Efficient Parametrisation of Rational Normal Curves

Ant Lei

September 21, 2006

Abstract

This is the report for my research project[1] carried out during the summer of 2006. The objective of the project is to find an efficient parametrisation for rational normal curves over $\mathbb{Q}$. There is an algorithm implemented in the latest version (v2.13) of the computer algebra system Magma. We look into that as well as try to generalise techniques for conics.

1 Introduction

First of all, we give the definition of a rational normal curve.

Definition 1 *Let $C_0 \subset \mathbb{P}^d$ be the image of the map $v_d : \mathbb{P}^1 \to \mathbb{P}^d$ given by $(s : t) \mapsto (s^d : s^{d-1}t : \ldots : t^d)$. A rational normal curve of degree d is a curve projectively equivalent to C_0. C_0 is called the standard rational normal curve of degree d.*

Note that any basis of the space of homogeneous polynomials in s and t of degree d defines a rational normal curve. It is just a conic when $d = 2$, so we see that rational normal curves generalise the idea of conics to higher dimensions. If we denote the homogeneous coordinates of $\mathbb{P}^d$ by $(x_0 : x_1 : \ldots : x_d)$, it is not hard to see that C_0 is defined by the following equation.

$$\operatorname{rank} \begin{pmatrix} x_0 & x_1 & \ldots & x_{d-1} \\ x_1 & x_2 & \ldots & x_d \end{pmatrix} = 1 \tag{1}$$

In fact, the ideal of C_0 is generated by the following polynomials.

$$f_{i,j}(\mathbf{x}) = x_i x_j - x_{i-1}x_{j+1}, 1 \leq i \leq j \leq d-1 \tag{2}$$

We want to parametrise a rational normal curve over $\mathbb{Q}$. From now on, unless otherwise stated, everything is defined over $\mathbb{Q}$. If we have the projective equivalence with C_0, we are done. However, such an equivalence is not unique

[1] I was supervised by Tom Fisher and the project was funded by Trinity College, Cambridge.

and there is no obvious way to find one when our curve is defined by a set of quadrics. So our goal is, given a set of quadrics defining a rational normal curve, we would like to find an algorithm that parametrises the curve if it exists. Note that the existence of a parametrisation can be decided using Hasse principle. By clearing the denominators, we may assume the projective equivalence is over $\mathbb{Z}$ and all coefficients of the quadrics are in $\mathbb{Z}$.

2 Use of Syzygies

There is an algorithm by Frank-Olaf Schreyer implemented on Magma v2.13. The idea is to use a function called adjoint map that sends a rational normal curve of degree d to one of degree $d-2$. Recursively applying this function, the curve is sent to a conic or $\mathbb{P}^1$. If the latter, we are done. For the former, there is an polynomial-time algorithm in [2] that parametrises a conic[2]. Therefore, it will give us a parametrisation for the rational normal curve.

To apply the adjoint map, one needs to work out the minimal free resolution of the quotient module of the ideal of the curve. Magma has a built-in function that does the job. When dealing with ideals in Magma, the Groebner basis will be computed. This is in general a good idea. For example, we can find out if a polynomial is in the ideal easily using Groenber basis. However, when computing the Groebner basis, the coefficients of the polynomials increase rapidly. That makes the process of finding minimal free resolution slow. In fact, we don't actually need to find the Groebner basis in order to find the minimal free resolution. It is shown in [1] that for a rational normal curve of degree d, all the kernels are generated by linear forms and the length of the complex is d. Therefore, the minimal free resolution can be found using linear algbera. Not only are the coefficients much smaller, but the time saved is also significant. See section 5 for examples.

3 Minimisation Method

3.1 Algorithm for Conics

In [2], an algorithm that parametrises conics is introduced. The algorithm consists of two parts, namely minimisation and indefinite LLL. The minimisation method works as follows. Given a conic with integer coefficients, we work out the factorisation of the discriminant which tells us what the 'bad' primes are, ie those primes p which give singularities when the curve is reduced mod p. For each of these primes, we can find a change of basis over $\mathbb{Z}$ such that the new conic has coefficients divisible by p. On dividing by p, the modulus of the discriminant decreases by $1/p$. We can therefore assume the matrix representing the quadratic form defined by the conic has determinant ± 1. We can then use indefinite LLL to find a non-trivial solution, hence a parametrisation.

[2]The algorithm runs in polynomial time only when the factorisation of the discriminant is known.

The most important step of the algorithm is applying the minimisation method. Roughly speaking, it makes a singularity 'worse', so bad that we can factor out p altogether. We have to choose our change of basis matrix carefully so that after cancelling p, the singularity is actually 'better' than the one before. We would like to see if we can do a similar trick for a general rational normal curve.

3.2 'Bad' Primes

Given a rational normal curve of degree 3, the adjoint map will send it to $\mathbb{P}^1$, hence a parametrisation. So we consider the degree 4 case instead.

We would like to mimic the minimisation method. So we will have to find the 'bad' primes first. But this is not easy without an analogue of discriminant. The difficulty is to find a well-defined quantity which is independent of the choice of quadrics. Nonetheless, we can find the 'bad' primes by brute force as follows.

