The CompGIT package: a computational tool for Geometric Invariant Theory quotients

Robert Hanson and Jesus Martinez-Garcia

April 2025

Abstract

We describe `CompGIT`, a SageMath package to describe Geometric Invariant Theory (GIT) quotients of projective space by simple groups. The implementation is based on algorithms described by Gallardo–Martinez-Garcia–Moon–Swinarski [Gal+23]. In principle the package is sufficient to describe any GIT quotient of a projective variety by a simple group — in practice it requires that the user can construct an equivariant embedding of the polarised variety into projective space. The package describes the non-stable and unstable loci up to conjugation by the group, as well as describing the strictly polystable loci. We discuss potential applications of the outputs of `CompGIT` to algebraic geometry problems, a well as suggesting directions for future developments.

1 Introduction

Understanding quotients of projective varieties by group actions is a fundamental problem in mathematics that goes back to the 19th century German algebra and geometry school and Hilbert's 14th problem [Hil02]. Such quotients are of interest in ring theory and algebraic geometry, but also find applications in fields as diverse as graph theory, coding theory, dynamical systems, and computer vision [DK15]. Within algebraic geometry, this often comes in the form of a Geometric Invariant Theory (GIT) quotient — an essential tool for constructing *moduli spaces* that parametrise objects such as algebraic curves [DM69], K-polystable Fano varieties [LX19; GM19; GMS21; MPZ24; Abb+25] and K3 surfaces [Sha80; GMZ18; Laz09].

`CompGIT` is a Sagemath package [The25] which aims to significantly simplify the description of GIT quotients of the form $\mathbb{P}^n /\!\!/ G$ where G is a simple group. In principle — but unfortunately not always in practice — such quotients can recover any G-structured GIT problem. The code in `CompGIT` is based on algorithms and code that first appeared in [Gal+23].

1.1 GIT quotients

We recall the fundamentals of GIT and their role in our code. Let k be an algebraically closed field of characteristic 0. If a projective variety $X = \operatorname{Proj}(R)$ is acted upon by a linear algebraic group G, we want to consider the quotient X/G. For this quotient to be well behaved in general, it is necessary to assume that G is reductive [Nag60]. In Hilbert's approach we consider a G-linearised very ample line bundle L so that $R = \bigoplus_m H^0(X, L^{\otimes m})$ and consider the invariant subring

$$R^G = \bigoplus_m H^0(X, L^{\otimes m})^G \subseteq R.$$

If R^G is finitely generated (which is the case when G is reductive), then the inclusion $R^G \subseteq R$ induces a rational map $\pi \colon X \dashrightarrow X /\!\!/_L G =: \operatorname{Proj}(R^G)$. The central characteristic of GIT is that $X /\!\!/_L G$ is projective variety, called the *GIT quotient* of the G-space X [MFK94].

A geometric description of the quotient is determined by the following stability conditions. The *semistable locus* $X^{ss} \subset X$ is the open subset where π is defined. The *unstable locus* is the complementary closed subset $X^{us} = X \setminus X^{ss}$. The *stable locus* $X^s \subseteq X^{ss}$ is the subset of all semistable closed orbits with finite stabilisers. The *polystable locus* X^{ps} is the subset of all orbits that are closed in the semistable locus. Thus we have

the natural inclusions $X^s \subseteq X^{ps} \subseteq X^{ss}$. Then, by construction, the polystable orbits are in one-to-one correspondence with points in $X /\!\!/_L G$, which is why, in geometric applications, we are primarily interested in finding stable orbits and strictly polystable orbits. However, finding GIT semistable orbits has some interest too, e.g. when working modulo S-equivalence, or when describing the semistable substack of some moduli stack [AHH23].

As L is very ample it determines an embedding $X \subseteq \mathbb{P}^N$ for $N = h^0(X, L) - 1$, so that $L = O_X(1)$. Being G-linearised implies that we have an embedding $G < \mathrm{Aut}(\mathbb{P}^N)$ as a subgroup, so that the action of G on $\mathbb{P}^N$ restricts to the action of G on X. Thus, we can define GIT (semi/poly)stability for G-orbits of $\mathbb{P}^N$ and, in fact, a point $p \in X \subset \mathbb{P}^N$ is G-(semi/poly)stable as point of X if and only if it is G-(semi/poly)stable as a point of $\mathbb{P}^N$. Thus we have the identifications

$$X^{ss} = X \cap \left(\mathbb{P}^N\right)^{ss}, \qquad X^{ps} = X \cap \left(\mathbb{P}^N\right)^{ps}, \qquad X^s = X \cap \left(\mathbb{P}^N\right)^s.$$

To solve a GIT problem, one may then simply replace X by $\mathbb{P}^N$, determine the (semi/poly)stable orbits and then restrict to X. This is the approach that the computations of `CompGIT` enables. In practice, given arbitrary X, it may be difficult to solve the required G-equivariant embedding problem, although many applications do come with natural embeddings, for instance when $X \subseteq \mathbb{P}^k$ is a hypersurface. Having a solution for $\mathbb{P}^N$ also works well to describe étale covers of $\mathbb{P}^N$ branched at hypersurfaces, since the moduli of hypersurfaces can be interpreted as the moduli of étale covers of $\mathbb{P}^N$ branched at a hypersurface.

Further considerations need to be made with regards to the group G. The algorithms presented in [Gal+23], which `CompGIT` implements, are optimised for a simple group G, making use of the rich action of the Weyl group, which simplifies both outputs and computations. In [Gal+23], the authors explain how a simplification of the algorithms can be obtained for general reductive group G. Since often in applications G is simple, we make that assumption in the implementation of `CompGIT`.

In view of the above, we assume that $X = \mathbb{P}^N = \mathbb{P}(V)$, where V is a vector space and G is a simple group acting linearly on X. In other words, V is a G-representation. The package `CompGIT` computes:

- The unstable locus $X^{us} = X \setminus X^{ss}$.

- The *non-stable locus* $X^{ns} = X \setminus X^s$

- The *strictly polystable locus* $X^{sps} = X^{ps} \setminus X^s$ with respect to a maximal torus T of G.

The details of the algorithms in `CompGIT` can be found in [Gal+23] and we omit them here. This document serves as a user manual for the package, with an emphasis on concrete examples of computations and suggestions for future improvements, in the hope that they attract interest from the community. Further examples and outputs are also available on the `GitHub` homepage of our package [HM].

One way to explore `CompGIT` is to go through the examples below and the ones online and refer back to the text below for additional explanations. While the code in the examples is mostly self-explanatory, some basic familiarity with `SageMath` is helpful, as it is to have some understanding of the geometry behind the examples and GIT more generally.

