Some applications of the LLL algorithm

Kevin Hare *

Abstract: The LLL algorithm of Lenstra, Lenstra and Lovasz (see [5]), is normally implemented for integer vectors in $\mathbb{R}^n$. It can also be implemented in a more general setting, of the vector space of real valued functions, with an inner product defined as $\int f(t)g(t)\,d\mu(t)$ for some measure μ. In this paper, we will discuss how we can use a more general implementation of LLL to explore some properties of real valued functions.

Keywords: LLL, Lattice Basis Reduction, Integer Chebyshev Problem, Prouhet-Tarry-Escott problem.

Introduction

This paper talks about the Lenstra, Lenstra, Lovasz, latice basis reduction algorithm, commonly refered to as LLL has been found to have numerous applications, since it was introduced in 1982 [5]. It was the basis for the first polynomial time algorithm for factoring a polynomial in $\mathbb{Q}[x]$. Other applications include finding integer relations between real numbers, computing the integer kernel and image of a matrix, and finding minimal polynomials of real numbers [2]. These applications require an implementation of LLL to work on integer vectors over $\mathbb{R}^n$. The applications that we will look at in this paper include:

1. Finding polynomials with integer coefficients with minimal 2 norm on $[0,1]$.
2. Finding polynomials with coefficients $-1, 0, 1$ which are divisible by $(1-x)^N$ for some integer N.
3. Finding integer relationships between functions.

These applications differ from those above, in that they used an implementation of LLL where the innerproduct is defined for real valued functions, as opposed to vectors in $\mathbb{R}^n$.

Background

Definition 1 (Lattice) *Let V be a vector space. A set $L \subset V$ is a lattice if there exist finitely many vectors $b_1, ..., b_n$ such that*

$$L = \{\sum_{i=1}^{n} r_i b_i : r_i \in \mathbb{Z}\}$$

If the set $\{b_1, ..., b_n\}$ is minimal, then it is called a basis for L of rank n.

*Centre for Experimental and Constructive Mathematics, Simon Fraser University, Burnaby, British Columbia, Canada, kghare@cecm.sfu.ca

For example, $\{(x_1, x_2, x_3) : x_1 \in \mathbb{Z}, x_2 \in 2\mathbb{Z}, 3x_3 \in \mathbb{Z}\}$, is a lattice. One possible basis for this lattice would be $(1,0,0)$, $(0,2,0)$, $(0,0,\frac{1}{3})$.

Theorem (Gram Schmidt Orthogonalization) If $b_1, ..., b_n$ is a basis for a vector space V, define $b_1^*, ..., b_n^*$ by:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$$

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$$

Then the new basis $b_1^*, ..., b_n^*$ is an orthogonal basis.

The proof of this can be found in any introductory linear algebra text. See [4]

The Problem

A given lattice can have many different bases. Some bases for a lattice L are nicer than others. For example $(2,0,1)$, $(0,5,0)$, $(2,0,-6)$ is a "nicer" basis than $(2,150,-48)$, $(0,5,98)$, $(0,0,7)$ which both lead to the same lattice structure. To this end we define what nice is:

Definition 2 (Reduced) *We say that a basis $b_1, ..., b_n$ for a lattice L is reduced if*

$$\sum_{i=1}^{n} ||b_i||$$

is minimal.

The problem of finding the reduced basis given an arbitrary basis is NP-complete, (hence the only known algorithms are exponential in time).

A major step in solving this problem was made in 1982 by Lenstra, Lenstra and Lovasz [5]. They weakened the requirements of what "reduced" meant sufficiently that the problem could be solved in polynomial time. To this end we redefine what nice is:

Definition 3 (LLL Reduced) *We say $b_1, ..., b_n$ is an LLL-reduced basis, if, with $\mu_{i,j}$, b_i^* are defined as in the Gram Schmidt Orthogonalization Theorem we have:*

$$|\mu_{i,j}| \leq \frac{1}{2} \quad 1 \leq j < i \leq n$$

and

$$||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4}||b_{i-1}^*||^2 \quad 1 < i \leq n$$

(or equivalently)

$$||b_i^*||^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2)||b_{i-1}^*||^2$$

Some Properties of LLL

The reason we would define LLL Reduced in this way is that despite being less restrictive than Definition 2 (Reduced), we still have some fairly restrictive properties. Namely, we know that:

