Projective Geometry and Homological Algebra

David Eisenbud*

We provide an introduction to many of the homological commands in *Macaulay 2* (modules, free resolutions, Ext and Tor...) by means of examples showing how to use homological tools to study projective varieties.

In this chapter we will illustrate how one can manipulate projective varieties and sheaves, using the rich collection of tools *Macaulay 2* provides. One of our goals is to show how homological methods can be effective in solving concrete geometric problems.

The first four sections can be read by anyone who knows about projective varieties at the level of a first graduate course and knows the definitions of Ext and Tor. The last section assumes that the reader is familiar with the theory of curves and surfaces roughly at the level of the books of Hartshorne [7] and Harris [6].

We will work with projective schemes over a field `kk`. *Macaulay 2* can work over any finite field of characteristic at most 32749, and also a variety of fields in characteristic 0 (except for the primary decomposition commands, which at this writing are still restricted to positive characteristics). Our main interest is in geometry over an algebraically closed field of characteristic 0. Nevertheless, it is most convenient to work over a large prime field. It is known that the intermediate results in Gröbner basis computations (as in the Euclidean Algorithm computations they generalize) often involve coefficients far larger than those in the input data, so that work in characteristic zero essentially requires infinite precision arithmetic, a significant additional overhead. If we work over a finite field where the scalars can be represented in one machine word, we avoid this coefficient explosion. Experience with the sort of computations we will be doing shows that working over $\mathbb{Z}/p$, where p is a moderately large prime, gives results identical to the results we would get in characteristic 0. Of course one still has to be careful about the fact that our fields are not algebraically closed, especially when using primary decomposition. The largest prime p we can work with being 32749, we choose the field $\mathbf{Z}/32749$. The name of the *Macaulay 2* constant representing the integers is `ZZ`, and by analogy we will call our field `kk`:

```
i1 : kk = ZZ/32749

o1 = kk

o1 : QuotientRing
```

In *Macaulay 2* we will represent projective space $\mathbb{P}^n$ by its homogeneous coordinate ring $\texttt{ringPn} = \texttt{kk}[x_0, \ldots, x_n]$. A projective scheme X in $\mathbb{P}^n$ may

* Supported by the NSF.

be most conveniently represented, depending on the situation, by its homogeneous ideal `idealX` or its homogeneous coordinate ring, represented either as a ring `ringPn/idealX` or as a module `OX` over `ringPn`. Coherent sheaves on the projective space, or on its subvarieties, will be represented by finitely generated graded modules over `ringPn`, using the Serre correspondence. For example, the structure sheaf $\mathcal{O}_X$ of the subvariety X would be represented by the module `ringPn^1/idealX`; here `ringPn^1` denotes the free module of rank one over the ring `ringPn`.

1 The Twisted Cubic

As a first illustration, we give three constructions of the twisted cubic curve in $\mathbb{P}^3$. We represent $\mathbb{P}^3$ by

```
i2 : ringP3 = kk[x_0..x_3]

o2 = ringP3

o2 : PolynomialRing
```

The twisted cubic is the image of the map $\mathbb{P}^1 \to \mathbb{P}^3$ sending a point with homogeneous coordinates (s, t) to the point with homogeneous coordinates (s^3, s^2t, st^2, t^3). We can compute its relations directly with

```
i3 : ringP1 = kk[s,t]

o3 = ringP1

o3 : PolynomialRing

i4 : cubicMap = map(ringP1,ringP3,{s^3, s^2*t, s*t^2, t^3})

                          3   2       2   3
o4 = map(ringP1,ringP3,{s  , s t, s*t  , t })

o4 : RingMap ringP1 <--- ringP3


i5 : idealCubic = kernel cubicMap

             2                     2
o5 = ideal (x  - x x , x x  - x x , x  - x x )
             2    1 3   1 2    0 3   1    0 2

o5 : Ideal of ringP3
```

We could also use *Macaulay 2*'s built-in facility, and say

```
i6 : idealCubic2 = monomialCurveIdeal(ringP3,{1,2,3})

                          2          2
o6 = ideal (x x  - x x , x  - x x , x  - x x )
             1 2    0 3   2    1 3   1    0 2

o6 : Ideal of ringP3
```

which uses precisely the same method.

Of course we might remember that the ideal of the twisted cubic is generated by the 2×2 minors of the matrix

$$\begin{pmatrix} x_0 & x_1 & x_2 \\ x_1 & x_2 & x_3 \end{pmatrix},$$

which we can realize with the commands

```
i7 : M = matrix{{x_0,x_1,x_2},{x_1,x_2,x_3}}

o7 = | x_0 x_1 x_2 |
     | x_1 x_2 x_3 |

                     2            3
o7 : Matrix ringP3  <--- ringP3


i8 : idealCubic3 = minors(2, M)

                  2                           2
o8 = ideal (- x  + x x , - x x  + x x , - x  + x x )
                  1    0 2     1 2    0 3     2    1 3

o8 : Ideal of ringP3
```

We can get some useful information about the ideal `idealCubic` with

```
i9 : codim idealCubic

o9 = 2


i10 : degree idealCubic

o10 = 3
```

This shows that we do indeed have a cubic curve. Note that the command

```
i11 : dim idealCubic

o11 = 2
```

gives 2, not 1; it represents the dimension of the ideal in `ringP3`, the dimension of the affine cone over the curve.

We can easily assure ourselves that these ideals are the same. For example, to see whether the ideal `idealCubic` is contained in the ideal of minors of `M`, we can reduce the former modulo the latter, and see whether we get zero. The reduction operator `%` takes two maps with the same target as its arguments, so we must replace each ideal by a matrix whose entries generate it. This is done by the function `gens` as in

```
i12 : gens idealCubic

o12 = | x_2^2-x_1x_3 x_1x_2-x_0x_3 x_1^2-x_0x_2 |

                      1            3
o12 : Matrix ringP3  <--- ringP3
```

Thus for one of the inclusions we check

```
i13 : 0 == (gens idealCubic)%(gens idealCubic3)

o13 = true
```

Both inclusions can be checked automatically in this way with

```
i14 : idealCubic == idealCubic3

o14 = true
```

2 The Cotangent Bundle of $\mathbb{P}^3$

Many invariants of varieties are defined in terms of their tangent and cotangent bundles. We identify a bundle with its sheaf of sections, which is locally free. Any coherent locally free sheaf arises this way. (One can also regard a bundle as a variety in its own right, but this view is used in algebraic geometry more rarely.) In this section and the next we construct the cotangent bundle $\Omega_{\mathbb{P}^3}$ of $\mathbb{P}^3$ and its restriction to the twisted cubic above.

