COMPUTING DISCRETE INVARIANTS OF VARIETIES IN POSITIVE CHARACTERISTIC

Notes about the Magma implementation

Ben Moonen

This note provides documentation for the Magma implementation of the method described in the paper *Computing discrete invariants of varieties in positive characteristic, I: Ekedahl–Oort type of curves*. We refer to this paper by the acronym CDI1.

For the time being, there is only an implementation for the case of a (non-singular) plane curve C. Such a curve is given by an equation $f = 0$ with $f \in k[X_0, X_1, X_2]$ irreducible and homogeneous of some degree d. The goal of the program is to calculate the Ekedahl–Oort type of C, which is represented by a permutation in $\mathfrak{S}_{2g}$, where $g = g(C) = (d-1)(d-2)/2$. It is assumed that $p > d \geq 3$, where p is the characteristic of the base field k.

The code is available in a file `EOType`, which should be loaded into Magma. The user should define a field k and a homogeneous polynomial, say f, in three variables. After this, the user can invoke the command `EOType(k,f)`, which returns a permutation that represents the Ekedahl–Oort type of the plane curve $C = \mathscr{Z}(f) \subset \mathbb{P}^2$. An error message is returned if f is not homogeneous, if $d < 3$ or $p \leq d$, or if C is singular.

Acknowledgement. The Magma implementation has been realised with much help from Wieb Bosma, who in fact wrote most of the code. I would like to express my sincerest thanks to Wieb for his patient help.

1. Overview of the method

The following situation is assumed:

- k is a field of characteristic $p \geq 5$.
- $\sigma\colon k \to k$ is the absolute Frobenius, i.e., the map $x \mapsto x^p$. In CDI1 the field k is assumed to be perfect, which means that σ is invertible. The Magma code is written in such a way that σ^{-1} is never used.
- $f \in k[X_0, X_1, X_2]$ is homogeneous of degree d with $3 \leq d < p$ such that the subscheme $C \subset \mathbb{P}^2$ defined by $f = 0$ is a smooth curve over k.

After checking if the above conditions on f and C are satisfied, the function `EOType` successively calls three further functions `HWtriple`, `DieudMod`, and `WeylGrElt`, which correspond to the main steps in the calculation. The Magma code for these functions is reviewed in detail in the next sections; here is a quick summary of what we are doing.

As explained in CDI1, we can associate to C a "Hasse–Witt triple", which is a triple (Q, Φ, Ψ) consisting of a finite dimensional k-vector space, a σ-linear map $\Phi\colon Q \to Q$, and a σ-linear bijective map $\Psi\colon \mathrm{Ker}(\Psi) \to \mathrm{Coker}(\Phi)^\vee$. The function `HWtriple` computes the Hasse–Witt triple associated with C, following the Theorem that is stated in the Introduction of CDI1.

As a next step, we need to convert the Hasse–Witt triple into a Dieudonné module, following the method explained in Section 2 of CDI1. This is what the function `DieudMod` does.

Finally, we need to compute the Weyl group coset that represents the isomorphism class of the Dieudonné module, under the bijective correspondence given in Theorem 2.3 of CDI1. This is done in `WeylGrElt`.

Some issues arising in the Magma implementation are the following.

- In CDI1, σ-linear maps play an important role. In the Magma code these are represented by ordinary matrices, with respect to some chosen bases for the spaces involved. For instance, `APhi` and `APsi` refer to matrices that represent the maps that in CDI1 are called Φ, resp. Ψ.
- In CDI1, the Hasse–Witt triple associated with C is described using certain subspaces of the graded k-vector space
$$\mathsf{T} := k[X_0^{\pm 1}, X_1^{\pm 1}, X_2^{\pm 1}]/L\,, \tag{1.1}$$
where L is the k-linear span of all monomials $X^e = X_0^{e_0}X_1^{e_1}X_2^{e_2}$ for which at least one of the exponents e_i is non-negative. Elements of T can be represented by Laurent polynomials, but at several steps in the Magma code this turns out to be inconvenient, for instance because Magma functions such as `MonomialCoefficient` are not available for Laurent polynomials. The solution we use is to multiply Laurent polynomials by a sufficiently high power of $(X_0X_1X_2)$ to ensure that we obtain ordinary polynomials.

2. The function `HWtriple`

Inputs:

- A field k.
- An integer $d \geq 3$.
- A homogeneous polynomial f of degree d with coefficients in k.