1. Take an affine piece, eg put $x_0 = 1$, work out the jacobian of the six equations, J say.
2. Let I be the ideal generated by the 3×3 minors of J over $\mathbb{Z}$.
3. Work out the intersection of I and $\mathbb{Z}$.

The idea is that at a singular point, J has rank less than 3, ie all 3×3 minors vanish. So if $I =< a >$ say, any prime factors of a are 'bad'. However, we will have to do this for all 5 affine pieces to ensure we have found all the 'bad' primes. To work out $I \cap \mathbb{Z}$, we use a function called EliminationIdeal in Magma which is slow when working over $\mathbb{Z}$.

If we can eliminate 'bad' primes for higher degree curves, we would like to make sure they will not reappear after applying the adjoint map. To do this, we have to consider how the adjoint map works. We denote the polynomial ring of $\mathbb{P}^4$ by R and the ideal of the curve by I. The minimal free resolution looks like this.

$$0 \to R^3 \xrightarrow{B} R^8 \xrightarrow{K} R^6 \to R \to R/I \to 0 \tag{3}$$

B is a 3×8 matrix over R, each entry is linear. We then construct a 5×8 matrix (denoted by A) over the polynomial ring $\mathbb{P}^2$, with variables a_0, a_1 and a_2 say, s.t. the (i, j)-entry of A is a linear form where the a_k coefficient is the x_{i-1} coefficient of $(k + 1, j)$-entry of B. The annihilator of A is generated by a quadric which will be the rational normal curve of degree 2 we want. In particular, if we start with the standard rational normal curves defined by (2), then the quadric would be $a_1^2 - a_0 a_2$. We denote A_0 for the 5×8 matrix in this case.

Given a rational normal curve C of degree 4, let M be a matrix over $\mathbb{Z}$ sending C onto C_0. If we write $\mathbf{y} = M\mathbf{x}$, then $x \in C$ iff $\mathbf{y}$ satisfies (2). So, C is defined by the following polynomials.

$$g_{i,j}(\mathbf{x}) = f_{i,j}(M\mathbf{x}) = f_{i,j}(\mathbf{y}), 1 \leq i \leq j \leq 3 \tag{4}$$

Note that from [1], the minimal free resolution is unique up to isomorphisms. So the minimal free resolution of C is isomorphic to that of C_0 after a change of coordinates by M. Let X be the isomorphism from R^8 (under the $\mathbf{x}$ coordinates) to R^8 (under the $\mathbf{y}$ coordinates). Similarly, Y denotes the map for R^3. By writing it out, one can see that $A = M^T A_0 (X^T)^{-1}$ with a change of coordinates in $\mathbb{P}^2$ by Y^T. A and $M^T A_0$ have the same image. M^T is invertible, so A and A_0 have the same annihilator. Therefore, the quadric we get is $a_1^2 - a_0 a_2$ with a change of coordinates by Y^T. If p is a prime not dividing $\det(M)$, then the two copies of R/I reduced mod p are isomorphic. Hence by the uniqueness of minimal free resolution, Y is invertible mod p. Hence, p is not a 'bad' prime for the final quadric. In other words, the 'bad' primes of the final quadric can only be those of C.

If we apply the adjoint map using the Groebner basis, a lot of 'bad' primes can be introduced because of the increase in coefficients. It's another reason why we should work out the minimal free resolution directly using linear algebra.

3.3 A Generalisation

Given a curve defined by (4), it is clear that the 'bad' primes can only be the prime factors of $\det(M)$. Indeed, by chain rule, we have $J_g(\mathbf{x}) = J_f(\mathbf{y})M$.

If p is a prime dividing $\det(M)$, there exits $\mathbf{x} \neq 0$ s.t. $M\mathbf{x} = \mathbf{y} \equiv 0 \pmod{p}$. Such a point, say $\mathbf{x_0}$, is a singular point of C reduced mod p with zero Jacobian. There exists a unimodular matrix over $\mathbb{Z}$, M_1 say, s.t. $M_1\mathbf{e_0} = \mathbf{x_0}$ where $e_0 = (1:0:0:0:0)$. Therefore, $MM_1e_0 \equiv 0 \pmod{p}$, hence the first column of MM_1 is divisible by p. Let M_2 be the diagonal matrix with entries $1, p, p, p, p$. Then MM_1M_2 is divisible by p. If we apply the change of coordinates defined by M_1M_2 to C, we can factor out a factor of p^2 from the new equations.

On continuing, we will eventually eliminate all the 'bad' primes. In fact, the same trick would work for a rational normal curve of any degrees. However, in reality, when given a rational normal curve, we don't know how to find the equations in the form of (4) as assumed above. If the six equations in (4) are $\mathbf{q} = (q_1, \ldots, q_6)$, for any 6×6 invertible matrix N, the equations $N\mathbf{q}$ defines C. Another problem is that after each step, we have to compute all the 'bad' primes all over again to see if the 'bad' prime has been eliminated. If the coefficients become large, the whole process would become very slow.

3.4 Pure Lattice

As we can see from above, a lot of 'unnecessary' singularity can be introduced by N. For example, if N is a scalar matrix pI, then p can become a 'bad' prime which is not introduced by M. To avoid this situation, we make use of the concept of pure lattice.