Consulting Hartshorne [7, Theorem II.8.13], we find that the cotangent bundle to $\mathbb{P}^n$ can be described by the *cotangent sequence*:

$$0 \longrightarrow \Omega_{\mathbb{P}^n} \longrightarrow \mathcal{O}_{\mathbb{P}^n}(-1)^{n+1} \xrightarrow{f} \mathcal{O}_{\mathbb{P}^n} \longrightarrow 0$$

where f is defined by the matrix of variables $(x_0, \ldots, x_n)$. We can translate this description directly into the language of *Macaulay 2*, here in the case $n = 3$:

```
i15 : f = vars ringP3

o15 = | x_0 x_1 x_2 x_3 |

                     1            4
o15 : Matrix ringP3  <--- ringP3


i16 : OmegaP3 = kernel f

o16 = image {1} | 0    0    0    -x_1 -x_2 -x_3 |
            {1} | 0    -x_2 -x_3 x_0  0    0    |
            {1} | -x_3 x_1  0    0    x_0  0    |
            {1} | x_2  0    x_1  0    0    x_0  |

                                             4
o16 : ringP3-module, submodule of ringP3
```

Note that the module which we specified as a kernel is now given as the image of a matrix. We can recover this matrix with

```
i17 : g=generators OmegaP3

o17 = {1} | 0    0    0    -x_1 -x_2 -x_3 |
      {1} | 0    -x_2 -x_3 x_0  0    0    |
      {1} | -x_3 x_1  0    0    x_0  0    |
      {1} | x_2  0    x_1  0    0    x_0  |

                     4            6
o17 : Matrix ringP3  <--- ringP3
```

and we could correspondingly write

```
i18 : OmegaP3=image g

o18 = image {1} | 0    0    0    -x_1 -x_2 -x_3 |
            {1} | 0    -x_2 -x_3 x_0  0    0    |
            {1} | -x_3 x_1  0    0    x_0  0    |
            {1} | x_2  0    x_1  0    0    x_0  |

                                        4
o18 : ringP3-module, submodule of ringP3
```

An even more elementary way to give a module is by generators and relations, and we can see this "free presentation" too with

```
i19 : presentation OmegaP3

o19 = {2} | x_1  0    0    x_0  |
      {2} | x_3  x_0  0    0    |
      {2} | -x_2 0    x_0  0    |
      {2} | 0    x_2  x_3  0    |
      {2} | 0    -x_1 0    x_3  |
      {2} | 0    0    -x_1 -x_2 |

                    6          4
o19 : Matrix ringP3  <--- ringP3
```

The astute reader will have noticed that we have just been computing the first few terms in the free resolution of the cokernel of the map of free modules corresponding to `f`. We could see the whole resolution at once with

```
i20 : G = res coker f

            1           4           6           4           1
o20 = ringP3  <-- ringP3  <-- ringP3  <-- ringP3  <-- ringP3  <-- 0

      0           1           2           3           4           5

o20 : ChainComplex
```

and then see all the matrices in the resolution with

```
i21 : G.dd

                1                              4
o21 = 0 : ringP3  <---------------------- ringP3  : 1
                    | x_0 x_1 x_2 x_3 |

                4                                                  6
      1 : ringP3  <---------------------------------------- ringP3  : 2
                    {1} | -x_1 -x_2 0    -x_3 0    0    |
                    {1} | x_0  0    -x_2 0    -x_3 0    |
                    {1} | 0    x_0  x_1  0    0    -x_3 |
                    {1} | 0    0    0    x_0  x_1  x_2  |

                6                                        4
      2 : ringP3  <------------------------------ ringP3  : 3
                    {2} | x_2  x_3  0    0    |
                    {2} | -x_1 0    x_3  0    |
                    {2} | x_0  0    0    x_3  |
                    {2} | 0    -x_1 -x_2 0    |
                    {2} | 0    x_0  0    -x_2 |
                    {2} | 0    0    x_0  x_1  |
```

```
                  4                          1
         3 : ringP3  <---------------- ringP3  : 4
                        {3} | -x_3 |
                        {3} | x_2  |
                        {3} | -x_1 |
                        {3} | x_0  |

                  1
         4 : ringP3  <----- 0 : 5
                        0

o21 : ChainComplexMap
```

or just one of them, say the second, with

```
i22 : G.dd_2

o22 = {1} | -x_1 -x_2 0    -x_3 0    0    |
      {1} | x_0  0    -x_2 0    -x_3 0    |
      {1} | 0    x_0  x_1  0    0    -x_3 |
      {1} | 0    0    0    x_0  x_1  x_2  |

                     4            6
o22 : Matrix ringP3  <--- ringP3
```

Note that this matrix does not look exactly the same as the matrix produced by computing the kernel of `f`. This is because when *Macaulay 2* is asked to compute a whole resolution, it does not do the "obvious" thing and compute kernels over and over; it defaults to a more efficient algorithm, first proposed by Frank Schreyer [10, Appendix].

Any graded map of free modules, such as a map in a graded free resolution of a graded module, comes with some numerical data: the degrees of the generators of the source and target free modules. We can extract this information one module at a time with the command `degrees`, as in

```
i23 : degrees source G.dd_2

o23 = {{2}, {2}, {2}, {2}, {2}, {2}}

o23 : List

i24 : degrees target G.dd_2

o24 = {{1}, {1}, {1}, {1}}

o24 : List
```

Macaulay 2 has a more convenient mechanism for examining this numerical data, which we take time out to explain. First, for the resolution just computed, we can call

```
i25 : betti G

o25 = total: 1 4 6 4 1
          0: 1 4 6 4 1
```

The diagram shows the degrees of the generators of each free module in the resolution in coded form. To understand the code, it may be helpful to look at a less symmetric example, say the free resolution of `ringP3^1/I` where I is the ideal generated by the minors of the following 2×4 matrix.

```
i26 : m = matrix{{x_0^3, x_1^2, x_2,x_3},{x_1^3,x_2^2,x_3,0}}

o26 = | x_0^3 x_1^2 x_2 x_3 |
      | x_1^3 x_2^2 x_3 0   |

                   2             4
o26 : Matrix ringP3  <--- ringP3
```

We do this with

```
i27 : I = minors(2,m)

                 5    3 2     3      3       3    2      3      2      2
o27 = ideal (- x  + x x , - x x  + x x , - x  + x x , -x x , -x x , -x )
                 1    0 2     1 2    0 3     2    1 3    1 3    2 3    3

o27 : Ideal of ringP3

i28 : F = res(ringP3^1/I)

             1            6            8            3
o28 = ringP3  <-- ringP3  <-- ringP3  <-- ringP3  <-- 0

      0            1            2            3            4

o28 : ChainComplex

i29 : betti F

o29 = total: 1 6 8 3
          0: 1 . . .
          1: . 1 . .
          2: . 2 2 .
          3: . 2 2 .
          4: . 1 4 3
```

The resulting Betti diagram should be interpreted as follows. First, the maps go from right to left, so the beginning of the resolution is on the left. The given Betti diagram thus corresponds to an exact sequence of graded free modules

$$F_0 \longleftarrow F_1 \longleftarrow F_2 \longleftarrow F_3 \longleftarrow 0.$$

The top row of the diagram, 1,6,8,3, shows the ranks of the free modules F_i in the resolution. For example the 1 on the left means that F_0 has rank 1 (and, indeed, the module `ringP3^1/I` we are resolving is cyclic). The 6 shows that the rank of F_1 is 6, or equivalently that the ideal `I` is minimally generated by 6 elements—in this case the $6 = \binom{4}{2}$ minors of size 2 of the 2×4 matrix m.