Outputs:

- A string s.
- A matrix $A(\Phi)$ of size $g \times g$, where $g = (d-1)(d-2)/2$.
- A basis $\kappa = \{\kappa_1, \ldots, \kappa_h\}$ of the kernel of the k-linear map $k^g \to k^g$ given by $A(\Phi)$. (The integer h is not known a priori).
- A matrix $A(\Psi)$ of size $g \times h$.

Before this function is called, it has been tested if k, d and f define a situation as described at the beginning of Section 1. (If not, an error message is given.)

The purpose of the string s is only to avoid unnecessary calculations: it will be assigned one of the values `ordinary`, `superspecial` or `interesting`. In the first two cases (which are detected as soon as we have the Hasse–Witt matrix $A(\Phi)$), no further work is required and we can directly output the Ekedahl–Oort type.

In the Theorem that is stated in the Introduction of CDI1 the following notation is used:

- $\mathsf{S} = k[X_0, X_1, X_2]$ with its natural grading
- $\mathsf{T} = \oplus_{m \leq -3}\, \mathsf{T}_m$ is the space defined in (1.1) with its natural grading.
- $Q = \mathsf{T}_{-d}$
- $Q' = \left\{\xi \in \mathsf{T}_{-2d} \;\middle|\; \frac{\partial f}{\partial X_j} \cdot \xi = 0 \text{ in } \mathsf{T}_{-d-1}, \text{ for all } j = 0, 1, 2\right\}$
- $\mathsf{U} = \left\{\xi \in \mathsf{T}_{-3d+3} \;\middle|\; \frac{\partial f}{\partial X_j} \cdot \xi = 0 \text{ in } \mathsf{T}_{-2d+2}, \text{ for all } j = 0, 1, 2\right\}$

The space U is 1-dimensional, and if we choose a generator $0 \neq \mathsf{u} \in \mathsf{U}$ we have an isomorphism $\mathsf{S}_{d-3} \xrightarrow{\sim} Q'$ by $g \mapsto g \cdot \mathsf{u}$. The bilinear map $Q \times Q' \to k$ that sends $(q, g \cdot \mathsf{u})$ to the coefficient of $(X_0X_1X_2)^{-1}$ in $g \cdot q$ is a perfect pairing. Let $\theta \colon Q' \xrightarrow{\sim} Q^\vee$ be the associated isomorphism.

The Hasse–Witt triple that we want to compute is the triple (Q, Φ, Ψ), where $\Phi \colon Q \to Q$ and $\Psi \colon \mathrm{Ker}(\Phi) \to Q^\vee$ are given by

$$\Phi[A] = \left[f^{p-1} \cdot A^p\right], \qquad \Psi[A] = \theta\left[f^{p-2} \cdot A^p\right].$$

A basis for the space Q is given by the classes of the monomials $m_i^{-1} \cdot (X_0X_1X_2)^{-1}$, where $m_1, \ldots, m_g$ are all monomials in $k[X_0, X_1, X_2]$ of degree $d-3$. These monomials m_i are stored in a sequence called `Md`.

Next the Hasse–Witt matrix $A(\Phi)$ with respect to this basis is calculated. First we store $F = f^{p-2}$. (For $A(\Phi)$ we need f^{p-1}; but we again need f^{p-2} later.) The matrix coefficient $A(\Phi)_{ij}$ is the coefficient of $m_j^p \cdot (X_0X_1X_2)^{(p-1)}$ in $f^{p-1} \cdot m_i = f \cdot F \cdot m_i$. If $A(\Phi)$ is either invertible (ordinary case) or zero (superspecial case), we can immediately stop.

If we are not in the ordinary or superspecial case, we go on to store a basis $\kappa = \{\kappa_1, \ldots, \kappa_h\}$ of the kernel of the linear map $A(\Phi) \colon k^g \to k^g$. For later use, note that k^g represents the space Q through the chosen basis of Q, and that a basis of the kernel of the σ-linear map $\Phi \colon Q \to Q$ is given by the vectors ${}^\tau\kappa_j$, where $\tau = \sigma^{-1}$.

Next we store bases for the spaces T_{-2d+2} and T_{-3d+3}. As explained above, we want to work with ordinary polynomials instead of Laurent polynomials; for this reason, the elements that we use are in fact $(X_0X_1X_2)^{3d-3}$ times a basis.

