^I)d^2 \leq n\alpha^n d^2$. The standard choice of $\alpha = 4/3$ gives $\|s_n\|^2 \leq 2^n d^2$. All of this is formalized in the final section, which culminates in the main theorem.

To avoid having to prove that a shortest vector exists, we use the definition $\inf\{\|u - t\| : u \in L\}$ for d instead of $\min\{\|u - t\| : u \in L\}$ and rephrase the arguments above to allow $\|v\|$ to exceed d by a small constant factor ϵ . This workaround and its details are contained in the section on the closest distance and negligibly change the rest of the proof.

theory *Babai-Algorithm*

imports

LLL-Basis-Reduction.LLL-Impl

HOL.Archimedean-Field

HOL-Analysis.Inner-Product

begin

fun *calculate-c*:: *rat vec* \Rightarrow *rat vec list* \Rightarrow *nat* \Rightarrow *int* **where**

*calculate-c s L1 n = round ((s * (L1!((dim-vec s) - n))) / (sq-norm-vec (L1!((dim-vec s) - n))))*

fun *update-s*:: *rat vec* \Rightarrow *rat vec list* \Rightarrow *rat vec list* \Rightarrow *nat* \Rightarrow *rat vec* **where**

update-s sn M Mt n = ((rat-of-int (calculate-c sn Mt n)) \cdot_v M!((dim-vec sn) - n))

fun *babai-help*:: *rat vec* \Rightarrow *rat vec list* \Rightarrow *rat vec list* \Rightarrow *nat* \Rightarrow *rat vec* **where**

babai-help s M Mt 0 = s |

babai-help s M Mt (Suc n) = (let B = (babai-help s M Mt n) in B - (update-s B M Mt (Suc n)))

This assumes an LLL-reduced input and outputs a short vector of the form $v + t$, with $v \in L$

fun *babai-of-LLL* :: *rat vec* \Rightarrow *rat vec list* \Rightarrow *rat vec* **where**
babai-of-LLL *s M* = *babai-help* *s M* (*gram-schmidt* (*dim-vec s*) *M*) (*dim-vec s*)

This begins with a non-reduced basis and outputs a vector v in L which is close to t

fun *full-babai* :: *int vec list* \Rightarrow *rat vec* \Rightarrow *rat* \Rightarrow *int vec* **where**
full-babai *fs target* α =
map-vec int-of-rat
(*babai-of-LLL*
(*uminus target*)
(*LLL.RAT* (*LLL-Impl.reduce-basis* α *fs*))
+ *target*)

end

theory *Babai-Correct*

imports *Babai-Algorithm*

LLL-Basis-Reduction.LLL-Impl

Jordan-Normal-Form.DL-Rank

BenOr-Kozen-Reif.More-Matrix

begin

This theory contains the proof of correctness of the algorithm. The main theorem is “theorem babai-correct”, under the locale “babai-with-assms”. To use the theorem, one needs to show that lattice, the vectors in the lattice basis, and the target vector all have the same dimension, that the lattice basis vectors are linearly independent, and that the lattice basis is LLL-weakly-reduced for some parameter $\alpha \geq 4/3$.

2 Locale setup for Babai

locale *babai* =
fixes *M* :: *int vec list*
fixes *t* :: *rat vec*
fixes α :: *rat*
assumes *length-M*: *length M* = *dim-vec t*
begin

abbreviation *n* **where** $n \equiv \text{length } M$

sublocale *LLL* *n n M* α *<proof>*

abbreviation *coset*::*rat vec set* **where** $\text{coset} \equiv \{(map-vec \text{rat-of-int } x) - t \mid x. x \in L\}$

abbreviation *Mt* **where** $Mt \equiv \text{gram-schmidt } n \text{ (RAT } M)$

definition *s* :: *nat* \Rightarrow *rat vec* **where**

$s \ i = \text{babai-help } (uminus \ t) \ (\text{RAT } M) \ Mt \ i$

definition *closest-distance-sq*:: *real* **where**

$\text{closest-distance-sq} = \text{Inf } \{\text{real-of-rat } (\text{sq-norm } x::\text{rat}) \mid x. x \in \text{coset}\}$

end

Locale setup with additional assumptions required for main theorem. Contains an arbitrary extra constant ϵ which appears in the final bound in this locale, giving a theorem quantified over all $\epsilon > 1$. The next locale, “babai-with-assms,” uses this theorem to prove a theorem without ϵ .

