1 Hermitian curve example

Consider the *Hermitian curve* in characteristic 2:

$$\mathcal{X} := \{(X : Y : Z) \in \mathcal{P}^2(\overline{\mathbf{F}}_2) \ : \ X^5 - Y^4Z - YZ^4 = 0\},$$

of interest in coding theory because it has overly many points (64) rational over $\mathbf{F}_{16}$. This is in *one-point* or *special form* because the *homogeneous, rational functions* $y := Y/Z$ and $x := X/Z$ have all of their *poles* at the single point $P_\infty := (0 : 1 : 0)$. There is a ring $\mathcal{L}(\infty P_\infty)$ of all the homogeneous, rational functions modulo $\mathcal{X}$ with no poles except possibly at P_∞.

The quotient ring, A , whether written as

$$\overline{\mathbf{F}}_2[x, y]/\langle x^5 - y^4 - y\rangle$$

or

$$\overline{\mathbf{F}}_2[y, x]/\langle y^4 + y - x^5\rangle$$

is only guaranteed to be a subring of this ring, with its *integral closure* being the ring $\mathcal{L}(\infty P_\infty)$ in question.

This example turns out to be *integrally closed*, and the *presentation* as an $\overline{\mathbf{F}}_2[f_4]$-*algebra* with $\overline{\mathbf{F}}_2[f_4]$-*module basis* $(f_0 := 1, f_5 := y, f_{10} := y^2, f_{15} := y^3)$ and $f_4 := x$, has *ideal of relations* with *minimal, reduced Gröbner basis*

$$f_5^2 - f_{10}$$

$$f_{10}f_5 - f_{15}$$

$$f_{10}^2 - (f_4^5 - f_5)$$

$$f_{15}f_5 - (f_4^5 - f_5)$$

$$f_{15}f_{10} - (f_5f_4^5 - f_{10})$$

$$f_{15}^2 - (f_{10}f_4^5 - f_{15}),$$

describing the $\overline{\mathbf{F}}_2[f_4]$-*algebra* multiplication rules. It also emphasizes that there are *canonical* functions f_4^i, $f_5f_4^i$, $f_{10}f_4^i$, and $f_{15}f_4^i$, $i \geq 0$, with poles of any order except $15-4 = 11$, $15-8 = 7$, $15-12 = 3$, $10-4 = 6$, $10-8 = 2$, and $5-4 = 1$; meaning the curve has *genus* 6.

I suppose it can be argued that

$$f_5^4 - (f_4^5 - f_5)$$

is shorthand for the above algebra presentation; but perhaps it gives the wrong message that one should always expect such a shorthand presentation. More likely though, it obscures the fact that there even is such an algebra presentation rather than highlighting that fact.

2 Disguised version

Now disguise this using $f_9 := f_5 f_4$ and $f_8 := f_4^2$ and arbitrary characteristic to get

$$A := \mathbf{F}[f_9; f_8]/\langle f_9^8 - f_8^9 + 2f_9 f_8^6 - f_9^2 f_8^3 \rangle$$

This is not integrally closed, as it lacks 22 of the functions above. [This example was used to show that the `normal` function in SINGULAR 3-0-4 and the `integralClosure` function in MACAULAY2 VERSION 1.1 both terminated prematurely most of the time. It also showed that MAGMA's `IntegralClosure` function used (as it still does) a much too restrictive separability condition.]

It should be noted that this ring A is of little importance, except as a way of defining the integral closure above. In fact, any two functions with pole orders relatively prime to each other could be used to define a similar A of little interest, yet give this same integral closure. What is important about A is that it has a *weight function* that induces a *weight function* on its integral closure, corresponding to the pole orders of the related homogeneous, rational functions.

Let's see how various implementations of standard algorithms fare at producing a useful presentation such as the one given above.

What should bother anyone about this is that there seems to be no agreement as to what should be an acceptable presentation, let alone a canonical one. [I clearly believe in one such as I produced above that gives a strict affine P-algebra presentation relative to a best Noether normalization P, with a weight function and corresponding monomial ordering. While my implementations of my qth-power algorithm give a presentation relative to the given Noether normalization, P, there is a minimization function that will suggest a best P. Together these give the desired canonical presentation.]

3 SINGULAR, version 3-1-3

Chapter 3 of the SINGULAR book has a canonical view of the proper presentation of an integral closure of such a ring A as a *strict, affine A-algebra* (*strict* being my terminology), meaning having ideal of relations with a minimal, reduced Gröbner basis having quadratic relations defining an A-algebra multiplication, linear relations defining possible A-syzygies, and the relations defining A itself.

This has the advantage of having A explicitly as a subring. But when the particular A is of little importance, as in this disguised example, being just a means to an end, inducing the proper structure and defining the proper integral closure, this doesn't usually produce a very good presentation.