1.2   Related implementations

There have been other attempts at describing the unstable loci computationally, most notably by A'Campo and Popov [DK15, Appendix C] using the Computer Algebra Systems (CASs) `pari-gp` and `LiE`. Mathematically, the A'Campo-Popov implementation focuses on the affine cone (the *nullcone*, in their language) $\mathcal{N}_{G,V}$ over the unstable locus $(\mathbb{P}^N)^{us} = (\mathbb{P}(V))^{ss}$ where V is the underlying affine vector space. The nullcone can be given a stratification according to 'how unstable' each orbit is and it plays an important role in the theory of invariants [CM93]. We refer to [DK15, Appendix C] for the details, where algorithms to determine the stratification of the Nullcone $\mathcal{N}_{G,V}$ are presented for any reductive group G, while their implementation in `pari-gp` and `LiE` assumes G is semisimple. To the best of our understanding, their algorithms do not consider the non-stable/semistable nor the strictly polystable loci and may take longer, since they need to distinguish within the different orbits of $(\mathbb{P}^N)^{ss}$. In practice, the most important difference between their work and ours is probably that `Sagemath` is currently a more accessible CAS. It would be desirable if portability between `pari-gp`, `LiE` and `Sagemath` was explored, so that their output could be compared to ours.

1.3   Organisation of the paper

We give a quick summary of what follows. In §2 we give a quick summary of the installation procedure for `CompGIT`. Section 3 gives a quick summary of conventions and facts from simple groups needed to understand the use of `Sagemath` code. In §4, we consider a very simple well-known example of GIT quotient (moduli of plane cubic curves) to illustrate how `CompGIT` solves a GIT problem. Then, we provide a list of examples of geometric interest that `CompGIT` can attack, some of which have never been considered in the literature, discussing their feasibility. Finally, in §5, we discuss possible generalisations and optimisations, in the hope that other researchers consider improving `CompGIT`. At this point is worth mentioning that `CompGIT` has a GNU General Public License 3.0, a free, copyleft license. Thus, the developers welcome improvements and changes to the software and algorithms by the community.

1.4   Acknowledgements

2   Installation

`CompGIT` is a `SageMath` package written in `Python 3.0`. The `SageMath` computer algebra system is a free open-source mathematics software system designed for applications in many areas of mathematics, including algebra, geometry and combinatorics. `SageMath` builds on top of many existing packages, such as `NumPy`, `SciPy`, `matplotlib`, `Maxima`, `FLINT` and `R`. The `SageMath` installation guide [Sag] contains binaries and source code — note that building from source code has the advantage of various computer-specific optimisations. Our `GitHub` homepage [HM] contains detailed installation instructions via pip. After successful installation each session in `CompGIT` starts with

```
sage: from CompGIT import *
```

and if using `CompGIT` within a sage file, simply include

```
from CompGIT import *
```

at the top of the file.

   All sample code in this article assumes first the execution of this command. We also note that the prefix *sage:* is a prompt provided by the `SageMath` console that allows one to distinguish between the inputs and the outputs.

3   Simple groups

A reductive linear algebraic group G is called *simple* when every smooth connected normal subgroup is trivial or equal to G. Such groups are fundamental building blocks in group theory, as demonstrated by decomposition results such as the Jordan–Hölder theorem.

3.1 Root systems

The fundamental properties of simple groups are encoded by a set of combinatorial data known as a *root system*. This consists of a choice of maximal torus $T \subset G$, a lattice of *one-parameter subgroups* $N = \mathrm{Hom}(\mathbb{G}_m, T)$, a set of *characters* $M = \mathrm{Hom}(T, \mathbb{G}_m)$ and a perfect pairing $\langle \cdot, \cdot \rangle : M \times N \to \mathbb{Z}$. The symmetries of N are represented by the *Weyl group*, whose action admits generating regions $F \subset N$ called *fundamental chambers*. Let U be a G-representation. As a T-module U decomposes as a direct sum of eigenspaces $U_\chi = \{u \in U : t \cdot u = \chi(t)u\}$, where $\chi : T \to \mathbb{G}_m$ is a character. If $U_\chi \neq 0$, then χ is called a *weight*, and $u \in U_\chi$ is a *weight vector*. In the special case where $U = \mathrm{Lie}(G)$, the non-zero weights χ are called the *roots* of G. We then denote the finite set of roots by Φ. Figure 3.1 displays the root system of G_2.
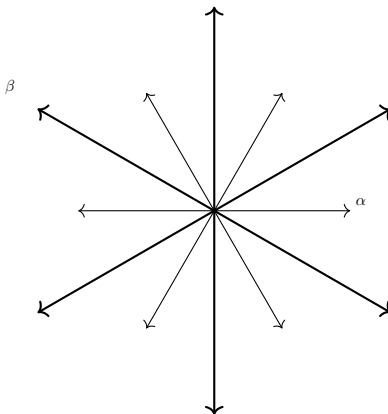

Figure 1: The root system of the exceptional group G_2. Our choice of simple roots (as per Section 3.3 and Table 1) are labeled by α and β.

Classical works of Cartan [Car29], Chevalley [Che48] and Killing [Kil89] show that reductive simple groups over an algebraically closed field are classified by the *Dynkin type* of their root system, which takes values in one of A_n, B_n, C_n, D_n, E_6, E_7, E_8, F_4, G_2 where $n \geq 2$ is an integer. Weyl [Wey46] named the Dynkin types A_n, B_n, C_n and D_n *classical*, whereas E_6, E_7, E_8, F_4, G_2 are called *exceptional*.

In order to understand the input and output of the programme, it is helpful to recall some additional representation theory of simple groups. Let G be a simple group and V a G-representation. Up to conjugation, G has a unique Borel subgroup $B \cong T \ltimes U$ where T is a maximal torus and U is the unipotent radical. When G is embedded into GL_n, we can assume it is written in matrix form and, up to conjugation, we can write B as upper triangular matrices, with T given as diagonal elements and U given as strict upper triangular matrices. The Borel subgroup B leaves some one-dimensional subspaces of V invariant. If $v \in V$ is a non-zero vector of such an invariant one-dimensional subspace, then we can write $g \cdot v = t_g v$ for some $t_g \in \mathbb{G}_m$. This defines a group homomorphism $t : B \to \mathbb{G}_m$. The *weight* w of an invariant vector v is the character $t_v : T \to \mathbb{G}_m$ obtained by restricting t to T. Note we can view $w \in \mathbb{Z}^n$ where $n = \dim(T)$ is the rank of G. The subgroup B acts on the eigenspaces V_χ, sending them to other eigenspaces. It can be show that if V is irreducible there is a unique eigenspace V_w corresponding to a unique weight w such that $B \cdot V_w \subseteq V_w$. The weight w is called the *highest* weight. Moreover, if a weight w is the highest weight of an irreducible G-representation V, then V is uniquely determined by w up to isomorphism.

Thus, when considering the action of G on $X = \mathbb{P}^N = P(V)$, we may assume that V is a G-reducible representation and it is determined by a (highest) weight $w \in \mathbb{Z}^n$ where n is the rank of G and it is sufficient to give the highest weight w of V to determine V.