The first column of the diagram shows degrees. The successive columns indicate how many generators of each degree occur in the successive F_i. The free module F_0 has a single generator in degree 0, and this is the significance of the second column. Note that F_1 could not have any generators of degree less than or equal to zero, because the resolution is minimal! Thus for compactness, the diagram is skewed: in each successive column the places correspond to larger degrees. More precisely, a number a occurring opposite the degree indication "`i:`" in the column corresponding to F_j signifies that F_j has a generators in degree $i + j$. Thus for example the 1 in the third column opposite the one on the left corresponds to a generator of degree 2 in the free module F_1; and altogether F_1 has one generator of degree 2, two generators of degree 3, two of degree 4 and one of degree 5.

Returning to the diagram

```
i30 : betti G

o30 = total: 1 4 6 4 1
          0: 1 4 6 4 1
```

we see that the successive free modules of `G` are each generated in degree 1 higher than the previous one; that is, the matrices in `G.dd` all have linear entries, as we have already seen.

3 The Cotangent Bundle of a Projective Variety

It is easy to construct the cotangent bundle Ω_X of a projective variety X starting from the cotangent bundle of the ambient projective space. We use the *conormal sequence* (Hartshorne [7, Proposition II.8.12] or Eisenbud [4, Proposition 16.3]). Writing I for the ideal of a variety X in $\mathbb{P}^n$ there is an exact sequence of sheaves

$$I \xrightarrow{\delta} \Omega_{\mathbb{P}^n} \otimes \mathcal{O}_X \longrightarrow \Omega_X \longrightarrow 0$$

where the map δ takes a function f to the element $df \otimes 1$. If I is generated by forms $f_1, \ldots, f_m$ then δ is represented by the Jacobian matrix (df_i/dx_j).

First of all, we must compute a module corresponding to $\Omega_{\mathbb{P}^n} \otimes \mathcal{O}_X$, the restriction of the sheaf $\Omega_{\mathbb{P}^n}$ to X. The simplest approach would be to take the tensor product of graded modules representing $\Omega_{\mathbb{P}^n}$ and $\mathcal{O}_X$. The result would represent the right sheaf, but would not be the module of twisted global sections of $\Omega_{\mathbb{P}^n} \otimes \mathcal{O}_X$ (the unique module of depth two representing the sheaf). This would make further computations less efficient.

Thus we take a different approach: since the cotangent sequence given in the previous section is a sequence of locally free sheaves, it is locally split, and thus remains exact when tensored by $\mathcal{O}_X$. Consequently $\Omega_{\mathbb{P}^n} \otimes \mathcal{O}_X$ is also represented by the kernel of the map $f \otimes \mathcal{O}_X$, where f is the map used in the definition of the cotangent bundle of $\mathbb{P}^n$. In *Macaulay 2*, working on $\mathbb{P}^3$, with X the twisted cubic, we can translate this into

```
i31 : OmegaP3res = kernel (f ** (ringP3^1/idealCubic))

o31 = subquotient ({1} | -x_3 0    0    -x_2 -x_3 0    -x_1 -x_2 -x_3  ...
                   {1} | x_2  -x_3 0    x_1  0    -x_3 x_0  0    0     ...
                   {1} | 0    x_2  -x_3 0    x_1  0    0    x_0  0     ...
                   {1} | 0    0    x_2  0    0    x_1  0    0    x_0   ...

                                                 4
o31 : ringP3-module, subquotient of ringP3
```

(The operator `**` is *Macaulay 2*'s symbol for tensor product.) Since the map is a map between free modules over `ringP3/idealCubic`, the kernel has depth (at least) two.

Next, we form the Jacobian matrix of the generators of `idealCubic`, which represents a map from this ideal to the free module `ringP3^4`.

```
i32 : delta1 = jacobian idealCubic

o32 = {1} | 0    -x_3 -x_2 |
      {1} | -x_3 x_2  2x_1 |
      {1} | 2x_2 x_1  -x_0 |
      {1} | -x_1 -x_0 0    |

                     4            3
o32 : Matrix ringP3  <--- ringP3
```

We need to make this into a map to `OmegaP3res`, which as defined is a subquotient of `ringP3^4`. To this end we must first express the image of `delta1` in terms of the generators of `OmegaP3res`. The division command `//` does this with

```
i33 : delta2 = delta1 // (gens OmegaP3res)

o33 = {2} | 0  1  0  |
      {2} | 2  0  0  |
      {2} | 0  0  0  |
      {2} | 0  0  2  |
      {2} | 0  1  0  |
      {2} | -1 0  0  |
      {2} | 0  0  0  |
      {2} | 0  0  -1 |
      {2} | 0  -1 0  |

                     9            3
o33 : Matrix ringP3  <--- ringP3
```

Once this is done we can use this matrix to form the necessary map $\delta : I \to \Omega_{\mathbb{P}^3} \otimes \mathcal{O}_X$:

```
i34 : delta = map(OmegaP3res, module idealCubic, delta2)

o34 = {2} | 0  1  0  |
      {2} | 2  0  0  |
      {2} | 0  0  0  |
      {2} | 0  0  2  |
      {2} | 0  1  0  |
      {2} | -1 0  0  |
      {2} | 0  0  0  |
      {2} | 0  0  -1 |
      {2} | 0  -1 0  |

o34 : Matrix
```

A minimal free presentation of Ω_X — or rather of one module over `ringP3` that represents it — can be obtained with

```
i35 : OmegaCubic = prune coker delta

o35 = cokernel {2} | -10917x_3 0    -10917x_3 x_2       0         0      ...
               {2} | 0         0    x_2       0         16374x_3  0      ...
               {2} | 0         -x_3 0         16373x_3  0         x_2    ...
               {2} | x_3       x_2  0         0         0         0      ...
               {2} | 0         0    0         0         0         -2x_3  ...
               {2} | 0         0    0         0         x_2       0      ...

                                            6
o35 : ringP3-module, quotient of ringP3
```

We have used the function `prune` to compute minimal presentation matrices; these often make subsequent computations faster, and also allow us to inspect the final answer more easily.

The module `OmegaCubic` represents the sheaf Ω_X, where X is the cubic, but it is not the simplest possibility. A better representative is the graded module $\oplus_{d\in\mathbb{Z}}\mathrm{H}^0(\Omega_X(d))$. We can at least find a minimal presentation of the submodule $\oplus_{d\geq 0}\mathrm{H}^0(\Omega_X(d))$ with

```
i36 : prune HH^0((sheaf OmegaCubic)(>=0))

o36 = cokernel {1} | 16374x_3 16374x_2 16374x_1 |
               {1} | x_2      x_1      x_0      |

                                      2
o36 : ringP3-module, quotient of ringP3
```

The large coefficients appearing in the matrix arise in finite characteristic as the result of chance division by small integers. We see from the degrees labeling the rows of the matrix in the output of this command that the generators of the submodule are in degree 1, so in particular $\mathrm{H}^0(\Omega_X) = 0$. It follows that that $\mathrm{H}^0(\Omega_X(d)) = 0$ for all $d \leq 0$, so the submodule we computed was actually the whole module that we wanted! (If this had not been the case we could have tried `HH^0((sheaf OmegaCubic)(>=d))` to compute the cohomology of all the twists greater than a given negative integer d, or simply used the submodule we had already computed, since it also represents the sheaf Ω_X.)

