421 lines (357 sloc) | 14.8 KB

```
/*
This code finds explicit polynomials that define the
involution of the surface X given by the equation
 - x^3w - x^2zw + x^2w^2 - xy^3 + xz^3 - y^2w^2 + z^4 + zw^3 = 0.

Moreover, we will calculate different models of the surface X
*/

Q := RationalField();
P3<x,y,z,w> := ProjectiveSpace(Q,3);

// Quartic in P3
h := -x*y^3 + x*z^3 + z^4 - x^3*w - x^2*z*w
        + x^2*w^2 - y^2*w^2 + z*w^3;
X := Scheme(P3,h);

// Counting points on reduction mod p=2
p := 2;
list := [];
for k in [1..10] do
  P3p := ProjectiveSpace(GF(p^k),3);
  Xp := Scheme(P3p,h);
  pts := #Points(Xp);
  Append(~list,pts-1-p^(2*k)); //list Frobenius traces
end for;

// Calculating Weil polynomail and show that rho(X)=2
R<t> := PolynomialRing(Rationals());
wp := FrobeniusTracesToWeilPolynomials(list, p, 2, 22: KnownFactor := t-p);
assert #wp eq 1;
assert WeilPolynomialToRankBound(wp[1], 2) eq 2;

```

```
33
34  // Curve C on X
35  eqC:= [-y*z + x*w,
36      x*z^2 + z^3 + x*y*w + w^3,
37      x*y^2 + x^2*z + x*z^2 + y*w^2];
38  C := Scheme(P3,eqC);
39  assert C subset X;
40
41  // Curve D on X
42  eqD := [-y*z + x*w,
43      y^2 - z*w,
44      x*y - z^2];
45  D := Scheme(P3,eqD);
46  assert D subset X;
47
48  // Divisors
49  DivC := Divisor(X,C);
50  DivD := Divisor(X,D);
51  DivH := Divisor(X,w);
52
53  assert IsLinearlyEquivalent(DivD,2*DivH-DivC);
54  // Basis of Riemann-Roch space of -H+5C ~ 9H-5D
55  LB := RiemannRochBasis(9*DivH-5*DivD);
56  // N.b. takes some time to calculate: ~1min
57  // Calculating RRB of -H+5C takes a bit longer
58
59  /*
60  We search for a coordinate transformation such that the image
61  under the rational map defined by this basis is again X.
62  First we search for some "simple" points in X.
63  */
64
65  PinX := [];
66  for a,b,c in [-1..5] do
67      P := [a,b,c,1];
68      if P in X eq true then
69          PinX := Include(PinX,P);
70      end if;
71  end for;
72
73  /* We take six points on X out of these which are pairs that
74  map to each other under the involution:
75          [ 1,-1,-1,1] <--> [0,0,-1,1]
76          [ 1,-1, 0,1] <--> [1,0,-1,1]
77          [-1,-1, 0,1] <--> [5,2, 3,1]
78  Note that all points are on the affine where w doesnt vanish.
79
80  For those that want to know how we found that these points
81  indeed map to each other, it can be checked as follows (this
82  idea comes from Ronald van Luijk):
83
```

```
The double cover X -> P2 corresponds to the map X -> |C|^*
that sends a point P in X to the hyperplane in |C| of
effective divisors that contain P. As dim |C| = 2, this
hyperplane is a line and it is a one-dimensional sub-linear
system |C|_P of |C| consisting of those curves in |C| going
through P. If any two curves in a one-dimensional linear
system intersect in a point, then all curves in that system
go through that point as well.

In our case if we take any two curves C_1 and C_2 in |C|_P,
they will intersect in 2 points (including P), and hence all
curves in |C|_P contain these 2 points. That means these
2 points have the same image under the double cover as P.
The associated involution must send the point P on X to R.

