Computing Conjugacy Classes of Elements in Matrix Groups

Alexander Hulpke

Department of Mathematics, Colorado State University, 1874 Campus Delivery, Fort Collins, CO, 80523-1874, USA

Abstract

This article describes a setup that – given a composition tree – provides functionality for calculation in finite matrix groups using the Trivial-Fitting approach that has been used successfully for permutation groups. It treats the composition tree as a black-box object. It thus is applicable to other classes of groups for which a composition tree can be obtained. As an example, we consider an effective algorithm for determining conjugacy class representatives.

Keywords: Matrix Group, Calculation, Solvable Radical, Conjugacy Classes

1. Introduction

The algorithms developed in the context of the matrix group recognition project [O'B11] produce as their principal output a composition tree, a data structure that represents a composition series of the group. Such a tree also provides homomorphisms from the subnormal subgroups in this series onto the simple composition factors. Implementations of algorithms determining a composition tree have recently reached a point at which it becomes feasible to use this tree for calculations beyond only the group order or composition structure, see [BHLGO].

One approach for such higher level calculation is the so-called Trivial-Fitting method (also called the "Solvable Radical" method) that has been used successfully for permutation groups and is the basis of many algorithms provided by the computer algebra systems GAP [GAP13] and MAGMA [BCP97].

This method sits at the heart of the black-box group approach of [BB99]. For permutation groups it has been used to design effective algorithms for determining conjugacy classes of elements [CS97, Hul00, CH06], conjugacy classes of subgroups [CCH01], normal subgroups [Hul98], maximal subgroups [CH04, EH01], and automorphism groups [CH03].

Email address: `hulpke@math.colostate.edu` (Alexander Hulpke)
URL: `http://www.math.colostate.edu/~hulpke` (Alexander Hulpke)

The Trivial-Fitting method requires determining the *solvable radical* $\mathrm{Rad}(G)$ (the largest solvable normal subgroup of G), a representation of its factor group $G/\mathrm{Rad}(G)$ and an effective natural homomorphism $\rho\colon G \to G/\mathrm{Rad}(G)$. (We shall call a homomorphism *effective* if we can determine the images of elements of the domain as well as preimages of elements of the image in time and storage requirement sub-linear in $|G|$).

The Trivial-Fitting method also requires a means for performing linear algebra in the elementary abelian layers of $\mathrm{Rad}(G)$. Following [HS08] it is possible to modify an existing composition tree to expose $\mathrm{Rad}(G)$ as a subgroup in the composition series, though this does not necessarily expose ρ as the top-most homomorphism of the tree. It therefore provides no immediate way to represent the radical factor.

The main result of this paper is an alternative approach that uses a composition tree to construct a new group $H \cong G/\mathrm{Rad}(G)$, together with an effective epimorphism $\rho\colon G \to H$ (Section 3). This is done by considering the tree as a black-box object without having to refer to the geometric structures that gave rise to it. The natural input size measurement for working with such a tree is the number of composition factors, the natural cost is operations on the composition tree (Definition 6). We shall show in Theorem 9 that with these metrics the construction of H and ρ can be performed in polynomial time; evaluating ρ on an element of G can be performed in linear time.

We obtain $\mathrm{Rad}(G)$ as kernel of ρ.

We then show how to adapt methods for solvable permutation groups [Sim90] to obtain a polycyclic generating set for $\mathrm{Rad}(G)$ (which implicitly allows the use of linear algebra for the elementary abelian layers), Section 4. We finally describe how these data structures allow us to handle arbitrary subgroups of G without the need to perform a new recognition procedure, Section 5. As an illustration of this approach, we describe a practical implementation of an algorithm that computes representatives of conjugacy classes of elements, as well as their centralizers, Section 6.

The composition tree approach we shall use follows [NS06], as implemented in the recog package for GAP, though other composition tree methods would work as long as they provide support for a set of basic tree operations that we will list in Section 2. In particular, if a composition tree (providing homomorphisms as well as constructive recognition of the leaves) is given for a group in a different representation, e.g. an automorphism group, then the approach described here would apply immediately.

