Vietnam Journal of Mathematics (2022) 50:545–556
https://doi.org/10.1007/s10013-022-00549-4

ORIGINAL ARTICLE

Equations for GL Invariant Families of Polynomials

Paul Breiding[1] · **Reuven Hodges**[2] · **Christian Ikenmeyer**[3] · **Mateusz Michałek**[1,4]

Received: 29 April 2021 / Accepted: 17 November 2021 / Published online: 24 February 2022

Abstract

We provide an algorithm that takes as an input a given parametric family of homogeneous polynomials, which is invariant under the action of the general linear group, and an integer d. It outputs the ideal of that family intersected with the space of homogeneous polynomials of degree d. Our motivation comes from Question 7 in Ranestad and Sturmfels (Le Math. **75**, 411–424, 2020) and Problem 13 in Sturmfels (2014), which ask to find equations for varieties of cubic and quartic symmetroids. The algorithm relies on a database of specific Young tableaux and highest weight polynomials. We provide the database and the implementation of the database construction algorithm. Moreover, we provide a Julia implementation to run the algorithm using the database, so that more varieties of homogeneous polynomials can easily be treated in the future.

Keywords Defining equations · Highest weight vector · Young tableau · Image of a map · GL-action

Mathematics Subject Classification (2010) 14Q15 · 14M20 · 20G05 · 20C40

1 Introduction

Many mathematical models are defined by nonlinear maps $f : V \to W$ between vector spaces. For instance, such models are common in statistics and physics. The setting allows to generate possible outcomes of the model, by evaluating f. This is called the *forward problem*. On the other hand, the *inverse problem* is to decide if a point $w \in W$ belongs to the image of f, and if so, to determine its preimage.

Dedicated to Professor Bernd Sturmfels on the occasion of his 60th birthday.

✉ Mateusz Michałek
mateusz.michalek@uni-konstanz.de

[1] Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

[2] University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

[3] University of Liverpool, Liverpool, UK

[4] University of Konstanz, Konstanz, Germany

In this article we focus on the case when f is a polynomial map. Under this assumption the forward problem consists in evaluating a system of polynomials, and the inverse problem is to solve a system of polynomial equations.

Our main aim is to describe the closure of the image of f, when V and W are complex vector spaces. The goal is to describe the polynomial equations that vanish on the image of f. Having such equations at hand decouples the inverse problem: for the decision problem, whether or not w is in the image of f, one can evaluate the polynomials at w instead of solving a system of equations. The former is much simpler than the latter.

The classical method to find equations relies on the computation of a lexicographic Gröbner basis [15, 27] to perform elimination of variables. It is a symbolic method, that in practice may be used only on small examples. Thus, the motivation for us is to describe an alternative algorithm that can go beyond these small cases. In general, this task is too ambitious. But, if we assume that the problem has some underlying symmetries, we can use the power of *representation theory* to reduce complexity. In this paper, we make the following assumption for f: we require it to be mapping into a vector space of polynomials and we assume that the image of f is GL-invariant.

Assumption 1.1 We assume that $W = S^c(\mathbb{C}^n)$ is the space of homogeneous polynomials of degree c in n variables. Furthermore, we assume that the image of f is invariant under $\mathrm{GL}(\mathbb{C}^n)$, which acts by variable substitution.

Our motivation comes from Question 7 in [31] and Problem 13 in [33], which ask to find equations for varieties of cubic and quartic symmetroids. These are subvarieties of the vector space of homogeneous polynomials in $n = 4$ variables of degree respectively $c = 3$ and $c = 4$. They are GL(4)-invariant. We address these problems in Section 5.

We note that the GL action both gives us many advantages and is very natural. Our ambient space $S^c(\mathbb{C}^n)$ of polynomials may be regarded as a space of varieties. Following Felix Klein's Erlangen program [24] geometric quantities should be group-invariant. Thus, very often when studying sets of polynomials, we would like those sets not to depend on the choice of the basis. This is precisely the GL invariance. Furthermore, the space of polynomials vanishing is often huge, but the GL action reduces the complexity and allows us to describe it using just a few generators.

2 Contributions

We present an algorithm to study the image of f under Assumption 1.1. This algorithm produces the following: Let

$$X := \overline{\mathrm{im}(f)}$$

be the closure of the image of f and let I be the ideal of polynomials that vanish on X. Given f and any d we return the minimal set of polynomials that under the GL action span I_d. This algorithm is exact, i.e., does not rely on any approximations. However, instead of a purely symbolic algorithm that works with the parametrized variety X directly, a much more efficient implementation just samples from X (without approximations) and uses only the sampled points as input, which reduces the finding problem to a linear algebra problem. The details are given in Section 4. This means that due to unlucky sampling in principle the algorithm could output equations that are not actually equations. In practice the probability of this is extremely low and can be further reduced to an inverse exponentially small probability by running the algorithm several times. Furthermore, a posteriori, it is easy to check

if the equations actually vanish on $\mathrm{im}(f)$. One of the algorithm's central ingredients is a *database* which contains bases of highest weight spaces for different plethysms.