The current `normal` function is an implementation of *de Jong's algorithmic approach*, consistent with the above theory, unlike the previous one, now called `normalC`. [The latter is still kept around though it is undoubtedly inferior in all respects to the former.] `normalP` is a stripped-down version of the *qth-power algorithm* in the spirit of Singh and Swanson. [That is, all of the structure of the qth-power approach is ignored in order to make the basic idea of a descending chain of modules apply to a general input, at the price of getting a general unstructured output.] If ``` ``withRing'' ``` and ``` ``noRed'' ``` are used as options, it seems that the same type of presentation results.

Using a weighted monomial order

```
                     SINGULAR                                 /
 A Computer Algebra System for Polynomial Computations       /   version 3-1-3
                                                            0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   March 2011
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \
> LIB "normal.lib";
> ring r=0,(y,x),wp(9,8);
> ideal i=y8-y2x3+2yx6-x9;
> list nor=normal(i);nor;
//   characteristic : 0
//   number of vars : 7
//        block   1 : ordering dp
//                  : names    T(1) T(2) T(3) T(4) T(5)
//        block   2 : ordering wp
//                  : names    y x
//                  : weights  9 8
//        block   3 : ordering C
[2]:
   [1]:
      _[1]=y5x2
      _[2]=y6x
      _[3]=y3x5-y4x2
      _[4]=y8
      _[5]=yx8-y2x5
      _[6]=yx6-y2x3
>  def R=nor[1][1];
>  setring R;
> option(redSB);
> ideal s=std(norid);s;
s[1]=y^8-x^9+2*y*x^6-y^2*x^3
s[2]=T(5)-x^2
s[3]=T(4)*y-x^3+y
s[4]=T(4)*x^6-y^7-x^6+y*x^3
s[5]=T(3)*x-y^2
s[6]=T(3)*y^5-T(4)*x^5+x^5-y*x^2
s[7]=T(2)*y^2-T(4)*x
s[8]=T(2)*x^4-T(2)*y*x-T(3)*y^3
s[9]=T(1)*y-T(2)*x
s[10]=T(1)*x^3-T(2)*x-T(3)*y^2
s[11]=T(4)^2-T(2)*y*x^2+T(4)
s[12]=T(3)*T(4)+T(3)-y*x^2
s[13]=T(2)*T(4)-T(1)*x^2+T(2)
s[14]=T(1)*T(4)-T(3)*y
s[15]=T(3)^2-T(1)*x^2+T(2)
```

```
s[16]=T(2)*T(3)-T(4)
s[17]=T(1)*T(3)-T(2)*y
s[18]=T(2)^2-T(3)
s[19]=T(1)*T(2)-y
s[20]=T(1)^2-x
```

```
> int q=7;
> ring r=q,(y,x),wp(9,8);
> ideal i=y8-y2x3+2yx6-x9;
> list norp=normalP(i,"withRing","noRed");norp;
//   characteristic : 7
//   number of vars : 6
//        block   1 : ordering dp
//                  : names    T(1) T(2) T(3) T(4)
//        block   2 : ordering wp
//                  : names    y x
//                  : weights  9 8
//        block   3 : ordering C
[2]:
   [1]:
      _[1]=y5x
      _[2]=y6
      _[3]=y3x4-y4x
      _[4]=x8-y2x2
      _[5]=yx5-y2x2
> def Rp=norp[1][1];
>  setring Rp;
> option(redSB);
> ideal sp=std(norid);sp;
sp[1]=y^8-x^9+2*y*x^6-y^2*x^3
sp[2]=T(4)*y-x^3-y
sp[3]=T(4)*x^6-y^7-3*x^6+y*x^3
sp[4]=T(3)*x-y^2
sp[5]=T(3)*y^5-T(4)*x^5+3*x^5-y*x^2
sp[6]=T(2)*y^2-T(4)*x+2*x
sp[7]=T(2)*x^4-T(2)*y*x-T(3)*y^3
sp[8]=T(1)*y-T(2)*x
sp[9]=T(1)*x^3-T(2)*x-T(3)*y^2
sp[10]=T(4)^2-T(2)*y*x^2-3*T(4)+2
sp[11]=T(3)*T(4)-T(3)-y*x^2
sp[12]=T(2)*T(4)-T(1)*x^2-T(2)
sp[13]=T(1)*T(4)-2*T(1)-T(3)*y
sp[14]=T(3)^2-T(1)*x^2+T(2)
sp[15]=T(2)*T(3)-T(4)+2
sp[16]=T(1)*T(3)-T(2)*y
sp[17]=T(2)^2-T(3)
sp[18]=T(1)*T(2)-y
sp[19]=T(1)^2-x
```