locale *babai-with-assms- ϵ* = *babai* +
fixes *mat-M mat-M-inv*:: *rat mat*
fixes ϵ :: *real*
assumes *basis*: *lin-indep M*
defines *mat-M* \equiv *mat-of-cols n (RAT M)*
defines *mat-M-inv* \equiv
 (if (invertible-mat mat-M) then SOME B. (inverts-mat B mat-M) \wedge (inverts-mat
mat-M B) else (0_m n n))
assumes *inv*: *invertible-mat mat-M*
assumes *reduced*: *weakly-reduced M n*
assumes *non-trivial*: *0 < n*
assumes *alpha*: $\alpha \geq 4/3$
assumes ϵ : $\epsilon > 1$
begin

lemma *dim-vecs-in-M*:
shows $\forall v \in \text{set } M. \text{dim-vec } v = \text{length } M$
 $\langle \text{proof} \rangle$

lemma *inv1:mat-M * mat-M-inv = 1_m n*
 $\langle \text{proof} \rangle$

lemma *inv2:mat-M-inv * mat-M = 1_m n*
 $\langle \text{proof} \rangle$

sublocale *rats*: *vec-module TYPE(rat) n* $\langle \text{proof} \rangle$

lemma *M-dim*: *dim-row mat-M = n dim-col mat-M = n*
 $\langle \text{proof} \rangle$

lemma *M-inv-dim*: *dim-row mat-M-inv = n dim-col mat-M-inv = n*
 $\langle \text{proof} \rangle$

lemma *babai-to-help*:
shows $s \ n = \text{babai-of-LLL } (\text{uminus } t) \ (\text{RAT } M)$
 $\langle \text{proof} \rangle$

3 Coordinates

This section sets up the use of the lattice basis and its GS orthogonalization as coordinate systems and some properties of that coordinate system. The

important lemma here is coord-invariance, which shows that after step i of the algorithm, all coordinates (in both systems) after $n-i$ are invariant.

definition *lattice-coord* :: *rat vec* \Rightarrow *rat vec*
where *lattice-coord* $a = \text{mat-}M\text{-inv} \cdot_v a$

lemma *dim-preserve-lattice-coord*:
fixes $v::\text{rat vec}$
assumes $\text{dim-vec } v = n$
shows $\text{dim-vec } (\text{lattice-coord } v) = n$ $\langle \text{proof} \rangle$

lemma *vec-to-col*:
assumes $i < n$
shows $(\text{RAT } M)!i = \text{col mat-}M\ i$
 $\langle \text{proof} \rangle$

lemma *unit*:
assumes $i < n$
shows $\text{lattice-coord } ((\text{RAT } M)!i) = \text{unit-vec } n\ i$
 $\langle \text{proof} \rangle$

lemma *linear*:
fixes $i::\text{nat}$
fixes $v1::\text{rat vec}$
and $v2::\text{rat vec}$
and $q::\text{rat}$
assumes $\text{dim-vec } v1 = n$
assumes $\text{dim-}2: \text{dim-vec } v2 = n$
assumes $0 \leq i$
assumes $\text{dim-}i: i < n$
shows $(\text{lattice-coord } (v1 + (q \cdot_v v2)))\$i = (\text{lattice-coord } v1)\$i + q * ((\text{lattice-coord } v2)\$i)$
 $\langle \text{proof} \rangle$

lemma *sub-s*:
fixes $i::\text{nat}$
assumes $0 \leq i$
assumes $i < n$
shows $s (\text{Suc } i) = (s\ i) -$
 $((\text{rat-of-int } (\text{calculate-c } (s\ i)\ Mt\ (\text{Suc } i))) \cdot_v (\text{RAT } M)!((\text{dim-vec } (s\ i)) - (\text{Suc } i)))$
 $\langle \text{proof} \rangle$

lemma *M-locale-1*:
shows $\text{gram-schmidt-fs-Rn } n\ (\text{RAT } M)$
 $\langle \text{proof} \rangle$

lemma *M-locale-2*:
shows $\text{gram-schmidt-fs-lin-indpt } n\ (\text{RAT } M)$
 $\langle \text{proof} \rangle$