The variety of quartic (resp. cubic) symmetroids consists of polynomials that are determinants of symmetric four by four (resp. three by three) matrices with entries that are linear forms in four variables. To distinguish these varieties from the general X we will use another symbol:

$$\begin{aligned} Q_3 &:= \left\{ \det(x_0A_0 + x_1A_1 + x_2A_2 + x_3A_3) \mid A_i \in \mathbb{C}^{3\times 3},\ A_i^T = A_i,\ 0 \le i \le 3 \right\}, \\ Q_4 &:= \left\{ \det(x_0A_0 + x_1A_1 + x_2A_2 + x_3A_3) \mid A_i \in \mathbb{C}^{4\times 4},\ A_i^T = A_i,\ 0 \le i \le 3 \right\}. \quad (1) \end{aligned}$$

The varieties are $\mathrm{GL}(\mathbb{C}^4)$ invariant subvarieties of $S^4(\mathbb{C}^4)$ (resp. $S^3(\mathbb{C}^4)$) of codimension 10 (resp. 4). We apply our algorithm to this variety, and we obtain the following result.

Theorem 2.1 *There are no equations for Q_4 in degrees up to (including)* 8.

There are no equations for Q_3 in degrees up to (including) 10. *In degree* 11 *the vector space $I(Q_3)_{11}$ is an irreducible representation of dimension* 220 *corresponding to the weight* [15, 6, 6, 6]. *The unique highest weight vector has* 23824 *many terms. In degree* 12 *the vector space $I(Q_3)_{12}$ has four irreducible components corresponding to weights* [15, 9, 6, 6], [16, 8, 6, 6], [17, 7, 6, 6] *and* [18, 6, 6, 6].

Our second contribution is a numerical algorithm that computes the degree of variets as described above. We applied it to Q_3 confirming the known result [34] that the variety has degree 305. However, we were not able to compute the degree of Q_4, which we leave as a future challenge.

We combine numerical and symbolic methods in our algorithms. Both the numerical and the symbolic algorithm appeared (explicitly or implicitly by using highest weight polynomials as images of symmetrizations over the wreath product) in particular examples before [1–3, 9–14, 16, 17, 21, 22, 26, 28–30]. However, to our knowledge, this is the first general implementation and the first one with which it is possible to check for equations of degree 8 on $S^4(\mathbb{C}^4)$ or find equations of Q_3. This is made possible by the use of an idea that we call *equivariant hash functions*, see Section 4. We provide the source code of our implementation and an easy to use user interface for future researchers to build upon.

We remark that new algorithms for evaluating highest weight polynomials have been developed very recently in [4]. No open source implementation of these algorithms is available, but in a special case (see [17]) the running time improvements seem to be of practical importance.

3 Representation Theory

Representation theory can be beneficial for large computations. In one line, it allows one to replace a possibly high dimensional irreducible representation, by a one dimensional subspace—the span of the highest weight vector. We briefly recall the relevant concepts for our setting. For more details we refer to [20, 27].

Every irreducible, polynomial representation $V = V_\lambda$ of $\mathrm{GL}(\mathbb{C}^n)$ is associated to a Young diagram λ with at most n rows. Fixing the torus $T \subset \mathrm{GL}(\mathbb{C}^n)$ of diagonal matrices the representation V of T is decomposable $V = \oplus_{\chi\in\mathbb{Z}^n} V_\chi$, where $tv = \chi(t)v$ for $v \in V_\chi$ and $\mathbb{Z}^n$ is the lattice of characters of the torus T. The lexicographically largest χ, say $\chi_0 =$

$(l_1, \ldots, l_n)$ is called the highest weight of V. The Young diagram λ has l_i boxes in the ith row. We have $\dim V_{\chi_0} = 1$. The unique up to scaling element of V_{χ_0} is called the *highest weight vector*.

Example 3.1 Let $V = S^d(\mathbb{C}^n)$ be the dth symmetric power of $\mathbb{C}^n$. It is an irreducible representation. The characters of the torus χ appearing in the representation correspond to n-tuples of nonnegative integers summing up to d. The highest weight is $(d, 0, \ldots, 0) \in \mathbb{Z}^n$. The highest weight vector is $e_1 \cdots e_1$. The associated Young diagram is a row with d boxes.