Sagemath's standard library is already able to deal with G-representations when G is a simple group and CompGIT builds on it. For instance, to introduce a Spin(8)-representation of highest weight ω_3, we would write:

```
Phi = WeylCharacterRing("C4")
representation= Phi(Phi.fundamental_weights()[3])
```

In order to create a Spin(4)-representation of highest weight $(2, 0)$, one may write:

```
Sp4 = WeylCharacterRing("C2")
representation3 = Sp4(2,0);
```

It may be worth to keep in mind a more geometric example. Consider $G = \mathrm{SL}_{n+1}$ acting naturally on $\mathbb{P}^n$. Let $V = H^0(\mathbb{P}^n, \mathcal{O}_{\mathbb{P}^n}(d))$ where $d \geqslant 1$, i.e. V represents homogeneous polynomials of degree d, i.e. hypersurfaces of degree d in $\mathbb{P}^n$. Note that the G-action on $\mathbb{P}^n$ induces a G-representation on V and, in fact, its maximal weight is $w = (d, 0, \ldots, 0) \in \mathbb{Z}^{n+1}$. Let $X = \mathbb{P}(V)$. The GIT problem $X /\!\!/ \mathrm{SL}_{n+1}$ describes equivalent hypersurfaces of degree d up to PGL_{n+1}-isomorphism.

3.2 Implementation in CompGIT

The combinatorial characterisation of simple groups provided by root systems is both pervasive throughout group theory [Bou02] and essential to computations within `CompGIT`. We represent various properties of the root system as attributes within the class `SimpleGroup`. Our code utilises inputs from the `Weyl Groups` standard package included in `SageMath` — the canonical method of representing root system data within `SageMath`. Below are some elementary computations with the `SimpleGroup` class applied to the root system A_2.

```
sage: G=SimpleGroup("A", 2)
sage: G.group_type()
'A'
sage: G.rnk()
2
sage: G.fundamental_chamber_generators()
[2 1]
[1 2]
sage: G.pairing(1, [1,2])
(-1, 2)
```

3.3 Root system conventions

We specify various coordinate systems that we use to represent root system data. We closely follow the terminology outlined by Fulton and Harris [FH13]. The *H-coordinates* on the lattice N of one-parameter subgroups are defined by basis elements given by the set $\{H_i\}_{i=1,\ldots,n+1}$ of $(n+1) \times (n+1)$ matrices with only one non-zero (i, i)th element of unitary size. The *L-coordinates* on the set M of characters are dual to the H-coordinates. For type A root systems we introduce an auxiliary coordinate system given by $\{T_i := H_i - H_{i+1}\}_{i=1,\ldots,n}$, where the reduction from $n+1$ to n basis coordinates is to account for the fact that type A weights of a one parameter subgroup add to 0.

We implement the symmetries of the Weyl group with respect to certain choices that we now describe. We begin by selecting a set of *simple roots* $\Delta \subset \Phi$ which are roots that cannot be written as a sum of two positive roots. To our choice of Δ the corresponding fundamental chamber F_Δ is then given by vectors $v \in \mathbb{R}^n$ that satisfy the positivity condition $\langle \alpha, v \rangle \geq 0$ for all $\alpha \in \Delta$. We then define *gamma coordinates* that are defined by a basis set of *gamma rays* — a class of vectors whose positive span coincides with F_Δ. To specify our choice of Δ, consider the standard Euclidean basis $e_1, \ldots e_n$ for $\mathbb{R}^n$ and refer to the choices of simple roots provided by Table 1. Our simple roots for G_2 are also depicted in Figure 3.1.

4 Finding the unstable and non-stable loci

Having explained the conventions we use to represent group data, we can now demonstrate how `CompGIT` solves GIT problems of geometric interest. We first run a toy example: the moduli space of plane cubics in $\mathbb{P}^2$ (§4.1), to familiarise the reader with `CompGIT` and demonstrate how one reads and interprets the outputs (§4.2). We then discuss more complex applications (§4.3) that either recover problems covered in the literature or provide new problems of geometric interest.

Dynkin type	Simple roots
A_n	$e_i - e_{i-1}$ for $i = 1, \dots n$
B_n	$e_i - e_{i-1}$ for $i = 1, \dots n-1$ and e_n
C_n	$e_i - e_{i-1}$ for $i = 1, \dots n-1$ and $2e_n$
D_n	$e_i - e_{i-1}$ for $i = 1, \dots n-1$ and $e_{n-1} + e_n$
F_4	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \\ -1/2 \end{bmatrix}$
E_6	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \sqrt{3}/2 \end{bmatrix}$
E_7	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ -1/2 \end{bmatrix}, \begin{bmatrix} -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ \sqrt{2}/2 \end{bmatrix}$
E_8	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ -1/2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1/2 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}$
G_2	$\begin{bmatrix} 1 \\ -3/2 \end{bmatrix}, \begin{bmatrix} 0 \\ \sqrt{3}/2 \end{bmatrix}$

Table 1: Choice of simple roots for simple groups

4.1   Moduli of plane cubics

A plane cubic curve in $\mathbb{P}^2$ is in one-to-one correspondence with cubic forms in 3 variables (prior to any identification via isomorphisms). Thus, their parameter space can be identified with the vector space $V =$

$H^0(\mathbb{P}^2, \mathcal{O}_{\mathbb{P}^2}(3))$. An element $f \in V$ can be seen as as a homogeneous polynomial

$$f = \sum a_I x^I$$

where the sum is over all $I = (d_0, d_1, d_2) \in \mathbb{Z}_{\geqslant 0}^3$ such that $d_0 + d_1 + d_2 = 3$ and such that $x_I = x_0^{d_0} x_1^{d_1} x_2^{d_2}$ and $a_I \in \mathbb{C}$. The natural action of $G = \mathrm{SL}_3(\mathbb{C})$ on $\mathbb{P}^2$ (by acting on the coordinates (x_0, x_1, x_2) induces an action on V (where $g \in \mathrm{SL}_3(\mathbb{C})$ sends $f(x_0, x_1, x_2)$ to $f(g \cdot (x_0, x_1, x_2)))$. Thus V is a G-representation and we have that $\mathbb{P}(V) /\!\!/ G \cong \mathbb{P}(V) /\!\!/ \mathrm{Aut}(\mathbb{P}^2) = \mathbb{P}(V) /\!\!/ \mathrm{PGL}_3$. From the exposition in §3.1, it follows that x_0^3 is a weight. In fact, under the right choice of coordinates for $B \cong T \ltimes U$ (e.g. by choosing T to be the diagonal entries in SL_3 and U to be the lower-diagonal entries), x_0^3 is the highest weight. Recall that the highest weight uniquely determines the G-representation. After loading `CompGIT` (see §2), we can create this representation by determining the type of the group `A2` and the weight `(3,0,0)`

```
sage: Phi = WeylCharacterRing("A2")
sage: representation= Phi(3,0,0)
```

The next line is to initialise the problem:

```
sage: P = GITProblem(representation)
```

The object `P`, of class `GITProblem`, has all the information required to 'solve' the problem. It contains a lot of information that can be 'requested' from the class via different class methods. It is possible that some of these can be used to obtain further information from the quotient $\mathbb{P}(V) /\!\!/ G$ (e.g. information about its cohomology, 'a la Kirwan' [Kir82]). The information inside `P` includes a list of one-parameter subgroups that suffices, via the Hilbert-Mumford criterion [MFK94, Theorem 2.1], to determine its stability.

