From the equations to the Chow form

Jiayue Qi

Research Institute for Symbolic Computation,
Johannes Kepler University, Linz, AUSTRIA

1 Introduction

The Chow form is a device for assigning invariant 'geometric' coordinates to any subvariety of projective space. It was introduced by Cayley for curves in 3-spaces and later generalized by Chow and van der Waerden.

In this report I will introduce the basic definitions of Chow form and later we will reach an algorithm how to compute te Chow form in primary Plücker coordinates of a projective variety.

2 Definition of the Chow form

Definition 1 (Grassmannian). *The set of all d-dimensional linear subspaces of P^n, is called the **Grassmannian** and is denoted by $G(d, n)$.*

Definition 2. *In a 3-dimensional projective space P^3, let L be a line through distinct points x and y with homogeneous coordinates $(x_0 : x_1 : x_2 : x_3)$ and $(y_0 : y_1 : y_2 : y_3)$. The **(primary) Plücker coordinates** of L: p_{ij} are defined as follows:*

$$p_{ij} = |x_i, y_i; x_j, y_j| = x_i y_j - x_j y_i.$$

Remark 3. The definition implies $p_{ii} = 0$ and $p_{ij} = -p_{ji}$, reducing the possibilities to only six independent quantities. The sixtuple is uniquely determined by L up to a common nonzero scale factor. Also not all six components can be zero.

Alternatively, a line can be described as the intersection of two planes.

Definition 4. *Let L be a line contained in distinct planes **a** and **b** with homogeneous coefficients $(a^0 : a^1 : a^2 : a^3)$ and $(b^0 : b^1 : b^2 : b^3)$,respectively.(The first plane equation is $\sum_K a^K x_K = 0$, for example.) The **dual Plücker coordinate** p^{ij} is*

$$p^{ij} = |a^i, a^j; b^i, b^j|.$$

Remark 5. Dual coordinates are equivalent to primary coordinates:

$$(p_{01} : p_{02} : p_{03} : p_{23} : p_{31} : p_{12}) = (p^{23} : p^{31} : p^{12} : p^{01} : p^{02} : p^{03}).$$

Here equality means the number on the right side are equal to the numbers on the left side up to some scaling factor λ. Specially, let (i, j, k, l) be an even permutation of $(0, 1, 2, 3)$; then

$$p_{ij} = \lambda p^{kl}.$$

Similarly, we can define Pl$\ddot{u}$cker coordinates for general cases: Given a d-dimensional linear subspace L of n-dimensional complex projective space P^n, we can write L as the intersection of $n-d$ hyperplanes. Each hyperplane corresponds to a point in the dual projective space, and we write the coordinates of these points as the rows of an $(n-d)\times(n+1)$ matrix M. Left multiplication gives a $GL(n-d,\mathbf{C})$-action on the hyperplanes that preserves the subspace L. The invariant of this action are the maximal minors of M, and these minors determine L uniquely. Conversely, L determines the vector of minors up to multiplication by a nonzero constant. Thus we can represent L by the projective vector of maximal minors of M, which we call the **(primary) Pl$\ddot{u}$cker coordinates** or **brackets** of L.

Let X be an arbitary irreducible projective variety: $X=\{x\in P^n : f_1(x)=...=f_r(x)=0\}$, where f_i are homogeneous polynomials in $k[x_0,...,x_n]$ and k is a subfield of the complex numbers. Let L be an (n-d-1)-dimensional linear subspace of P^n, let Y be the set of all (n-d-1)-dimensional linear subspaces L of P^n such that $X\cap L\neq\phi$.

Theorem 6. *The set Y is an irreducible hypersurface in the Grassmannian $G(n-d-1,n)$.*

Remark 7. Every hypersurface in the Grassmannian is defined by a single polynomial equation, the corresponding irreducible polynomial of Y is called the **Chow form** of X, we denote it as R_X. We can express R_X as a polynomial in brackets. While this representation is not unique, the Chow form itself is unique up to multiplication by a nonzero constant.

When X is reducible, we have the following definition for its Chow form:

Definition 8. *If $X=X_1\cup X_2\cup...\cup X_r$, where X_i is irreducible and appears m_i times in the above equation, then the Chow form of X is $R_X:=R_{X_1}^{m_1}...R_{X_r}^{m_r}$.*

Remark 9. In this case, R_X has coefficient in k, the field of definition of X, while the factors R_{X_i} have coefficients in some algebraic field extension k' of k.

3 From equations to the Chow form

In this section, we consider an algorithm computing the corresponding Chow form in primary Pl$\ddot{u}$cker coordinates for a projective variety.

