

Mordell-Weil Problem for Cubic Surfaces, Numerical Evidence

Bogdan G. Vioreanu

ABSTRACT. Let V be a plane smooth cubic curve over a finitely generated field k . The Mordell-Weil theorem for V states that there is a finite subset $P \subset V(k)$ such that the whole $V(k)$ can be obtained from P by drawing secants and tangents through pairs of previously constructed points and consecutively adding their new intersection points with V . In this paper we present numerical data regarding the analogous statement for cubic surfaces. For the surfaces examined, we also test Manin's conjecture relating the asymptotics of rational points of bounded height on a Fano variety with the rank of the Picard group of the surface.

1. Introduction

Let V be a smooth cubic surface over a field k in \mathbb{P}^3 . If $x, y, z \in V(k)$ are three points (with multiplicities) lying on a line in \mathbb{P}^3 not belonging to V we write $x = y \circ z$. Thus \circ is a partial and multivalued composition law on $V(k)$. Note that $x \circ x$ is defined as the set of points in the intersection of $V(k)$ with the tangent plane at x . If x does not lie on a line, this is a cubic curve $C(x)$ with double point $x \in V(k)$. This whole set must be considered as the domain of the multivalued expression $x \circ x$, because geometrically all its points can be obtained by drawing tangents with k -rational direction to x . This means that an important source for generating new rational points on the cubic surface will be doubling the points that were already generated. The analogue of the Mordell-Weil theorem for cubic surfaces states that $(V(k), \circ)$ is finitely generated, i.e., there is a finite subset $P \subset V(k)$ such that the whole $V(k)$ can be obtained from P by drawing secants and tangent planes through pairs of (not necessarily distinct) previously constructed points, and consecutively adding their new intersection points with V . By drawing secants we can add only one rational point to P while tangent sections give us an infinite number of points that can be generated, by the note above. For a more thorough discussion of various versions of finite generation cf. [KM01]. Note that, by Theorem 11.7 of [Man86], finite generation of $(V(k), \circ)$ implies that the universal quasi-group of $(V(k), \circ)$, as defined in [Man86], chapter II, is finite and has $2^n 3^m$ elements for some $n, m \in \mathbb{Z}_{\geq 0}$.

2000 *Mathematics Subject Classification*. Primary 11G35, Secondary 11G50, 14J26.

©2009 Bogdan Vioreanu

In the following, we present the procedure we used to test whether $(V(\mathbb{Q}) \circ)$ is finitely generated, and the results we obtained for thirteen diagonal cubic surfaces, six of them having the rank of their Picard group equal to 1, and seven of them mentioned in [PT01], illustrating the cases of surfaces with ranks 2 and 3 of the Picard group. We also bring numerical evidence supporting Manin's conjecture for the asymptotics of rational points of bounded height on a Fano variety. Note that John Slater and Sir Peter Swinnerton-Dyer have proved in [SP98] a one-sided estimate for the conjecture in the case when V contains two rational skew lines. All the computations were done using the Magma computer algebra system (cf. [BCP97].)

2. Description of the procedure

Let $ax^3 + by^3 + cz^3 + du^3 = 0$, where a, b, c, d are nonzero integers, be a diagonal cubic surface. Using a program due to Dan Bernstein (see [Ber01]), we find all rational points on this surface up to height $H = 10^5$ or $H = 1.5 \cdot 10^5$, where the height of a rational point $P = (x : y : z : u)$, with $x, y, z, u \in \mathbb{Z}$ and $\gcd(x, y, z, u) = 1$ is defined as

$$h_{\max}(P) := \max\{|x|, |y|, |z|, |u|\}$$

We consider also another height function $h_{\text{sum}} : V(\mathbb{Q}) \rightarrow \mathbb{R}_+$ defined by

$$h_{\text{sum}}(P) := |x| + |y| + |z| + |u|$$

Note that a rational point P can be uniquely written in the above form up to a sign change of the coordinates. So, if we assume, in addition, that the first nonzero coordinate of P is positive, then there is a unique such 'canonical' form corresponding to each point P . We order the rational points by increasing h_{sum} . If there are two or more points having the same height h_{sum} , then we order them lexicographically according to their coordinates in the canonical form. This defines a total order on the set of rational points. We will write $P < Q$ if P precedes Q in the sorted list, and use the number of a point in this list as its name. We will also refer to this number as the *index* of a rational point.

We will use the h_{\max} height function only to study the asymptotics of the number of rational points on a cubic surface, while for the ordering of the points and in the implementation of the main function we will use h_{sum} .