2. Composition Trees

We assume that we have already computed a composition tree for the group G and have established simplicity of the leaves. (If G is a matrix group, such a tree is exactly the result of matrix group recognition [O'B11].)

Concretely, this means that if G is simple we have proven so and have an effective isomorphism to a "natural" representation for this simple group.

Remark 1. *By applying constructive recognition [O'B11, Section 8] to the simple group we may assume that for groups of Lie Type this natural representation is an absolutely irrreducible, projective representation of small degree in defining characteristic. In particular, if the simple group is $PSX(d,q)$, we assume that the representation is by $d \times d$ matrices over the field with q elements.*

Otherwise we have constructed an effective homomorphism φ on G, together with generators for $\ker\varphi$ and for G^{φ}. Recursively, $\ker\varphi$ and G^{φ} are treated in the same way as G was, i.e. we either have established simplicity or a new decomposing homomorphism.

This produces recursively a tree structure for G (which is called a *composition tree*) in which each node corresponds to a subgroup, or factor group of a subgroup, of G. The root of the tree is G. The leaves of the tree correspond to the simple composition factors of G.

Such a composition tree implies a composition series $\langle 1\rangle = C_l \lhd \cdots \lhd C_1 \lhd C_0 = G$ for G with C_i/C_{i+1} simple and nontrivial. (The subgroups in this series are the kernels of the homomorphisms for the tree, respectively preimages of these kernels under homomorphisms from nodes higher in the tree.)

We note that the homomorphisms of the composition tree yield effective homomorphisms $\gamma_i \colon C_i \to C_i/C_{i+1}$, identifying this factor group with the corresponding leaf of the composition tree.

To enable a reduction to calculation in composition factors, we shall use the following definition:

Definition 2. *A group G, in a particular representation on the computer, is called* benign*, if the following tasks can be performed:*

- I *One can determine the order of a subgroup of G, given by a generating set.*
- II *Given a generating set for G, it is possible to write an arbitrary element of G as a product of these generators.*
- III *For given $g_1, \ldots, g_k, h_1, \ldots, h_k \in G$, one can effectively find $x \in G$ such that $g_i^x = h_i$, or show that no such element exist.*

Remark 3. *Task II does not make assumptions about particular generating sets or length of word expressions. In concrete applications, or for a complexity analysis, short words (or short straight line programs [Ser03, p.10]) will often be desirable and might necessitate choice of particular generating sets.*

Remark 4. *If G is represented as a permutation group or as a matrix group of moderate degree, then stabilizer chains and backtrack search [HEO05, chapter 4] provide effective solutions to these tasks: Tasks I, II immediately follow from the existence of stabilizer chains. Task III can be solved using multiple backtrack searches for conjugating elements. (Find $x_1 \in G$ that maps g_1 to h_1. Then find $x_2 \in C_G(h_1)$ that will map $g_2^{x_1}$ to h_2, and so on. The desired element is $\prod_i x_i$.)*

We finally shall assume that for each non-abelian leaf T of the composition tree we have an effective homomorphism $T \to \mathrm{Aut}(T)$, with $\mathrm{Aut}(T)$ benign.

Remark 5. *Representing T as a subgroup of $\operatorname{Aut}(T)$ rarely changes the complexity of the problem: From [CCN+85, Table 1, p. viii and Table 5, p. xvi] we deduce that for every finite simple group T there are normal subgroups $D, F \lhd \operatorname{Aut}(T)$, $T \leq D \leq F \leq \operatorname{Aut}(T)$. Here D is generated by T and diagonal automorphisms (such as automorphisms induced by GL on SL) and F is in addition generated by field automorphisms. If T is represented in its "natural" representation (see Remark 1) in defining characteristic, this representation extends to D. Field automorphisms can be treated by forming a semidirect product with a Galois group acting in a natural way. Finally, $\operatorname{Aut}(T)/F$ is generated by graph automorphisms and automorphisms of alternating and sporadic groups. Thus its order is universally bounded by $m = 6$. We can therefore embed $\operatorname{Aut}(T)$ in a wreath product $F \wr S_m$.*

In the current implementation the benignness of $\operatorname{Aut}(T)$ comes from representing it as a permutation group. On a theoretical level of course the expectation is that constructive recognition and the theory of simple groups will provide a benign representation with efficient algorithms for the tasks in Definition 2.