Suppose X is an irreducible projective variety, presented by a finite set of generators for the corresponding homogeneous prime ideal $I=I(X)$ in $k[x_0,...,x_n]$.

step 0: Compute $d=dim(X)$.

step 1: Add $d+1$ linear forms $l_i=u_{i0}x_0+u_{i1}x_1+...+u_{in}x_n$ with indeterminate coefficients, consider the ideal $J=I+\langle l_0,...,l_d\rangle\subset k[x_i,u_{ji}:i=0,...,d,j=0,...,n]=:S$.

step 2: Replace J by $J'=(J:(x_0,...,x_n)^{\infty})=\{f\in S|\forall i\exists d_i:x_i^{d_i}f\in J\}$.

step 3: Compute the elimination ideal $J'\cap k[u_{00},u_{01},...,u_{dn}]$, since it's in a Euclidean domain, the ideal is principal; let $R(u_{00},...,u_{dn})$ be its principal generator.

step 4: Rewrite the polynomial R in terms of brackets. The result is the Chow form R_X.

Here are some intuitive explanation of this algorithm: In step 1 we form the natural incidence correspondence $\{(x, L) : x \in X \cap L\}$ between X and the Grassmannian. In step 2 we remove trivial solutions with all x-coordinates zero, for which there is no point in P^n. In step 3 and 4 we project the incidence correspondence onto the Grassmannian.

Remark 10. This algorithm works not only for prime ideals but for all unmixed homogenous ideals.

I have implemented this algorithm in Maple, and I will list some examples in the next section.

4 Examples

In the program, we only need to input the corresponding ideal of our variety X and the dimension of whole space, then we will get the Chow form of X in bracket polynomial. In this section we go through most of the examples listed in Dalbec and Sturmfels' paper with a verification from our "chowform" program.

Example 11. Now let's consider the twisted cubic curve $X = \{(s^3 : s^2t : st^2 : t^3) \in P^3 : (s : t) \in P^1\}$. It's the intersection of three quatratic surfaces: $X = \{(x_0 : x_1 : x_2 : x_3) \in P^3 : x_0x_2 - x_1^2 = x_0x_3 - x_1x_2 = x_1x_3 - x_2^2 = 0\}$. So we input this variety into the program "chowform", then the out put is the same as what's listed in the paper "Introsuction to Chow forms" by John Dalbec and Bernd Sturmfels,which is: $R_X = -[03]^3 - [03]^2[12] + 2[02][03][13] - [01][13]^2 - [02]^2[23] + [01][03][23] + [01][12][23]$ up to modulo the syzygy $[01][23] - [02][13] + [03][12]$.

Example 12. Consider the ideal $I = \langle x_1^2 - x_0x_2, x_2^2 - x_1x_3\rangle$. It's variety consists of the twisted cubic and the line $x_1 = x_2 = 0$. Then add the generic linear forms $u_{00}x_0+u_{01}x_1+u_{02}x_2+u_{03}x_3$ and $u_{10}x_0+u_{11}x_1+u_{12}x_2+u_{13}x_3$. Saturate it with respect to $\langle x_0, x_1, x_2, x_3\rangle$ and eliminate the x-variables, then we get a homogeneous polynomial R of bidegree (4,4) in the u_{ij}. To obtain a polynomial in brackets, we may take the ideal generated by R together with the polynomial $u_{0i}u_{1j} - u_{0j}u_{1i} - [ij]$ for $0 \le i < j \le 3$, and eliminate the u-variables. Finally, we compute the normal form of the resulting polynomial with respect to the syzygy $[01][23]-[02][13]+[03][12]$. The output is same as $R_X = [03](-[03]^3-[03]^2[12]+2[02][03][13]-[01][13]^2-[02]^2[23]+[01][03][23]+[01][12][23])$ up to the syzygy $[01][23] - [02][13] + [03][12]$.

Example 13. Here is an example with $k = Q$ and $k' = Q(\omega)$, where ω is a primitive cube root of unity. Consider the point set $\{(1 : \omega : \omega^2), (1 : \omega^2) : \omega\}$ in the projective plane. It's defining ideal is $\langle x_0 + x_1 + x_2, x_1^2 + x_1x_2 + x_2^2\rangle$. Input it into our program, we get the Chow form: $[0]^2 - [0][1] + [1]^2 - [0][2] - [1][2] + [2]^2$.

Example 14. Another example is the Fermat cubic surface $x_0^3 + x_1^3 + x_2^3 + x_3^3 = 0$ in P^3, its Chow form is $[012]^3 - [013]^3 + [023]^3 - [123]^3$.

Example 15. Now let's consider the intersection of the twisted cubic with the hyperplane $x_0 + x_1 + x_2 + x_3 = 0$, after factoring over complex numbers, it coincides with the result in their paper: $-([0] - [1] + [2] - [3])([0] + i[1] - [2] - i[3])([0] - i[1] - [2] + i[3])$.