For testing whether a given set of rational points is generating, we use the procedure *Test Generating Set (TGS)*, which is described below.

The procedure implements essentially a descent method. Given an index bound n and a set of points *GeneratedSet* that is presumably generating, we perform the following iterative process. In one iteration of loop, we consider all points in the range $\{1, \dots, n\}$ that are not in *GeneratedSet* and test whether they can be decomposed as $x \circ y$, with $x, y \in \text{GeneratedSet}$. Every point that can be decomposed in such a way is added to the *GeneratedSet* and at the end of the loop, the procedure is reiterated. As now *GeneratedSet* is bigger, there may be additional points in the range $\{1, \dots, n\}$ that can be generated because we can choose the points x, y for a possible decomposition from a bigger set. The procedure is repeated until *GeneratedSet* stabilizes, i.e., until some iteration of the loop does not add any new points to the *GeneratedSet*.

In order to avoid repeating some operations of composing points, we use the additional variables *OldGeneratedSet*, *JustAdded* and *Decomp. OldGeneratedSet*

stores the value of *GeneratedSet* at the beginning of the iteration of the loop. At the end of the preceding loop, a number of points will have been added to *GeneratedSet*. These points are stored in the set variable *JustAdded*. During an iteration of the loop, we store in *Decomp* decompositions of the type $i = j \circ k$, with $i, j, k \leq n$, $i, k \in \text{GeneratedSet}$ and $j \in \text{GeneratedSet}$. These are the only decompositions that we could further use. Indeed, if, at some point, k was added to *GeneratedSet*, then by searching in *Decomp*, we would find the decomposition $j \circ k$ of i and we would add i to *GeneratedSet* without performing any composition of points (which requires multiplications, so is computationally expensive) because we know, by the way we constructed *Decomp*, that $j \in \text{GeneratedSet}$ already.

Receiving as input the parameters *GeneratedSet* (a set of points in $V(\mathbb{Q})$ that is assumed to be generating), and n (the index bound for the points used in the decompositions), the *TGS* procedure does the following:

- (1) Set $\text{Decomp} = \emptyset$, $\text{OldGeneratedSet} = \emptyset$.
- (2) Set $\text{JustAdded} = \text{GeneratedSet} \setminus \text{OldGeneratedSet}$,
 $\text{OldGeneratedSet} = \text{GeneratedSet}$.
- (3) If $\text{JustAdded} = \emptyset$, return *GeneratedSet*.
- (4) For every point $i \in \{1, 2, \dots, n\} \setminus \text{GeneratedSet}$ do:
 search in *Decomp* for decompositions of i as $x \circ y$ with $y \in \text{JustAdded}$
 if such a decomposition exists, add i to *GeneratedSet*
 else for every point j in *JustAdded* do:
 $k = i \circ j$
 if $k \in \text{JustAdded}$
 add i to *GeneratedSet*
 break
 else if $k \leq n$ add the decomposition $(j \circ k)$ of i to *Decomp*
 end for
 end for.
- (5) Go to step 2.

Let us explain in more detail the way the algorithm works. Suppose that an iteration of the outer loop has just finished, and we are in step 2. We set $\text{JustAdded} = \text{GeneratedSet} \setminus \text{OldGeneratedSet}$ and test whether this is the empty set. If this is so, then during the last iteration we could not generate any new points, so the maximum set of points that can be generated is the current *GeneratedSet*. If *JustAdded* is not empty, then during the last iteration we found a number of new points that could be generated and added them to *GeneratedSet* (these are the elements of *JustAdded*), so there is hope of generating other points. We consider a point $i \in \text{GeneratedSet}$. Since we have already tested during the previous iteration whether we could decompose i as $x \circ y$, with $x, y \in \text{OldGeneratedSet}$, all we have to check now is whether we can write $i = x \circ y$ for $x \in \text{JustAdded}$ and either $y \in \text{OldGeneratedSet}$ or $y \in \text{JustAdded}$. At the previous iterations of the loop all compositions of i with points in *OldGeneratedSet* that could further be used (i.e., compositions whose result is not bigger than n) were stored in *Decomp*, so we can check for the first possibility by searching in the vector *Decomp*. Since by construction we only store in *Decomp* decompositions of the type $x \circ y$, with $x \in \text{GeneratedSet}$, all we have to check in the beginning of step 4 is whether $y \in \text{JustAdded}$ —we are sure that $x \in \text{GeneratedSet}$. In order to check for the second possibility, we have to compose i with every point $j \in \text{JustAdded}$. If the result k of