More generally, for any representation V of $\mathrm{GL}(\mathbb{C}^n)$ a vector $v \in V$ is called a *highest weight vector* if it is an image of a highest weight vector in some irreducible representation V_χ under a $\mathrm{GL}(V)$-equivariant map. If $V = \bigoplus_\lambda V_\lambda^{\oplus a_\lambda}$ is the decomposition of V, then a_λ equals the dimension of the vector space of highest weight vectors in V of weight λ. Furthermore, any highest weight vector of weight λ uniquely determines a subrepresentation $V_\lambda \subset V$. In other words, representation theory allows to replace a possibly large representation V by much smaller spaces of highest weight vectors.

The main observation is that if $X \subset S^c(\mathbb{C}^n)$ is $\mathrm{GL}(\mathbb{C}^n)$ invariant, then I_d is a representation of $\mathrm{GL}(\mathbb{C}^n)$, which is a subrepresentation of $S^d(S^c((\mathbb{C}^n)^*))$. The representation $S^d(S^c(\mathbb{C}^n)^*)$ is known as a plethysm. In general the formulas for its decomposition into irreducible representations are not known, and determining a combinatorial description for the multiplicities of irreducibles is Problem 9 in Stanley's list of open problems in algebraic combinatorics [32]. However, they are known up to $d \leq 5$ [23] and for fixed d and c there are algorithms to find such decompositions. For general d and $c = 3$ even the task of deciding positivity of a_λ is NP-hard, see [19].

If $S^d(S^c((\mathbb{C}^n)^*)) = \bigoplus_{\lambda \vdash dc}(S^\lambda)^{\oplus a_\lambda}$ is the decomposition, then we seek to find subrepresentations $(S^\lambda)^{\oplus b_\lambda} \subset (S^\lambda)^{\oplus a_\lambda}$ such that $I_d = \bigoplus_{\lambda \vdash dc}(S^\lambda)^{\oplus b_\lambda}$. This is equivalent to finding a b_λ-dimensional linear subspace in the space of highest weight vectors in $(S^\lambda)^{\oplus a_\lambda}$. We provide a database of polynomials in

$$S^d(S^c) := S^d(S^c((\mathbb{C}^n)^*))$$

that for each λ provides a basis of the highest weight space of $(S^\lambda)^{\oplus a_\lambda}$. Finally, we apply exact linear algebra methods to find which combinations of those vectors vanish on X. This is done by finding exact random points of X giving linear conditions on highest weight spaces.

To generate a basis of the highest weight vectors in $S^d(S^c)$ one may first generate a basis of highest weight vectors of weight λ in $(S^c)^{\otimes d}$. This is obtained by applying the Pieri rule. As writing this basis in terms of tensors is quite memory and time consuming, it is much better to simply remember it in terms of semistandard Young tableaux. The symmetrizing operator $(S^c)^{\otimes d} \to S^d(S^c)$ maps this basis to a generating set. Out of that set one chooses a basis, using linear algebra. There are many choices to pick a basis out of a generating set. We choose a random initial element in the generating set and add it to our basis. Then we choose another random element in the generating set, and check if it is linearly independent to the current basis. If it is we add this new element to the basis. We repeat this process until the number of basis elements equals the multiplicity of S^λ in $S^d(S^c)$. To check linear independence it is enough to be able to evaluate a polynomial corresponding to a given Young tableaux at many points. We apply a method that allows fast evaluation, without the necessity to expand the whole highest weight vector.

4 Algorithm

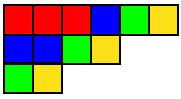
We describe here how to convert a Young tableau into a highest weight polynomial over the monomial basis. Evaluation at a point in X is then straightforward. In this way, if we can sample efficiently from X, we can evaluate the basis of a_λ many highest weight polynomials at a_λ sampled points, obtain a square matrix A of evaluations, and use linear algebra to compute $b_\lambda = \dim \ker A$.

We are given two natural numbers $d, c \in \mathbb{N}$. Moreover, we are given a so-called *isobaric Young tableau*. This is a top-left justified arrangement of dc many boxes with entries from $\{1, \ldots, d\}$ such that every entry appears exactly c times, see this example with $d = 4$, $c = 3$:

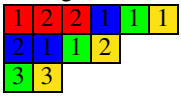
1	1	1	2	3	4
2	2	3	4		
3	4				

In fact, we may assume that the tableau is *semistandard*, which means that the entries are increasing within each column from top to bottom and they are nondecreasing within each row from left to right. The example above is semistandard. It is an open question whether or not it is possible to use semistandardness to get a speed-up in the running time, see [5] for the details.