We can now ask `CompGIT` to find all non-stable elements. In this example, this is fast but in some cases this can take a long time (see [Gal+23, Table 1] for a discussion of times for a large number of examples).

```
sage: P.solve_non_stable()
```

The next step is to print the output. Since it is already computed in the previous step, this is fast (although it may be lengthy, depending on the problem).

```
sage: P.print_solution_nonstable()
```

```
****************************************
SOLUTION TO GIT PROBLEM: NONSTABLE LOCI
****************************************
Group: A2
Representation  A2(3,0,0)
Set of maximal non-stable states:
(1) 1-PS = (1, 1, -2) yields a state with 7 characters
Maximal nonstable state={(1, 2, 0), (2, 1, 0), (1, 1, 1), (0, 2, 1),
(0, 3, 0), (2, 0, 1), (3, 0, 0)}
(2) 1-PS = (1, -1/2, -1/2) yields a state with 6 characters
Maximal nonstable state={(1, 2, 0), (1, 0, 2), (2, 1, 0), (1, 1, 1), (2, 0, 1),
(3, 0, 0)}
```

The output above needs to be read as follows. There are 2 families (`Maximal nonstable states`) that parameterise (up to G-conjugacy) all non-stable elements and the weights generating this family are listed. For each family there is a one-parameter subgroup λ such that if $f \in V$ belongs to non-stable family with weights in $\Xi_{V, \lambda \geqslant 0}$, then the Hilbert-Mumford weight $\mu_{(}f, \lambda) \geqslant 0$ and any $\widetilde{f}$ with $\mu_{(}\widetilde{f}, \lambda) \geqslant 0$ must be generated by weights in $\Xi_{V, \lambda \geqslant 0}$. The output assumes that the maximal torus to which all weights belong is in diagonal form in SL_3, so the first one-parameter subgroup must be read as $\lambda_1(t) := \mathrm{Diag}(t^1, t^1, t^{-2}) \in \mathrm{SL}_3(\mathbb{C})$. Since the set of all weights corresponds to monomials of degree 3, a non-stable f in this family can be read as:

$$f = a_0 x_0 x_1^2 + a_1 x_0^2 x_1 + a_2 x_0 x_1 x_2 + a_3 x_1^2 x_2 + a_4 x_1^3 + a_5 x_0^2 x_2 + a_6 x_0^3.$$

Sometimes the one-parameter subgroups appear as fractional vectors, such as in the second family. One can read it like an actual one-parameter subgroup by clearing denominators, i.e. $\lambda_2 := \mathrm{Diag}(t^2, t^{-1}, t^{-1})$, since positive rescaling does not affect the Hilbert-Mumford criterion.

Just as above, we can obtain the non-stable loci and the strictly T-polystable loci where T is a maximal one-dimensional torus (recall that there is precisely one such torus up to conjugacy):

```
sage: P.solve_unstable()
sage: P.print_solution_unstable()



****************************************
SOLUTION TO GIT PROBLEM: UNSTABLE LOCI
****************************************
Group: A2 RepresentationA2(3,0,0)
Set of maximal unstable states:
(1) 1-PS = (1, 1/4, -5/4) yields a state with 5 characters
Maximal unstable state={(1, 2, 0), (2, 1, 0), (0, 3, 0), (2, 0, 1), (3, 0, 0)}

sage: P.solve_strictly_polystable()
sage: P.print_solution_strictly_polystable()



****************************************************************
SOLUTION TO GIT PROBLEM: STRICTLY POLYSTABLE LOCI
****************************************************************
Group: A2 RepresentationA2(3,0,0)
Set of strictly polystable states:
(1) A state with 1 characters
Strictly polystable state={(1, 1, 1)}
(2) A state with 3 characters
Strictly polystable state={(0, 2, 1), (2, 0, 1), (1, 1, 1)}
```

4.2 Interpretation of the outputs

The code above exemplifies how `CompGIT` can provide, in terms of representation theory, a full description of the non-stable, unstable and strictly T-polystable families in a GIT problem. However, in applications we usually require an answer in terms that are more meaningful to the problem at hand. In the case above, for instance, we would like a classification in terms of singularities of the curves involved.

Let $p = (0 : 0 : 1) \in \mathbb{P}^2$. From the output in the previous section and by grouping elements, we see that nonstable elements destabilised by λ_1 can be written as

$$f = x_2 \left(f_2(x_0, x_1) + f_3(x_0, x_1) \right),$$

where f_2 is a quadric form and f_3 is a cubic form linear form and any element written in that form is destabilised by λ_1. By taking partial derivatives, it follows that f is of this form, if and only if its vanishing locus $V(f)$ is singular at p.

Similarly, nonstable elements characterised by λ_2 are precisely those that can be written as

$$f = x_0 \left(f_2(x_0, x_1) + x_2 l(x_0, x_1, x_2) \right),$$

where f_2 is a quadric form and l is a linear form and any element written in that form is destabilised by λ_2. One readily checks that f is of this form, if and only if its vanishing locus $V(f)$ is reducible and two of the components contain p.

It follows from [Gal+23] that our code precisely characterises all the nonstable elements up to G-action. Thus we have shown that a cubic form is nonstable if and only if it is singular (as reducible curves in $\mathbb{P}^2$ are always singular). By taking complements, we conclude:

Lemma 4.1. *A cubic form f in 3 variables is GIT-stable if and only if $V(f)$ is smooth.*

In the case of strictly polystable loci, we have that f is polystable if and only if, up to conjugation by G,

$$f = x_0 x_1 x_2.$$

Thus:

Lemma 4.2. *A cubic form f in 3 variables is strictly GIT-polystable if and only if $V(f)$ is the union of 3 lines which do not intersect at a point.*

Proof. Note that our output only gives us that f is strictly GIT-T-polystable where T is a maximal torus. However any strictly GIT-polystable element must be GIT-T-polystable and since we only have one such element and GIT-stable elements form a non-compact subset while the GIT quotient is compact, we must have at least one strictly GIT-polystable element and thus $f = x_0 x_1 x_2$ must be GIT-polystable. □

Using the programme's output, one can further characterise GIT unstable elements as those which have a singularity that is worse than an ordinary double point (i.e. an A_2-singularity or worse, in Arnold's language). By taking complements, one deduces that semistable elements are those which are smooth or have normal crossings. Since the only polystable element is the union of 3 lines not intersecting at a point, any strictly semistable curve is either a nodal cubic or the union of a conic and a line intersecting normally. To fill in the details, one must make use of the theory of A_n and D_n singularities and their degenerations (cf. [GM19, Table 3 and Fig. 2]), as well as the fact that local degenerations of singularities of cubic curves are unobstructed [ST99, Theorem 1].

4.3 Applications

In [Gal+23, Table 1], a number of applications to modelling moduli spaces are discussed, with references to the literature. We summarise and expand them here. We do not see this section as a description of full-worked out applications, but as a list of problems one can solve using the output of `CompGIT`.