Without ‘‘noRed’’,

```
> int q=7;
> ring r=q,(y,x),wp(9,8);
> ideal i=y8-y2x3+2yx6-x9;
> int t=timer;
> list norp=normalP(i,"withRing");norp;
//   characteristic : 7
//   number of vars : 2
//        block   1 : ordering dp
//                  : names    T(1) T(3)
//        block   2 : ordering C

      _[1]=y5x
      _[2]=y6
      _[3]=y3x4-y4x
      _[4]=x8-y2x2
      _[5]=yx5-y2x2

> def Rp=norp[1][1];
> setring Rp;
> option(redSB);
> ideal sp=std(norid);sp;
sp[1]=T(1)^10-2*T(1)^5*T(3)^2+T(3)^4-T(3)
```

And without ‘‘noRed’’ but using a grevlex-over-weight monomial ordering,

```
> intmat a[2][2]=1,0,9,8;
> int q=7;
> ring r=q,(y,x),M(a);
> ideal i=y8-y2x3+2yx6-x9;
> int t=timer;
> list norp=normalP(i,"withRing");norp;
//   characteristic : 7
//   number of vars : 2
//        block   1 : ordering dp
//                  : names    T(2) T(4)
//        block   2 : ordering C

      _[1]=y2x12
      _[2]=y4x11
      _[3]=y6x10
      _[4]=y7+y6x3+y5x6+y4x9-yx3+x6
      _[5]=x13

> def Rp=norp[1][1];
> setring Rp;
> option(redSB);
> ideal sp=std(norid);sp;
sp[1]=T(4)^20+3*T(2)*T(4)^15-T(2)^2*T(4)^10+3*T(2)^3*T(4)^5+T(2)^4-T(2)
```

We know, in this disguised example, that there is no reason to keep the input variables, and that a strict affine A-algebra presentation is not the preferred one. But what is more discouraging about all of the above is that there is no realization that there is a natural monomial ordering for the output induced by that on the input.

And what are we to think of the reduced versions in `normalP`? These have nothing to do with the qth-power algorithm, they could equally well be applied to the `normal` output. Is the point to ignore the theory in the SINGULAR book in an attempt to remove variables based on no real mathematics other than the fact that it can be done? This is surprising in and of itself, given that `normal` almost always produces extraneous variables that could easily be removed for *good* mathematical reasons. For instance, in the example above, the relation s[2] is a tacit admission that T[5] is clearly unnecessary. So are we to believe that the affine A-algebra presentation is more important or that minimizing the number of variables is? There is clearly no consistent answer that can be given to either of these based even on this one example.

And we'll leave it as an open question whether there is any way to determine if $(-yx + x^4)/y^3 \in Q(A)$ is in any of these integral closures, with the only appropriate answer being a "direct" method that would apply to testing all elements of $Q(A)$.

4 Macaulay2, version 1.4

In Macaulay2, there is a loaded package `IntegralClosure` containing the functions `integralClosure` and `icFracP`, again implementing the de Jong and stripped-down qth-power algorithms.

`integralClosure` produces an affine A-algebra presentation, but almost never strict. As with `normal` it produces a product order, but chooses *lex* on the new variables over the input order on the input variables. `icFracP` is the code used in the Singh-Swanson, 2009 paper.

```
Macaulay2, version 1.4
with packages: ConwayPolynomials, Elimination, IntegralClosure, LLLBases,
               PrimaryDecomposition, ReesAlgebra, TangentCone

i1 : R0=QQ[y,x,MonomialOrder=>{Weights=>{9,8},Weights=>{1,0}}];
i2 : I0=ideal(y^8-y^2*x^3+2*y*x^6-x^9);
i3 : A0=R0/I0;
i4 : ICf0=icFractions A0; toString ICf0

o5 = {(y^4)/(x^4-y*x), (x^4-y*x)/(y^3), y, x}

i6 : G0=transpose gens gb presentation integralClosure A0

o6 = {-9} | y8-x9+2yx6-y2x3                |
     {-4} | w_(10,0)y3-x4+yx               |
     {-5} | w_(10,0)x5-w_(10,0)yx2-y5      |
     {-2} | w_(10,0)^2x-y2                 |
     {-3} | w_(10,0)^3y-x3+y               |
     {-3} | w_(10,0)^5+w_(10,0)^2-yx2      |
     {-1} | w_(12,0)y-w_(10,0)x            |
     {0}  | w_(12,0)x2-w_(10,0)^4-w_(10,0) |
     {-1} | w_(12,0)w_(10,0)-y             |
     {-1} | w_(12,0)^2-x                   |

i7 : Rq=ZZ/7[y,x,MonomialOrder=>{Weights=>{9,8},Weights=>{1,0}}];
i8 : Iq=ideal(y^8-y^2*x^3+2*y*x^6-x^9);
i9 : Aq=Rq/Iq
i10 : ICfq=icFractions Aq; toString ICfq

o11 = {(y^4)/(x^4-y*x), (x^4-y*x)/(y^3), y, x}

i12 : Gq=transpose gens gb presentation integralClosure Aq

o12 = {-9} | y8-x9+2yx6-y2x3                |
      {-4} | w_(10,0)y3-x4+yx               |
      {-5} | w_(10,0)x5-w_(10,0)yx2-y5      |
```

```
      {-2} | w_(10,0)^2x-y2                 |
      {-3} | w_(10,0)^3y-x3+y               |
      {-3} | w_(10,0)^5+w_(10,0)^2-yx2      |
      {-1} | w_(12,0)y-w_(10,0)x            |
      {0}  | w_(12,0)x2-w_(10,0)^4-w_(10,0) |
      {-1} | w_(12,0)w_(10,0)-y             |
      {-1} | w_(12,0)^2-x                   |

i13 : ICq=icFracP Aq; toString(ICq)

o14 = {1, (y^4)/(x^4-y*x), (x^4-y*x)/(y^3), (y^2)/(x), (x^3-2*y)/(y)}
```