We color the boxes in the same color if and only if they have the same number, and then we remove the numbers:



Let μ_i denote the number of boxes in column i. Let $\mathfrak{S}_k$ denote the symmetric group on k letters. A *column permutation assignment* is defined as an assignment of numbers to the boxes such that in each column i each number from $\{1, \ldots, \mu_i\}$ appears exactly once. For example, this is a column permutation assignment:

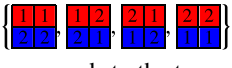


Each column in a column permutation assignment specifies a permutation, so we can define the *sign* of a column permutation assignment to be the product of the signs of the permutations that correspond to the columns. The example above has sign $1 \cdot (-1) \cdot (-1) \cdot 1 \cdot 1 \cdot 1 = 1$.

To each column permutation assignment T we assign the word $w(T)$ that is obtained by reading from T first all entries from one color, then from the next, and so on. The order of colors and the order in which we read entries from the same color does not matter, because we define two words of length cd to be *equivalent* if they arise from each other by permuting symbols within the block $\{1, \ldots, c\}$ or within $\{c + 1, \ldots, 2c\}$, and so on, or if they arise by permuting the d many blocks (in other words, they are equivalent if and only if they lie in the same orbit under the action of the wreath product $\mathfrak{S}_c \wr \mathfrak{S}_d$). The equivalence class of words $w(T)$ in the example above is $\{\{1, 2, 2\}, \{1, 1, 2\}, \{1, 1, 3\}, \{1, 2, 3\}\}$. To every column permutation assignment T, let $\kappa(T)$ denote the equivalence class of $w(T)$. Consider the vector space spanned by all possible $\kappa(T)$, where we interpret distinct $\kappa(T)$ to be linearly independent unit vectors.

Up to a simple rescaling of the basis, the highest weight polynomial in monomial presentation is then

$$\sum_{\text{column permutation assignment } T} \operatorname{sgn}(T)\kappa(T). \tag{†}$$

For example, let $c = d = 2$ and take the tableau  , then the set of column permutation assignments (written into the tableau) is  , so the sum (†) becomes $2\{\{1,1\},\{2,2\}\} - 2\{\{1,2\},\{1,2\}\}$. This corresponds to the tensor $2(e_1 \odot e_1) \odot (e_2 \odot e_2) - 2(e_1 \odot e_2) \odot (e_1 \odot e_2)$ in $S^2(S^2)$, where $a \odot b := \frac{1}{2}(a \otimes b + b \otimes a)$.

We compute the sum (†) in a brute force way and store the result in a file. This means that the file then contains the highest weight polynomial in the monomial basis. In particular, the evaluation of a highest weight polynomial from a file at any point is very efficient. A bottleneck in the computation (†) is the number of column permutation assignments. For example, if $d = 8$, $c = 4$, then for the Young diagram with row lengths $\lambda = (8, 8, 8, 8)$ we have 110 075 314 176 many column permutation assignments. Therefore, it is imperative to perform as few operations as possible for each summand. Here are a few points which accelerate the computation:

1. We use a Gray code to iterate through the sum so that the sign alternates for every summand. A Gray code is a way to iterate over the set of all permutations so that each permutation differs from the next by only a transposition. Therefore in each step the sign of the permutation flips and hence we never have to compute the sign of a permutation. The Gray code we use for each column is *Algorithm P* in [25, Section 7.2.1.2.] and we hardcode its list of permutations for each column.
2. We do not compute $\kappa(T)$, because it would require sorting a list of lists. Instead we use an *equivariant hash function*, which is a function that takes list of d lists of numbers that are each of length c and assigns this list a number (its so-called *hash value*) such that every reordering under the wreath product $\mathfrak{S}_c \wr \mathfrak{S}_d$ has the same hash value and in a way that lists of lists that are *not* equivalent under the wreath product action get different hash values (this last property is called *collision-freeness*). We can efficiently compute the hash value for a column permutation assignment and the equivariance of the hash function guarantees that words that are equivalent under the wreath product action are mapped to the same hash value. The collision-free hash function is chosen in a precomputation step.
3. To crucially speed up to computation the hash value is not computed for each summand, but the hash value is just *adjusted* at each step. This is possible, because the hash function is chosen as follows. Let $T_{i,j}$ be the jth entry in the ith colored block of T. Then, the hash function h is

$$h(T) := \sum_{i=1}^{d} \left(\sum_{j=1}^{c} \iota(T_{i,j}) \right)^k \bmod p$$

for a suitable $k \in \mathbb{N}$ and prime p, where $\iota(i)$ is the ith entry in a fixed array of random numbers from $\{0, \ldots, p-1\}$. Raising to the kth power is done by repeated squaring. The Gray code ensures that only two blocks are changed and only one entry in each block, which makes updating the hash value very efficient. To give a rough idea of the performance, after the precomputation of the hash function the summation over the 110 075 314 176 entries for $\lambda = (8, 8, 8, 8)$ takes only a few hours on a laptop.