Definition 6. *Motivated by [BB99], our approach towards the radical factor now considers the composition tree as a black-box structure. For working with such a tree we will use the following basic operations:*

Depth *For $x \in G$, determine the maximum d such that $x \in C_d$. We call this value d the* depth *of x.*

Leaf Image *For $x \in C_i$, with $T = C_i/C_{i+1}$ and $\gamma_i \colon C_i \to T$, determine the image x_i^γ as an element of $\operatorname{Aut}(T)$.*

Leaf Operations *We shall assume for all simple leaves T of the tree that $\operatorname{Aut}(T)$ is benign; thus the tasks I,II,III in $\operatorname{Aut}(T)$ each are considered a basic composition tree operation.*

Remark 7. *Determining the depth of an element can be done by determining images $x^{\gamma_0}, x^{\gamma_1}, \ldots$ until an image is nontrivial. We shall denote a large numerical value of the depth d as "deep" and a small numerical value as "shallow". That is, a deep element lies in a small subgroup in the composition series, while shallow elements only occur in the largest subgroups of the series.*

Remark 8. *One could argue about the appropriateness of counting a* Leaf Image *as of comparable cost to a conjugacy tests of element sequences in a leaf. The data structure of a composition tree however means that the leaf image might require a sequence of homomorphisms, together with constructive recognition, but a conjugacy test in a classical group might only involve normal forms. We thus use the coarse granularity of composition tree operations to hide any particularities of the implementation of the tree and its leaves.*

3. Determining the radical factor

Recent algorithms for finite groups have emphasized the following structure [BB99]. A finite group G possesses characteristic subgroups $R \leq S^* \leq Pker \leq G$ where $R = \mathrm{Rad}(G) = O_\infty(G)$ is the solvable radical (i.e. the largest solvable normal subgroup of G), $S = S^*/R$ is the socle of G/R and $Pker$ is the kernel of the action π of G permuting the direct factors of S^*/R. (By the proof of Schreier's conjecture [Fei80] $Pker/S^*$ is solvable.) For a permutation group G, the algorithms of [LS97, Hol97] provide a means for constructing a representation of G/R as a permutation group, together with the natural homomorphism $\rho\colon G \to G/R$.

See [BHLGO] for discussion on how to retreive these subgroups from a composition tree.

Our approach differs in that it constructs a representation of G/R first, never building a new composition tree. We shall prove the following.

Theorem 9. *Let G be a group for which a composition tree has been determined such that for all nonabelian leaves T of this tree $\mathrm{Aut}(T)$ is benign. Denote by m the number of nonabelian composition factors of G and by n the maximum of the total number of composition factors of G and the number of generators of G.*

Then there exists a group $H \cong G/\mathrm{Rad}(G)$ and an effective homomorphism $\rho\colon G \to H$ with $\ker \rho = \mathrm{Rad}(G)$, such that for $g \in G$ an image g^ρ can be determined in $2m$ element conjugation operations in G and $4m$ basic composition tree operations.

The initial setup for constructing H and ρ requires at most $4n^3$ composition tree operations.

To construct H, we observe that $S := \mathrm{Soc}(G/R)$ is a direct product of nonabelian simple groups, and $C_{G/R}(S) = \langle 1 \rangle$. The action of G/R on its socle thus is faithful and we can identify G/R with a subgroup of $\mathrm{Aut}(S)$. The action of G/R on S induces an action of G on S (with kernel R) and we will build an effective version of ρ from this action.

By Jordan-Hölder, the direct factors of S are (with multiplicities) a subset of the nonabelian leaves of the composition tree. For ease of description we first consider the case where the radical factor lies in a single wreath product and $G/Pker$ is solvable, and then show how to generalize.