This is where we used the following function (further
details are omitted):

function InvK3(f,P) // input: f of curve C in P1xP1
    Q:=RationalField();
    P1xP1<x0,x1,y0,y1>:=ProductProjectiveSpace(Q,[1,1]);
    g:= -x1^2*y0 + x0^2*y1;
    D:=Scheme(P1xP1,g);
    Dim<x,y,z,w>:=SegreEmbedding(D);
    q:=DefiningEquations(Dim);

    C:=Scheme(P1xP1,f);
    CD:=Union(C,D);
    CDim<x,y,z,w>:=SegreEmbedding(CD);
    h:=DefiningEquations(CDim)[2];

    R<x,y,z,w>:=PolynomialRing(Q,4);
    P3:=ProjectiveSpace(R);
    X:=Scheme(P3,h);
    if P in X then
        M:=Matrix(Q,#q,1,
            [Evaluate(q[i],P):i in [1..#q]]);
        kerM:=Basis(Kernel(M));
        if Dimension(Kernel(M)) ne 2 then
            return("P? kerM not 2");
        end if;
        p1:=&+[R!kerM[1,i]*R!q[i]: i in [1..3]];
        p2:=&+[R!kerM[2,i]*R!q[i]: i in [1..3]];
        Y1:=Scheme(P3,p1);
        Y2:=Scheme(P3,p2);
        I1:=Intersection(Y1,X);
        I2:=Intersection(Y2,X);
        Dd:=Scheme(P3,q);
        C1:=Difference(I1,Dd);
        C2:=Difference(I2,Dd);
        IP:=Intersection(C1,C2);
```

```
        if Dimension(IP) eq 0 then
            RP:=RationalPoints(IP);
            if #RP eq 2 then
                if RP[1] eq RP!P then
                    return(RP[2]);
                end if;
                return(RP[1]);
            end if;
            return("P? #RP not 2");
        end if;
        return("Something went wrong");
    end if;
    return("point not on X");
end function;

N.B. One can also try to apply this construction on the generic
point to find equations for the involution, but this gives more
complicated defining polynomials for the involution.

*/


preinv := map < X -> P3 | LB >;
Pts := [P3!PinX[i] : i in [5,3,6,7,1,10]];
ev := [preinv(Pts[i]) : i in [1..5]];

M1 := TranslationOfSimplex(P3,[Pts[i] :
          i in [2,1,4,3,6]]);
N1 := TranslationOfSimplex(P3,ev);
i1 := Expand(preinv*N1^(-1)*M1);

"The K3 surface X: ";
X;
"The involution on X is defined by the polynomials: ";
[ DefiningPolynomials(i1)[i]/8 : i in [1..4] ];
// apparently we can devide by 8...

/*

Magma is not able to check if i1 is an involution, because the
polynomials are too involved. But one can be convinced that this
is indeed the involution, by what we will do in the remainder of
these calculation...

We will calculate different models. Recall that C+D is linearly
equivalent to 2H, so |C|=|2H-D|, |3H-C|=|H+D| and |3C|=|6H-3D|.

------------------------------------------------------------*/

// Model as double cover

```

```
L2HmD := RiemannRochBasis(2*DivH-DivD);
// equations for double cover map
P2<s,t,u> := ProjectiveSpace(Q,2);
phi := map < X -> P2 | L2HmD >; // double cover

L6Hm3D := RiemannRochBasis(6*DivH-3*DivD);
// gives model in P10, splits via P(1,1,1,3) in
// the following way
f1 := L6Hm3D[3];
Func := L2HmD;
Func1 := Append(Func,f1);
// equations for map to P(1,1,1,3)

WPS<xx,yy,zz,ww> := ProjectiveSpace(Q,[1,1,1,3]);
prepi := map < X -> WPS | Func1 >;
// embedding of X in P(1,1,1,3),

/* The above embedding gives surface:
         x^5y - x^3y^2z + y^5z + x^4z^2 + xy^3z^2
           - y^4z^2 + y^2z^4 + xz^5 + x^3w + x^2yw
           - y^3w + w^2 = 0
in P(1,1,1,3) so there are still linear terms

Needed a linear map to adjust this embedding
 pi2 := map < WPS -> WPS |
    [x,y,z,(w+(x^3+x^2*y-y^3/2))/2]>;
 Image(pi*pi2);
 Gives surface:
        x^6 - 2x^5y + x^4y^2 - 2x^3y^3
        - 2x^2y^4 + y^6 + 4x^3y^2*z - 4y^5z - 4x^4z^2
        - 4xy^3z^2 + 4y^4z^2 - 4y^2z^4 - 4xz^5 - w^2
*/

f2 := (f1+(Func[1]^3+Func[1]^2*Func[2]
          -Func[2]^3)/2)*2;
Func2 := Append(Func,f2);
pi := map < X -> WPS | Func2 >;
// embedding of X in P(1,1,1,3)
X2 := Image(pi); // model in P(1,1,1,3),
    //equation is given by w^2=f(x,y,z) with f of degree 6
pires := map < X -> X2 | Func2 >; // restricted pi to image

"Model of X as double cover: ";
X2;

"where the isomorphism is given by the map: ";
pires;

/*
We can calculate the inverse with
IsInvertible(pires); (takes ~2mpmin)
```

```
But this gives some ugly polynomials, and we can do better!