The sequence of commands we have used to construct the cotangent sheaf can be obtained also with the following built-in commands.

```
i37 : Cubic = Proj(ringP3/idealCubic)

o37 = Cubic

o37 : ProjectiveVariety

i38 : cotangentSheaf Cubic

o38 = cokernel {1} | x_2  x_1  x_0  |
               {1} | -x_3 -x_2 -x_1 |

                                                 2
o38 : coherent sheaf on Cubic, quotient of OO      (-1)
                                             Cubic
```

Since X is a smooth curve, its cotangent bundle is equal to its *canonical bundle*, and also to its *dualizing sheaf* (see Hartshorne [7, sections II.8 and III.7] for definitions). We will see another (generally more efficient) method of computing this dualizing sheaf by using `Ext` and duality theory.

4 Intersections by Serre's Method

To introduce homological algebra in a simple geometric context, consider the problem of computing the intersection multiplicities of two varieties X and Y in $\mathbb{P}^n$, assuming for simplicity that $\dim X + \dim Y = n$ and that the two meet in a zero-dimensional scheme. Beginning in the 19th century, many people struggled to make a definition of local intersection multiplicity that would make *Bézout's Theorem* true: the product of the degrees of X and Y should be the number of points of intersection, each counted with its local intersection multiplicity (multiplied by the degree of the point, if the point is not rational over the ground field). In the simplest case, where the two varieties are Cohen-Macaulay, the right answer is that a point p should count with multiplicity equal to the length of the local ring $\mathcal{O}_{X,p} \otimes_{\mathcal{O}_{\mathbb{P}^n,p}} \mathcal{O}_{Y,p}$, and at first it was naively assumed that this would be the right answer in general.

Here is a famous example in $\mathbf{P}^4$ showing that the naive value can be wrong: in it, the scheme X is a 2-plane and the scheme $Y = L_1 \cup L_2$ is the union of two 2-planes. The planes L_1 and L_2 meet at just one point p, and we assume that X passes through p as well, and is general enough so that it meets Y only in p. Since $\text{degree}(X) = 1, \text{degree}(Y) = 2$, Bézout's Theorem requires that the multiplicity of the intersection at p should be 2. However, we have:

```
i39 : ringP4 = kk[x_0..x_4]

o39 = ringP4

o39 : PolynomialRing

i40 : idealX = ideal(x_1+x_3, x_2+x_4)

o40 = ideal (x  + x , x  + x )
              1    3   2    4

o40 : Ideal of ringP4

i41 : idealL1 = ideal(x_1,x_2)

o41 = ideal (x , x )
              1   2

o41 : Ideal of ringP4

i42 : idealL2 = ideal(x_3,x_4)

o42 = ideal (x , x )
              3   4

o42 : Ideal of ringP4

i43 : idealY = intersect(idealL1,idealL2)

o43 = ideal (x x , x x , x x , x x )
              2 4   1 4   2 3   1 3

o43 : Ideal of ringP4

i44 : degree(idealX+idealY)

o44 = 3
```

That is, the length of $\mathcal{O}_{X,p} \otimes_{\mathcal{O}_{\mathbb{P}^n,p}} \mathcal{O}_{Y,p}$ is 3 rather than 2. (We can do this computation without first passing to local rings because there is only one point of intersection, and because all the constructions we are using commute with localization.)

It was the happy discovery of Jean-Pierre Serre [11, V.B.3] that the naive measure of intersection multiplicity can be fixed in a simple way that works for all intersections in smooth varieties. One simply replaces the length of the tensor product

$$\mathcal{O}_{X,p} \otimes_{\mathcal{O}_{\mathbb{P}^n,p}} \mathcal{O}_{Y,p} = \mathrm{Tor}_0^{\mathcal{O}_{\mathbb{P}^n,p}}(\mathcal{O}_{X,p}, \mathcal{O}_{Y,p})$$

with the alternating sum of the Tor functors

$$\sum_i (-1)^i \,\mathrm{length}\, \mathrm{Tor}_i^{\mathcal{O}_{\mathbb{P}^n,p}}(\mathcal{O}_{X,p}, \mathcal{O}_{Y,p}).$$

In *Macaulay 2* we can proceed as follows:

```
i45 : degree Tor_0(ringP4^1/idealX, ringP4^1/idealY)

o45 = 3

i46 : degree Tor_1(ringP4^1/idealX, ringP4^1/idealY)

o46 = 1

i47 : degree Tor_2(ringP4^1/idealX, ringP4^1/idealY)

o47 = 0
```

The other Tor's are 0 because the projective dimension of `ringP4^1/idealX` is only two, as we see from

```
i48 : res (ringP4^1/idealX)

            1           2           1
o48 = ringP4  <-- ringP4  <-- ringP4  <-- 0

      0           1           2           3

o48 : ChainComplex
```

Thus, indeed, the alternating sum is 2, and Bézout's Theorem is upheld.

5 A Mystery Variety in $\mathbb{P}^3$

In the file `mystery.m2` is a function called `mystery` that will compute the ideal of a subvariety X of $\mathbb{P}^3$. We'll reveal what it does at the end of the chapter. Let's run it.

```
i49 : ringP3 = kk[x_0..x_3];

i50 : load "mystery.m2"

i51 : idealX = mystery ringP3

              4       2       2        2 2   2 2           2          ...
o51 = ideal (x  - 2x x x  - x x x  + x x , x x  - 10915x x x  - 10917x ...
              1     0 1 3    1 2 3    0 3   0 1         0 1 2          ...

o51 : Ideal of ringP3
```

We can't see all the generators of the ideal; the same file contains a function `prettyPrint` which will display the generators visibly.

```
i52 : prettyPrint gens idealX
x_1^4-2*x_0*x_1^2*x_3-x_1^2*x_2*x_3+x_0^2*x_3^2,
x_0^2*x_1^2-10915*x_0*x_1^2*x_2-10917*x_0^3*x_3+10916*x_0^2*x_2*x_3-
   10916*x_0*x_2^2*x_3-10916*x_1*x_3^3,
x_0*x_1^2*x_2^2+11909*x_0^4*x_3+5954*x_0^3*x_2*x_3+2977*x_0^2*x_2^2*x_3+
   11910*x_0*x_2^3*x_3-2978*x_1^3*x_3^2+14887*x_0*x_1*x_3^3+
   11910*x_1*x_2*x_3^3,
x_0*x_1^3*x_2-13099*x_1^3*x_2^2-6550*x_0^3*x_1*x_3-
   13100*x_0^2*x_1*x_2*x_3-6550*x_0*x_1*x_2^2*x_3+13099*x_1*x_2^3*x_3+
   13100*x_1^2*x_3^3+13099*x_0*x_3^4,
x_0^5+5*x_0^2*x_2^3+5*x_0*x_2^4-3*x_0*x_1^3*x_3-4*x_1^3*x_2*x_3+
   4*x_0^2*x_1*x_3^2+10*x_0*x_1*x_2*x_3^2+5*x_1*x_2^2*x_3^2,
x_1^2*x_2^4-8932*x_0^4*x_2*x_3+11909*x_0^3*x_2^2*x_3+5954*x_0^2*x_2^3*x_3-
   8934*x_0*x_2^4*x_3-x_2^5*x_3+2*x_0*x_1^3*x_3^2-5952*x_1^3*x_2*x_3^2-
   x_0^2*x_1*x_3^3-2979*x_0*x_1*x_2*x_3^3-8934*x_1*x_2^2*x_3^3+x_3^6
```

Imagine that you found yourself looking at the scheme X in $\mathbb{P}^3$ defined by the 6 equations above.

```
i53 : X = variety idealX

o53 = X

o53 : ProjectiveVariety
```

How would you analyze the scheme X? We will illustrate one approach.