3.1. Case 1: Minimal Normal Socle

We shall assume first that $G/Pker$ is solvable and that S is minimally normal in G/R. This means [BB99] that $S = T_1 \times \cdots \times T_n \cong T^n$ for a simple nonabelian group T. Then $\mathrm{Aut}(S) \cong \mathrm{Aut}(T) \wr S_n$ and the direct factors of S correspond exactly to the nonabelian composition factors of G. If $\mathrm{Aut}(T)$ is given as a permutation group or matrix group of degree k, then $\mathrm{Aut}(T) \wr S_n$ can be represented in degree $k \cdot n$. Assume that $\rho\colon G \to \mathrm{Aut}(T) \wr S_n$ is the (induced) action of G on S. We will describe how to make this action effective:

If we represent $t \in T_i$ by a preimage $x \in S^*$, the depth of x corresponds to the index i of the direct factor.

We first choose the nonabelian leaf in the composition tree that is maximally deep (denote the depth by d). This choice ensures that the image of the d-th subgroup in the composition series under ρ is one of the direct factors of S (and not a diagonal subgroup). We set ${C_d}^{\rho} = T_1$ and therefore represent the simple group T by the image ${C_d}^{\gamma_d}$.

We will now determine which composition factors correspond to the other copies of T in S. As each of these copies T_i corresponds to a composition factor of a different depth, we will record the bijection between depths of nonabelian composition factors and direct factors of S in two lists, v and w. That is, C_{v_i}/C_{v_i+1} corresponds to the i-th copy T_i and $w_{(v_i)} = i$ for all i. The initial setup gives $v_1 = d$ and $w_d = 1$. Other values will be determined as the calculation progresses.

By [AG84] every finite simple group can be generated by two elements. We thus take a generating sequence $\underline{\mathbf{t}} = (t_1, t_2)$ (we shall use underlined bold letters to denote generating sequences) for $T \cong C_d/C_{d+1}$.

Let $\underline{\mathbf{x}}_1 = (x_{1,1}, x_{1,2}) \subset C_d$ be preimages of these generators under γ_d. As we have chosen a generating pair for T, we know that $x_{1,i} \notin C_{d+1}$. Conjugates of $\underline{\mathbf{x}}_1$ under G (with elements $x_{i,j}$) will then be used as generating sets for the other T_i.

To allow the identification of socle factors with composition factors of particular depths, we will need to ensure that no element of G we use for representing factor T_i has *shallower* depth than the corresponding depth v_i. (By definition of depth such an element can never be deeper, as it has a nontrivial image in C_{v_i}/C_{v_i+1}.)

Such a situation could arise for example if

$$G = (A \times B) \wr S_2 = ((A_1 \times B_1) \times (A_2 \times B_2)) \rtimes S_2$$

and if the subgroups in the composition series are (in ascending order) $C_5 = B_1$, $C_4 = B_1 \times A_1$, $C_3 = (B_1 \times A_1) \times A_2$, $C_2 = (B_1 \times A_1) \times (A_2 \times B_2)$ and $C_1 = G$. Assume that we have chosen $x \in C_4$ to represent a nontrivial element of A_1 (i.e. x has depth 4), but that x also projects nontrivially to B_1. If we now conjugate x with $(1,2) \in S_2$, we would want $y = x^{(1,2)}$ to represent the composition factor A_2; however y also has a nontrivial image in B_2 and therefore is of depth 2 and not depth 3 as expected.

If such a situation arises, the algorithm will resolve it by modifying the chosen generating sets $\underline{\mathbf{x}}_i$. We will describe this correction process in Section 3.1.1 after the initial description of the algorithm which we give here:

Denote by $\underline{\mathbf{g}}$ a generating sequence of G. We now perform (by forming conjugation images of $\underline{\mathbf{x}}_1$) an orbit algorithm for the action of G on the indices $\{1, \ldots, n\}$ of the direct factors of S, determining the permutation image $G^{\pi} = G/Pker$.

During this calculation, we will also determine, once we encounter factor i for the first time, the corresponding depth v_i in the composition series (as well as the inverse $w_{v_i} = i$).