1. $\det(L) \leq \prod_{i=1}^{n} ||b_i|| \leq 2^{\frac{n(n-1)}{4}} \det(L)$
2. $||b_j|| \leq 2^{\frac{i-1}{2}} ||b_i^*||, \ 1 \leq j \leq i \leq n$
3. $||b_1|| \leq 2^{\frac{n-1}{4}} \det(L)^{\frac{1}{n}}$
4. $\forall x \in L, x \neq 0, ||b_1|| \leq 2^{\frac{n-1}{2}} ||x||$
5. $||b_j|| \leq 2^{\frac{n-1}{2}} max(||x_1||, ..., ||x_t||)$ for $x_1, ..., x_t \in L$, $1 \leq j \leq t$, and x_j linearly independent.

For a proof of these properties see [2].

There does exist an algorithm which will find an LLL Reduced basis, given an arbitrary basis. See [2, 5, 6] for detailed implementations. A detailed analysis would show that its running time is $\mathcal{O}(n^6 \ln^3 B)$ where $||b_i||^2 \leq B \ \forall i$ [2]. Further, restricted to integer lattices, there exist variations of LLL which operate in $\mathcal{O}(n^{2.381} \ln B)$, [8].

There are some things to note about this algorithm. They are:

1. This method will work over any inner product space
   (a) $\mathbb{R}^n$ with the standard inner product.
   (b) Set of functions on [0,1] with $\langle f, g \rangle = \int f(t)\, g(t)\, d\mu(t)$.
   (Useful for finding integer combinations of functions with small 2 norm.)
   (c) Set of polynomials on [0,1] with $\langle p(x), q(x) \rangle = p(\alpha)q(\alpha)$ for some point $\alpha \in \mathbb{R}$. *
   (Useful for finding minimal polynomials)
2. The constant $\frac{3}{4}$ in Definition 3 (LLL Reduced) can be replaced by any c such that:

$$0 < c - \mu_{i,i-1}^2 < 1$$

$$\frac{1}{4} < c < 1$$

The condition for LLL-reduced is then modified to:

$$||b_i^*||^2 \geq (c - \mu_{i,i-1}^2)||b_{i-1}^*||^2$$

See [2] as to why this value can be altered. If c is close to 1, we have tighter restrictions on the properties 1 to 5 of the functions. We can replace the value 2 in the properties 1 to 5 by $\frac{1}{c-\frac{1}{4}}$. If c is close to $\frac{1}{4}$ then we have an algorithm which theoretically runs faster. (Although in practice the speedup is not that great.) Ideally, we would like to run it with $c = 1$ but in this case, the proof that the code will terminate does not work.

To this end the code presented in the appendix will allow for an arbitrary inner product, as well as any value of c.

Small 2 norm of polynomials

The goal of this problem is to find a polynomial, with integer coefficients, of degree 21, such that the integral of the square from 0 to 1 is small. This is applicable to the study of the Integer Chebyshev Problem, see [1].

To do this we consider a basis $\{1, x, ..., x^{21}\}$, with the inner product of polynomials v(x), w(x) defined by $\int_0^1 \mathrm{v}(x)\, \mathrm{w}(x)\, dx$. An LLL-Reduced basis for this has basis elements of very small norm. Also, these basis elements are "sorted" (roughly) so that the first basis vector has the smallest norm. So we need only look at the first element of the LLL-reduced basis as our candidate for the polynomial with very small norm.

```
> read LLLPoly;
```

```
> Basis := [seq(x^i, i=0..21)];
```

$$Basis := [1, x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{10}, x^{11}, x^{12}, x^{13}, x^{14}, x^{15}, x^{16}, x^{17}, x^{18}, x^{19}, x^{20}, x^{21}]$$

```
> Small := LLLPoly( Basis, x, c=0.99, iprod=(
> (v,w)->int( v*w, x= 0..1))) [1];
```

$$Small := -x^7 + 23\,x^8 - 240\,x^9 + 1508\,x^{10} - 6383\,x^{11} + 19279\,x^{12} - 42903\,x^{13} + 71545\,x^{14} - 89938\,x^{15} + 84898\,x^{16} - 59293\,x^{17} + 29735\,x^{18} - 10130\,x^{19} + 2100\,x^{20} - 200\,x^{21}$$

*Note: we must make sure that $p(\alpha) \neq 0$, or we only have a semi-inner product, under which the algorithm will not work. If this is the case, we simply add the semi-inner product to a standard inner product to give a valid inner product.

```
> int(Small^2,x=0..1);
```