Constructing inverse:
First define involution on model X2
*/

iX2 := map< X2 -> X2 | [x,y,z,-w]>;

// Then take divisor -H+5C
E := (pires*iX2)(Scheme(X,w));

// This divisor embeds X2 back to P3
LSys := LinearSystem(LinearSystem(WPS,5),E);
prepiinv := map < X2 -> P3 | Sections(LSys) >;

// Now we make sure that the image in P3 is equal to X
comp := pires*prepiinv;
ev2 := [comp(Pts[i]) : i in [1,2,3,4,6]];
M2 := TranslationOfSimplex(P3,[Pts[i] :
        i in [1,2,3,4,6]]);
N2 := TranslationOfSimplex(P3,ev2);
piinv := Expand(prepiinv*N2^(-1)*M2);

"with inverse: ";
piinv;

/*
In particular piinv, will be the inverse of pi;
Magma cannot check this, because calculating the image takes
to much time. But we can check it by calculating what it does
on the generic point:
*/

// creating function field of X
RR<xx,yy,zz,ww>:=PolynomialRing(Rationals(),4);
hh:=-xx^3*ww - xx^2*zz*ww + xx^2*ww^2 - xx*yy^3
      + xx*zz^3 - yy^2*ww^2 + zz^4 + zz*ww^3;
II:=ideal< RR | hh >;
RR:=RR/II;
K<x0,y0,z0,w0>:=FieldOfFractions(RR);

// basechange to function field and generic point
XK:=BaseChange(X,K);
P3K<xx,yy,zz,ww>:=AmbientSpace(XK);
genP:=XK![x0,y0,z0,w0];

// checking if it is the identity on generic point
d1:=DefiningEquations(pires);
d2:=DefiningEquations(piinv);
PP:=[Evaluate(d2[i],d1): i in [1..4]];
assert genP eq
```

```
288         XK![Evaluate(PP[i],[x0,y0,z0,w0]):i in [1..4]];
289
290 i2 := pires*iX2*piinv;
291 assert i1 eq i2;
292
293 /*
294 Hence, the involution is given by composing the three maps
295         pi, iX2 and the inverse piinv:
296 So this gives the involution as a composition of much easier
297 maps. This also shows that indeed the involution i1 is well
298 defined and its own inverse.
299 */
300
301 // Branch locus is given by Z:
302 Zp := Scheme(P2,Evaluate(Equation(X2)
303         ,[s,t,u,0]));
304 Z := Zp @@ phi; // branched curve in P3;
305
306 // -----------------------------------------------------
307
308 // Model in P5
309
310 LHD := RiemannRochBasis(3*DivH-DivC);
311 // equations for map to P5
312 P5<x0,x1,x2,x3,x4,x5> := ProjectiveSpace(Q,5);
313 chi := map < X -> P5 | LHD >;
314 // embedding of X in P5
315
316
317 assert IsIrreducible(X) eq true; // needs to be runned to find
318 X8p := Image(chi); // the image of chi
319 eq8 := Equations(X8p);
320 for pol in eq8 do
321   if Degree(pol) ne 2 then
322     eq8:=Exclude(eq8,pol);
323   end if;
324 end for;