lemma *more-dim*: $\text{length } (\text{RAT } M) = n$
<proof>

lemma *Mt-gso-connect*:
fixes $j :: \text{nat}$
assumes $j < n$
shows $\text{Mt!}j = \text{gs.gso } j$
<proof>

lemma *access-index-M-dim*:
assumes $0 \leq i$
assumes $i < n$
shows $\text{dim-vec } (\text{map of-int-hom.vec-hom } M ! i) = n$
<proof>

lemma *s-dim*:
fixes $i :: \text{nat}$
assumes $i \leq n$
shows $\text{dim-vec } (s \ i) = n \wedge (s \ i) \in \text{carrier-vec } n$
<proof>

lemma *dim-vecs-in-Mt*:
fixes $i :: \text{nat}$
assumes $i < n$
shows $\text{dim-vec } (\text{Mt!}i) = n$
<proof>

lemma *upper-tri*:
fixes $i :: \text{nat}$
and $j :: \text{nat}$
assumes $j > i$
assumes $j < n$
shows $((\text{RAT } M)!i) \cdot (\text{Mt!}j) = 0$
<proof>

lemma *one-diag*:
fixes $i :: \text{nat}$
assumes $0 \leq i$
assumes $i < n$
shows $((\text{RAT } M)!i) \cdot (\text{Mt!}i) = \text{sq-norm } (\text{Mt!}i)$
<proof>

lemma *coord-invariance*:
fixes $j :: \text{nat}$
fixes $k :: \text{nat}$
fixes $i :: \text{nat}$
assumes $k \leq j$
assumes $j + i \leq n$
assumes $k > 0$

shows $(\text{lattice-coord } (s \ (j+i)))\$(n-k) = (\text{lattice-coord } (s \ j))\$(n-k)$
 $\wedge (s \ (j+i)) \cdot \text{Mt!}(n-k) = (s \ j) \cdot \text{Mt!}(n-k)$
 $\langle \text{proof} \rangle$

lemma *small-orth-coord*:

fixes $i::\text{nat}$
assumes $1 \leq i$
assumes $i \leq n$
shows $\text{abs } ((s \ i) \cdot \text{Mt!}(n-i)) \leq (\text{sq-norm } (\text{Mt!}(n-i))) * (1/2)$

$\langle \text{proof} \rangle$

lemma *lattice-carrier*: $L \subseteq \text{carrier-vec } n$

$\langle \text{proof} \rangle$

4 Lattice Lemmas

lemma *lattice-sum-close*:

fixes $u::\text{int vec}$ **and** $v::\text{int vec}$
assumes $u \in L \ v \in L$
shows $u + v \in L$

$\langle \text{proof} \rangle$

lemma *lattice-smult-close*:

fixes $u::\text{int vec}$ **and** $q::\text{int}$
assumes $u \in L$
shows $q \cdot_v u \in L$

$\langle \text{proof} \rangle$

lemma *smult-vec-zero*:

fixes $v :: 'a::\text{ring vec}$
shows $0 \cdot_v v = 0_v (\text{dim-vec } v)$

$\langle \text{proof} \rangle$

lemma *coset-s*:

fixes $i::\text{nat}$
assumes $i \leq n$
shows $s \ i \in \text{coset}$

$\langle \text{proof} \rangle$

lemma *subtract-coset-into-lattice*:

fixes $v::\text{rat vec}$
fixes $w::\text{rat vec}$
assumes $v \in \text{coset}$
assumes $w \in \text{coset}$
shows $v - w \in \text{of-int-hom.vec-hom } L$

$\langle \text{proof} \rangle$

lemma *t-in-coset*:

shows $\text{uminus } t \in \text{coset}$

<proof>

5 Lemmas on closest distance

lemma *closest-distance-sq-pos*: *closest-distance-sq* ≥ 0

<proof>

definition *witness*:: *rat vec* \Rightarrow *rat* \Rightarrow *bool*

where *witness* *v* ε -*closest* = (*sq-norm* *v* $\leq \varepsilon$ -*closest* $\wedge v \in$ *coset* \wedge *dim-vec* *v* = *n*)

definition *close-condition*::*rat* \Rightarrow *bool*

where *close-condition* ε -*closest* \equiv

(*if* *closest-distance-sq* = 0 *then* $0 \leq$ *real-of-rat* ε -*closest*

else *real-of-rat* ε -*closest* $>$ *closest-distance-sq*)