Then we calculate the partial derivatives $\partial f/\partial X_i$ and we compute the matrix `Multdf` which represents the linear map $\mathsf{T}_{-3d+3} \to \mathsf{T}_{-2d+2}^{\oplus 3}$ given by

$$\xi \mapsto \Big(\frac{\partial f}{\partial X_0} \cdot \xi,\ \frac{\partial f}{\partial X_1} \cdot \xi,\ \frac{\partial f}{\partial X_2} \cdot \xi\Big).$$

By definition, U is the kernel of this map. We choose a generator; but for the same reason as above, what we store is not a generator u of U but rather $\tilde{\mathsf{u}} = (X_0X_1X_2)^{3d-3} \cdot \mathsf{u}$, which in the code is called `utilde`. The elements $m_i \cdot \mathsf{u}$ form a basis of the space $Q' \cong Q^\vee$ which is dual to the chosen basis $\{m_i^{-1} \cdot (X_0X_1X_2)^{-1}\}$ of Q.

The final step of `HWtriple` is the calculation of the $g \times h$ matrix $A(\Psi)$. If we write κ_j as

$$\kappa_j = \begin{pmatrix} \kappa_1^{(j)} \\ \vdots \\ \kappa_g^{(j)} \end{pmatrix},$$

the jth column of the matrix $A(\Psi)$ is obtained by solving

$$A(\Psi)_{1j} \cdot m_1 \cdot \mathsf{u} + \cdots + A(\Psi)_{gj} \cdot m_g \cdot \mathsf{u} = f^{p-2} \cdot \Big(\kappa_1^{(j)} \cdot m_1^{-p} \cdot X^{-p\cdot\mathbf{1}} + \cdots + \kappa_g^{(j)} \cdot m_g^{-p} \cdot X^{-p\cdot\mathbf{1}}\Big). \quad (2.1)$$

(This is an equation in T_{-2d}, and $X^{-p\cdot\mathbf{1}}$ means $(X_0X_1X_2)^{-p}$. Note that the jth column of the matrix $A(\Psi)$ is the vector $\Psi({}^\tau\kappa_j)$; as $\Psi \colon \mathrm{Ker}(\Phi) \to Q'$ is given by $[A] \mapsto [f^{p-2} \cdot A^p]$, this leads to equation (2.1) for the coefficients of $A(\Psi)$.)

Let $c = (2d-1)(2d-2)/2$, which is the number of monomials in $k[X_0, X_1, X_2]$ of degree $2d-3$, and let $M_1, \ldots, M_c$ be those monomials. Because Magma's default is to let matrices act from the right, (2.1) is written as the matrix equation

$${}^{\mathrm{t}}A(\Psi) \cdot B = \kappa \cdot C, \quad (2.2)$$

where B and C are the matrices of size $g \times c$ whose rows express the $m_i \cdot \mathsf{u}$ (resp. the $f^{p-2} \cdot m_i^{-p} \cdot (X_0X_1X_2)^{-p}$) as vectors with respect to the basis $\{M_j^{-1} \cdot (X_0X_1X_2)^{-1}\}_{j=1,\dots,c}$ of T_{-2d}, and where κ now is the matrix of size $h \times g$ whose rows give the vectors κ_j. Concretely, B_{ji} is the coefficient of $(X_0X_1X_2)^{3d-4}$ in $M_i \cdot m_j \cdot \tilde{\mathsf{u}}$, and C_{ji} is the coefficient of $(X_0X_1X_2)^{p-1} \cdot m_j^p$ in $f^{p-2} \cdot M_i$. (Recall that $F = f^{p-2}$ has been calculated before and that we have stored $\tilde{\mathsf{u}} = (X_0X_1X_2)^{3d-3} \cdot \mathsf{u}$.) Then (2.2) is solved using Magma's function `IsConsistent`.

3. The function `DieudMod`

Inputs:

- A field k and a positive integer d.
- A matrix $A(\Phi)$ of size $g \times g$, where $g = (d-1)(d-2)/2$.
- A basis $\{\kappa_1, \dots, \kappa_h\}$ of the kernel of the k-linear map $k^g \to k^g$ given by $A(\Phi)$.
- A matrix $A(\Psi)$ of size $g \times h$.

Output:

- A matrix $A(F)$ of size $2g \times g$ whose columns are linearly independent.

3.1 Steps that are carried out.

(1) Find a subset $I = \{i_1, \dots, i_{g-h}\} \subset \{1, \dots, g\}$ such that $\mathrm{Span}(e_i; i \in I)$ is a complement of $\{\kappa_1, \dots, \kappa_h\}$ inside k^g.