4.3.1 Type A_n

For G of type A_n (i.e. if $G = \mathrm{SL}_{n+1}$), the moduli of hypersurfaces in $\mathbb{P}^n$ can be modeled as $\mathbb{P}(V) /\!/ \mathrm{SL}_{n+1}$ where $V = H^0(\mathbb{P}^n, \mathcal{O}_{\mathbb{P}^n}(d))$. The highest weight for this representation is $d\omega_1$, where ω_i denote the fundamental weights for the root system of G. For plane curves of degree 7 and higher, the output has never been studied. For quintic surfaces and threefolds there are only partial results [Gal19; Gal14; Lak10] with no results for degree 6 or higher. Even the GIT quotient of quartic threefolds is largely unexplored and in higher dimensions we only have results for cubics.

One may consider linear systems $\mathcal{L}_d^k$ generated by k hypersurfaces of degree d in $\mathbb{P}^n$. Extending the action to V as above , we get an action on $\Lambda_{i=1}^k V$ with highest weight $d\omega_1 \wedge \cdots \wedge d\omega_1$. When $n = 3, d = 2, k = 2$, an example is worked out in [Gal+23], cf. [Pap23; Pap24].

4.3.2 Type B_n

Let G be of type B_n. Then $G = \mathrm{SO}_{2n+1}$. Consider a smooth quadric $Q \subset \mathbb{P}^{2n}$. Since Q is a Fano hypersurface, any automorphism of Q must fix $-K_Q = \mathcal{O}_{\mathbb{P}^{2n}}(2n + 1 - 2)|_Q \cong \mathcal{O}_Q(2n - 1) = (2n - 1)\mathcal{O}_Q(1)$, i.e. any automorphism of Q is induced by an automorphism of $\mathbb{P}^{2n}$ that fixes Q. By diagonalising Q, we have that the automorphism group $\mathrm{Aut}(Q) \cong \mathrm{PO}_{2n+1}(Q)$, i.e. the projective orthogonal group fixing Q. But since we diagonalised Q,

$$\mathrm{PO}_{2n+1}(Q) = \mathrm{PO}_{2n+1} \cong \mathrm{SO}_{2n+1} = G.$$

Now let $d \geqslant 2$ consider the moduli of hypersurfaces $X \in |\mathcal{O}_Q(d)| = \mathbb{P}(V)$, where V is defined by the exact sequence

$$0 \longrightarrow H^0(\mathbb{P}^{2n}, \mathcal{O}_{\mathbb{P}^{2n}}(d - 2)) \longrightarrow H^0(\mathbb{P}^{2n}, \mathcal{O}_{\mathbb{P}^{2n}}(d)) \longrightarrow V \longrightarrow 0.$$

Then the GIT quotient $\mathbb{P}(V) /\!/ G$ is a model of the moduli of complete intersections of type $(2, d)$. In the case when $n = 2$ and $d \geqslant 3$ (i.e. when $X \subset Q$ are surfaces of general type) the GIT stack compactifies such surfaces

with at worst semi-log canonical singularities (thus providing a model for the KSBA compactification) and for $d \geqslant 4$, these surfaces are actually all in the GIT stable locus, as proven by Byun-Lee [BL15]. Thus, the GIT quotient provides a model for the KSBA compactification. Interestingly, the description of the GIT quotient in terms of the singularities of the surfaces is not provided even for $d = 3$ and our program could provide the initial step to describe such quotient (i.e. not just 'semi-log canonical' but the explicit singularity types that can be realised on this quotient), addressing a concern of [BL15]: "If S is a complete intersection defined by hypersurfaces with arbitrary degree in P4, then GIT stability analysis is hard to describe". As it can be seen from Table 4.3.2, the number of families to consider is relatively small for the first few degrees, so this is a doable problem. The notation in Table 4.3.2 (and subsequent Tables 3, 4, 5) follows that in [Gal+23]; namely the 'Rep.' column denotes the irreducible representation $V = \Gamma_w$ determined by its highest weight w and the following elements denote the size of certain sets of interest. $|\Xi_V|$ is the number of weights of the representation V and serves as a measure of the complexity of the input. The numbers $|P_s^F|$, $|P_{ss}^F|$, $|P_{ps}^F|$ is the number of non-stable, unstable and strictly polystable families to analyse. In fact, note that the GIT

	Type	Rep.	$|\Xi_V|$	$|P_s^F|$	$|P_{ss}^F|$	$|P_{ps}^F|$
$d = 3$	B2	$\Gamma_{3\omega_1}$	25	3	2	4
$d = 4$	B2	$\Gamma_{4\omega_1}$	41	4	3	5
$d = 5$	B2	$\Gamma_{5\omega_1}$	61	6	5	7
$d = 6$	B2	$\Gamma_{6\omega_1}$	85	7	6	8
$d = 7$	B2	$\Gamma_{7\omega_1}$	113	10	9	11
$d = 8$	B2	$\Gamma_{8\omega_1}$	145	12	11	13

Table 2: Complexity of solutions for problem of type B_2.

quotient with respect to SL_5-action is well-understood [All03], so comparing it to the SO_5 quotient may prove both useful and interesting.

If we consider $n = 3$ (representing fourfolds obtained as a complete intersection of a smooth quadric and a hypersurface of degree d in $\mathbb{P}^6$), the complexity grows significantly (see Table 3) and it may not be possible to say much about it except for $d = 3$. In particular, the programme is not able to obtain the strictly polystable data in a reasonable time.

	Type	Rep.	$|\Xi_V|$	$|P_s^F|$	$|P_{ss}^F|$
$d = 3$	B3	$\Gamma_{3\omega_1}$	14	8	7
$d = 4$	B3	$\Gamma_{4\omega_1}$	26	20	26
$d = 5$	B3	$\Gamma_{5\omega_1}$	48	68	102
$d = 6$	B3	$\Gamma_{6\omega_1}$	70	141	227

Table 3: Complexity of solutions for problem of type B_3.

4.3.3 Type C_n

Consider general complex curve C of genus 9 such that C has no g_5^1. By [Muk10, Theorem B], we can embed C into the symplectic Grassmannian $X = \mathrm{SpG}(3,6) \subset \mathbb{P}^{13}$. We have that C is a transversal intersection of X with a linear $P \cong \mathbb{P}^8 \subset \mathbb{P}^{13}$ and any two such C and P are unique up to the action of $G = \mathrm{Sp}(6)$, the subgroup of PGL_6 fixing the one-dimensional space generated by the symplectic form. Here G is a group of type C_3. Thus, we can see any such curve C as a Zariski-dense open set of $\overline{M} := \mathrm{Gr}(9, V) /\!\!/ \mathrm{Sp}_6$ where $V \cong \mathbb{C}^{14}$ is the irreducible representation of Sp_6 with highest weight ω_3. In particular $\overline{M}_9$ is birationally equivalent to $\overline{M}$.

While the dimension of the GIT quotient is not high, the boundary is expected to be surprisingly complicated. Indeed, in [Gal+23], we showed that the unstable locus and the non-stable locus where expected to have 186 and 142 components. It is possible many of these are isomorphic to each other, but in any case it is a considerable enterprise to describe them explicitly. While our software is primarily created to help describe

GIT quotients, this example demonstrates it is also helpful to determine a priory the viability of providing an explicit description of a moduli problem via GIT.