Definition 2 *Let A be an $m \times n$ matrix with integer entries. If B is its Smith normal form, then $B_{i,i}$ where $1 \leq i \leq \min(m, n)$ are called the elementary divisors of A.*

Definition 3 *Given a lattice L with integer entries. The pure lattice of L, P say, is such that P and L generate the same $\mathbb{Q}$-vector space and the elementary divisors of the basis matrix of P are trivial.*

There is a built-in function called 'PureLattice' in Magma. Given six equations defining C, it defines a lattice of degree 6 in $\mathbb{R}^{15}$. By finding its pure lattice, we can eliminate the trivial 'bad' primes introduced by N.

If we have equations $N\mathbf{q}$ after applying PureLattice, the Jacobian becomes NJ_g. If N has integer entries, then N is unimodular. Plus, the singularity is unchanged by N when reduced mod p for any primes p. The method mentioned above will still work. However, if N has non-integer entries, ie we can cancel some factors out from the quadrics, then the singularity is changed. Of course x_0 can still be found as there are only finitely many points in $\mathbb{P}^d(\mathbb{F}_p)$. But it is not obvious if this point is still a singularity or even on the reduced curve.

3.5 Further Generalisations

We want to know how well the above method works in general. But we will need a notion of 'good'. Assume we have 6 quadrics. We perform a change of coordinates using a 5×5 matrix M. After that, we performe a change of basis for the 6 quadrics by a 6×6 matrix N. We want to compare $\det(M)$ and $\det(N)$ to see the overall effect of the changes. If M is a scalar matrix pI, setting $N = p^{-2}I$ would give us the original equations. So this suggests we should consider the quantity $\det(M)^{12}\det(N)^5$. When we apply the method in 3.3, this is $(p^4)^{12}(p^{-12})^5 = p^{-12}$. Now, we consider what would happen when our equations are not of the form in (4).

When there is a singularity reduced mod p, the Jacobian has rank less than 3. As above, after a change of coordinates by a unimodular matrix, we may assume the singular point is e_0 so that the coefficients of x_0^2 are divisible by p. Note that the Jacobian mod p at e_0 only depends on the coefficients of x_0x_1, x_0x_2, x_0x_3 and x_0x_4. If the Jacobian has rank less than 3, we can make at least two more terms (x_0x_3 and x_0x_4 say) vanish mod p after a further change of coordinates. If we perform another change of coordinates with the diagonal matrix with entries $1, p, p, p, p$ as in 3.3, all cooefficients not involving x_0 are divisible by p^2. All terms involving x_0 have coefficients divisible by p, at most three of them (x_0^2, x_0x_1 and x_0x_2) are not divisible by p^2. Amongst the 6 new equations, we can cancel out at least a factor of p^9. If we have a factor of p^{10} or more, then the quantity we defined above would be at most $(p^4)^{12}(p^{-10})^5 = p^{-2}$. We are actually doing 'better'. In particular, we see that this works when the Jacobian has rank 1 or 0. It's also not clear how well it works for higher degrees.

4 Use of LLL Algorithm

We will see from the examples in section 5 that the coefficients increase when we apply the adjoint map. Therefore, we want to make sure the increase is

under control so that our algorithm terminates in reasonable times. The LLL algorithm returns 'short' basis vectors for a lattice. Although it doesn't always return the shortest basis vectors, it runs in polynomial time and the results are usually satisfactory. One natural thing to do would be to apply it to the lattice defined by the equations of the curve. This works fine, but it's actually similar in effect to the pure lattice method introduced in 3.4. We would like to see if more can be done.

4.1 Indefinite LLL

A quadric defines a lattice when treated as a Gram matrix. In [2], a generalised LLL algorithm is introduced and it is applied to the lattice defined by the Gram matrix obtained from the conic. It will work even for a indefinite quadratic form, but it will stop if an isotropic vector is found. To try to apply this to a general rational normal curve, we will have to choose a non-degenerate quadratic form in the ideal. We cannot find a canonical way to do so. When a random quadric is chosen, the algorithm doesn't guarantee that the coefficients in a generating set for the ideal will be small. This is therefore not satisfactory.

4.2 Real Parametrisation

Given a rational normal curve of degree 4, C say, we can obtain a parametrisation over $\mathbb{R}$ as follows.

1. Choose a hyperplane in $\mathbb{P}^4$, H say, defined by some linear form.
2. Eliminate one variable in the six equations of C using the equation of H.
3. Use EliminationIdeal function to eliminate 2 more variables in 2.
4. We get a homogeneous polynomial in 2 variables from above.
5. Find a root in $\mathbb{R}$ or choose another hyperplane.
6. Have a real point on C, P say.
7. Find a linear map sending P to e_0. With this new basis, equations of C have no x_0^2 terms.
8. Cancelling terms involving x_0, we get three equations, defining a rational normal curve of degree 3. The tangent at P is sent to a point on the new curve.
9. Repeat until we get down to $\mathbb{P}^1$.
10. Substitue back and get a parametrisation.