In outline, we will first look at the topological invariants: the number and dimensions of the irreducible components, and how they meet if there is more than one; the topological type of each component; and the degree of each component in $\mathbb{P}^3$. We will then see what we can say about the analytic invariants of X using adjunction theory (we give some references at the end).

Since we are interested in the projective scheme defined by `idealX` we could work with any ideal having the same saturation. It is usually the case that working with the saturation itself greatly eases subsequent computation so, as a matter of good practice, we begin by checking whether the ideal is saturated. If not, we should replace it with its saturation.

```
i54 : idealX == saturate idealX

o54 = true
```

Thus we see that `idealX` is already saturated. Perhaps the most basic invariant of X is its dimension:

```
i55 : dim X

o55 = 1
```

This shows that `X` consists of a curve, and possibly some zero-dimensional components. The command

```
i56 : idealXtop = top idealX

              4       2       2        2 2   2 2           2           ...
o56 = ideal (x  - 2x x x  - x x x  + x x , x x  - 10915x x x  - 10917x ...
              1     0 1 3    1 2 3    0 3   0 1         0 1 2          ...

o56 : Ideal of ringP3
```

returns the ideal of the largest dimensional components of `X`. If there were 0-dimensional components (or if idealX were not saturated) then `idealXtop` would be larger than `idealX`. To test this we reduce `idealXtop` modulo `idealX` and see whether we get 0:

```
i57 : (gens idealXtop)%(gens idealX) == 0

o57 = true
```

Thus `X` is a purely one-dimensional scheme.

Is `X` singular?

```
i58 : codim singularLocus idealX

o58 = 4
```

A variety of codimension 4 in $\mathbb{P}^3$ must be empty, so X is a nonsingular curve.

A nonsingular curve in $\mathbb{P}^3$ could still be reducible, but since the intersection of two components would be a singular point, the curve would then be disconnected. A straightforward way to decide is to use the command `decompose`, which returns a list of irreducible components defined over `kk`. The length of this list,

```
i59 : # decompose idealX

o59 = 1
```

is thus the number of irreducible components that are defined over `kk`, and we see there is only one. (Warning: at this writing (December 2000), the command "decompose" works only in positive characteristic).

Often what we really want to know is whether `X` is *absolutely irreducible* (that is, irreducible over the algebraic closure of `kk`). The property of being smooth transfers to the algebraic closure, so again the question is the number of connected components we would get over the algebraic closure. For any reduced scheme `X` over a perfect field (such as our finite field `kk`) this number is $\mathrm{h}^0\,\mathcal{O}_X := \dim_{\mathtt{kk}} \mathrm{H}^0\,\mathcal{O}_X$. We compute it with

```
i60 : HH^0 OO_X

        1
o60 = kk

o60 : kk-module, free

i61 : rank oo

o61 = 1
```

This command works much faster than the decompose command. (You can compute the time by adding the command `time` to the beginning of the line where the command to be timed starts.) Since we already know that `idealX` is saturated, this also shows that `idealX` is prime.

We next ask for the genus of the curve X. Here the *genus* may be defined as the dimension of the space $\mathrm{H}^1\,\mathcal{O}_X$. We can get this space with

```
i62 : HH^1 OO_X

        6
o62 = kk
```

```
o62 : kk-module, free
```

The genus of the curve is the dimension of this space, which we can see to be 6. Next, the cohomology class of X in $\mathbb{P}^3$ is determined by the degree of X:

```
i63 : degree idealX

o63 = 10
```

In sum: X is a smooth, absolutely irreducible curve of genus 6 and degree 10.

We next ask for analytic information about the curve and the embedding. A reasonable place to start is with the relation between the line bundle defining the embedding and the canonical sheaf ω_X. Notice first that the degree of the hyperplane divisor (the degree of the curve) is $10 =$ 2g-2, the same as the canonical bundle. By Riemann-Roch the embedding line bundle either is the canonical bundle or has first cohomology 0, which we can check with

```
i64 : P3 = Proj ringP3

o64 = P3

o64 : ProjectiveVariety

i65 : HH^1((OO_P3(1)/idealX)(>=0))

o65 = cokernel | x_3 x_2 x_1 x_0 |

                                         1
o65 : ringP3-module, quotient of ringP3
```

Let's examine the degree of the generator of that module.

```
i66 : degrees oo

o66 = {{0}}

o66 : List
```

From that and the presentation matrix above we see that this cohomology module is the residue class field $\mathtt{ringP3}/(\mathtt{x}_0, \mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3)$, concentrated in degree 0. Thus the embedding line bundle $\mathcal{O}_X(1)$ is isomorphic to ω_X. On the other hand the dimension of the space of sections of this line bundle has already been computed; it is $g = 6$. The curve is embedded in $\mathbb{P}^3$, so only 4 of these sections were used—the embedding is a projection of the same curve, embedded in $\mathbb{P}^6$ by the *canonical map*.

We next ask more about the curve itself. After the genus, the gonality and the Clifford index are among the most interesting invariants. Recall that the *gonality* of X is the smallest degree of a mapping from X to $\mathbb{P}^1$. To define the Clifford index of X we first define the *Clifford index of a line bundle* L on X to be $\mathrm{degree}(L) - 2(\mathrm{h}^0(L) - 1)$. For example, the Clifford indices of the structure sheaf $\mathcal{O}_X$ and the canonical sheaf ω_X are both equal to 0. The *Clifford index of the curve* X is defined to be the minimum value of the Clifford index of a line bundle L on X for which both $\mathrm{h}^0(L) \geq 2$ and $\mathrm{h}^1(L) \geq 2$. The Clifford index of a curve of genus g lies between 0 (for a hyperelliptic curve) and $\lfloor (g-1)/2 \rfloor$ (for a general curve). The Clifford index of any curve is bounded above by the gonality minus 2.

For a curve of genus 6 such as X, the gonality is either 2 (the hyperelliptic case), 3 (the trigonal case) or 4 (the value for general curves). The Clifford index, on the other hand is either 0 (the hyperelliptic case) or 1 (the case of a trigonal curve OR a smooth plane quintic curve—which is necessarily of gonality 4) or 2 (the case of a general curve). Thus for most curves (and this is true in any genus) the Clifford index is equal to the gonality minus 2.

We can make a start on distinguishing these cases already: since our curve is embedded in $\mathbb{P}^3$ by a subseries of the canonical series, X cannot be hyperelliptic (for hyperelliptic curves, the canonical series maps the curve two-to-one onto a rational curve.)

To make further progress we use an idea of Mark Green (see Green and Lazarsfeld [5]). Green conjectured a formula for the Clifford index that depends only on numerical data about the free resolution of the curve in its complete canonical embedding (where the hyperplanes cut out all the canonical divisors). The conjecture is known for genus 6 and in many other cases; see for example Schreyer [9].