As with SINGULAR above, the most important failing here is the recognition that the input monomial ordering induces a natural monomial ordering on the output, far superior to a generic lex/input choice.

`integralClosure`, as with `normal` also suffers from the requirement that the input ring A be explicit as part of the presentation, which we know is counterproductive in this example. `icFractions` finds fewer fractions, the two necessary to describe the integral closure as a ring over A, not as an A-module (as well as printing the two variables from A for some reason). `icFracP` produces appropriate fractions to describe a minimizes P-module presentation, though this seems not to be the case in general.

In fact, if one changes the weights to grevlex-over-weight instead of weight-over-grevlex in the input monomial order

```
i13 : ICq=icFracP Aq; toString(ICq)

o14 = {1, (-y^4)/(y*x-x^4), (-y^5)/(y*x^2-x^5), (-y^2)/(x), (-2*y+x^3)/(y),
      (y^6+x^3)/(x^3), (y^6-2*y+x^3)/(y), (y^6+y^4*x^6+x^3)/(x^3),
      (y^6+y^4*x^6-2*y+x^3)/(y), (y^6+y^4*x^6+y^2*x^12+x^3)/(x^3),
      (-3*y^6-3*y^4*x^6-3*y^2*x^12-y-3*x^3)/(y)}
```

There is no attempt at giving a presentation of any sort relative to the fractions produced by `icFracP`, and as mentioned below in the timings section, it necessarily has problems dealing with even moderate sized prime characteristics. It also doesn't try to simplify the fractions it does produce, though this may be impossible given how elements of frac(A) are micro-managed by MACAULAY2.

Since there are no common denominators in any of these sets of fractions, determining membership of, say $(y^7+y^6x^3+y^5x^6+y^4x^9-yx^3+x^6)/x^{13} \in Q(A)$ produced in SINGULAR above must be a problem.

5 Magma

Let's try Magma's, `IntegralClosure` function, since it works on separable extensions over one free variable.

There is no way to use weights; (and we know it would complain about separability in char 2, since the derivative with respect to y there is 0). It necessarily produces an $\mathbf{F}[x]$-module basis, as elements of $\mathbf{F}(x)[y]$.

```
F:=GF(107);
FF<x>:=FunctionField(F);
PP<y>:=PolynomialRing(FF);
f:=y^8-y^2*x^3+2*y*x^6-x^9;
A<Y>:=RationalExtensionRepresentation(FunctionField(f));
C<X>:=CoefficientRing(A);
INT:=Integers(C);
ICA:=IntegralClosure(INT,A);
B:=Basis(ICA);
for i in [1..#B] do i-1,B[i]; end for;

0 1
1 Y
2 1/X*Y^2
3 1/X*Y^3
4 1/X^2*Y^4
5 1/X^2*Y^5
6 1/X^3*Y^6
7 1/X^13*Y^7 + 1/X^10*Y^6 + 1/X^7*Y^5 + 1/X^4*Y^4 + 106/X^10*Y + 1/X^7
```

```
F:=Rationals();
FF<x>:=FunctionField(F);
PP<y>:=PolynomialRing(FF);
f:=y^8-y^2*x^3+2*y*x^6-x^9;
A<Y>:=RationalExtensionRepresentation(FunctionField(f));
C<X>:=CoefficientRing(A);
INT:=Integers(C);
ICA:=IntegralClosure(INT,A);
B:=Basis(ICA);
for i in [1..#B] do i-1,B[i]; end for;

0 1
1 Y
2 1/X*Y^2
3 1/X*Y^3
4 1/X^2*Y^4
```

```
5 1/X^2*Y^5
6 1/X^3*Y^6
7 1/X^13*Y^7 + 1/X^10*Y^6 + 1/X^7*Y^5 + 1/X^4*Y^4 - 1/X^10*Y + 1/X^7
```

In both cases, it gives an $\mathbf{F}[f_8]$-module basis with elements of weights $0, 9, 10, 19, 20, 29, 30, 4$, obscuring the function of weight 5. And the integral closure is treated as a subset of, maybe, $\mathbf{F}(X)[Y]/\langle f \rangle$, with no explicit version given.