Those ideas are incorporated into our implementation:

function WRITEHIGHESTWEIGHTPOLYTOFILE(isobaric Young tableau $\mathscr{T}$)
  Repeatedly choose k and p until the hash function h is collision free
  Initialize an array A of p many integers
  Initialize the hash value γ for the first summand in the sum (†)
  $\alpha := 1$
  for column permutation assignment T **do**
    $A[\gamma] := A[\gamma] + \alpha$
    efficiently update γ to be the hash value of the next summand
    $\alpha := -\alpha$
  end for
  Start a new empty highest weight polynomial file
  for basis vector v in $S^d(S^c)$ **do**
    Set $\beta := A[h(v)]$
    Append "+" and β and "$\cdot$" and v to the highest weight polynomial file
  end for
end function

This works well as long as p many integers can be stored in the memory and no hash collisions appear. For extremely large cases this is a problem, but then we just accept some hash collisions are store them, and whenever a hash collision appears the values are hashed again with a second hash function. This is also done with more than 2 hash functions for extremely large problems. The number of such hash functions is determined before running the algorithm and an estimate of the number of hash functions is obtained based on estimating the number of hash collisions using the birthday problem formula.

5 Numerical Methods

The algorithm that we have described in the last section is symbolic. It is based on exact computations, thus yielding exact results. As we have demonstrated by obtaining Theorem 2.1, it can go beyond the cases that the classical method relying on Gröbner basis [15, 27] can cope with.

Nevertheless, there are still limits to our algorithm with the current technology that numerical methods can surpass. For instance, our main theorem (Theorem 2.1) shows that no equations of degree at most 8 vanish on the variety of quartic symmetroids Q_4. But we could not find the minimal degree d, for which there are equations; i.e., such that $I_d \neq \emptyset$. Numerical methods, although not exact, can help to make an educated guess for those numbers. In this last section we want to explain this.

We first explain an approach on how to compute the degree of X. Thereafter, we will discuss that one can in principle extract the minimal d, such that $I_d \neq \emptyset$, from this computation. This poses new numerical challenges, however.

5.1 Cubic Symmetroids

Our algorithm was successful in case of the variety of cubic symmetroids Q_3. First we verified that there are no equations up to degree 10. This can be performed without problems on a laptop. In degree 11, the longest part is to transform the database of highest weight

vectors from tableau to polynomials. This takes a few days. However, this should be considered as a precomputation and this database, once created, can be used in future for any other problem concerning $S^{11}(S^3(\mathbb{C}^4))$. In particular, in this step we create six linearly independent polynomials of weight [15, 6, 6, 6]. Once this is done, our algorithm finds fast (within hours) the unique highest weight polynomial of degree 11 in the ideal. This is a unique linear combination of the six highest weight vectors of weight [15, 6, 6, 6] that vanishes on Q_3. We do not present this polynomial in the article, as it is quite large. It may be downloaded from (http://www.math.uni-konstanz.de/~michalek/eqnfile.txt). Using the same algorithm we also determined all of the degree 12 polynomials that appear in the ideal of Q_3. The obtained representations corresponded exactly to those that appear in the tensor product [15, 6, 6, 6] $\otimes$ [3] (and correspond to partitions with at most four entries). This motivates the following definition and question.

Definition 5.1 A variety $X \subset \mathbb{C}^n$ with a G action is called *G-principle* if the ideal $I(X)$ is generated by one, irreducible representation of G.

We note that to provide the whole ideal of a G-principle variety, it is enough to know the group action and provide only one polynomial that generates the irreducible representation.

Question 5.2 Is the variety Q_3 $GL(4)$-principle?

The highest weight polynomials in degree 12 may also be downloaded from (http://www.math.uni-konstanz.de/~michalek/eqnfile.txt).

5.2 A Numerical Approach for Computing the Degree

We make a numerical computation to determine the degree of the symmetroid Q_3 (1). The approach described in this section can easily be generalized to the general situation involving X, but for simplicity we will stick to the special situation with Q_3.

We use coordinates by setting the first matrix to be $A_0 = \operatorname{diag}(a_1, a_2, a_3)$, where a_1, a_2, a_3 are variables. Then, we have the following situation:

$$\begin{aligned} f : V &\to W, \\ a = (a_1, a_2, a_3, A_1, A_2, A_3) &\mapsto \text{coefficients of } \det(x_0A_0 + x_1A_1 + x_2A_2 + x_3A_3) \end{aligned}$$

and $\dim V = 21$ and $\dim W = 20$. Here, coefficients means the coefficients of a polynomial in x. To determine the dimension of Q_3 we compute the rank of the Jacobian matrix of f at a random point. We get that this rank is 16 (see [6]). Therefore, $\dim Q_3 = 16$. This implies that the dimension of the fibers of f for a general point $h \in Q_3$ is $\dim f^{-1}(h) = 5$. The degree of Q_3 multiplied by the degree of a general fiber $f^{-1}(h)$, $h \in Q_3$, is thus the number of isolated complex solutions of the following system of 16 polynomial equations in the 16 variables $b = (b_1, \ldots, b_{16})$:

$$R \cdot f(a) = r \quad \text{and} \quad a = S \cdot b + s, \tag{2}$$

where $R \in \mathbb{C}^{16\times 20}$, $r \in \mathbb{C}^{16}$, $S \in \mathbb{C}^{21\times 16}$ and $s \in \mathbb{C}^{21}$ are chosen randomly. We can exploit monodromy by varying the coefficients of R and r in loops and numerically tracking the solutions along those loops. This produces new solutions for (2). The details of this technique are, for instance, explained in [18]. An initial solution for this system can easily be generated. In the computation it is enough to keep one point in each fiber $f^{-1}(h)$, $h \in Q_3$,

so that we do not have to compute the degree of a general fiber to get the degree of Q_3. A concrete implementation is at [6].

We can use the same method for the quartic symmetroid Q_4. In this case we have $\dim V = 34$, $\dim W = 35$, and $\dim Q = 25$. The dimension of the fibers of f for a general point is 9. However, we could not finish this computation. We stopped the computation manually. At this point 849998 solutions for (2) had been found.

5.3 The Degree of Q_3

We used `HomotopyContinuation.jl` [8] for the computation of degrees of symmetroids. The code for the cubic symmetroid can be found at [6]. We compute 305 solutions of the system (2) for Q_3. On a laptop this takes about two minutes. We use the certification method [7] implemented in `HomotopyContinuation.jl` to show that the 305 solutions we have found correspond to 305 true distinct solutions for (2). This confirms that the degree of Q_3 is at least 305. It was proven in [34] that the degree is equal to 305.

5.4 Further Directions

Here, we explain an approach for answering the following question: Given a homogeneous polynomial map $f : \mathbb{C}^a \to \mathbb{C}^b$ what is the dimension of the vector space I_d of polynomials of degree d that vanish on the image?

The basic idea is this: suppose that we have run the algorithm from the previous section. Then, we have found a linear space L in $W = S^c(\mathbb{C}^n)$ and points $w_1, \ldots, w_\delta \in X \cap L$, such that δ is the degree of X. Any equation that vanishes on X also vanishes on the w_i. We now discuss when the reverse is true. If this holds, we can check numerically by solving a system of linear equations, whether or not there are equations of a fixed degree d vanishing on the X. Note that this does *not* yield equations for X. Furthermore, we can use coordinates for L for doing the linear algebra. This kind of *dimensionality reduction* can provide a significant reduction in computational complexity. These ideas first appeared in [21].

Let us write $b := \dim(S^c(\mathbb{C}^n))$ and the image of f is invariant under the action of $\mathrm{GL}(\mathbb{C}^n)$. We ask for the dimension of I_d, that is the degree d part of I. It should be emphasized that this is naturally a problem in linear algebra, as I_d is a vector space. Each point $x \in X$ determines a linear condition on the space $S^d(\mathbb{C}^b)$, giving rise to a hyperplane containing I_d. In fact, I_d is the intersection of all such hyperplanes. For dimensional reasons, it would be enough to pick consecutively random $x \in X$ and intersect the hyperplanes in $S^d(\mathbb{C}^b)$, until the intersection stabilizes. This is indeed sometimes done in practice, but the main problem is the large dimension $\binom{d+b-1}{d}$ of the space $S^d(\mathbb{C}^b)$. The method we describe is particularly useful if:

1. the codimension of X is small,
2. the degree of X is small.

From now on we work in the projective space $\mathbb{P}(\mathbb{C}^b)$ and consider X as a projective variety. Let $e := \operatorname{codim} X$.

We pick a random subspace $L = \mathbb{P}^e \subset \mathbb{P}(\mathbb{C}^b)$. By Bertini's theorem $\mathbb{P}^e$ intersects X in $\delta = \deg X$ many smooth points

$$S = L \cap X.$$

A random linear form h_1 is not a zero divisor in the ring $\mathbb{C}[y_1, \ldots, y_b]/I$, hence we have an exact sequence:

$$0 \to \mathbb{C}[y_1, \ldots, y_b]/I \to \mathbb{C}[y_1, \ldots, y_b]/I \to \mathbb{C}[y_1, \ldots, y_b]/(I + (h_1)) \to 0,$$

where the first map is multiplication by h_1. Hence, the Hilbert series of $I + (h_1)$ equals $(1 - t)$ times the Hilbert series of I. In particular, the numerators of the Hilbert series are the same. The number of linear $h_1, \ldots, h_l$ such that h_{i+1} is not a zero divisor modulo $I + (h_1, \ldots, h_i)$ for every $0 \leq i < l$ is governed by the depth of the (localization of the) ring $\mathbb{C}[y_1, \ldots, y_b]/I$. Depth is always at most equal to the dimension and the cases when equality holds are called Cohen–Macaulay.