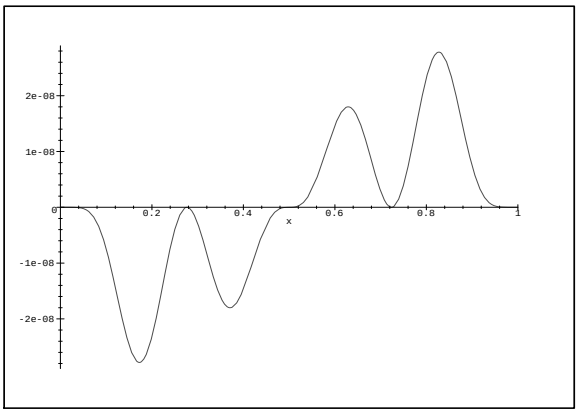
$$\frac{1}{5843416971961800}$$

```
> factor(Small);
```

$$-x^7\,(1-5\,x+5\,x^2)^2\,(2\,x-1)^3\,(x-1)^7$$

Here we notice that the function is small near 1, 0 and $\frac{1}{2}$ because of the factors of x, $x-1$ and $2x-1$.

```
> plot(Small,x=0..1);
```



Small coefficient norm of polynomials

The goal of this problem is to find a polynomial of small degree with coefficients which are in the set $\{0, \pm 1\}$ that is divisible by $(1-x)^N$ for some integer N. If in addition to this, the polynomial contains only $2\,N$ nonzero coefficients, then it is also a solution of the Prouhet-Tarry-Escott problem. (The one type of problem is not provably a subset of the other. See [1].)

To do this we define the inner product of the two polynomials to be the sum of the product of the coefficients of the same degree. Thus a small basis vector will have few coefficients, each of small modulus. To ensure that the polynomial has the desired divisibility requirement, we simply start with a basis of M vectors where $(1-x)^N$ divides each basis element. Then we run the worksheet with larger and larger sized starting bases (altering M), until the desired polynomial is found.

```
> read LLLPoly;

> innerprod := proc(v,w)
>     RETURN(add((coeff(v,x,i)*coeff(w,x,i))^1,
>         i=0..max(degree(v,x),degree(w,x))));
> end:

> N := 3: M := 3: Basis :=
> [seq(expand(x^(i)*(1-x)^N),i=0..M)]:

> LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$x^4+x^5-x^6+1-x-x^2$$

```
> factor(");
```

$$-(x+1)\,(x^2+x+1)\,(x-1)^3$$

```
> N := 4: M := 7: Basis :=
> [seq(expand(x^(i)*(1-x)^N),i=0..M)]:

> LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$-x^9-x^{10}+x^{11}-x-x^2+x^4+x^7+1$$

```
> factor(");
```

$$(x+1)\,(x^2+x+1)\,(x^4+x^3+x^2+x+1)\,(x-1)^4$$

```
> N := 5: M := 10: Basis :=
> [seq(expand(x^(i)*(1-x)^N),i=0..M)]:

> LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$1-x+x^5-x^2+x^6+x^7-x^8-x^9-x^{10}+x^{13}+x^{14}-x^{15}$$

```
> factor(");
```

$$-(x^2+x+1)\,(x^4+x^3+x^2+x+1)\,(x^2+1)\,(x+1)^2\,(x-1)^5$$

```
> N := 6: M := 16: Basis :=
> [seq(expand(x^(i)*(1-x)^N),i=0..M)]:

> LLLPoly(Basis,x,iprod=innerprod,c=0.99)[1];
```

$$1-x+x^5-x^2+x^6-x^{10}-x^{21}+x^{22}-x^{20}+x^{17}+x^{16}-x^{12}$$

```
> factor(");
```

$$(x^2+x+1)\,(x^4+x^3+x^2+x+1)\,(x^6+x^5+x^4+x^3+x^2+x+1)\,(x^2+1)\,(x+1)^2\,(x-1)^6$$

It can be shown that these examples above are in fact the minimal examples. An example for $N=7$, of degree 102, but the methods used in this example are unable to find it. It would be interesting to see if different starting bases would be able to lead to this example.

Relationships between functions

The goal of this problem is to find an integer relationship between some known functions. As the functions are analytic, it suffices to look at minimizing the 2 norm between 1 and 2, for this example. First we put the known functions into the basis, as well as a very small constant. We will work out what the constant is afterwards.