We therefore begin by computing the canonical embedding of X. We could proceed to find the canonical bundle as in the computation for $\mathbb{P}^3$ above, or indeed as $\mathcal{O}_X(1)$, but instead we describe the general method that is most efficient: duality, as described (for example) in the book of Altman and Kleiman [1]. The module $\oplus_{d\in\mathbb{Z}} \mathrm{H}^0(\omega_X(d))$ can be computed as

```
i67 : omegaX = Ext^(codim idealX)(ringP3^1/idealX, ringP3^{-4})

o67 = cokernel {0}  | 9359x_3            -4677x_3          -10105x_1     ...
               {0}  | 12014x_1           2552x_1           2626x_0       ...
               {-1} | x_0x_3-2553x_2x_3 x_1^2-1702x_2x_3 x_0x_1-8086x_  ...

                                              3
o67 : ringP3-module, quotient of ringP3
```

To find the equations of the canonical embedding of X, we first compute a basis of $\mathrm{H}^0(\omega_X)$, which is the degree 0 part of the module `omegaX`. The desired equations are computed as the algebraic relations among the images of this basis under any monomorphism $\omega_X \to \mathcal{O}_X$.

As the ring `ringP3/idealX` is a domain, and ω_X is the module corresponding to a line bundle, any nonzero map from ω_X to `ringP3/idealX` will be an embedding. We can compute the module of such maps with

```
i68 : dualModule = Hom(omegaX, ringP3^1/idealX)

o68 = subquotient ({0} | x_0^3x_2^2+10915x_0^2x_2^3+807x_0x_2^4+4043x_ ...
                   {0} | 10105x_0x_1x_2^3+6063x_1x_2^4+11820x_0x_1^2x_ ...
                   {1} | 10105x_0^2x_2^2-11322x_0x_2^3+11322x_2^4+8396 ...

                                                 3
o68 : ringP3-module, subquotient of ringP3
```

and examine it with

```
i69 : betti prune dualModule

o69 = relations : total: 10 26
                      3:  3  2
```

```
4:  6 14
5:  1  9
6:  .  1
```

For want of a better idea we take the first generator, `dualModule_{0}`, which we can turn into an actual homomorphism with

```
i70 : f = homomorphism dualModule_{0}

o70 = | x_0^3x_2^2+10915x_0^2x_2^3+807x_0x_2^4+4043x_2^5+7655x_0_1x_2 ···

o70 : Matrix
```

The image of a basis of ω_X is given by the columns of the matrix

```
i71 : canGens = f*basis(0,omegaX)

o71 = | x_0^3x_2^2+10915x_0^2x_2^3+807x_0x_2^4+4043x_2^5+7655x_0_1x_2 ···

o71 : Matrix
```

regarded as elements of

```
i72 : ringX = ringP3/idealX

o72 = ringX

o72 : QuotientRing
```

Because of the particular homomorphism we chose, they have degree 5.

We can now compute the defining ideal for X in its canonical embedding as the relations on these elements. We first define a ring with 6 variables corresponding to the columns of `canGens`

```
i73 : ringP5 = kk[x_0..x_5]

o73 = ringP5

o73 : PolynomialRing
```

and then compute the canonical ideal as the kernel of the corresponding map from this ring to `ringX` with

```
i74 : idealXcan = trim kernel map(ringX, ringP5,
                                  substitute(matrix canGens,ringX),
                                  DegreeMap => i -> 5*i)

                2                                                    ···
o74 = ideal (x  + 5040x x  - 8565x x  - 11589x x , x x  - 6048x x  - 1 ···
              3        0 5        2 5         4 5   1 3        0 5     ···

o74 : Ideal of ringP5
```

Here the command `trim` is used to extract a minimal set of generators of the desired ideal, and the command `matrix` replaces the map of (nonfree) modules `canGens` by the matrix that gives its action on the generators. The `DegreeMap` option specifies a function which transforms degrees (represented as lists of integers) as the ring homomorphism does; using it here makes the ring map homogeneous.

To get information about the Clifford index, we examine the free resolution with

```
i75 : betti res idealXcan

o75 = total: 1 9 16 9 1
          0: 1 .  . . .
          1: . 6  8 3 .
          2: . 3  8 6 .
          3: . .  . . 1
```

Quite generally, for a non-hyperelliptic curve of genus $g \geq 3$ the ideal of the canonical embedding requires $g - \binom{2}{2}$ quadratic generators, in our case 6. It is known that the curve is trigonal (Clifford index 1) if and only if the ideal also requires cubic generators, that is, the first term in the free resolution requires generators of degree $3 = 1 + 2$; and Green's conjecture says in general that the curve has Clifford index c if the $c - 1$ term in the resolution does not require generators of degree $(c - 1) + 2 = c + 1$ but the c term does require generators of degree $c + 2$. Thus from the Betti diagram above, and the truth of Green's conjecture in low genus, we see that our curve has Clifford index 1 and is thus either trigonal or a plane quintic.

If X is trigonal, that is, X has a map of degree 3 to $\mathbb{P}^1$, then the fibers of this map form a linear series whose elements are divisors of degree three. The geometric form of the Riemann-Roch theorem says that if

$$p_1, \ldots, p_d \in X \subset \mathbb{P}^g$$

are points on a canonically embedded curve X, then the dimension of the linear system in which the divisor $p_1 + \cdots + p_d$ moves is the amount by which the points fail to be linearly independent: $d - 1$ minus the dimension of the projective plane spanned by the points. In particular, the 3 points in the fiber of a three-to-one map to $\mathbb{P}^1$ are linearly dependent, that is, they span a projective line. This "explains" why the ideal of a trigonal curve requires cubic generators: the quadrics all contain three points of these lines and thus contain the whole lines! It is known (see St-Donat [8]) that, in the trigonal case, the 6 quadrics in the ideal of the canonical curve generate the defining ideal of the variety which is the union of these lines, and that variety is a rational normal scroll. In case X is a plane quintic, the *adjunction formula* (Hartshorne [7, II.8.20.3]) shows that the canonical embedding of X is obtained from the plane embedding by composing with the Veronese embedding of the plane in $\mathbb{P}^5$ as the Veronese surface; and the 6 quadrics in the ideal of the canonical curve generate the defining ideal of the Veronese surface.

Thus if we let S denote the variety defined by the quadrics in the ideal of X, we can decide whether X is a trigonal curve or a plane quintic by deciding whether S is a rational normal scroll or a Veronese surface. To compute the ideal of S we first ascertain which of the generators of the ideal of the canonical curve have degree 2 with

```
i76 : deg2places = positions(degrees idealXcan, i->i=={2})

o76 = {0, 1, 2, 3, 4, 5}

o76 : List
```

and then compute

```
i77 : idealS= ideal (gens idealXcan)_deg2places

                2                                                        ...
o77 = ideal (x  + 5040x x  - 8565x x  - 11589x x , x x  - 6048x x  - 1 ...
              3        0 5        2 5         4 5   1 3        0 5     ...

o77 : Ideal of ringP5
```

One of the scrolls that could appear is singular, the cone over the rational quartic in $\mathbb{P}^4$. We check for singularity first:

```
i78 : codim singularLocus idealS

o78 = 6
```

Since the codimension is 6, the surface S is nonsingular, and thus must be one of the nonsingular scrolls or the Veronese surface (which is by definition the image of $\mathbf{P}^2$, embedded in $\mathbf{P}^5$ by the linear series of conics.)