4.3.4 Type D_n

We recall two applications for D_n. For the first one, let $\mathrm{OG}(5,10) \subset \mathbb{P}^{15}$ be the naturally-embedded 10-dimensional orthogonal Grassmannian. If C is a smooth genus-7 curve with no g_4^1, it has an ideal generated by 10 quadrics in $\mathbb{P}^6$. Such a curve can be embedded in $\mathrm{OG}(5,10)$ by sending each point $p \in C$ to the value of the Jacobian matrix of the quadrics, as proven by Mukai in [Muk95], where equations for the quadrics are provided, cf. [Swi23] for a more contemporaneous treatment. In fact, Mukai shows that the image of C in $\mathrm{OG}(5,10) \subset \mathbb{P}^{15}$ is the transversal intersection in $\mathbb{P}^1 5$ of $\mathrm{OG}(5,10)$ with a 6-dimensional subspace in $\mathbb{P}^{15}$ and that any such transversal intersection provides a curve of genus 7 with no g_4^1, with that one-to-one correspondence being unique up to action of SO_{10}, which is a group of type D_5.

Since Spin_{10} is the universal cover of SO_{10}, and curves with no g_4^1 are general in the moduli of genus 7 curves, the GIT quotient

$$\mathrm{Gr}(7,16) /\!\!/ \mathrm{Spin}_{10} \subset \mathbb{P}(S^+) /\!\!/ \mathrm{Spin}_{10}$$

is a birational model of $\overline{M}_7$, where S^+ is the 16-dimensional half-spin representation of Spin_{10} with highest weight ω_4. As argued in [Gal+23], modern desktop computers do not seem to be able to run `CompGIT` for this problem, as the output is expected to be of the order of 10^6 families to analyse. We must emphasise that even if a supercomputer could run `CompGIT` for this problem in a reasonable time (if, say, parallelisation and a smart use of memory was carried out), the output would be too complex to analyse by hand. Nonetheless, in [Muk95], Mukai uses this construction to classify tetragonal curves of genus 7 and in [Swi23], uses the above analysis as a first step to further use invariant theory to establish the GIT semistability of 7-cuspidal curves, the balanced ribbon and a family of highly reducible nodal curves.

The second application is the odd-dimensional version of what we covered in §4.3.2. Let $n \geqslant 2$ and $G = \mathrm{SO}_{2n-1}$. Then G is of type D_n and, by the same reasoning, we have a natural action of G on $\mathbb{P}(H^0(\mathbb{P}^{2n-1}, \mathcal{O}_{\mathbb{P}^{2n-1}}(d)))$, where the latter parametrises complete intersections (of dimension $2n-3$) of a smooth quadric and a hypersurface of degree d in $\mathbb{P}^{2n-1}$ with $\mathbb{P}(V) /\!\!/ G$ giving a model of their moduli space. In Table 4 we give the complexity for curves. Since $-K_Q = \mathcal{O}_Q(2)$, for each d, this natural moduli problem parametrises the moduli of curves in $Q \cong \mathbb{P}^1 \times \mathbb{P}^1$ of bidegree (d,d).

	Type	Rep.	$|\Xi_V|$	$|P_s^F|$	$|P_{ss}^F|$	$|P_{ps}^F|$
$d = 2$	D2	$\Gamma_{2\omega_1}$	3	3	2	4
$d = 3$	D2	$\Gamma_{3\omega_1}$	3	3	4	3
$d = 4$	D2	$\Gamma_{4\omega_1}$	5	5	4	6
$d = 5$	D2	$\Gamma_{5\omega_1}$	7	7	8	7
$d = 6$	D2	$\Gamma_{6\omega_1}$	9	9	8	10

Table 4: Complexity of solutions for problem of type D_2.

The next dimension still has some interest while being accessible to classification. The GIT quotient $\mathbb{P}(V) /\!\!/ G$ parametrises threefolds obtained as complete intersections of a smooth quadric Q and a hypersurface of degree d in $\mathbb{P}^5$.

	Type	Rep.	$|\Xi_V|$	$|P_s^F|$	$|P_{ss}^F|$	$|P_{ps}^F|$
$d = 2$	D3	$\Gamma_{2\omega_1}$	7	4	2	6
$d = 3$	D3	$\Gamma_{3\omega_1}$	11	11	14	10
$d = 4$	D3	$\Gamma_{4\omega_1}$	22	18	24	16

Table 5: Complexity of solutions for problem of type D_3.

4.3.5 Exceptional types

One of the most important additions to `CompGIT` since the prototype code of [Gal+23] is the inclusion of exceptional Lie groups, i.e those of type G_2, F_4, E_6, E_7 and E_8. Explicit examples of their GIT problems are harder to find, beyond canonical examples such as their fundamental representations. Related examples come from gauge theory, in which one can take exceptional structure group G and study GIT problems modulo the corresponding gauge group [Ram96a; Ram96b].

5 Possible future developments

In [Gal+23, §7], we detailed a number of potential improvements to the algorithms and the code introduced there, which we have completed here. It is difficult to elaborate on those potential improvements without going into the details of the implementation. Moreover, there is little we can add to what was written there and we refer the reader to [Gal+23] for some ideas on generalisations to variations of GIT quotients (cf. [GM18; GM17] for previous work by the second author). Nonetheless, we can offer some additional changes not considered in [Gal+23] that could be considered and that were not considered in [Gal+23], as well as elaborating slightly on [Gal+23, §7.3 Parallel computing].

5.1 Parallel computing

The main computational hurdle of our algorithms is finding solutions to one-parameter subgroups. Solutions do not depend on previous computations and this is an aspect that could be easily parallelised using the `sage.parallel` library. The reason we have not carried out this implementation is because for current applications, the time saved is neglectable when compared to the time to process the output in geometric terms. There is one exception where it may be worth considering parallelisation. The computation of the strictly polystable loci requires determining if a given point is in the interior or the boundary of a convex hull of points. The current implementation uses the class `Polyhedron` — `sage.geometry.polyhedron.constructor` — as a black box to obtain this information. When the number of points to consider is large, the current implementation takes too long, even for a relatively small number of families (e.g. for only 8 families in the case of B_4 acting on Γ_{ω_3}, the computation was stopped after 48 hours on a desktop computer, see [Gal+23, Table 1]).

5.2 Generalisation to semisimple groups and other groups

Our implementation works for irreducible G-representations where G is a simple group, while our algorithms (once some optimisations detailed in [Gal+23] are removed) works for any reductive algebraic group. The reason for this is that the implementation relies on two pre-existing classes in `Sagemath`, namely `WeylCharacterRing` and `WeylGroup`.

`WeylCharacterRing` should work for general semisimple groups without changing the code. This class allows us to take representations as a black box, but the only bit of this class that we use is the method `weight_multiplicities` in `Git.py` within the constructor of the class `GITProblem` to obtain a list of non-zero weights of the representation. Our implementation could easily be modified to request from the user a list of the weights of a representation, thus allowing us to work with reducible representations and more general reductive groups.