\wedge (*real-of-rat* ε -*closest* $\leq \varepsilon * \text{closest-distance-sq}$)

lemma *close-rat*:

obtains ε -*closest*::*rat*

where *close-condition* ε -*closest*

<proof>

definition ε -*closest*::*rat*

where ε -*closest* = (*if* $\exists r$. *close-condition* *r* *then* *SOME* *r*. *close-condition* *r* *else* 0)

lemma ε -*closest-lemma*: *close-condition* ε -*closest*

<proof>

lemma *rational-tri-ineq*:

fixes *v*::*rat vec*

fixes *w*::*rat vec*

assumes *dim-vec* *v* = *dim-vec* *w*

shows (*sq-norm* (*v*+*w*)) $\leq 4 * (\text{Max} \{(\text{sq-norm } v), (\text{sq-norm } w)\})$

<proof>

lemma *witness-exists*:

shows $\exists v$. *witness* *v* ε -*closest*

<proof>

6 More linear algebra lemmas

lemma *carrier-Ms*:

shows *mat-M* \in *carrier-mat* *n n* *mat-M-inv* \in *carrier-mat* *n n*

<proof>

lemma *carrier-L*:

fixes *v*::*rat vec*

assumes $\text{dim-vec } v = n$
shows $\text{lattice-coord } v \in \text{carrier-vec } n$
 $\langle \text{proof} \rangle$

lemma *sumlist-index-commute*:

fixes $Lst::\text{rat vec list}$
fixes $i::\text{nat}$
assumes $\text{set } Lst \subseteq \text{carrier-vec } n$
assumes $i < n$
shows $(\text{gs.sumlist } Lst)\$i = \text{sum-list } (\text{map } (\lambda j. (Lst!j)\$i) [0..<(\text{length } Lst)])$
 $\langle \text{proof} \rangle$

lemma *mat-mul-to-sum-list*:

fixes $A::\text{rat mat}$
fixes $v::\text{rat vec}$
assumes $\text{dim-vec } v = \text{dim-col } A$
assumes $\text{dim-row } A = n$
shows $A*_v = \text{gs.sumlist } (\text{map } (\lambda j. v\$j \cdot_v (\text{col } A \ j)) [0 ..< \text{dim-col } A])$
 $\langle \text{proof} \rangle$

lemma *recover-from-lattice-coord*:

fixes $v::\text{rat vec}$
assumes $\text{dim-vec } v = n$
shows $v = \text{gs.sumlist } (\text{map } (\lambda i. (\text{lattice-coord } v)\$i \cdot_v (\text{RAT } M)\$i) [0 ..< n])$
 $\langle \text{proof} \rangle$

lemma *sumlist-linear-coord*:

fixes $Lst::\text{int vec list}$
assumes $\bigwedge i. i < \text{length } Lst \implies \text{dim-vec } (Lst!i) = n$
shows $\text{lattice-coord } (\text{map-vec rat-of-int } (\text{sumlist } Lst)) = \text{gs.sumlist } (\text{map lattice-coord } (\text{RAT } Lst))$
 $\langle \text{proof} \rangle$

lemma *integral-sum*:

fixes $l::\text{nat}$
assumes $\bigwedge j1. j1 < l \implies \text{map } f [0..<l] ! j1 \in \mathbf{Z}$
shows $\text{sum-list } (\text{map } f [0..<l]) \in \mathbf{Z}$
 $\langle \text{proof} \rangle$

lemma *int-coord*:

fixes $i::\text{nat}$
assumes $0 \leq i$
assumes $i < n$
fixes $v::\text{int vec}$

assumes $v \in L$
assumes $\dim\text{-vec } v = n$
shows $(\text{lattice-coord } (\text{map-vec rat-of-int } v)) \$i \in \mathbb{Z}$
 $\langle \text{proof} \rangle$

lemma *int-coord-for-rat*:
fixes $i :: \text{nat}$
assumes $0 \leq i$
assumes $i < n$
fixes $v :: \text{rat vec}$
assumes $v \in \text{of-int-hom.vec-hom } L$
assumes $\dim\text{-vec } v = n$
shows $(\text{lattice-coord } v) \$i \in \mathbb{Z}$
 $\langle \text{proof} \rangle$

7 Coord-Invariance

This section shows that the algorithm output matches true closest (or near-closest) vector in some trailing coordinates.