Example 16. For the surface $(x_1 - x_2)^3 - x_0x_3(x_0 - 3x_1 + 3x_2 - x_3) = 0$, it's Chow form in P^3 is: $-[012][123]([012] + 3[013] + 3[023] + [123]) + ([013] + [023])^3$.

Example 17. The Chow form of the planar curve defined by $3x_0^3 + 2x_0^2x_1 + 13x_0^2x_2 - 12x_0x_1^2 - 4x_0x_1x_2 + 13x_0x_2^2 + 8x_1^3 - 12x_1^2x_2 + 2x_1x_2^2 + 3x_2^3$ is $3[01]^3 - 2[01]^2[02] + 13[01]^2[12] - 12[01][02]^2 + 4[01][02][12] + 13[01][12]^2 - 8[02]^3 - 12[02]^2[12] - 2[02][12]^2 + 3[12]^3$.

The last example in their paper is quite time consuming in my program, so I didn't manage to verify it.

5 Further remark

In my program I only considers Pl*ü*cker relation when the dimension of the variety is 1 and dimension of the whole space is 3, but for other situations I didn't manage to add this into consideration for the moment, it could be one of the future works (to improve the present "chowform" program). So as we mentioned above, the representation of bracket polynomial may not be unique sometimes. Actually even if we consider the Pl*ü*cker relations, the representation may also not be unique but the result could be more concise in some cases.

6 Reference

- Dalbec J., Sturmfels B. (1995) Introduction to Chow Forms. In: White N.L. (eds) Invariant Methods in Discrete and Computational Geometry. Springer, Dordrecht

- Bernd Sturmfels. Algorithms in Invariant Theory. Springer Verlag, Vienna, 1993.

7 Appendix

The program "chowform.txt" is written in Maple language. Input should be a homogeneous ideal s (denote its corresponding projective variety as X), and the dimension of the projective space that we are working in n. Output should be the Chow form for X in bracket polynomial.

7.1 Program: from equations to the Chow form

```
with(PolynomialIdeals):
with(Groebner):
with(combinat):


ChowForm:= module()
      export chow;



chow:= proc(s,n)  local d,b,c,g,i,j,a,J,k,K,l1,l2,K1,B,R,i1,j1,R1,S1,S2,u,br,Q
                       ,R3,S,R2,N,t0,t1,t2,t3,t4,t5,t6,p,e,t,i5;

                  b:= {};
                  c:= {};
                  g:= {};

                  B:= {};
                  t:= {};
                  t6:= {};
                  t3:= {};
                  K:= <1>;

                for k from 0 by 1 to n do
                    c:= `union`({x[k]}, c);
                end do;
                    d:=HilbertDimension(s,c)-1;

                for i from 0 by 1 to d do
                    a[i]:=0;
                    for j from 0 by 1 to n do

                        a[i]:= a[i] + u[i,j]*x[j];
                     end do;
                     b:= `union`({a[i]}, b);

                end do;
                J:= Add(s, <b>);
```

```
for i5 from 0 by 1 to n do
   K:= Intersect(K,Saturate(J,x[i5]));

end do;



 for l1 from 0 by 1 to d do
      for l2 from 0 by 1 to n do
          g:= `union`({u[l1,l2]}, g);
         end do;
  end do;


 K1:= EliminationIdeal(K, g);
 R:= Basis(K1,plex(seq(g)));
 S:= {seq(R)};



 e:=binomial(n+1,d+1);
 p:=firstcomb(n+1,d+1);

for t0 from 1 by 1 to e do

  for t2 from 0 by 1 to d do
    for t1 from 1 by 1 to d+1 do
       t:=`union`({u[t2,p[t1]-1]} ,t);
       t6:=`union`({p[t1]-1},t6);
    end do;

    t3:=`union`({[seq(t)]} ,t3);
    t:={};
  end do;

  t4:= Matrix([seq(t3)]);

  t5:= LinearAlgebra:-Determinant(t4);

  B:=`union`({br[seq(t6)]} ,B);

  S:=`union`({t5-br[seq(t6)]} ,S);

 p:=nextcomb(p,n+1);
 t3:={};
```

```
                t6:={};
                end do;



                S1:= seq(S);
                S2:= <S1>;
                R1:= EliminationIdeal(S2, B);

                R3:= Basis(R1,plex(seq(B)));

                 if d=1 and n=3 then

                Q:= br[0,1]*br[2,3]-br[0,2]*br[1,3]+br[0,3]*br[1,2];

                N:=NormalForm(R3,{Q},tdeg(seq(B)));

                R2:=Basis(<N>,plex(seq(B)));

                return seq(R2);

                else


                return seq(R3);

                end if
            end proc;
end module;
```