`WeylGroup` is used within `SimpleGroup.py` to obtain a list of all the elements of the Weyl group of the simple group G to potentially simplify the output and reduce the number of computations (this is what we call *Weyl optimisation* in [Gal+23]). To generalise to more general reductive groups, some additional work must be carried out here. There are two options. The first option is to describe the Weyl group for more general G. We do not recommend this, as it will be highly dependent on the choice of group, making it difficult to provide a uniform approach. A simpler option is to stop carrying out any Weyl optimisation at all. The solution methods (`solve_non_stable`, `solve_unstable`) already include a parameter (`Weyl_optimisation`), set by default to `False`, not to carry out this optimisation. However, the constructors assume that the group G is simple and the Weyl group is computed nonetheless.

It is worth noting that it is actually believed that the output is not simplified at all by the use of the Weyl group (see [Gal+23, Conjecture 7.4]), although the programme may be faster when Weyl optimisation is used.

5.3 Non-reductive groups

As exemplified by Nagata [Nag60], the GIT of Mumford (and more generally invariant theory) does not work well for non-reductive groups. The more recent theory of *non-reductive GIT* due to Kirwan and her collaborators [BDK18] does allow for GIT quotients by non-reductive groups under an additional choice of *graded linearisation*. To the best of our knowledge no systematic algorithmical study has been carried out, although some seminal work for hypersurfaces has been considered [Bun24]. A non-reductive version of our setting (a non-reductive group G acting on a vector space V) is a natural starting point.

5.4 Generalisations to other schemes

As discussed in §1.1 a G-equivariant embedding of a G-scheme X into $\mathbb{P}^N$ is necessary to apply the outputs of our code. Implementing solutions to this embedding problem into `CompGIT` appears to be out of reach, as for general X, one does not expect a uniform computational approach to finding such an embedding. Nonetheless, it may be possible to give an ad-hoc treatment for some schemes of geometric interest. For instance, one can consider Grassmannians, or hypersurfaces, or products and complete intersections of either of the former (note that $\mathbb{P}^N$ is a special case of Grassmannian). Indeed, some seminal work for $G = \mathrm{SL}_{n+1}$ acting on $V = H^0(\mathbb{P}^n, \mathcal{O}(d)) \times \mathbb{P}^n$ naturally was carried out in [GM18] and quotients for SL_n actions on a Grassmannian and on a product of a Grassmannian and projective space were considered in [Pap23; Pap24].

References

[Abb+25]   H. Abban, I. Cheltsov, E. Denisova, E. Etxabarri-Alberdi, A. Kaloghiros, D. Jiao, J. Martinez-Garcia, and T. Papazachariou. "One-dimensional components in the K-moduli of smooth fano 3-folds". *J. Algebraic Geom.* 34.3 (2025), pp. 489–534. ISSN: 1056-3911,1534-7486.

[All03]   D. Allcock. "The moduli space of cubic threefolds". *J. Algebraic Geom.* 12.2 (2003), pp. 201–223. ISSN: 1056-3911,1534-7486. DOI: 10.1090/S1056-3911-02-00313-2. URL: https://doi.org/10.1090/S1056-3911-02-00313-2.

[AHH23]   J. Alper, D. Halpern-Leistner, and J. Heinloth. "Existence of moduli spaces for algebraic stacks". *Inventiones mathematicae* 234.3 (2023), pp. 949–1038.

[BDK18]   G. Bèrczi, B. Doran, and F. Kirwan. "Graded linearisations". *Modern geometry: a celebration of the work of Simon Donaldson.* Vol. 99. Proc. Sympos. Pure Math. Amer. Math. Soc., Providence, RI, 2018, pp. 1–22. ISBN: 978-1-4704-4094-7. DOI: 10.1090/pspum/099/01. URL: https://doi.org/10.1090/pspum/099/01.

[Bou02]   N. Bourbaki. "Lie groups and Lie algebras. Chapters 4-6. Translated from the 1968 French original by A. Pressley". *Graduate Texts in Mathematics, Springer-Verlag* (2002).

[Bun24]   D. Bunnett. "On the moduli of hypersurfaces in toric orbifolds". *Proc. Edinb. Math. Soc. (2)* 67.2 (2024), pp. 577–616. ISSN: 0013-0915,1464-3839. DOI: 10.1017/S0013091524000166. URL: https://doi.org/10.1017/S0013091524000166.

[BL15]   S. Byun and Y. Lee. "Stability of hypersurface sections of quadric threefolds". *Sci. China Math.* 58.3 (2015), pp. 479–486. ISSN: 1674-7283,1869-1862. DOI: 10.1007/s11425-014-4918-8. URL: https://doi.org/10.1007/s11425-014-4918-8.

[Car29]   E. Cartan. "Groupes simples clos et ouverts et géométrie riemannienne". *Journal de mathématiques pures et appliquées* 8 (1929), pp. 1–33.

[Che48]   C. Chevalley. "Sur la classification des algèbres de Lie simples et de leurs représentations". *CR Acad. Sci. Paris* 227 (1948), pp. 1136–1138.

[CM93]   D. H. Collingwood and W. M. McGovern. *Nilpotent orbits in semisimple Lie algebras.* Van Nostrand Reinhold Mathematics Series. Van Nostrand Reinhold Co., New York, 1993, pp. xiv+186. ISBN: 0-534-18834-6.

[DM69]   P. Deligne and D. Mumford. "The irreducibility of the space of curves of given genus". *Inst. Hautes Études Sci. Publ. Math.* 36 (1969), pp. 75–109. ISSN: 0073-8301,1618-1913. URL: `http://www.numdam.org/item?id=PMIHES_1969__36__75_0`.

[DK15]   H. Derksen and G. Kemper. *Computational invariant theory.* enlarged. Vol. 130. Encyclopaedia of Mathematical Sciences. With two appendices by Vladimir L. Popov, and an addendum by Norbert A'Campo and Popov, Invariant Theory and Algebraic Transformation Groups, VIII. Springer, Heidelberg, 2015, pp. xxii+366. ISBN: 978-3-662-48420-3. DOI: `10.1007/978-3-662-48422-7`. URL: `https://doi.org/10.1007/978-3-662-48422-7`.

[FH13]   W. Fulton and J. Harris. *Representation theory: a first course.* Vol. 129. Springer Science & Business Media, 2013.

[Gal14]   P. Gallardo. *On the moduli space of quintic surfaces.* Thesis (Ph.D.)–State University of New York at Stony Brook. ProQuest LLC, Ann Arbor, MI, 2014, p. 99. ISBN: 978-1-321-11737-0. URL: `http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqm&rft_dat=xri:pqdiss:3632411`.

[Gal19]   P. Gallardo. "On the GIT quotient space of quintic surfaces". *Trans. Amer. Math. Soc.* 371.6 (2019), pp. 4251–4276. ISSN: 0002-9947,1088-6850. DOI: `10.1090/tran/7493`. URL: `https://doi.org/10.1090/tran/7493`.