The ideals defining any rational normal scroll of codimension 3, and the ideal of a Veronese surface all have free resolutions with the same Betti diagrams, so we need a subtler method to determine the identity of S. The most powerful tool for such purposes is adjunction theory; we will use a simple version.

The idea is to compare the embedding bundle (the "hyperplane bundle") with the canonical bundle. On the Veronese surface, the canonical bundle is the bundle associated to -3 lines in $\mathbf{P}^2$, while the hyperplane bundle is associated to 2 lines in $\mathbf{P}^2$. Thus the inverse of the square of the canonical bundle is the cube of the hyperplane bundle, $\mathcal{O}_S(3)$. For a scroll on the other hand, these two bundles are different.

As before we follow the homological method for computing the canonical bundle:

```
i79 : omegaS = Ext^(codim idealS)(ringP5^1/idealS, ringP5^{-6})

o79 = cokernel {2} | 4032x_5  0       14811x_5 -4032x_3    6549x_3     ...
               {2} | x_3      x_2     x_1      -x_4        x_0-14291x_ ...
               {2} | -6852x_5 6549x_3 362x_5   x_1-6248x_3 0           ...

                                        3
o79 : ringP5-module, quotient of ringP5

i80 : OS = ringP5^1/idealS

o80 = cokernel | x_3^2+5040x_0x_5-8565x_2x_5-11589x_4x_5 x_1x_3-6048x_ ...

                                        1
o80 : ringP5-module, quotient of ringP5
```

We want the square of the canonical bundle, which we can compute as the tensor square

```
i81 : omegaS**omegaS

o81 = cokernel {4} | 4032x_5  0       14811x_5 -4032x_3    6549x_3     ...
               {4} | x_3      x_2     x_1      -x_4        x_0-14291x_ ...
               {4} | -6852x_5 6549x_3 362x_5   x_1-6248x_3 0           ...
               {4} | 0        0       0        0           0           ...
```

```
                 {4} | 0        0      0      0      0         ...
                 {4} | 0        0      0      0      0         ...
                 {4} | 0        0      0      0      0         ...
                 {4} | 0        0      0      0      0         ...
                 {4} | 0        0      0      0      0         ...

                                      9
o81 : ringP5-module, quotient of ringP5
```

But while this module represents the correct sheaf, it is hard to interpret, since it may not be (is not, in this case) the module of all twisted global sections of the square of the line bundle. Since the free resolution of `OS` (visible inside the Betti diagram of the resolution of `idealXcan`) has length 3, the module `OS` has depth 2. Thus we can find the module of all twisted global sections of `omega2S` by taking the double dual

```
i82 : omega2S = Hom(Hom(omegaS**omegaS, OS),OS)

o82 = cokernel {3} | x_3^2+5040x_0x_5-8565x_2x_5-11589x_4x_5 x_1x_3-60 ...

                                      1
o82 : ringP5-module, quotient of ringP5
```

We see from the output that this module is generated by 1 element of degree 3. It follows that $\omega_S^2 \cong \mathcal{O}_S(-3)$. This in turn shows that S is the Veronese surface.

We now know that the canonical embedding of the curve X is the Veronese map applied to a planar embedding of X of degree 5, and we can ask to see the plane embedding. Since the *anticanonical bundle* ω_S^{-1} on S corresponds to 3 lines in the plane and the hyperplane bundle to 2 lines, we can recover the line bundle corresponding to 1 line, giving the isomorphism of X to the plane, as the quotient

```
i83 : L = Hom(omegaS, OS**(ringP5^{-1}))

o83 = subquotient ({-1} | 14401x_2+16185x_4     x_0-14291x_4 -5359x_1+1 ...
                   {-1} | -1488x_1-10598x_3     -6549x_3     -11789x_5  ...
                   {-1} | x_0+7742x_2-15779x_4 x_2          x_1+6551x_ ...

                                         3
o83 : ringP5-module, subquotient of ringP5
```

and the line bundle on `Xcan` that gives the embedding in $\mathbb{P}^2$ will be the restriction of `L` to `Xcan`. To realize the map from X to $\mathbb{P}^2$, we proceed as before:

```
i84 : dualModule = Hom(L, OS)

o84 = subquotient (| x_0+7742x_2-15779x_4 14401x_2+16185x_4 x_1-301x_3 ...
                   | x_2                  x_0-14291x_4      4032x_3    ...
                   | x_1+6551x_3          -5359x_1+14409x_3 -9874x_5   ...

                                         3
o84 : ringP5-module, subquotient of ringP5

i85 : betti generators dualModule

o85 = total: 3 3
          0: 3 3
```

Again, we may choose any homomorphism from `L` to `OS`, for example

```
i86 : g = homomorphism dualModule_{0}

o86 = | x_0+7742x_2-15779x_4 x_2 x_1+6551x_3 |

o86 : Matrix

i87 : toP2 = g*basis(0,L)

o87 = | x_0+7742x_2-15779x_4 x_2 x_1+6551x_3 |

o87 : Matrix

i88 : ringXcan = ringP5/idealXcan

o88 = ringXcan

o88 : QuotientRing

i89 : ringP2 = kk[x_0..x_2]

o89 = ringP2

o89 : PolynomialRing

i90 : idealXplane = trim kernel map(ringXcan, ringP2,
                                    substitute(matrix toP2,ringXcan))

               5          4            3 2         2 3            4         5 ···
o90 = ideal(x   + 13394x x   - 13014x x   + 9232x x   + 12418x x   - 2746x   ···
               0          0 1          0 1         0 1          0 1         1 ···

o90 : Ideal of ringP2
```

We have effectively computed the square root of the line bundle embedding X in $\mathbb{P}^3$ with which we started, and exchanged a messy set of defining equations of an unknown scheme for a single equation defining a smooth plane curve whose properties are easy to deduce. The same curve may also be defined by a much simpler plane equation (see Appendix A below). I do not know any general method for choosing a coordinate transformation to simplify a given equation! Can the reader find one that will work at least in this case?

There is not yet a textbook-level exposition of the sort of methods we have used (although an introduction will be contained in a forthcoming elementary book of Decker and Schreyer). The reader who would like to go further into such ideas can find a high-level survey of how adjunction theory is used in the paper of Decker and Schreyer [3]. For a group of powerful methods with a different flavor, see Aure, Decker, Hulek, Popescu, and Ranestad [2].

Appendix A. How the "Mystery Variety" was Made

For those who would like to try out the computations above over a different field (perhaps the field of rational numbers `QQ`), and for the curious, we include the code used to produce the equations of the variety X above.