325 X8 := Scheme(P5,eq8);
326 assert X8 eq X8p;
327
328 "Model as intersection of quadrics: ";
329 X8; // model in P5
330 assert chi(Scheme(X,eqD)) eq Scheme(P5,[x1,x2,x4,x5]); // streched twisted cubic cur
331
332 reschi := map < X -> X8 | LHD >;
333 chiinv := IsInvertible(reschi);
334
335 "where the isomorphism is given by ";
336 reschi;
337
338 /* ---------------------------------------------------------------------
```

```
Output file:

The K3 surface X:
Scheme over Rational Field defined by
-x^3*w - x^2*z*w + x^2*w^2 - x*y^3 + x*z^3 - y^2*w^2 + z^4 + z*w^3
The involution on X is defined by the polynomials:
[
    x^4*y^5 - 5*x^3*y^4*z^2 + 2*x^2*y^5*w^2 + 10*x^2*y^3*z^4 - 8*x^2*y^3*z*w^3 - x^2
        6*x^2*y*z^2*w^4 - x^2*z^6*w + x^2*z^2*w^5 - x^2*z*w^6 + x*y^8 - 7*x*y^6*z*w
        x*y^3*z*w^4 - 10*x*y^2*z^6 - 6*x*y^2*z^3*w^3 - x*z^7*w + x*z^6*w^2 - 3*x*z^4
        y^3*z^6 - 8*y^3*z^4*w^2 + y^2*z^4*w^3 + 2*y^2*z*w^6 + 5*y*z^8 + 6*y*z^5*w^3
    x^3*y^6 + x^3*y^5*z + x^2*y^7 + x^2*y^6*w - 4*x^2*y^5*z^2 - 5*x^2*y^4*z^3 - x^2*
        2*x^2*y^4*w^3 - 3*x^2*y^3*z^3*w + 9*x^2*y^3*z^2*w^2 + 2*x^2*y^3*z*w^3 - 2*x^
        3*x^2*y^2*z^3*w^2 + 7*x^2*y^2*z^2*w^3 - 4*x^2*y^2*z*w^4 - 3*x^2*y*z^4*w^2 +
        x^2*y*z*w^5 + x^2*y*w^6 - 2*x^2*z^5*w^2 + x^2*z^3*w^4 - x^2*z^2*w^5 - x*y^7*
        x*y^6*z*w + 2*x*y^6*w^2 - 3*x*y^5*z^2*w + 4*x*y^5*z*w^2 + x*y^5*w^3 + 5*x*y^
        2*x*y^4*z*w^3 - x*y^4*w^4 + 10*x*y^3*z^5 - x*y^3*z^3*w^2 + x*y^3*z^2*w^3 - x
        5*x*y^2*z^5*w - 8*x*y^2*z^4*w^2 + 5*x*y*z^6*w + x*y*z^3*w^4 + x*y*z*w^6 - x*
        y^9 - 5*y^7*z*w - y^6*z*w^2 + y^6*w^3 - y^5*z^4 + 5*y^5*z^2*w^2 - y^5*z*w^3
        2*y^4*z^2*w^3 + 3*y^4*z*w^4 + 3*y^3*z^6 + 3*y^3*z^5*w - y^3*z^4*w^2 - 9*y^3*
        2*y^2*z^7 + 3*y^2*z^6*w - 4*y^2*z^5*w^2 + 3*y^2*z^3*w^4 - 3*y^2*z^2*w^5 + 7*
        9*y*z^4*w^4 - y*z^2*w^6 + y*z*w^7 + z^8*w - z^6*w^3 - z^3*w^6,
    -x^4*y^5 + 5*x^3*y^4*z^2 + x^2*y^7 - x^2*y^5*w^2 + 2*x^2*y^4*z^2*w + 2*x^2*y^4*z
        10*x^2*y^3*z^4 - 4*x^2*y^3*z^3*w + 7*x^2*y^3*z^2*w^2 - 3*x^2*y^3*z*w^3 - 6*x
        x^2*y^2*z^2*w^3 + x^2*y^2*z*w^4 + x^2*y^2*w^5 - 4*x^2*y*z^5*w + 2*x^2*y*z^3*
        - x^2*z^2*w^5 - x*y^7*z - x*y^7*w - 4*x*y^6*z^2 + 3*x*y^6*z*w + x*y^6*w^2 -
        x*y^5*w^3 - x*y^4*z^3*w + 2*x*y^4*z^2*w^2 - x*y^4*z*w^3 - x*y^4*w^4 - 9*x*y^
        10*x*y^2*z^6 + x*y^2*z^3*w^3 + x*y^2*z*w^5 - x*y*z^5*w^2 + 2*x*y*z^3*w^4 + x