Note that in order to carry out the cancellation needed, we have to work with precise rings on a computer algebra system. Hence, we need to work with a number field. We can only approximate the numbers by real numbers until the

end. This could make the calculation slow since the coefficients can become large even though the real numbers itself are not.

If we have a parametrisation over $\mathbb{R}$, we have a linear map in $\mathbb{P}^4(\mathbb{R})$ sending C onto C_0. If we can find a basis s.t. the this linear map has small coefficients, one could hope that this will lead to smaller coefficients when parametrised over $\mathbb{Q}$ as well. To do this, one could apply LLL algorithm to the lattice with basis defined by this real linear map. However, there is no guarantee that it would make the coefficients of the original equations smaller. In fact, it could increase the coefficients which makes the subsequent calculations actually slower.

5 Numerical Examples

All examples are generated with Magma v2.31. It is run on an Intel Pentium 4 1.9 GHz processor. All times shown below are in seconds.

5.1 Minimal Free Resolution

We take a random matrix, denoted by Ran below, to perform a change of basis to generate a rational normal curve of degree 4. We take another random matrix to change the six equations for the curve and then apply pure lattice. We compare the built-in function MinimalFreeResolution with mykernel, a user-defined function that computes the minimal free resolution of a rational normal curve using linear algebra. We only give the times taken and the first entries of the boundary maps in the last step to save space.

```
> Ran;
[-18 -76 -99  -4  82]
[-48  75  -2 -23   4]
[-14  -9 -66  62  99]
[ 92  22 -92  88 -90]
[-59 -74 -52  -8  42]

> time R:=MinimalFreeResolution(Q);
Time: 0.180
> BoundaryMap(R,3)[1,1];
16517009868393976004083 4/249275258804049862737795*$.3 -
    61193716018582603485112/249275258804049862737795*$.4 -
    23186630103642516264714 8/249275258804049862737795*$.5

> time for i in [1..d-2] do
time|for> M:=mykernel(d,M);
time|for> end for;
Time: 0.040
> M[1,1];
$.2 + 42640390759*$.4 + 74704111444*$.5
```

The time difference is even more significant when we consider a degree 5 curve.

```
> Ran;
[-8 -5 -3 10 -7 -6]
[ 6  6 -7  5 -8  9]
[ 9  9  4  2  2  0]
[-2  4 -6 10  1  2]
[-9  1  9 -7 -8 -1]
[ 3 10 -2  1 -6 10]

> time R:=MinimalFreeResolution(M);
Time: 21.920
> BoundaryMap(R,4)[1,1];
185803903218211477007/891758206608616183 24*$.1 +
    232171908783573386039/891758206608616183 24*$.4 -
    192438708595163675845/891758206608616183 24*$.6

> time for i in [1..d-2] do
time|for> M:=mykernel(d,M);
time|for> end for;
Time: 0.510
> M[1,1];
$.3 + $.4 + 1265731085*$.6
```

5.2 Adjoint Map

We have seen that when we apply the adjoint map directly, the 'bad' primes of the final quadrics can only be those of the initial rational normal curve. In fact, most of them don't come up. However, if we use Groebner basis, some 'bad' primes can be introduced. Below are two examples. C4toQ is a user-defined function modifying the built-in adjoint map with mykernel, C2 is the conic obtained using the built-in function.

```
> Ran;
[-14 -36  65  44 -74]
[ 42 -70  30 -81 -30]
[ 14 -73  81  22 -16]
[ 71 -45  61  27 -18]
[ 42 -42 -47  -7 -95]
> Factorization(Determinant(Ran));
[ <3, 1>, <7, 1>, <11, 1>, <15152303, 1> ]

> Q:=C4toQ(H);
> Q;
32490325025313*$.1^2 + 80294854266884*$.1*$.2 + 84895544519200*$.1*$.3 +
    49609103761790*$.2^2 + 104903157628711*$.2*$.3 + 55456920433978*$.3^2
```

```
> C1:=Conic(P2,Q);
> Discriminant(C1);
-1

> C2;
Conic over Rational Field defined by
$.1^2 + 3156221224155952733143 25/2945786067357799987436538*$.1*$.2 -
    1008456133016370301712377519/3575595128558897624750469824 4*$.2^2 +
    4926919765996045250092549/6873500823834866637351922*$.1*$.3 +
    3042156797720302649036715631/5959325214264829374584116374*$.2*$.3 +
    1506066617670458573282641733/3972883476176552916389410916*$.3^2
> a:=Discriminant(C2);
> Factorization(Numerator(a));
[ <3, 2>, <503, 4>, <1951, 4>, <6147251178389, 4> ]
> Factorization(Denominator(a));
[ <2, 2>, <7, 3>, <17, 2>, <31, 3>, <2647, 3>, <5983210994304365639, 3> ]



> Ran;
[   1245456      91728    9418968 -915959680 1218560320]
[      4698       -468       8892      76320    -159480]
[   1148958          0   10301850 -491552800  647671480]
[   1366632          0   12337650 -585880200  771954420]
[      -522          0          0     -18360      23700]
> Factorization(Determinant(Ran));
[ <2, 16>, <3, 10>, <5, 5>, <7, 2>, <13, 2>, <19, 1> ]

> Q:=C4toQ(H);
> Q;
756027478657473639889715846595 44*$.1^2 +
    6458742425852638637881618523534 93*$.1*$.2 +
    2271123649065397332100998986055823*$.1*$.3 +
    1379425844335230037755529354870455*$.2^2 +
    9701104179853301952847416109851221*$.2*$.3 +
    1705626705031780492981982052279935 0*$.3^2
> Discriminant(C);
-30

> a:=Discriminant(C2);
> Factorization(Numerator(a));
[ <3, 2>, <5, 4>, <13, 6>, <29, 4>, <17393, 4>, <577831, 4>,
<3253975951960348437733356420097028 17, 4> ]
> Factorization(Denominator(a));
[ <2, 4>, <7, 3>, <4643, 3>, <128903, 3>, <272257, 3>,
<2483152394230306023254465751554690982119400006722819, 3> ]
```