Then try MAGMA's `Normalisation` function. This produces a lex-ordered presentation, occassionally, as here, properly reduced, but sometimes bad enough to be unreadable.

```
Magma V2.15-9     Sun Jan 30 2011 12:43:46 on dell-desktop

Q:=GF(107);
R<y,x>:=PolynomialRing(Q,2,"weight",[9,8,1,0]);
I:=ideal<R|y^8-y^2*x^3+2*y*x^6-x^9>;
IC:=Normalisation(I);
x@IC[1][2];
y@IC[1][2];
G:=GroebnerBasis(IC[1][1]);G;

$.1^2
$.1*$.2
[
    $.1^5 + 106*$.2^4 + 106*$.2
]

Q:=Rationals();
R<y,x>:=PolynomialRing(Q,2,"weight",[9,8,1,0]);
I:=ideal<P|y^8-y^2*x^3+2*y*x^6-x^9>;
IC:=Normalisation(I);
x@IC[1][2];
y@IC[1][2];
G:=GroebnerBasis(IC[1][1]);G;

$.1^2
$.1*$.2
[
    $.1^5 - $.2^4 - $.2
]
```

Here `Normalisation` produces a minimized version as a ring, though this is far from the truth in other examples. Its default choice of *lex monomial ordering* causes major problems in other examples, not especially noticeable here, though still backwards from optimal.