7.2 Example verifications

```
    |\^/|     Maple 2017 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2017
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> read"chowform.txt";
                 ChowForm := module() export chow;  end module

> h:=ChowForm;
                                 h := ChowForm

> h:-chow(<x[0]*x[2]-x[1]^2,x[0]*x[3]-x[1]*x[2],x[1]*x[3]-x[2]^2>,3);
memory used=2.6MB, alloc=40.3MB, time=0.20
memory used=3.8MB, alloc=72.3MB, time=0.34
```

```
-2 br[0, 1] br[0, 3] br[2, 3] - br[0, 1] br[1, 2] br[2, 3]

                         2            2
     + br[0, 1] br[1, 3]  + br[0, 2]  br[2, 3] - br[0, 2] br[0, 3] br[1, 3]

                3
     + br[0, 3]

> h:-chow(<x[1]^2-x[0]*x[2],x[2]^2-x[1]*x[3]>,3);
        2          2
br[0, 1]  br[2, 3]  - br[0, 1] br[0, 2] br[1, 3] br[2, 3]

                           2                                       2
     - 2 br[0, 1] br[0, 3]  br[2, 3] + br[0, 1] br[0, 3] br[1, 3]

                2                                      2                   4
     + br[0, 2]  br[0, 3] br[2, 3] - br[0, 2] br[0, 3]  br[1, 3] + br[0, 3]

> h:-chow(<x[0]+x[1]+x[2],x[1]^2+x[1]*x[2]+x[2]^2>,2);
           2                                     2                       2
      br[0]  - br[0] br[1] - br[0] br[2] + br[1]  - br[1] br[2] + br[2]

> h:-chow(<x[0]^3+x[1]^3+x[2]^3+x[3]^3>,3);
memory used=40.0MB, alloc=75.9MB, time=2.87
memory used=105.3MB, alloc=117.2MB, time=9.60
                      3               3               3               3
           br[0, 1, 2]  - br[0, 1, 3]  + br[0, 2, 3]  - br[1, 2, 3]

> h:-chow(<x[0]*x[2]-x[1]^2,x[0]*x[3]-x[\
> 1]*x[2],x[1]*x[3]-x[2]^2,x[0]+x[1]+x[2]+x[3]>,3);
     3        2              2              2                    2
br[0]  - br[0]  br[1] - br[0]  br[2] - br[0]  br[3] + br[0] br[1]

                                                                 2
     + 2 br[0] br[1] br[2] - 2 br[0] br[1] br[3] - br[0] br[2]

                                             2        3        2
     + 2 br[0] br[2] br[3] + br[0] br[3]  - br[1]  + br[1]  br[2]

           2                    2                                     2
     + br[1]  br[3] - br[1] br[2]  - 2 br[1] br[2] br[3] + br[1] br[3]

           3        2                   2        3
     + br[2]  - br[2]  br[3] + br[2] br[3]  - br[3]

> h:-chow(<(x[1]-x[2])^3-x[0]*x[3]*(x[0]-3*x[1]+3*x[2]-x[3])>,3);
```

```
memory used=169.3MB, alloc=117.2MB, time=14.00
memory used=213.2MB, alloc=120.0MB, time=18.55
memory used=292.4MB, alloc=133.8MB, time=28.47
             2
-br[0, 1, 2]  br[1, 2, 3] - 3 br[0, 1, 2] br[0, 1, 3] br[1, 2, 3]

                                                            2
      - 3 br[0, 1, 2] br[0, 2, 3] br[1, 2, 3] - br[0, 1, 2] br[1, 2, 3]

                   3                 2                                         2
      + br[0, 1, 3]  + 3 br[0, 1, 3]  br[0, 2, 3] + 3 br[0, 1, 3] br[0, 2, 3]

                   3
      + br[0, 2, 3]

> h:-chow(<3*x[0]^3+2*x[0]^2*x[1]+13*x[0\
> ]^2*x[2]-12*x[0]*x[1]^2-4*x[0]*x[1]*x[\
> 2]+13*x[0]*x[2]^2+8*x[1]^3-12*x[1]^2*x[2]+2*x[1]*x[2]^2+3*x[2]^3>,2);
          3              2                        2
3 br[0, 1]  - 2 br[0, 1]  br[0, 2] + 13 br[0, 1]  br[1, 2]

                               2
      - 12 br[0, 1] br[0, 2]  + 4 br[0, 1] br[0, 2] br[1, 2]

                               2             3              2
      + 13 br[0, 1] br[1, 2]  - 8 br[0, 2]  - 12 br[0, 2]  br[1, 2]

                              2             3
      - 2 br[0, 2] br[1, 2]  + 3 br[1, 2]
```