5.3 Finding 'Bad' Primes

Below is an example where an affine piece can miss out a 'bad' prime when using the method mentioned in 3.2. The second variable of the user-defined function, BadPrimes, defines which affine piece to work with. Also note the long running time despite the small coefficients.

```
> Ran;
[1125    0    0  -40   80]
[ -60    5   38   16  -40]
[ -60    0    8   16  -40]
[-280    0    0   10  -20]
[-291   25  190   80 -194]

> time Factorization(BadPrimes(H,1));
[
    <2, 2>,
    <5, 3>
]
Time: 5.220
> time Factorization(BadPrimes(H,2));
[
    <2, 2>,
    <3, 1>,
    <5, 3>
]
Time: 4.440
```

5.4 Example for 3.3

Below is an example using the method introduced in 3.3. We apply PureLattice after each step to make sure coefficients are under control. The determinant of Ran tells us that we should do twice for $p = 3$. H is the set of equations we start with. We will see that we can eliminate all the bad primes using this method, but the coefficients are increased.

```
> Ran;
[  3  37  91 -37 -23]
[ 90 -37  39 -88  15]
[-75  23  43  53 -42]
[ 47 -52  57 -58 -56]
[ 99 -12 -42 -76  80]
> Factorization(Integers()!Determinant(Ran));
[ <3, 2>, <11, 1>, <541, 1>, <727, 1> ]
```

```
> H;
[
    -998*$.1^2 + 3010*$.1*$.2 + 762*$.1*$.3 - 478*$.1*$.4 -
        4291*$.1*$.5 - 877*$.2^2 + 8126*$.2*$.3 - 4168*$.2*$.4 +
        1470*$.2*$.5 - 3548*$.3^2 - 1424*$.3*$.4 + 3818*$.3*$.5 +
        1725*$.4^2 + 3218*$.4*$.5 + 180*$.5^2,
    -1654*$.1^2 - 1011*$.1*$.2 - 7023*$.1*$.3 + 6409*$.1*$.4 +
        3867*$.1*$.5 - 565*$.2^2 + 472*$.2*$.3 + 279*$.2*$.4 -
        5059*$.2*$.5 + 2082*$.3^2 + 6231*$.3*$.4 - 2044*$.3*$.5 -
        4560*$.4^2 - 2761*$.4*$.5 - 1536*$.5^2,
    1190*$.1^2 - 5243*$.1*$.2 + 3632*$.1*$.3 - 2349*$.1*$.4 -
        3536*$.1*$.5 - 1685*$.2^2 + 1026*$.2*$.3 + 4437*$.2*$.4 -
        8199*$.2*$.5 + 4500*$.3^2 - 6468*$.3*$.4 - 3401*$.3*$.5 +
        1710*$.4^2 + 5121*$.4*$.5 - 3578*$.5^2,
    -1525*$.1^2 + 3496*$.1*$.2 + 1600*$.1*$.3 + 2676*$.1*$.4 +
        1741*$.1*$.5 - 4202*$.2^2 - 1440*$.2*$.3 - 2403*$.2*$.4 -
        6414*$.2*$.5 + 5823*$.3^2 - 4933*$.3*$.4 - 6995*$.3*$.5 -
        642*$.4^2 + 1284*$.4*$.5 - 563*$.5^2,
    1745*$.1^2 - 1793*$.1*$.2 - 4790*$.1*$.3 - 3594*$.1*$.4 -
        3053*$.1*$.5 + 3176*$.2^2 + 5609*$.2*$.3 + 485*$.2*$.4 +
        7725*$.2*$.5 - 2003*$.3^2 + 3128*$.3*$.4 - 2356*$.3*$.5 +
        2307*$.4^2 + 3367*$.4*$.5 + 4758*$.5^2,
    -4239*$.1^2 + 1263*$.1*$.2 + 2758*$.1*$.3 + 7352*$.1*$.4 -
        3232*$.1*$.5 + 2515*$.2^2 - 6669*$.2*$.3 + 126*$.2*$.4 +
        6364*$.2*$.5 - 2044*$.3^2 + 863*$.3*$.4 + 3640*$.3*$.5 -
        3975*$.4^2 + 628*$.4*$.5 - 562*$.5^2
]
> Factorization(BadPrimes(H,1));
[
    <3, 6>,
    <11, 1>,
    <541, 3>,
    <727, 3>
]
//****We start with p=11 and get HR****
//****Then we do the same for p=541 and get HR2****
//****Then p=727 and get HR3****
//****Then p=3 and get HR4****
//****and again gives HR5****
> HR5;
[
    -2285783*$.1^2 + 4631296*$.1*$.2 + 7743911*$.1*$.3 -
        10898403*$.1*$.4 - 10399373*$.1*$.5 + 243078*$.2^2 -
        24656921*$.2*$.3 + 16351109*$.2*$.4 + 20534946*$.2*$.5 +
        20742140*$.3^2 + 2062761*$.3*$.4 - 13440973*$.3*$.5 -
        16371680*$.4^2 - 9517846*$.4*$.5 + 17257882*$.5^2,
```

```
    8318826*$.1^2 - 26121675*$.1*$.2 + 1341928*$.1*$.3 +
        11925052*$.1*$.4 - 15481775*$.1*$.5 + 21252513*$.2^2 -
        6300433*$.2*$.3 - 25407532*$.2*$.4 + 31815090*$.2*$.5 +
        5705641*$.3^2 + 25810474*$.3*$.4 - 25666059*$.3*$.5 -
        19114857*$.4^2 - 820401*$.4*$.5 + 15302283*$.5^2,
    -8160116*$.1^2 + 27746135*$.1*$.2 - 9812189*$.1*$.3 -
        1547715*$.1*$.4 - 8160878*$.1*$.5 - 24544155*$.2^2 +
        22775212*$.2*$.3 - 2520262*$.2*$.4 + 4736997*$.2*$.5 -
        12500405*$.3^2 + 14329620*$.3*$.4 + 28668685*$.3*$.5 -
        3603412*$.4^2 - 42685817*$.4*$.5 + 14947243*$.5^2,
    3569285*$.1^2 - 5582597*$.1*$.2 - 18434642*$.1*$.3 +
        16037874*$.1*$.4 - 7073645*$.1*$.5 + 3981785*$.2^2 +
        3534822*$.2*$.3 - 15890475*$.2*$.4 + 22455215*$.2*$.5 +
        40047367*$.3^2 - 26787724*$.3*$.4 - 37290493*$.3*$.5
        - 5484905*$.4^2 + 12114559*$.4*$.5 + 18970364*$.5^2,
    -3445070*$.1^2 + 13089812*$.1*$.2 - 7101185*$.1*$.3 -
        10893956*$.1*$.4 + 19528636*$.1*$.5 - 12298221*$.2^2 +
        13516702*$.2*$.3 + 9627685*$.2*$.4 - 29984517*$.2*$.5 -
        5215558*$.3^2 + 27154826*$.3*$.4 - 1272098*$.3*$.5 -
        31553865*$.4^2 + 23266507*$.4*$.5 - 10593933*$.5^2,
    -7472758*$.1^2 + 21463969*$.1*$.2 + 4200023*$.1*$.3 +
        361124*$.1*$.4 + 1877855*$.1*$.5 - 18306637*$.2^2 +
        13064712*$.2*$.3 + 1049535*$.2*$.4 + 5012461*$.2*$.5 -
        32029264*$.3^2 - 8451360*$.3*$.4 - 27222783*$.3*$.5 +
        37665728*$.4^2 + 25751268*$.4*$.5 - 1231265*$.5^2
]
> BadPrimes(HR5,1);
1
> BadPrimes(HR5,2);
1
> BadPrimes(HR5,3);
1
> BadPrimes(HR5,4);
1
> BadPrimes(HR5,5);
1
```