Start with the Fermat quintic in the plane

```
i91 : ringP2 = kk[x_0..x_2]

o91 = ringP2

o91 : PolynomialRing

i92 : idealC2 = ideal(x_0^5+x_1^5+x_2^5)

              5     5     5
o92 = ideal(x  + x  + x )
              0     1     2

o92 : Ideal of ringP2
```

Embed it by the Veronese map in $\mathbb{P}^5$:

```
i93 : ringC2 = ringP2/idealC2

o93 = ringC2

o93 : QuotientRing

i94 : ringP5 = kk[x_0..x_5]

o94 = ringP5

o94 : PolynomialRing

i95 : idealC5 = trim kernel map(ringC2, ringP5,
              gens (ideal vars ringC2)^2)

               2                                    2                 ...
o95 = ideal (x  - x x , x x  - x x , x x  - x x , x  - x x , x x  - x ...
               4    3 5   2 4    1 5   2 3    1 4   2    0 5   1 2    0...

o95 : Ideal of ringP5
```

Finally, choose a projection into $\mathbb{P}^3$, from a line not meeting C5, which is an isomorphism onto its image. (This requires the image to be a smooth curve of degree 10).

```
i96 : ringC5 = ringP5/idealC5

o96 = ringC5

o96 : QuotientRing

i97 : use ringC5

o97 = ringC5

o97 : QuotientRing

i98 : idealC = trim kernel map(ringC5, ringP3,
              matrix{{x_0+x_1,x_2,x_3,x_5}})

               4       2       2        2 2   2 2            2            ...
o98 = ideal (x  - 2x x x  - x x x  + x x , x x  - 10915x x x  - 10917x ...
               1     0 1 3    1 2 3    0 3   0 1          0 1 2           ...

o98 : Ideal of ringP3
```

Let's check that this is the same ideal as that of the mystery variety.

```
i99 : idealC == idealX

o99 = true
```

Here is the code of the function `mystery`, which does the steps above.

```
i100 : code mystery

o100 = -- mystery.m2:1-13
       mystery = ringP3 -> (
          kk := coefficientRing ringP3;
          x := local x;
          ringP2 := kk[x_0..x_2];
          idealC2 := ideal(x_0^5+x_1^5+x_2^5);
          ringC2 := ringP2/idealC2;
          ringP5 := kk[x_0..x_5];
          idealC5 := trim kernel map(ringC2, ringP5,
               gens (ideal vars ringC2)^2);
          ringC5 := ringP5/idealC5;
          use ringC5;
          trim kernel map(ringC5, ringP3,
             matrix{{x_0+x_1,x_2,x_3,x_5}}))
```

And here is the code of the function `prettyPrint`.

```
i101 : code prettyPrint

o101 = -- mystery.m2:15-51
       prettyPrint = f -> (
          -- accept a matrix f and print its entries prettily,
          -- separated by commas
          wid := 74;
          -- page width
          post := (c,s) -> (
             -- This function concatenates string c to end of each
             -- string in list s except the last one
             concatenate \ pack_2 between_c s);
          strings := post_"," (toString \ flatten entries f);
          -- list of strings, one for each polynomial, with commas
          istate := ("",0);
          -- initial state = (out : output string, col : column number)
          strings = apply(
             strings,
             poly -> first fold(
                -- break each poly into lines
                (state,term) -> (
                   (out,col) -> (
                      if col + #term > wid -- too wide?
                      then (
                         out = out | "\n   ";
                         col = 3;
                         -- insert line break
                         );
                      (out | term, col + #term) -- new state
                      )
                   ) state,
                istate,
                fold( -- separate poly into terms
                   {"+","-"},
                   {poly},
                   (delimiter,poly) -> flatten(
                      post_delimiter \ separate_delimiter \ poly
                      ))));
          print stack strings;  -- stack them vertically, then print
          )
```

References

1. Allen Altman and Steven Kleiman: *Introduction to Grothendieck duality theory*. Springer-Verlag, Berlin, 1970. Lecture Notes in Mathematics, Vol. 146.
2. Alf Aure, Wolfram Decker, Klaus Hulek, Sorin Popescu, and Kristian Ranestad: Syzygies of abelian and bielliptic surfaces in $\mathbf{p}^4$. *Internat. J. Math.*, 8(7):849–919, 1997.
3. Wolfram Decker and Frank-Olaf Schreyer: Non-general type surfaces in $\mathbf{p}^4$: some remarks on bounds and constructions. *J. Symbolic Comput.*, 29(4-5):545–582, 2000. Symbolic computation in algebra, analysis, and geometry (Berkeley, CA, 1998).
4. David Eisenbud: *Commutative algebra*. Springer-Verlag, New York, 1995. With a view toward algebraic geometry.
5. Mark Green and Robert Lazarsfeld: On the projective normality of complete linear series on an algebraic curve. *Invent. Math.*, 83(1):73–90, 1985.
6. Joe Harris: *Algebraic geometry*. Springer-Verlag, New York, 1995. A first course, Corrected reprint of the 1992 original.
7. Robin Hartshorne: *Algebraic geometry*. Springer-Verlag, New York, 1977. Graduate Texts in Mathematics, No. 52.
8. B. Saint-Donat: On Petri's analysis of the linear system of quadrics through a canonical curve. *Math. Ann.*, 206:157–175, 1973.
9. Frank-Olaf Schreyer: Syzygies of canonical curves and special linear series. *Math. Ann.*, 275(1):105–137, 1986.
10. Frank-Olaf Schreyer: A standard basis approach to syzygies of canonical curves. *J. Reine Angew. Math.*, 421:83–123, 1991.
11. Jean-Pierre Serre: *Algèbre locale. Multiplicités*. Springer-Verlag, Berlin, 1965. Cours au Collège de France, 1957–1958, rédigé par Pierre Gabriel. Seconde édition, 1965. Lecture Notes in Mathematics, 11.

Index

% 3

32749 1

absolutely irreducible 14
adjunction theory 13

betti 6
Betti diagram 6
Bézout's Theorem 11

canonical bundle 10, 16
canonical embedding 15, 16
chain complex 5
Clifford index 15
codimension 3
coherent sheaf
– representation of 2
cohomology
– sheaf 10
coker 5
cokernel 5
complex 5
– differentials in 5
conormal sequence 8
cotangent bundle 4, 8, 10
cotangent sequence 4
cubic space curve 2
curve
– general 15
– hyperelliptic 15
– plane quintic 16
– trigonal 16

dd 5
dd_i, i-th differential of a complex 6
decompose 14
decompose a variety 14
degree of a projective variety 3
degrees 6
diagram, Betti 6
differentials of a complex 5
dimension of a projective variety 3
division command 9
duality 10
dualizing sheaf 10

Ext 16

finite fields
– use of 1

generators 3
gens 3
genus of a curve 14
gonality 15
Green's conjecture 16

HH 10
homogeneous coordinate ring 2
homogeneous ideal 2
homological methods
– introduction to 1
hyperelliptic curve 15

ideal 3
image 5
intersection multiplicities 10
intersection theory 10
irreducible
– absolutely 14
irreducible decomposition 14

jacobian 8
Jacobian matrix 8

kernel of a module map 4
kernel of a ring map 2

length of a module 11

map of modules 9
map of rings 2
matrix 2
minors 3
modules
– how to represent 4
monomialCurveIdeal 2

positions 18
presentation
– minimal 9
presentation 5
primary decomposition 14
projective space 1
prune 9

rational normal scroll 18
reduced form 3
res 5
resolution
– free 5
Riemann-Roch theorem
– geometric 18

saturate 13
saturation 13
Schreyer's algorithm for free resolutions 6
scroll
– rational normal 18
Serre correspondence 2
Serre's intersection formula 10
sheaf
– representation of 2
sheaf 10
sheaf cohomology 10
singular locus of a scheme 14
source 6

target 6

`time` 14

top dimensional part of an ideal 14

topology of a projective variety 13

`Tor` 12

trigonal curve 16

`trim` 17

twisted cubic 2

Veronese surface 18