6 My qth-power algorithm

My second implementation of my qth-power algorithm, written in the `QthPower` package for Macaulay2, is relatively fast for small q, and the char 0 version, necessarily slightly slower, can be specialized to all large q. It is set up to produce a strict affine P-module presentation relative to the given Noether normalization P, with monomial ordering induced by the required weight-over-grevlex monomial ordering on the input. There is a minimization function that will produce a best choice of Noether normalization when one exists, though the code may not be up to writing the new strict affine P-module presentation at this time. (It comes close enough that one can modify it by hand easily. In any case, it is probably better to redefine the ring so as to use new variable names that reflect the weights, as opposed to those involving the default variable p used by Macaulay2.)

```
i1 : loadPackage "QthPower";

i2 : q=17;

i3 : wtR=matrix{{9,8}};

i4 : Rq=ZZ/q[y,x,Weights=>entries weightGrevlex(wtR)];

i5 : Iq={y^8-y^2*x^3+2*y*x^6-x^9};

i6 : icq=qthIntegralClosure(wtR,Rq,Iq);
     toString(icq)

o7 = ({x^13,
       y^4*x^9+y^5*x^6+y^6*x^3+y^7+x^6-y*x^3,
       y^5*x^8+y^6*x^5+y^7*x^2+x^8-y*x^5,
       y*x^13,
       y^2*x^12,
       y^6*x^8+y^7*x^5+x^11-y*x^8,
       y^7*x^7-y*x^10,
       y^3*x^12},
      {p_0^2+p_1*p_7-p_2*p_7^3+p_7,
       p_0*p_1+2*p_0-p_3*p_7^3,
       p_0*p_2+p_3*p_7-p_4*p_7^3,
       p_0*p_3+p_2-p_5*p_7^3,
       p_0*p_4+p_5*p_7-p_6*p_7^3,
       p_0*p_5+p_4-p_7^3,
       p_0*p_6-p_1*p_7-p_7,
       p_1^2+3*p_1-p_2*p_7^2+2,
       p_1*p_2+2*p_2-p_5*p_7^3,
       p_1*p_3+2*p_3-p_4*p_7^2,
```

```
       p_1*p_4+2*p_4-p_7^3,
       p_1*p_5+2*p_5-p_6*p_7^2,
       p_1*p_6-p_0+p_6,
       p_2^2+p_5*p_7-p_6*p_7^3,
       p_2*p_3+p_4-p_7^3,
       p_2*p_4-p_1*p_7-p_7,
       p_2*p_5-p_0,
       p_2*p_6-p_3*p_7,
       p_3^2+p_5-p_6*p_7^2,
       p_3*p_4-p_0,
       p_3*p_5-p_1-1,
       p_3*p_6-p_2,
       p_4^2-p_3*p_7,
       p_4*p_5-p_2,
       p_4*p_6-p_5*p_7,
       p_5^2-p_3,
       p_5*p_6-p_4,
       p_6^2-p_7},
      (ZZ/17)[p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7],
      matrix {{19, 15, 14, 10, 9, 5, 4, 8}})

i8 : minq=minimization(icq);

i9 : minicq=qthIntegralClosure(minq#0,minq#1,minq#2);
     toString(minicq)

o10 = ({1,
        p_5,
        p_3,
        p_1},
       {p_0^2+3*p_0-p_1*p_3^5+2,
        p_0*p_1+2*p_1-p_2*p_3^5,
        p_0*p_2+2*p_2-p_3^5,
        p_1^2+p_2-p_3^5,
        p_1*p_2-p_0-1,
        p_2^2-p_1},
       (ZZ/17)[p_0, p_1, p_2, p_3],
       matrix {{15, 10, 5, 4}})
```

```
Macaulay2, version 1.4

i1 : loadPackage "QthPower";

i2 : q=7;

i3 : wtR=matrix{{9,8}};

i4 : Rq=ZZ/q[y,x,Weights=>entries weightGrevlex(wtR)];

i5 : Iq={y^8-y^2*x^3+2*y*x^6-x^9};

i6 : icq=qthIntegralClosure(wtR,Rq,Iq);
     toString(icq)

o7 = ({x^13,
       y^4*x^9+y^5*x^6+y^6*x^3+y^7+x^6-y*x^3,
       y^5*x^8+y^6*x^5+y^7*x^2+x^8-y*x^5,
       y*x^13,
       y^2*x^12,
       y^6*x^8+y^7*x^5+x^11-y*x^8,
       y^7*x^7-y*x^10,
       y^3*x^12},
      {p_0^2+p_1*p_7-p_2*p_7^3+p_7,
       p_0*p_1+2*p_0-p_3*p_7^3,
       p_0*p_2+p_3*p_7-p_4*p_7^3,
       p_0*p_3+p_2-p_5*p_7^3,
       p_0*p_4+p_5*p_7-p_6*p_7^3,
       p_0*p_5+p_4-p_7^3,
       p_0*p_6-p_1*p_7-p_7,
       p_1^2+3*p_1-p_2*p_7^2+2,
       p_1*p_2+2*p_2-p_5*p_7^3,
       p_1*p_3+2*p_3-p_4*p_7^2,
       p_1*p_4+2*p_4-p_7^3,
       p_1*p_5+2*p_5-p_6*p_7^2,
       p_1*p_6-p_0+p_6,
       p_2^2+p_5*p_7-p_6*p_7^3,
       p_2*p_3+p_4-p_7^3,
       p_2*p_4-p_1*p_7-p_7,
       p_2*p_5-p_0,
       p_2*p_6-p_3*p_7,
       p_3^2+p_5-p_6*p_7^2,
       p_3*p_4-p_0,
       p_3*p_5-p_1-1,
       p_3*p_6-p_2,
       p_4^2-p_3*p_7,
```

```
       p_4*p_5-p_2,
       p_4*p_6-p_5*p_7,
       p_5^2-p_3,
       p_5*p_6-p_4,
       p_6^2-p_7},
      (ZZ/7)[p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7],
      matrix {{19, 15, 14, 10, 9, 5, 4, 8}})

i8 : minq=minimization(icq);

i9 : minicq=qthIntegralClosure(minq#0,minq#1,minq#2);
     toString(minicq)


o10 = ({1,
        p_5,
        p_3,
        p_1},
       {p_0^2+3*p_0-p_1*p_3^5+2,
        p_0*p_1+2*p_1-p_2*p_3^5,
        p_0*p_2+2*p_2-p_3^5,
        p_1^2+p_2-p_3^5,
        p_1*p_2-p_0-1,
        p_2^2-p_1},
       (ZZ/7)[p_0, p_1, p_2, p_3],
       matrix {{15, 10, 5, 4}})

i11 : R0=QQ[y,x,Weights=>entries weightGrevlex(wtR)];

i12 : I0={y^8-y^2*x^3+2*y*x^6-x^9};

i13 : ic0=rationalIntegralClosure(wtR,R0,I0);
      toString(ic0)

o14 = ({x^13,
        y^4*x^9+y^5*x^6+y^6*x^3+y^7+x^6-y*x^3,
        y^5*x^8+y^6*x^5+y^7*x^2+x^8-y*x^5,
        y*x^13,
        y^2*x^12,
        y^6*x^8+y^7*x^5+x^11-y*x^8,
        y^7*x^7-y*x^10,
        y^3*x^12},
       {p_0^2+p_1*p_7-p_2*p_7^3+p_7,
        p_0*p_1+2*p_0-p_3*p_7^3,
        p_0*p_2+p_3*p_7-p_4*p_7^3,
        p_0*p_3+p_2-p_5*p_7^3,
```

```
         p_0*p_4+p_5*p_7-p_6*p_7^3,
         p_0*p_5+p_4-p_7^3,
         p_0*p_6-p_1*p_7-p_7,
         p_1^2+3*p_1-p_2*p_7^2+2,
         p_1*p_2+2*p_2-p_5*p_7^3,
         p_1*p_3+2*p_3-p_4*p_7^2,
         p_1*p_4+2*p_4-p_7^3,
         p_1*p_5+2*p_5-p_6*p_7^2,
         p_1*p_6-p_0+p_6,
         p_2^2+p_5*p_7-p_6*p_7^3,
         p_2*p_3+p_4-p_7^3,
         p_2*p_4-p_1*p_7-p_7,
         p_2*p_5-p_0,
         p_2*p_6-p_3*p_7,
         p_3^2+p_5-p_6*p_7^2,
         p_3*p_4-p_0,
         p_3*p_5-p_1-1,
         p_3*p_6-p_2,
         p_4^2-p_3*p_7,
         p_4*p_5-p_2,
         p_4*p_6-p_5*p_7,
         p_5^2-p_3,
         p_5*p_6-p_4,
         p_6^2-p_7},
        QQ[p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7],
        matrix {{19, 15, 14, 10, 9, 5, 4, 8}})

i15 : min0=minimization(ic0);

i16 : minic0=rationalIntegralClosure(min0#0,min0#1,min0#2);
      toString(minic0)

o17 = ({1,
         p_5,
         p_3,
         p_1},
        {p_0^2+3*p_0-p_1*p_3^5+2,
         p_0*p_1+2*p_1-p_2*p_3^5,
         p_0*p_2+2*p_2-p_3^5,
         p_1^2+p_2-p_3^5,
         p_1*p_2-p_0-1,
         p_2^2-p_1},
        QQ[p_0, p_1, p_2, p_3],
        matrix {{15, 10, 5, 4}})
```