5.5 Pure Lattice

Below is an example that shows PureLattice can make 3.3 not work. F is the set of equations obtained after the change of coordinates and H is the set of equations after applying PureLattice. In fact, the reduced (mod 13) curve is singular with all singular points having rank 2 Jacobians. Geometrically, it is just three lines.

```
> Ran;
```

```
[ 1  0  0  0  0]
[ 0 13  0  0  0]
[ 0  0 13  0  0]
[ 0  0  0  1  0]
[ 0  0  0  0  1]

> F;
[
    -13*$.1*$.3 + 169*$.2^2,
    -$.1*$.4 + 169*$.2*$.3,
    -13*$.2*$.4 + 169*$.3^2,
    -$.1*$.5 + 13*$.2*$.4,
    -13*$.2*$.5 + 13*$.3*$.4,
    -13*$.3*$.5 + $.4^2
]

> H;
[
    $.2*$.5 - $.3*$.4,
    -$.1*$.3 + 13*$.2^2,
    -$.2*$.4 + 13*$.3^2,
    -13*$.3*$.5 + $.4^2,
    $.1*$.5 - 13*$.2*$.4,
    $.1*$.4 - 169*$.2*$.3
]
```

5.6 Indefinite LLL

As above, we randomly pick a rational normal curve. We take a non-degenerate quadratic from (the sum of the basis) and apply the indefinite LLL algorithm and get a change of basis matrix A. We see that it decreases the coefficients of the chosen form, but increases those of the original quadrics.

```
> H;
[
    -2712*$.1^2 - 4860*$.1*$.2 - 3638*$.1*$.3 - 6290*$.1*$.4 +
        3498*$.1*$.5 - 3758*$.2^2 + 5988*$.2*$.3 + 545*$.2*$.4 +
        9014*$.2*$.5 + 7640*$.3^2 + 4057*$.3*$.4 + 9149*$.3*$.5 -
        1690*$.4^2 + 8592*$.4*$.5 + 3579*$.5^2,
    4456*$.1^2 + 4672*$.1*$.2 + 4730*$.1*$.3 + 9890*$.1*$.4 -
        7350*$.1*$.5 + 4002*$.2^2 - 641*$.2*$.3 + 4750*$.2*$.4 +
        3467*$.2*$.5 - 3333*$.3^2 + 2548*$.3*$.4 - 145*$.3*$.5 +
        4765*$.4^2 - 9307*$.4*$.5 - 286*$.5^2,
    -3124*$.1^2 - 3181*$.1*$.2 - 4222*$.1*$.3 - 4449*$.1*$.4 -
        3808*$.1*$.5 + 4823*$.2^2 + 1863*$.2*$.3 - 7319*$.2*$.4 -
        758*$.2*$.5 + 2851*$.3^2 + 4986*$.3*$.4 + 3868*$.3*$.5 +
```

```
        6110*$.4^2 + 6791*$.4*$.5 - 3153*$.5^2,
    6264*$.1^2 + 1038*$.1*$.2 - 717*$.1*$.3 + 6748*$.1*$.4 -
        10689*$.1*$.5 - 5632*$.2^2 - 13594*$.2*$.3 + 3112*$.2*$.4 +
        840*$.2*$.5 - 7437*$.3^2 + 408*$.3*$.4 + 3018*$.3*$.5 +
        2780*$.4^2 - 2101*$.4*$.5 - 2235*$.5^2,
    -6102*$.1^2 - 11059*$.1*$.2 + 3736*$.1*$.3 + 1312*$.1*$.4 +
        3199*$.1*$.5 - 1329*$.2^2 + 4930*$.2*$.3 - 5171*$.2*$.4 -
        1174*$.2*$.5 - 4072*$.3^2 - 13881*$.3*$.4 - 3103*$.3*$.5 -
        1151*$.4^2 - 7189*$.4*$.5 + 3328*$.5^2,
    1200*$.1^2 - 724*$.1*$.2 - 8922*$.1*$.3 - 4572*$.1*$.4 -