definition *I* where

$I = (\text{if } (\{i \in \{0..<n\}. ((\text{sq-norm } (Mt!i)::\text{rat})) \leq 4 * \epsilon\text{-closest}\} :: \text{nat set}) \neq \{\}} \\ \text{then } \text{Max } (\{i \in \{0..<n\}. ((\text{sq-norm } (Mt!i)::\text{rat})) \leq 4 * \epsilon\text{-closest}\} :: \text{nat set}) \text{ else } -1)$

lemma *I-geq*:
shows $I \geq -1$
 $\langle \text{proof} \rangle$

lemma *I-leq*:
shows $I < n$
 $\langle \text{proof} \rangle$

lemma *index-geq-I-big*:
fixes $i :: \text{nat}$
assumes $i > I$
assumes $i < n$
shows $((\text{sq-norm } (Mt!i)::\text{rat})) > 4 * \epsilon\text{-closest}$
 $\langle \text{proof} \rangle$

lemma *scalar-prod-gs-from-lattice-coord*:
fixes $i :: \text{nat}$
fixes $v :: \text{rat vec}$
assumes $\dim\text{-vec } v = n$
assumes $i < n$
shows $v \cdot Mt!i = \text{sum-list } (\text{map } (\lambda k. (\text{lattice-coord } v) \$k * (((\text{RAT } M)!k) \cdot Mt!i)) \\ [i..<n])$
 $\langle \text{proof} \rangle$

lemma *correct-coord-help*:
fixes $i::nat$
assumes $i < (int\ n) - I$
assumes *witness* v (ε -closest)
assumes $0 < i$
shows $(lattice\text{-}coord\ (s\ i))\$(n-i) = (lattice\text{-}coord\ v)\$(n-i)$
 $\wedge ((s\ i) \cdot Mt!(n-i) = v \cdot Mt!(n-i))$
 $\langle proof \rangle$

lemma *correct-coord*:
fixes $v::rat\ vec$
fixes $k::nat$
assumes *witness* v ε -closest
assumes $I < k$
assumes $k < n$
shows $(s\ n) \cdot Mt!(k) = v \cdot Mt!(k)$
 $\langle proof \rangle$

8 Bound on distance to target, with ε factor.

This section culminates in a bound (which includes the ε factor) on the output's distance to the target vector.

lemma *sq-norm-from-Mt*:
fixes $v::rat\ vec$
assumes $v\text{-carr}:v \in carrier\text{-}vec\ n$
shows $sq\text{-}norm\ v = sum\text{-}list\ (map\ (\lambda i. (v \cdot Mt!i)^2 / (sq\text{-}norm\ (Mt!i)))\ [0..<n])$
 $\langle proof \rangle$

lemma *basis-decay*:
fixes $i::nat$
fixes $j::nat$
assumes $i < n$
assumes $i + j < n$
shows $sq\text{-}norm\ (Mt!i) \leq \alpha^{\wedge j} * sq\text{-}norm\ (Mt!(i+j))$
 $\langle proof \rangle$

lemma *basis-decay-cor*:
fixes $i::nat$
fixes $j::nat$
assumes $i < n$
assumes $j < n$
assumes $i \leq j$
shows $sq\text{-}norm\ (Mt!i) \leq \alpha^{\wedge n} * sq\text{-}norm\ (Mt!j)$
 $\langle proof \rangle$

theorem *babai-correct*:
shows $real\text{-}of\text{-}rat\ ((sq\text{-}norm\ (s\ n))::rat) \leq (real\text{-}of\text{-}rat\ ((rat\text{-}of\text{-}int\ n) * \alpha^{\wedge n}) * \varepsilon * closest\text{-}distance\text{-}sq) \wedge s\ n \in coset$

<proof>

end

In this section we remove the arbitrary constant ϵ and clean up the assumptions and results.

locale *babai-with-assms* = *babai* +
 fixes *mat-M* :: *rat mat*
 assumes *basis: lin-indep M*
 defines *mat-M* \equiv *mat-of-cols n (RAT M)*
 assumes *reduced: weakly-reduced M n*
 assumes *non-trivial: 0 < n*
 assumes *alpha: $\alpha \geq 4/3$*
begin

sublocale *vec-space TYPE(rat) n <proof>*

definition *mat-M-inv* \equiv
 (*if (invertible-mat mat-M) then SOME B. (inverts-mat B mat-M) \wedge (inverts-mat mat-M B) else (0_m n n)*)

lemma *carrier-M*:
 shows *mat-M* \in *carrier-mat n n*
 <proof>

lemma *mat-M-inv-is-inv*:
 shows *invertible-mat mat-M*
 <proof>

abbreviation *babai-out* **where** *babai-out* \equiv *babai-of-LLL (-t) (RAT M)*

lemma *babai-with-assms- ϵ -connect*:
 shows *babai-with-assms- ϵ M t α 2*
 <proof>