7 qth-power algorithm in MAGMA

My original implementation of my qth-power algorithm was written in MAGMA code. Here it was possible to generate variable names reflecting the weights induced, at least when printing the results to the screen.

```
Magma V2.15-5
Loading file "qchar051109.mag"
Loading "closure_func040909.mag"
11111111111111111111111111111111111111111111111111111111111
q= 3
Delta= $.2^27
WT_MATRIX_T= [
    [ 10, 9, 7, 6, 5, 4, 3, 8 ]
]
time for q= 3 is 0.050 seconds
modulus= 3
11111111111111111111111111111111111111111111111111111111111
q= 5
11111111111111111111111111111111111111111111111111111111111
q= 7
Delta= $.2^24
WT_MATRIX_T= [
    [ 19, 15, 14, 10, 9, 5, 4, 8 ]
]
time for q= 7 is 0.030 seconds
11111111111111111111111111111111111111111111111111111111111
q= 11
Delta= $.2^24
WT_MATRIX_T= [
    [ 19, 15, 14, 10, 9, 5, 4, 8 ]
]
time for q= 11 is 0.040 seconds
modulus= 77
[
    f_15^2*f_8^12,
    f_15*f_8^13,
    f_15^7*f_8^6 + f_15^5*f_8^7 + f_15^6*f_8^4 - f_15^2*f_8^11 + f_15^7*f_8 +
        f_8^12 - f_15*f_8^9 + f_8^7 - f_15*f_8^4,
    f_15^6*f_8^7 + f_15^7*f_8^4 + f_15*f_8^12 + f_8^10 - f_15*f_8^7,
    f_15^5*f_8^8 + f_15^6*f_8^5 + f_15^7*f_8^2 + f_8^8 - f_15*f_8^5,
    f_15^4*f_8^9 + f_15^5*f_8^6 + f_15^6*f_8^3 + f_15^7 + f_8^6 - f_15*f_8^3,
    -f_15^6*f_8^6 - f_15^7*f_8^3 + f_15^3*f_8^10 + f_15*f_8^11 - f_8^9 +
        f_15*f_8^6,
    f_8^14,
    f_8^13
```

```
]
11111111111111111111111111111111111111111111111111111111111
q= 13
Delta= $.2^24
WT_MATRIX_T= [
    [ 19, 15, 14, 10, 9, 5, 4, 8 ]
]
time for q= 13 is 0.040 seconds
modulus= 1001
[
    f_9^3*f_8^12,
    f_9^7*f_8^7 - f_9*f_8^10,
    f_9^6*f_8^8 + f_9^7*f_8^5 + f_8^11 - f_9*f_8^8,
    f_9^2*f_8^12,
    f_9*f_8^13,
    f_9^5*f_8^8 + f_9^6*f_8^5 + f_9^7*f_8^2 + f_8^8 - f_9*f_8^5,
    f_9^4*f_8^9 + f_9^5*f_8^6 + f_9^6*f_8^3 + f_9^7 + f_8^6 - f_9*f_8^3,
    f_8^14,
    f_8^13
]
1 f_9^3*f_8^12
2 f_9^7*f_8^7 - f_9*f_8^10
3 f_9^6*f_8^8 + f_9^7*f_8^5 + f_8^11 - f_9*f_8^8
4 f_9^2*f_8^12
5 f_9*f_8^13
6 f_9^5*f_8^8 + f_9^6*f_8^5 + f_9^7*f_8^2 + f_8^8 - f_9*f_8^5
7 f_9^4*f_8^9 + f_9^5*f_8^6 + f_9^6*f_8^3 + f_9^7 + f_8^6 - f_9*f_8^3
8 f_8^14
9 f_8^13
newrelations= [
    f_4^2 - f_8,
    f_5^2 - f_10,
    f_5*f_4 - f_9,
    f_9^2 - f_10*f_8,
    f_9*f_5 - f_14,
    f_9*f_4 - f_5*f_8,
    f_10^2 - f_4*f_8^2 + f_5,
    f_10*f_9 - f_19,
    f_10*f_5 - f_15 - 1,
    f_10*f_4 - f_14,
    f_14^2 - f_4*f_8^3 + f_5*f_8,
    f_14*f_10 - f_8^3 + f_9,
    f_14*f_9 - f_15*f_8 - f_8,
    f_14*f_5 - f_19,
    f_14*f_4 - f_10*f_8,
    f_15^2 - f_14*f_8^2 + 3*f_15 + 2,
```

```
    f_15*f_14 - f_5*f_8^3 + 2*f_14,
    f_15*f_10 - f_9*f_8^2 + 2*f_10,
    f_15*f_9 - f_8^3 + 2*f_9,
    f_15*f_5 - f_4*f_8^2 + 2*f_5,
    f_15*f_4 - f_19 + f_4,
    f_19^2 - f_14*f_8^3 + f_15*f_8 + f_8,
    f_19*f_15 - f_10*f_8^3 + 2*f_19,
    f_19*f_14 - f_9*f_8^3 + f_10*f_8,
    f_19*f_10 - f_5*f_8^3 + f_14,
    f_19*f_9 - f_4*f_8^3 + f_5*f_8,
    f_19*f_5 - f_8^3 + f_9,
    f_19*f_4 - f_15*f_8 - f_8
]
```