(2) To obtain the jth column of the matrix $A(F)$, write the standard base vector e_j in the form

$$e_j = \sum_{\mu=1}^{g-h} a_\mu \cdot e_{i_\mu} + \sum_{\nu=1}^{h} b_\nu \cdot \kappa_\nu \,. \tag{3.1}$$

Then

$$A(F)_{rj} = \begin{cases} \sum_{\mu=1}^{g-h} a_\mu \cdot A(\Phi)_{r,i_\mu} & r = 1, \dots, g \\ \sum_{\nu=1}^{h} b_\nu \cdot A(\Psi)_{2g+1-r,\nu} & r = g+1, \dots, 2g. \end{cases}$$

3.2 Technical comments. The above is based on section 2.5 of CDI1. Let $e_1, \dots, e_g$ be the standard basis of k^g. The goal is to give the matrix of $F\colon M \to M$, where $M = Q \oplus Q^\vee$. However, F factors through the projection $M \to Q$, so we only need to give the first g columns. We are identifying Q with k^g via the basis $\{m_i^{-1} \cdot X^{-\mathbf{1}}\}_{i=1,\dots,g}$. The dual vector space $Q^\vee$ is identified with Q' as in the paper (choice of $0 \neq \mathsf{u} \in \mathsf{U}$), and the dual basis of Q' is $\{m_i \cdot \mathsf{u}\}_{i=1,\dots,g}$. However, as a preparation for the next step we want to use $e_1, \dots, e_g, \check{e}_g, \dots, \check{e}_1$ *(note the order!)* as a basis for $M = Q \oplus Q^\vee$.

We are choosing $I \subset \{1, \dots, g\}$ in such a way that $R_0 = \mathrm{Span}(e_i)_{i \in I}$ is a complement of ${}^\sigma R_1 := \mathrm{Span}(\kappa_1, \dots, \kappa_h)$ inside k^g. Then R_0 is also a complement of $R_1 = \mathrm{Ker}(\Psi) = \mathrm{Span}({}^\tau\kappa_1, \dots, {}^\tau\kappa_h)$. With notation as in (3.1),

$$e_j = \sum_{\mu=1}^{g-h} {}^\tau a_\mu \cdot e_{i_\mu} + \sum_{\nu=1}^{h} {}^\tau b_\nu \cdot {}^\tau\kappa_\nu \,.$$

is the decomposition of e_j corresponding to $k^g = R_0 \oplus R_1$. So the top half (first g coefficients) of the jth column of $A(F)$ is given by the vector

$$\Phi\Big(\sum_{\mu=1}^{g-h} {}^\tau a_\mu \cdot e_{i_\mu}\Big) = \sum_{\mu=1}^{g-h} a_\mu \cdot A(\Phi)_{*,i_\mu} \,.$$

Similarly, the lower half (last g coefficients) of the jth column of $A(F)$ is given by putting the vector

$$\Psi\Big(\sum_{\nu=1}^{h} {}^{\tau}b_\nu \cdot {}^{\tau}\kappa_\nu\Big) = \sum_{\nu=1}^{h} b_\nu \cdot \cdot A(\Psi)_{*,\nu}$$

upside down (because we now use the order $\check{e}_g, \ldots, \check{e}_1$).

4. The function `WeylGrElt`

Input:

- A field k and a positive integer d.
- A matrix $A(F)$ of size $2g \times g$ whose columns are linearly independent.

Output:

- An element $w \in \mathfrak{S}_{2g}$ (symmetric group on $2g$ letters).

There are three parts in this procedure. In the first part (steps (1)–(3)) we are going to (partially) fill a table, whose initial state is the following:

i	0	1	2	$\cdots$	$g-1$	g	$g+1$	$\cdots$	$2g$
Basis(i)	$\emptyset$					the g columns of $A(F)$			$\{e_1, \ldots, e_{2g}\}$
$f(i)$	0								g

If Basis(i) is defined, it consists of a set of i linearly independent vectors in k^{2g}, and if $f(i)$ is defined, it is an integer with $0 \leq f(i) \leq i$. (In the initial state, $\{e_1, \ldots, e_{2g}\}$ denotes the standard basis of k^{2g}.) The calculation involves finding the perpendiculars of certain subspaces $W \subset k^{2g}$ with respect to the symplectic form on k^{2g} that is represented by the matrix (in block form)

$$\begin{pmatrix} 0 & \mathsf{J} \\ -\mathsf{J} & 0 \end{pmatrix}$$

where J denotes the anti-diagonal matrix

$$\mathsf{J} = \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix}$$

of size $g \times g$.