This shows that the output, which is of the form $v-t$, with v in L , is short

lemma *babai-correct*:
 shows *real-of-rat (sq-norm (babai-out)) \leq (real-of-rat ((rat-of-int n)* α \hat{n}) * closest-distance-sq) \wedge (babai-out) \in coset*
 <proof>

This shifts the output, which is of the form $v-t$, with $v \in L$, back to v .

abbreviation *vL* **where** *vL* \equiv *map-vec int-of-rat (babai-out + t)*

lemma *shifted-babai-correct*:
 shows *vL* \in *L*
 \wedge *real-of-rat (sq-norm (map-vec rat-of-int (vL) - t)) \leq real-of-rat ((rat-of-int n)* α \hat{n})*closest-distance-sq*
 <proof>

end

Here we prove correctness of the full algorithm, which starts with a non-reduced basis and outputs a vector in L close to t .

```

locale full-babai-with-assms =
  fixes target::rat vec
  fixes n::nat
  fixes fs-init::int vec list
  fixes α::rat
  assumes non-triv:0<n
  assumes t-dim:dim-vec target = n
  assumes LLL-assms:LLL.LLL-with-assms n n fs-init α
begin

sublocale vec-space TYPE(rat) n ⟨proof⟩

abbreviation B::real where B ≡ (real-of-rat (α^n) * (n)) * babai.closest-distance-sq
fs-init target
abbreviation out where out ≡ (full-babai fs-init target α)

lemma LLL-output-babai-assms:
  shows babai-with-assms (map of-int-hom.vec-hom (LLL-Impl.reduce-basis α fs-init))
target α
⟨proof⟩

lemma full-babai-correct:
  shows real-of-rat (sq-norm ((map-vec rat-of-int out) – target))
    ≤ real-of-rat (α) ^ n * real n * babai.closest-distance-sq fs-init target ∧
    out ∈ vec-module.lattice-of n fs-init
⟨proof⟩
end

```

Here we prove correctness of the full algorithm, starting with a target vector of $TYPE(int)$.

```

lemma full-babai-correct-int-target:
  assumes full-babai-with-assms (map-vec rat-of-int target) n fs-init α
  shows real-of-int (sq-norm ( (full-babai fs-init (map-vec rat-of-int target) α) –
target))
    ≤ (real-of-rat(α) ^ n * (n)) * babai.closest-distance-sq fs-init (map-vec rat-of-int
target) ∧
    (full-babai fs-init (map-vec rat-of-int target) α) ∈ vec-module.lattice-of n
fs-init
⟨proof⟩

```

The literature typically states the main result using 2^n as the approximation constant, rather than $\alpha^n * n$. Here we show that $\alpha^n * n$ is stronger for $\alpha = 4/3$.

```
lemma clean-bound:  
  fixes n::nat  
  shows  $(4/3)^{\wedge n} * n \leq 2^{\wedge n}$   
  <proof>  
  
end
```

References

- [1] R. Bottesch, J. Divasón, and R. Thiemann. Two algorithms based on modular arithmetic: lattice basis reduction and Hermite normal form computation. *Archive of Formal Proofs*, March 2021. https://isa-afp.org/entries/Modular_arithmetic_LLL_and_HNF_algorithms.html, Formal proof development.
- [2] J. Divasón, S. J. C. Joosten, R. Thiemann, and A. Yamada. A Verified Factorization Algorithm for Integer Polynomials with Polynomial Complexity. *Archive of Formal Proofs*, February 2018. https://isa-afp.org/entries/LLL_Factorization.html, Formal proof development.
- [3] K. Kreuzer. Hardness of Lattice Problems. *Archive of Formal Proofs*, February 2023. https://isa-afp.org/entries/CVP_Hardness.html, Formal proof development.
- [4] N. Stephens Davidowitz. Lecture 5: CVP and Babais Algorithm, August 2016.