        13538*$.1*$.5 - 1915*$.2^2 + 4233*$.2*$.3 + 2592*$.2*$.4 +
        3579*$.2*$.5 + 1034*$.3^2 - 9560*$.3*$.4 + 2163*$.3*$.5 -
        6985*$.4^2 - 15414*$.4*$.5 - 167*$.5^2
]
> Q:=&+H;
> Q;
-18*$.1^2 - 14114*$.1*$.2 - 9033*$.1*$.3 + 2639*$.1*$.4 -
    28688*$.1*$.5 - 3809*$.2^2 + 2779*$.2*$.3 - 1491*$.2*$.4 +
    14968*$.2*$.5 - 3317*$.3^2 - 11442*$.3*$.4 + 14950*$.3*$.5 +
    3829*$.4^2 - 18628*$.4*$.5 + 1066*$.5^2

> Q^A;
-18*$.1^2 + 17*$.1*$.2 + 13*$.1*$.3 - 5*$.1*$.4 + $.1*$.5 +
    25848*$.2^2 - 7483*$.2*$.3 - 5518*$.2*$.4 + 6475*$.2*$.5 -
    44062*$.3^2 + 41463*$.3*$.4 - 17434*$.3*$.5 - 38569*$.4^2 +
    18127*$.4*$.5 + 143395*$.5^2
> [H[i]^A:i in [1..6]];
[
    -2712*$.1^2 - 40922*$.1*$.2 - 169338*$.1*$.3 + 253068*$.1*$.4 +
        614388*$.1*$.5 - 152700*$.2^2 - 1253258*$.2*$.3 +
        1893211*$.2*$.4 + 4616839*$.2*$.5 - 2625446*$.3^2 +
        7805471*$.3*$.4 + 19097796*$.3*$.5 - 5839668*$.4^2 -
        28533270*$.4*$.5 - 34730382*$.5^2,
    4456*$.1^2 + 64282*$.1*$.2 + 281118*$.1*$.3 - 413668*$.1*$.4 -
        1009852*$.1*$.5 + 234784*$.2^2 + 2014045*$.2*$.3 -
        2984473*$.2*$.4 - 7278203*$.2*$.5 + 4422299*$.3^2 -
        12988153*$.3*$.4 - 31800456*$.3*$.5 + 9544728*$.4^2 +
        46754858*$.4*$.5 + 57154375*$.5^2,
    -3124*$.1^2 - 39097*$.1*$.2 - 199992*$.1*$.3 + 283770*$.1*$.4 +
        705226*$.1*$.5 - 111744*$.2^2 - 1211418*$.2*$.3 +
        1755115*$.2*$.4 + 4395818*$.2*$.5 - 3163815*$.3^2 +
        9047226*$.3*$.4 + 22543547*$.3*$.5 - 6426914*$.4^2 -
        32001512*$.4*$.5 - 39788614*$.5^2,
    6264*$.1^2 + 77891*$.1*$.2 + 384141*$.1*$.3 - 559770*$.1*$.4 -
        1403171*$.1*$.5 + 249908*$.2^2 + 2382666*$.2*$.3 -
        3482562*$.2*$.4 - 8716353*$.2*$.5 + 5880770*$.3^2 -
```

```
        17107206*$.3*$.4 - 42971021*$.3*$.5 + 12424554*$.4^2 +
        62529993*$.4*$.5 + 78497980*$.5^2,
    -6102*$.1^2 - 73191*$.1*$.2 - 344998*$.1*$.3 + 528967*$.1*$.4 +
        1345356*$.1*$.5 - 210920*$.2^2 - 2110495*$.2*$.3 +
        3211974*$.2*$.4 + 8122411*$.2*$.5 - 4902460*$.3^2 +
        15027548*$.3*$.4 + 38092536*$.3*$.5 - 11485087*$.4^2 -
        58349517*$.4*$.5 - 74166006*$.5^2,
    1200*$.1^2 + 11054*$.1*$.2 + 49082*$.1*$.3 - 92372*$.1*$.4 -
        251946*$.1*$.5 + 16520*$.2^2 + 170977*$.2*$.3 - 398783*$.2*$.4 -
        1134037*$.2*$.5 + 344590*$.3^2 - 1743423*$.3*$.4 -
        4979836*$.3*$.5 + 1743818*$.4^2 + 9617575*$.4*$.5 +
        13176042*$.5^2
]
```