```
> read LLLPoly;
```

```
> innerprod := proc(v,w)
>     RETURN(add((coeff(v,x,i)*coeff(w,x,i))^1,
>         i=0..max(degree(v,x),degree(w,x))));
> end:
```

```
> Basis := [lnGAMMA(3*z), 10^(-9),ln(3)*z,
> lnGAMMA(z), lnGAMMA(z+1/3), lnGAMMA(z+2/3)];
```

$$Basis := [\mathrm{lnGAMMA}(3\,z),\ \frac{1}{1000000000},\ \ln(3)\,z,\ \mathrm{lnGAMMA}(z),\ \mathrm{lnGAMMA}(z+\frac{1}{3}),\ \mathrm{lnGAMMA}(z+\frac{2}{3})]$$

```
> Formula := LLLPoly(Basis, x, iprod=((v,w)->
> evalf( int( v*w, z=1..2))), c=0.99)[2];
```

$$Formula := \mathrm{lnGAMMA}(z+\frac{2}{3}) + \mathrm{lnGAMMA}(z) - \frac{2387183211}{1000000000} - \mathrm{lnGAMMA}(3\,z) + 3\ln(3)\,z + \mathrm{lnGAMMA}(z+\frac{1}{3})$$

So now we need only figure out what this constant is. We can guess that it is a combination of ln's.

```
> Const[1] := (2387183211/1000000000);
```

$$Const_1 := \frac{2387183211}{1000000000}$$

```
> Const[2] := ln(2);
```

$$Const_2 := \ln(2)$$

```
> Const[3] := ln(3);
```

$$Const_3 := \ln(3)$$

```
> Const[4] := ln(5);
```

$$Const_4 := \ln(5)$$

```
> Const[5] := ln(Pi);
```

$$Const_5 := \ln(\pi)$$

```
> M := 10^5;
```

$$M := 100000$$

```
> Basis :=
> [seq(x^i-evalf(M*Const[i]),i=1..5)];
```

$$Basis := [x - 238718.3211,\ x^2 - 69314.71806,\ x^3 - 109861.2289,\ x^4 - 160943.7912,\ x^5 - 114472.9886]$$

```
> LLLPoly(Basis,x,iprod=innerprod);
```

$$[x^3 + .00002 + 2\,x^2 - 2\,x + 2\,x^5,\ 26\,x^2 + .56940 + 5\,x - 9\,x^4 - 12\,x^3 - 2\,x^5,\ x^5 + 19.26418 - 4\,x^3 - x^2 + 3\,x - 2\,x^4,\ -x^4 + 8.44414 + 19\,x^3 - 26\,x^2 - x + x^5,\ -2\,x + 13.05674 + 22\,x^2 + 3\,x^3 - 5\,x^4 - 5\,x^5]$$

So from this we can see that 2 times the constant is $2\ln(2) + \ln(3) + 2\ln(\pi)$, so we have that.

```
> C := Const[3]/2+Const[2]+Const[5];
```

$$C := \frac{1}{2}\ln(3) + \ln(2) + \ln(\pi)$$

So we see that we have an approximation for 0 given these known functions. (In fact it is identically 0). We can consider it an approximation, because the 2 norm of the integral between 1 and 2 is approximately 0 (on the order of 10^{-29}.)