        y^6*z*w^2 + y^6*w^3 - 4*y^5*z^2*w^2 + 3*y^5*z*w^3 - y^4*z^5 - y^4*z^4*w - 3*
        + 4*y^3*z^6 - 3*y^3*z^5*w + 4*y^3*z^3*w^3 - y^3*z^2*w^4 + 6*y^2*z^7 + 3*y^2*
        - y^2*z^2*w^5 + y^2*z*w^6 - y*z^8 - 2*y*z^6*w^2 - 2*y*z^3*w^5 - z^7*w^2,
    -x^3*y^6 - x^2*y^6*z + x^2*y^6*w + 3*x^2*y^5*z^2 + x^2*y^5*w^2 - 3*x^2*y^4*z^2*w
        3*x^2*y^3*z^2*w^2 + 3*x^2*y^3*z*w^3 - x^2*y^3*w^4 - 7*x^2*y^2*z^3*w^2 + 14*x
        2*x^2*y^2*w^5 - 9*x^2*y*z^4*w^2 - 2*x^2*y*z^3*w^3 + 8*x^2*y*z^2*w^4 - 6*x^2*
        2*x^2*z^3*w^4 + x^2*z^2*w^5 - x^2*z*w^6 + x^2*w^7 - 3*x*y^7*z - 3*x*y^6*z*w
        2*x*y^4*z^2*w^2 + 6*x*y^4*z*w^3 + x*y^4*w^4 - 12*x*y^3*z^3*w^2 + 8*x*y^3*z*w
        x*y^2*z^3*w^3 - x*y^2*z*w^5 - x*y^2*w^6 - 4*x*y*z^5*w^2 - 3*x*y*z^4*w^3 - 6*
        x*z^3*w^5 + x*z*w^7 - y^8*w + y^6*z^3 + 4*y^6*z*w^2 - 3*y^5*z*w^3 - 2*y^4*z^
        y^4*z*w^4 + y^4*w^5 - y^3*z^6 + 3*y^3*z^5*w + y^3*z^2*w^4 + 6*y^3*z*w^5 + 12
        y^2*z^4*w^3 + 10*y^2*z^3*w^4 + y^2*z^2*w^5 - y^2*w^7 + 2*y*z^6*w^2 - 6*y*z^5
        2*z^7*w^2 - z^6*w^3 - z^5*w^4 + 3*z^4*w^5 - z^2*w^7 + z*w^8
]
Model of X as double cover:
Scheme over Rational Field defined by
xx^6 - 2*xx^5*yy + xx^4*yy^2 - 2*xx^3*yy^3 - 2*xx^2*yy^4 + yy^6 + 4*xx^3*yy^2*zz - 4
    4*xx*yy^3*zz^2 + 4*yy^4*zz^2 - 4*yy^2*zz^4 - 4*xx*zz^5 - ww^2
where the isomorphism is given by the map:
Mapping from: Sch: X to Sch: X2
with equations :
x*y - z^2
x*w - y*z
```

```
y^2 - z*w
x^3*y^3 + x^3*y^2*w - x^3*w^3 + x^2*y^3*z - 3*x^2*y^2*z^2 - 2*x^2*y*z^2*w + 3*x^2*y*
    3*x*y^2*z^2*w + 2*x*y^2*w^3 + 3*x*y*z^4 - x*z^4*w + 2*x*z^3*w^2 - 2*x*z*w^4 + 2*
    5*y*z^5 + 6*y*z^2*w^3 + z^6
with inverse:
Mapping from: Sch: X2 to Prj: P3
with equations :
-1/2*xx^3*yy^2 - 1/2*xx^2*yy^3 + 1/2*yy^5 - 1/2*xx^4*zz + 1/2*xx^3*yy*zz + 1/2*xx*yy
xx*yy^4 - xx*yy^3*zz - 1/2*xx^3*zz^2 + 1/2*xx^2*yy*zz^2 + 1/2*yy^3*zz^2 + xx*yy*zz^3
xx^2*yy^3 + 1/2*xx^3*yy*zz - 1/2*xx^2*yy^2*zz - 1/2*yy^4*zz + xx^2*zz^3 + 1/2*yy*zz*
yy^5 + xx*yy^3*zz - yy^4*zz + yy^2*zz^3 + xx*zz^4
Model as intersection of quadrics:
Scheme over Rational Field defined by
x1^2 + x0*x2 + x2*x3 + x1*x4 - x2*x4 - x3*x4 - x1*x5,
x2^2 + x2*x3 + x1*x4 + x4^2 - x0*x5 - x3*x5,
x1*x3 - x0*x4 - x3*x4 - x5^2
where the isomorphism is given by
Mapping from: Sch: X to Sch: X8
with equations :
x*y^2 + x^2*z - y^2*z - z^3 + y*w^2 - w^3
-x*y*z + x^2*w
-y^2*z + x*y*w
y^2*z + x*z^2 + z^3 + w^3
-y*z^2 + x*z*w
-y*z*w + x*w^2
and inverse
x1
x2
x4
x5

*/
```