5.7 Real Parametrisation

We obtain a real parametrisation by intersecting with the hyperplane $x_4 = 0$. M represents the map from the standard rational normal curve onto the random curve we generated. We try to apply LLL to both M and M^{-1}. Both changes of coordinates increase the coefficients of the original quadrics. We only show the first polynomials of the generating sets because of space.

```
> Ran;
[ -99  -91   27    6  -96]
[  81   80   77   79  -71]
[  30  -90  -23  -22  -38]
[  88    8   82   46  -82]
[-100   55   84   88   97]
> H[1];
11142*$.1^2 + 3910*$.1*$.2 + 6189*$.1*$.3 + 4614*$.1*$.4 -
    4878*$.1*$.5 - 6472*$.2^2 - 1524*$.2*$.3 - 4732*$.2*$.4 -
    3344*$.2*$.5 - 3985*$.3^2 - 5245*$.3*$.4 + 8793*$.3*$.5 -
    2014*$.4^2 + 3468*$.4*$.5 - 5174*$.5^2,

> M;
[0.010976626670846010870413851741 9 -0.025840354021725991032949812233 0
    0.019466602645467528879609418411 1 -0.006008326911685933479705839111 83
    0.000661801815549585622955667573798]
[0.022871728873309603864969796376 5 -0.022674825663679218436196353371 9
    0.007463175568457015246941235240 06 -0.000812381225736292330234482571252
    3.417735902331657402441933468 46E-6]
[0.064771034842221108137473485766 2 -0.098716862540920603421527534958 2
    0.052670784634577622242135800068 9 -0.011678879373893466601579505758 9
    0.000904259196951546730798041661287]
[-0.109505375884754266776567078987 0.151214134498079577649471396674
    -0.075688997384703143467680092140 0 0.016067731066283496948106031913 5
```

```
    -0.0012057908202743619648161972652 8]
[0.033198702999494944537467473068 7 -0.059177342149405073062545293792 5
    0.037118921735100778830170352434 3 -0.009975203909030731215367429729 61
    0.000980790142194652336055128158293]
> _,A:=LLL(M);
> _,B:=LLL(M^-1);
> H[1]^ChangeRing(A,Rationals());
-126755530*$.1^2 - 155841195*$.1*$.2 - 157036729*$.1*$.3 -
    87806065*$.1*$.4 + 208015013*$.1*$.5 - 47080973*$.2^2 -
    95667416*$.2*$.3 - 52988845*$.2*$.4 + 128326976*$.2*$.5 -
    48410595*$.3^2 - 53870435*$.3*$.4 + 129096232*$.3*$.5 -
    14908948*$.4^2 + 72322547*$.4*$.5 - 85279389*$.5^2,
> H[1]^ChangeRing(B,Rationals());
-67085517*$.1^2 + 178368130*$.1*$.2 - 83081797*$.1*$.3 +
    15447511*$.1*$.4 - 888454*$.1*$.5 - 119896541*$.2^2 +
    113625627*$.2*$.3 - 21786523*$.2*$.4 + 1343072*$.2*$.5 -
    27548926*$.3^2 + 10964565*$.3*$.4 - 727579*$.3*$.5 -
    1151224*$.4^2 + 167515*$.4*$.5 - 6939*$.5^2,
>
```

References

[1] D. Eisenbud: *Geometry of Syzygies : A Second Course in Commutative Algebra and Algebraic Geometry*, Springer (2005).

[2] D. Simon: *Solving Quadratic Equations Using Reduced Unimodular Quadratic Forms*, Mathematics of Computation, vol. 74, No. 251 (2005), 1531-1543.