```
> Formula + Const[1] - C;
```

$$\mathrm{lnGAMMA}(z+\frac{2}{3}) + \mathrm{lnGAMMA}(z) - \mathrm{lnGAMMA}(3\,z) + 3\ln(3)\,z + \mathrm{lnGAMMA}(z+\frac{1}{3}) - \frac{1}{2}\ln(3) - \ln(2) - \ln(\pi)$$

We could also get this from taking the *ln* of the triplication formula for GAMMA, as given in [7].

References

[1] 9th international conference on approximation theory. *Some old problems on polynomials with integer coefficients, P. Borwein*, 1998.

[2] Henri Cohen. *A course in Computational Algebraic Number Theory.* Number 138 in Graduate Texts in Mathematics. Springer Verlag, New York, 1991.

[3] Kevin Hare. Home page. http://www.cecm.sfu.ca/~kghare.

[4] Kenneth Hoffman and Ray Kunze. *Linear Algebra, second edition.* Prentice-Hall, Inc, Toronto, 1971.

[5] A.K. Lenstra, H.W. Lenstra Jr, and L. Lovasz. Factoring polynomials with rational coefficients. *Math. Ann.*, 261, 1982.

[6] Maurice Mignotte. *Mathematics for Computer Algebra*. Springer Verlag, New York, 1991.

[7] Abramowitz Milton and Irene A. Stegun. *Handbook of Mathematical Functions, 9th edition*. Dover Publications, Inc, New York, 1992.

[8] Arne Storjohann. Algorithms for integer lattice basis reduction. Technical Report 249, Departement Informatik, ETH Zurich, July 1996.

Biography

Kevin Hare is an M.Sc. student working in the Centre for Experimental and Constructive Mathematics at Simon Fraser University. Before coming to Simon Fraser University, he earned his B.Math degree from the University of Waterloo in Pure Mathematics and Computer Science. He also worked with the Symbolic Computation Group at the University of Waterloo while working on his B.Math.

Appendix: Code

This follows the algorithm given in [6]. This code is also available at [3].

```
# This follows the algorithm given in Maurice Mignotte's
# "Mathematics for Computer Algebra"

LLLPoly := proc()
    local dim, basis, new_basis, local_iprod, local_c, i,
            j, u, B, k, h, l, param;

    basis := args[1];
    param := args[2];

    dim := nops(basis);

    local_iprod := subs(x=param,proc(v,w)
            RETURN(int(v*w,x=0..1)):
        end):

    local_c := 0.75;

    for i from 3 to nargs do
        if op(1,args[i]) = iprod then
            local_iprod := op(2,args[i]);
        elif op(1,args[i]) = c then
            local_c := op(2,args[i]);
        fi;
    od;

    for i from 1 to dim do
        new_basis[i] := basis[i];
        for j from 1 to i - 1 do
            u[i][j] := local_iprod(basis[i], new_basis[j])/B[j];
            new_basis[i] := new_basis[i] -u[i][j]*new_basis[j];
        od;
        B[i] := local_iprod(new_basis[i],new_basis[i]);
    od;

    for j from 1 to dim do
        new_basis[j] := basis[j];
    od;

    k := 2;

    while k <= dim do
        `LLLPoly/trans`(u, k-1, k, dim, [new_basis], local_iprod);

        if B[k] < (local_c-u[k][k-1]^2)*B[k-1] then
            `LLLPoly/modif`(k, u, B, dim, [new_basis], local_iprod);
            if k > 2 then k := k - 1; fi;
            next;
        fi;
```

```
        for h from k-2 by -1 to 1 do
            `LLLPoly/trans`(u, h, k, dim, [new_basis], local_iprod);
        od;

        k := k + 1;
    od;

    RETURN([seq(new_basis[i],i=1..dim)]);
end:

`LLLPoly/trans` := proc(u, h, k, dim, basis, iprod)
    local r, i, j, b, l;

    b := basis[1];
    if abs(u[k][h]) > 1/2 then
        r := floor(u[k][h] + 1/2);
        b[k] := b[k] - r*b[h];

        for j from 1 to h-1 do
            u[k][j] := u[k][j] - r*u[h][j];
        od;

        u[k][h] := u[k][h] - r;
    fi;
end:

`LLLPoly/modif` := proc(k, u, B, dim, b, iprod)
    local j, u_temp, B_temp, u_prime, b_temp, u_perm, i, basis;

    basis := b[1];
    u_temp := u[k][k-1];
    B_temp := B[k] + u_temp^2*B[k-1];
    u[k][k-1] := u_temp*B[k-1]/B_temp;
    B[k] := B[k-1]*B[k]/B_temp;
    B[k-1] := B_temp;
    b_temp := copy(basis[k]);
    basis[k] := copy(basis[k-1]);
    basis[k-1] := copy(b_temp);

    for j from 1 to k-2 do
        u_perm := u[k-1][j];
        u[k-1][j] := u[k][j];
        u[k][j] := u_perm;
    od;

    for i from k+1 to dim do
        u_prime := u[i][k-1] - u_temp*u[i][k];
        u[i][k-1] := u[i][k] + u[k][k-1]*u_prime;
        u[i][k] := u_prime;
    od;
end:

#savelib(``LLLPoly/trans``,``LLLPoly/modif``,``LLLPoly``,``LLLPoly.m``):
```