As we intend to form a wreath product $\mathrm{Aut}(T) \wr S_n$, we will represent each direct factor T_i by the *same* group T. To play the role of γ_{v_i} we determine for each i a homomorphism $\mu_i \colon C_{v_i} \to T$ and an element sequence $\underline{\mathbf{x}}_i \subset C_{v_i}$ such that $T = \langle \underline{\mathbf{x}}_i^{\mu_i} \rangle$. (Thus the same generating set $\underline{\mathbf{t}}$ has different preimages $\underline{\mathbf{x}}_i$ under the different projections μ_i.)

The following algorithm describes this process. Here v and w are lists that describe the index translation, μ is a list of homomorphisms, *rep* is a list of group elements mapping the first copy of T to the respective other copies (rep_i is required for the evaluation of μ_i).

```
1: $v := []$; $w := []$; $\mu := []$; $rep := []$;
2: $v_1 := d$; $w_d := 1$;$i := 1$; $k := 1$; $rep_1 := 1_G$; $\mu_1 := \gamma_d$.
3: while $i \leq k$ do
4:    for $g \in \underline{\mathbf{g}}$ do
5:       Let $\underline{\mathbf{y}} = \underline{\mathbf{x}}_i^g = (x^g \mid x \in \underline{\mathbf{x}}_i)$.
6:       Let $e$ be the depth of $y_1 \in \underline{\mathbf{y}}$. If the depths of elements of
         $\underline{\mathbf{y}}$ differ, or if $e$ is the depth of an abelian layer then correct
         all generating sets $\underline{\mathbf{x}}_k$ defined so far (and $\underline{\mathbf{y}}$) as described in
         Section 3.1.1 below.
7:       if $w_e$ is not known then
8:          $k := k + 1$; {The new direct factor found is labelled $T_k$}
9:          Set $v_k := e$; $w_e := k$;
10:         Set $\underline{\mathbf{x}}_k := \underline{\mathbf{y}}$; $rep_k := rep_i \cdot g$; {Thus $x_{k,j} := x_{1,j}^{rep_k}$.}
11:         Set $\mu_k \colon x \mapsto \left(x^{(rep_k^{-1})}\right)^{\gamma_d} \in T$.
12:      fi;
13:      Record that the permutation $g^\pi$ maps $i$ to $w_e$.
14:   od;
15:   $i := i + 1$
16: od;
```

The algorithm terminates when $k = n$ and $i = k + 1$. At that point we have determined the images g^π for all $g \in \underline{\mathbf{g}}$ and therefore the permutation image $G^\pi \leq S_n$. An obvious variant without the for-loop and the if-case can determine g^π for an arbitrary $g \in G$. Thus π is an effective homomorphism.

3.1.1. Ensuring correct depths

Note that (since d is maximally deep), when decomposing as elements of $\mathrm{Aut}(T) \wr S_n$, the image of $x_{1,j}$ in G/R, and thus also the image of every conjugate $x_{i,j} = x_{1,j}^g$ for $g \in G$, has identity components for all but one copy of T. Thus in step 6 of the algorithm, if the observed depth e of a conjugate $x_{i,j}^g$ is that of a nonabelian layer, it must be the correct depth. The depth can only be wrong if it is that of an abelian layer. This can be detected easily.

In this situation, as $\underline{\mathbf{x}}_1$ is a list of preimages of generators of C_d/C_{d+1}, the subgroup C_d^g (and therefore also C_d itself) must have an abelian quotient. This quotient of C_d arose in the chosen composition series *deeper* than d, and at least one of the chosen generators $x_{1,j}$ has a nontrivial image in this quotient.

We thus must modify $\underline{\mathbf{x}}_1$ so that it has a trivial image in this abelian quotient. This can be achieved by choosing new generators in the derived subgroup of $\langle \underline{\mathbf{x}}_1 \rangle$. As C_d/C_{d+1} is simple nonabelian, such a change does not affect generation of the factor group. Indeed, using pretabulated data, we may assume that for the chosen generating set $\underline{\mathbf{t}}$ of the simple group T, we know words in $\underline{\mathbf{t}}$ that express the elements of $\underline{\mathbf{t}}$ as words in commutators of $\underline{\mathbf{t}}$.