After choosing $e = \operatorname{codim} X$ many linear forms, we arrive at the ring

$$\mathbb{C}[y_1, \ldots, y_b]/(I + (h_1, \ldots h_e))$$

which describes S as a *projective* scheme. In general, the ideal $(I + (h_1, \ldots h_e))$ may have an embedded component at zero, however this again does not happen if X is arithmetically Cohen–Macaulay (which means that its coordinate ring is Cohen–Macaulay). In practice, we next choose an affine linear form and add it to the ideal to represent S as a finite subset of an affine space.

In particular, if our variety X is arithmetically Cohen–Macaulay then the Hilbert function of the finite set S in fact encodes the numerator of the Hilbert series of X. In any case a nonzero element in I_d gives rise to a nonzero element in $I(S)_d$. As long as $I(S)_d = 0$ we also have $I_d = 0$, hence we do not have to look for equations in those degrees. Furthermore, in smallest degree d such that $I(S)_d \neq 0$ we have $\dim I_d = \dim I(S)_d$ in the Cohen–Macaulay case.

Example 5.3 In the following example we construct a toric ring of small depth. Consider the map for $x = (x_1, x_2, x_3, x_4)$:

$$x \mapsto \left(x_1x_2^4, x_1x_2^3x_3, x_1x_2x_3^3, x_1x_3^4, x_1x_2^4x_4, x_1x_2^3x_3x_4, x_1x_2x_3^3x_4, x_1x_3^4x_4\right).$$

The image is a toric variety of projective dimension two and degree eight. It is minimally generated by nine quadrics and twelve cubics. If we intersect the image with two affine linear forms we obtain eight points. These eight points do not contribute to new linear equations, however their ideal has thirteen minimal generators in degree two.

Thus, if we know S, we may estimate $\dim I_d$ using linear algebra approach described above, but now we deal with points in the ambient space of dimension $e = \operatorname{codim} X$. Hence, we have to solve $\deg X$ many *linear* equations in $\binom{d+e-1}{d}$ many variables.

Numerical methods help us both: to obtain S and to solve the linear equations. To generate $\mathbb{P}^e$ we take a span of $e + 1$ many random/general points of X. We obtain $\mathbb{P}^e$ together with $e + 1$ many points of S. To generate all of $S = \{w_1, \ldots, w_\delta\}$ we apply the monodromy method from the previous subsection.

Now a new problem arises. As our points are just approximations of the points in S, if we ask for the rank of the matrix associated to the system of linear equations, symbolically it will always be nondegenerate. Furthermore, the matrix we obtain depends on the choice of the basis of degree d polynomials we take. The idea is to look at the singular values of the associated matrix in the basis. This allows us to discover the rank of the approximated matrix.

We plan to apply the approach, that we have just described, to computations involving GL invariant families of polynomials.

Acknowledgements P. Breiding has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 787840) and funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 445466444. C. Ikenmeyer was supported by the DFG grant IK 116/2-1.