[GM17]   P. Gallardo and J. Martinez-Garcia. *Variations of GIT quotients package v0.6.13.* `https://doi.org/10.15125/BATH-00458`. 2017.

[GM18]   P. Gallardo and J. Martinez-Garcia. "Variations of geometric invariant quotients for pairs, a computational approach". *Proc. Amer. Math. Soc.* 146.6 (2018), pp. 2395–2408. ISSN: 0002-9939,1088-6826. DOI: `10.1090/proc/13950`. URL: `https://doi.org/10.1090/proc/13950`.

[GM19]   P. Gallardo and J. Martinez-Garcia. "Moduli of cubic surfaces and their anticanonical divisors". *Rev. Mat. Complut.* 32.3 (2019), pp. 853–873. ISSN: 1139-1138,1988-2807. DOI: `10.1007/s13163-019-00298-y`. URL: `https://doi.org/10.1007/s13163-019-00298-y`.

[Gal+23] P. Gallardo, J. Martinez-Garcia, H-B Moon, and D. Swinarski. "Computation of GIT quotients of semisimple groups". *arXiv:2308.08049* (2023).

[GMS21]   P. Gallardo, J. Martinez-Garcia, and C. Spotti. "Applications of the moduli continuity method to log K-stable pairs". *J. Lond. Math. Soc. (2)* 103.2 (2021), pp. 729–759. ISSN: 0024-6107,1469-7750. DOI: `10.1112/jlms.12390`. URL: `https://doi.org/10.1112/jlms.12390`.

[GMZ18]   P. Gallardo, J. Martinez-Garcia, and Z. Zhang. "Compactifications of the moduli space of plane quartics and two lines". *Eur. J. Math.* 4.3 (2018), pp. 1000–1034. ISSN: 2199-675X,2199-6768. DOI: `10.1007/s40879-018-0248-7`. URL: `https://doi.org/10.1007/s40879-018-0248-7`.

[HM]   R. Hanson and J. Martinez-Garcia. *The CompGIT GitHub homepage.* `https://github.com/Robbie-H/CompGIT`.

[Hil02]   D. Hilbert. "Mathematical problems". *Bull. Amer. Math. Soc.* 8.10 (1902), pp. 437–479. ISSN: 0002-9904. DOI: `10.1090/S0002-9904-1902-00923-3`. URL: `https://doi.org/10.1090/S0002-9904-1902-00923-3`.

[Kil89]   W. Killing. "Die Zusammensetzung der stetigen endlichen Transformations-gruppen". *Mathematische Annalen* 34.1 (1889), pp. 57–122.

[Kir82]   F. Kirwan. "Sur la cohomologie des espaces quotients". *C. R. Acad. Sci. Paris Sér. I Math.* 295.3 (1982), pp. 261–264. ISSN: 0249-6291.

[Lak10]   C. Lakhani. *The GIT Compactification of Quintic Threefolds.* 2010. arXiv: `1010.3803 [math.AG]`. URL: `https://arxiv.org/abs/1010.3803`.

[Laz09]   R. Laza. "Deformations of singularities and variation of GIT quotients". *Trans. Amer. Math. Soc.* 361.4 (2009), pp. 2109–2161. ISSN: 0002-9947,1088-6850. DOI: `10.1090/S0002-9947-08-04660-6`. URL: `https://doi.org/10.1090/S0002-9947-08-04660-6`.

[LX19]   Y. Liu and C. Xu. "K-stability of cubic threefolds". *Duke Math. J.* 168.11 (2019), pp. 2029–2073. ISSN: 0012-7094,1547-7398. DOI: `10.1215/00127094-2019-0006`. URL: `https://doi.org/10.1215/00127094-2019-0006`.

[MPZ24]   J. Martinez-Garcia, T. S. Papazachariou, and J. Zhao. *K-moduli of log del Pezzo pairs and variations of GIT.* 2024. arXiv: `2406.20008 [math.AG]`. URL: `https://arxiv.org/abs/2406.20008`.

[Muk95]   S. Mukai. "Curves and symmetric spaces. I". *Amer. J. Math.* 117.6 (1995), pp. 1627–1644. ISSN: 0002-9327,1080-6377. DOI: `10.2307/2375032`. URL: `https://doi.org/10.2307/2375032`.

[Muk10]   S. Mukai. "Curves and symmetric spaces, II". *Ann. of Math. (2)* 172.3 (2010), pp. 1539–1558. ISSN: 0003-486X,1939-8980. DOI: `10.4007/annals.2010.172.1539`. URL: `https://doi.org/10.4007/annals.2010.172.1539`.

[MFK94]   D. Mumford, J. Fogarty, and F. Kirwan. *Geometric invariant theory*. Third. Vol. 34. Ergebnisse der Mathematik und ihrer Grenzgebiete (2) [Results in Mathematics and Related Areas (2)]. Springer-Verlag, Berlin, 1994, pp. xiv+292. ISBN: 3-540-56963-4.

[Nag60]   M. Nagata. "On the fourteenth problem of Hilbert". *Proc. Internat. Congress Math. 1958*. Cambridge Univ. Press, New York, 1960, pp. 459–462.

[Pap23]   T. S. Papazachariou. *K-moduli of log Fano_complete_intersections*. Thesis (Ph.D.)–The University of Essex (United Kingdom). 2023. URL: `https://repository.essex.ac.uk/35380/`.

[Pap24]   T. S. Papazachariou. *K-moduli of log Fano complete intersections*. 2024. arXiv: `2212.09332 [math.AG]`. URL: `https://arxiv.org/abs/2212.09332`.

[Ram96a] A. Ramanathan. "Moduli for principal bundles over algebraic curves: I". *Proc. Indian Acad. Sci.* Vol. 106. Springer. 1996, pp. 301–328.

[Ram96b] A. Ramanathan. "Moduli for principal bundles over algebraic curves: II". *Proc. Indian Acad. Sci.* Vol. 106. Springer. 1996, pp. 421–449.

[Sag]   Sage. *The SageMath installation guide*. `https://doc.sagemath.org/html/en/installation/index.html`.

[The25]   The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.6)*. `https://www.sagemath.org`. 2025.

[Sha80]   J. Shah. "A complete moduli space for $K3$ surfaces of degree 2". *Ann. of Math. (2)* 112.3 (1980), pp. 485–510. ISSN: 0003-486X. DOI: `10.2307/1971089`. URL: `https://doi.org/10.2307/1971089`.

[ST99]   E. Shustin and I. Tyomkin. "Versal deformation of algebraic hypersurfaces with isolated singularities". *Math. Ann.* 313.2 (1999), pp. 297–314. ISSN: 0025-5831,1432-1807. DOI: `10.1007/s002080050262`. URL: `https://doi.org/10.1007/s002080050262`.

[Swi23]   D. Swinarski. *Some singular curves in Mukai's model of $\overline{M}_7$*. 2023. arXiv: `2304.12936 [math.AG]`. URL: `https://arxiv.org/abs/2304.12936`.

[Wey46]   H. Weyl. *The classical groups: their invariants and representations*. Vol. 1. Princeton university press, 1946.