We thus replace $\underline{\mathbf{x}}_1$ by the values *of these same words* in $\underline{\mathbf{x}}_1$. These new generators therefore will lie in the derived subgroup $\langle \underline{\mathbf{x}}_1 \rangle'$ but still have the same images $\underline{\mathbf{t}}$ in C_d/C_{d+1}.

After such a modification, we can keep the data structures consistent by replacing *every* generating set $\underline{\mathbf{x}}_k$ computed so far by the values of these same words in $\underline{\mathbf{x}}_k$, because conjugation is an isomorphism. (This is the correction of generators referred to in step 6 of the algorithm.)

Since the newly chosen generators lie in the derived subgroup $\langle \underline{\mathbf{x}}_1 \rangle'$, this situation can happen at most once for every pair of abelian and nonabelian composition factor and thus at most $m(n-m)$ times. (Alternatively, one could immediately form $(n-m)$-fold repeated commutator expressions in the original generators and then select these commutator words for $\underline{\mathbf{x}}$. As this takes time, creates unnecessarily long words, and as this kind of exception occurs rarely, the single-step "correction" process described above is preferable in practice.)

3.1.2. Evaluating the homomorphism

This action homomorphism π, together with the homomorphisms μ_i, now lets us determine the action of G on S and thus yields an effective version of $\rho\colon G \to G/R$. For this, we need to determine for $g \in G$ its image g^ρ in $\mathrm{Aut}(T) \wr S_n$ that describes the action of g.

For each $i \in \{1, \ldots, n\}$ we calculate the images of the generators $\underline{\mathbf{x}}_\mathbf{i}$, representing the i-th copy of T, under conjugation by g. These images lie in the j-th copy of T where $j = i^{(g^\pi)}$. We obtain j from the depth of $x_{i,1}^g$ and thus construct g^π. The generator images in T thus are $\underline{\mathbf{y}} = (\underline{\mathbf{x}}_i^g)^{\mu_j}$. Using the conjugacy test in $\mathrm{Aut}(T)$ (fundamental task III) we find $\alpha_i \in \mathrm{Aut}(T)$ such that $\underline{\mathbf{t}}^{\alpha_i} = \underline{\mathbf{y}}$. The standard embedding into a wreath product gives the image

$$g^\rho = \left(\alpha_{1/g}, \alpha_{2/g}, \ldots, \alpha_{n/g}; g^\pi\right) \in T^n \rtimes S_n = \mathrm{Aut}(T) \wr S_n,$$

where i/g denotes the image of i under $(g^{-1})^\pi$. (The inverse g^{-1} arises, as the index denotes the entries that *are permuted into* the i-th component.) As we can compute π as an action homomorphism, and the α's by calculations in $\mathrm{Aut}(T)$, we can effectively evaluate this image g^ρ for arbitrary $g \in G$.

Calculating g^ρ this way for all $g \in \underline{\mathbf{g}}$ gives generators for $G^\rho \cong G/R$.

We note that this process – acting on generators of the simple factors and forming an image in a wreath product – always constructs a valid homomorphism on G. The only place where the conditions – S being minimally normal and $G/Pker$ being solvable – are used is in establishing that $\ker \rho = R$.

3.2. Case 2: Decomposable Socle

Next consider the case that S is not minimal normal, but $G/Pker$ still is solvable. The same process can be used to determine the action of G on the minimal normal subgroups of G/R in S separately. To this end we maintain a list of marks for all nonabelian leaves, having initially all unmarked. When selecting a nonabelian leaf that is maximally deep, we choose only amongst the so far unmarked leaves. In the orbit algorithm on indices we then mark all leaves whose depth is encountered (and obtain n as the value of k at termination). This construction then gives one homomorphism, call it ρ_1.