We thank Bernd Sturmfels for motivating questions. These questions not only inspired us, but also revealed bottlenecks of our algorithms. We also thank anonymous referees, whose suggestions helped us to improve the exposition of the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Abdesselam, A.: Feynman diagrams in algebraic combinatorics. Séminaire Lotharingien de Combinatoire B49c (2003)
2. Abdesselam, A., Ikenmeyer, C., Royle, G.: 16,051 formulas for Ottaviani's invariant of cubic threefolds. J. Algebra **447**, 649–663 (2016)
3. Bates, D.J., Oeding, L.: Toward a salmon conjecture. Exp. Math. **20**, 358–370 (2011)
4. Bläser, M., Dörfler, J., Ikenmeyer, C.: On the complexity of evaluating highest weight vectors. arXiv:2002.11594 (2020)
5. Bläser, M., Dörfler, J., Ikenmeyer, J.: On the complexity of evaluating highest weight vectors. In: Proceedings of the 36th Computational Complexity Conference, CCC '21, Dagstuhl, DEU (2021)
6. Breiding, P., Ikenmeyer, C., Michalek, M., Hodges, R.: Symmetroids. https://www.JuliaHomotopyContinuation.org/examples/symmetroids/ (2021)
7. Breiding, P., Rose, K., Timme, S.: Certifying zeros of polynomial systems using interval arithmetic. arXiv:2011.05000 (2020)
8. Breiding, P., Timme, S.: Homotopycontinuation.Jl: a package for homotopy continuation in Julia. In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds.) Mathematical Software – ICMS 2018. Lecture Notes in Computer Science, vol. 10931, pp. 458–465. Springer, Cham (2018)
9. Bürgisser, P., Christandl, M., Ikenmeyer, C.: Even partitions in plethysms. J. Algebra **328**, 322–329 (2011)
10. Bürgisser, P., Ikenmeyer, C.: Geometric complexity theory and tensor rank. In: Proceedings 43rd Annual ACM Symposium on Theory of Computing 2011, pp. 509–518. Association for Computing Machinery, New York (2011)
11. Bürgisser, P., Ikenmeyer, C.: Explicit lower bounds via geometric complexity theory. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC '13, pp. 141–150. Association for Computing Machinery, New York (2013)
12. Bürgisser, P., Ikenmeyer, C., Panova, G.: No occurrence obstructions in geometric complexity theory. J. Amer. Math. Soc. **32**, 163–193 (2019)
13. Cheung, M.-W., Ikenmeyer, C., Mkrtchyan, S.: Symmetrizing tableaux and the 5th case of the Foulkes conjecture. J. Symb. Comput. **80**, 833–843 (2017)
14. Chiantini, L., Hauenstein, J.D., Ikenmeyer, C., Landsberg, J.M., Ottaviani, G.: Polynomials and the exponent of matrix multiplication. Bull. Lond. Math. Soc. **50**, 369–389 (2018)
15. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra. Undergraduate Texts in Mathematics. Springer, New York (2013)

16. Daleo, N.S., Hauenstein, J.D., Oeding, L.: Computations and equations for Segre-Grassmann hypersurfaces. Port. Math. **73**, 71–90 (2016)
17. Dörfler, J., Ikenmeyer, C., Panova, G.: On geometric complexity theory: multiplicity obstructions are stronger than occurrence obstructions. In: 46th International colloquium on automata, languages, and programming, ICALP 2019, July 9–12, 2019, Patras, Greece, pp. 51:1–51:14, 2019. Journal Version: SIAM J. Appl. Algebra Geom. 4, 354–376 (2020)
18. Duff, T., Hill, C., Jensen, A., Lee, K., Leykin, A., Sommars, J.: Solving polynomial systems via homotopy continuation and monodromy. IMA J. Numer. Anal. **39**, 1421–1446 (2019)
19. Fischer, N., Ikenmeyer, C.: The computational complexity of plethysm coefficients. Comput. Complex. **29**, 8 (2020)
20. Fulton, W., Harris, J.: Representation Theory: a First Course Graduate Texts in Mathematics, vol. 129. Springer, New York (2013)
21. Hauenstein, J.D., Ikenmeyer, C., Landsberg, J.M.: Equations for lower bounds on border rank. Exp. Math. **22**, 372–383 (2013)
22. Ikenmeyer, C., Kandasamy, U.: Implementing geometric complexity theory: on the separation of orbit closures via symmetries. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, pp. 713–726. Association for Computing Machinery, New York (2020)
23. Kahle, T., Michałek, M.: Plethysm and lattice point counting. Found. Comput. Math. **16**, 1241–1261 (2016)
24. Klein, F.: A comparative review of recent researches in geometry. Bull. Amer. Math. Soc. **2**, 215–249 (1893)
25. Knuth, D.: The Art of Computer Programming, vol. 4a: Combinatorial Algorithms, Part 1. Addison-Wesley, Reading (2011)
26. Kumar, S.: A study of the representations supported by the orbit closure of the determinant. Compos. Math. **151**, 292–312 (2011)
27. Michałek, M., Sturmfels, B.: Invitation to Nonlinear Algebra. Graduate Studies in Mathematics, vol. 211. American Mathematical Society, Providence (2021)
28. Oeding, L., Raicu, C.: Tangential varieties of Segre–Veronese varieties. Collectanea Math. **65**, 303–330 (2014)
29. Ottaviani, G.: Five lectures on projective invariants. arXiv:1305.2749 (2013)
30. Raicu, C.: 3×3 minors of catalecticants. Math. Res. Lett. **20**, 745–756 (2013)
31. Ranestad, K., Sturmfels, B.: Twenty-seven questions about the cubic surface. Le Math. **75**, 411–424 (2020)
32. Stanley, R.P.: Positivity problems and conjectures in algebraic combinatorics. In: Mathematics: Frontiers and Perspectives, pp. 295–319. American Mathematical Society, Providence (2000)
33. Sturmfels, B.: Problems for the Einstein Program proposal. https://math.berkeley.edu/~bernd/Einstein2014.pdf (2014)
34. Vainsencher, I.: Hypersurfaces with up to six double points. Commun. Algebra **31**, 4107–4129 (2003)