in terms of a $\mathbf{Q}[f_8]$ module basis $f_0 := 1, f_9, f_{10}, f_{19}, f_4, f_5, f_{14}, f_{15}$.

Minimization gives:

```
> load "closure_interchange.mag";

[15,10,5,4]
[
    v_15^2+3*v_15-v_10*v_4^5+2,
    v_15*v_10+2*v_10-v_5*v_4^5,
    v_10^2+v_5-v_4^5,
    v_15*v_5+2*v_5-v_4^5,
    v_10*v_5-v_15-1,
    v_5^2-v_10
]
```

Note that $q = 5$ was considered a bad prime in the input, whereas $q = 3$ is also a bad prime in that the integral closure is clearly larger than the reduction *mod* 3 would imply. So the primes $q = 7, 11, 13$ were used to reconstruct the result in characteristic 0, from which it is then possible to specialize to all larger primes q.

8 Timings

Here are rough timing comparisons, all done on the same departmental computer. Times vary somewhat, even on the same machine with the same problem, and certainly are machine dependent. But this clearly points out how dependent certain implementations are on the characteristic, and whether they are competitive with each other time-wise. (* means that the integral closure mod q is larger than in characteristic 0, — means the method is not applicable, ** means it ran out of (time and/or storage) resources, and **** here meant the this was not separable as given, though the variables could have been interchanged to make it separable.)

char	*normal*	*normalP*	*icFract.*	*icFRacP*	*Normalis.*	*Int.Clos.*	*QthPower*	*Qth*
0	0.4	− − −	1.3	− − −	0.8	0.1	1.3	0.0
2	0.4	0.0	0.9	0.1	0.4	* * **	0.4	0.0
∗3	0.3	0.0	1.3	0.1	0.5	0.1	0.3	0.1
∗5	0.8	0.0	5.9	0.1	0.9	0.1	2.3	0.0
7	0.4	0.0	1.1	0.1	0.7	0.1	0.3	0.0
11	0.4	0.2	1.1	0.5	0.7	0.1	0.3	0.1
13	0.4	0.3	1.2	11.	0.7	0.1	0.3	0.0
17	0.4	34.	1.1	39.	0.7	0.1	0.3	0.1
19	0.4	0.8	1.1	82.	0.7	0.1	0.3	0.0
23	0.4	0.7	1.1	**	0.7	0.1	0.3	0.1
29	0.4	1.9	1.2		0.7	0.1	0.3	0.1
31	0.4	75.	1.2		0.7	0.1	0.3	0.1
37	0.4	**	1.2		0.7	0.1	0.4	0.1
41	0.4		1.3		0.7	0.1	0.4	0.1
101	0.4		1.3		0.8	0.1	0.6	0.1
4001	0.4		1.4		0.1	0.1	123.	35.

The failings of both `icFracP` and `normalP` for even relatively small characteristics is based on poor computational implementation of the crucial $f^q \ mod \ I$ step. If one naively computes f^q first, before modding out by I, then this will certainly require large resources of time and space, just as in cryptography when one attempts $a^b \ mod \ c$ for moderately large integers. The qth-power algorithm will always be characteristic-dependent, but not so much as these implementations would suggest.

I have no explanation for the glitch at $q = 17$ in `normalP`.