In the second part (step (4)), we are going to define $f(i)$ for all i. In the third part (step (5)) we are going to convert the sequence f into a permutation.

4.1 Steps that are carried out.

(1) Create a table as above.

(2) Search for the first index i such that Basis(i) is defined, but $f(i)$ is not yet defined. If there is no such i (in the range $1, \ldots, 2g$), go to step (4). If Basis(i) $= \{b_1, \ldots, b_i\}$, calculate the vectors $A(F)\big({}^{\sigma}b_j\big)$ ($j = 1, \ldots, i$), and let $f(i)$ be the dimension of their k-linear span. Store the value $f(i)$ in the table.

(3) If Basis$\big(f(i)\big)$ is already defined, again do step (2). If Basis$\big(f(i)\big)$ is not yet defined, do the following:

- Among the vectors $A(F)\big({}^{\sigma}b_1\big), \ldots, A(F)\big({}^{\sigma}b_i\big)$, find a maximal linearly independent subset, say $\{\beta_1, \ldots, \beta_{f(i)}\}$, and store this collection as $\mathrm{Basis}\big(f(i)\big)$.
- Find a basis for the space

$$\mathrm{Span}\big(\beta_1, \ldots, \beta_{f(i)}\big)^{\perp} = \left\{ y \in k^{2g} \;\middle|\; {}^{\mathsf{t}}\beta_j \cdot \begin{pmatrix} 0 & \mathsf{J} \\ -\mathsf{J} & 0 \end{pmatrix} \cdot y = 0 \quad \text{for all } j = 1, \ldots, f(i) \right\},$$

and store this as $\mathrm{Basis}\big(2g - f(i)\big)$.

After this, return to step (2).

(4) If $f(i)$ is defined for all i, go to step (5). Otherwise, find the first value a for which $f(a)$ is still undefined, and let b be the next value for which $f(b)$ is defined. Now assign the values $f(a), \ldots, f(b-1)$ as follows: it will be true that either $f(a-1) = f(b)$ or that $f(b) = f(a-1) + (b-a+1)$; in the first case, set $f(a), f(a+1), \ldots, f(b-1)$ all equal to $f(a-1)$, in the second case define $f(i)$ for $a \leq i < b$ by the rule $f(i) = f(a-1) + (i-a+1)$. Now repeat this step.

(5) Let $j_1 < j_2 < \cdots < j_g$ be the values in $\{1, 2, \ldots, 2g\}$ with the property that $f(j) = f(j-1)$. (There will be precisely g such values.) Let $i_1 < i_2 < \cdots < i_g$ be the remaining values. Define a function $w\colon \{1, 2, \ldots, 2g\} \to \{1, 2, \ldots, 2g\}$ by $w(j_m) = m$ and $w(i_m) = g + m$. Now output the message: `The Ekedahl-Oort type of the curve is given by the Weyl group element`

$$\begin{bmatrix} 1 & 2 & \cdots & g & g+1 & \cdots & 2g \\ w(1) & w(2) & \cdots & wg) & w(g+1) & \cdots & w(2g) \end{bmatrix}$$

(Further details in the output to be added.)

Note. In the Magma implementation, the index i in our table runs from 1 to $2g+1$, rather than from 0 to $2g$. So everything is shifted by 1.

4.2 Technical comments. In the table we keep track of the so-called canonical flag, as outlined in CDI1, Section 2.2. We build it using the operations F and $\perp$, so we avoid using the Verschiebung. (The result is the same.)

For the conversion to a Weyl group element, we follow [GSAS], Section 3.6. Note that the condition $f(j) = f(j-1)$ is equivalent to saying that the sequence η that is considered in loc. cit. jumps at j.

5. References

[GSAS] B. Moonen, Group schemes with additional structures and Weyl group cosets. In: Moduli of Abelian Varieties (C. Faber, G. van der Geer and F. Oort, eds.), Progress in Math. 195, Birkhäuser, Basel, 2001, 255–298.

[CDI1] B. Moonen, Computing discrete invariants of varieties in positive characteristic, I. *Ekedahl-Oort types of curves.* Preprint, April 2020.

b.moonen@science.ru.nl
Radboud University Nijmegen, IMAPP, Nijmegen, The Netherlands