If there are yet unmarked leaves remaining, we repeat the process, starting with the deepest so far unmarked leaf and obtain a second homomorphism ρ_2. (Since the nonabelian leaves leading to a ρ_i are obtained under a group action, the calculation of ρ_2 will never encounter a nonabelian leaf that already had been used in the construction of ρ_1.)

To keep the argument from Section 3.1.1 valid for the construction of ρ_2, we now need to ensure that the elements in the generating set $\underline{\mathbf{x}}_1$ have nontrivial images in only one direct factor of $\mathrm{Soc}(G/R)$. If we choose the new generating set $\underline{\mathbf{x}}_1$ as generators on the level of maximal unmarked depth d, then $x \in \underline{\mathbf{x}}_1$ could violate this condition only if its image in G/R projected nontrivially on a socle component corresponding to a deeper composition factor. We can test for such a condition by checking whether the image x^{ρ_1} under the previous homomorphism has a nontrivial socle part.

If this is the case, we find a preimage $a \in G$ under ρ_1 of x^{ρ_1} (see Section 3.4 on how to compute preimages). As ρ_1 was constructed from elements that project only on socle components for previously marked layers, this means that x/a will have trivial image under ρ_1, but still have the same image at depth d. It thus still represents the same image in the composition factor of depth d.

(If multiple previous homomorphisms ρ_i exist, then we consider all of them in sequence.)

In practice, this correction is only relevant if the algorithm would encounter a wrong depth for a conjugate. This can be detected, as this depth had been marked already. One thus can treat this correction in the same way as that of encountering abelian layers, correcting the generators only if a nonabelian layer is encountered, avoiding unnecessary evaluation of the previous ρ_i.

In this way, we obtain homomorphisms $\rho_1, \rho_2, \ldots$ until all nonabelian leaves are marked. Since R acts trivially on S, but every element outside R acts nontrivially, $R = \bigcap_i \ker \rho_i$.

The homomorphism ρ thus can be taken as the subdirect product of the ρ_i, i.e.

$$g^{\rho} = (g^{\rho_1}, g^{\rho_2}, \ldots) \in G^{\rho_1} \times G^{\rho_2} \times \cdots .$$

3.3. Case 3: Nonsolvable Socle Factor

Finally, consider the situation that $G/Pker$ is not solvable and assume that a (so far unmarked) nonabelian composition factor C_d/C_{d+1} corresponds to a nonabelian composition factor F of $G/Pker$. Then all socle factors of S that are moved by F must be *deeper* in the composition series than d.

The above process therefore first determines sufficiently many ρ_i such that the image of their subdirect product represents all socle factors of G/R which are permuted by F, as well as the action of G on these (which includes the contribution of F). When working on C_d/C_{d+1}, the construction then produces another homomorphism $\tilde{\rho}$ whose kernel contains $\bigcap \ker \rho_i$ for these ρ_i.

As it is a valid homomorphism, we can include $\tilde{\rho}$ in the subdirect product construction of ρ without changing $\ker \rho$. Thus the algorithm as described will work also in the case of a nonsolvable socle factor.

In practice of course the extra factor $\tilde{\rho}$ in the image of ρ is an unnecessary burden. When constructing ρ we can simply consider the order of the image group formed by the several ρ_i and eliminate those factors that do not increase the image order. This will automatically eliminate $\tilde{\rho}$.

3.4. Preimages

To make ρ effective, we finally need to be able to compute preimages of elements of G^ρ. Let $P := Pker^\rho \leq G/R$ be the base subgroup of G/R as subgroup of a direct product of wreath products and let $\varpi\colon G/R \to G/Pker$ be defined such that $\pi = \varpi \circ \rho$. By stabilizer chain methods applied to the permutation group $G/Pker$, we can find a generating set $\underline{\mathbf{p}}$ for P, together with elements $\underline{\mathbf{h}} \subset G$ such that $\underline{\mathbf{h}}^\rho = \underline{\mathbf{p}}$. Working in the direct product factors of P (each isomorphic to $\mathrm{Aut}(T)$ and thus permitting decomposition into generators by condition II) with this generating set, we can thus for $y \in P$ determine $g \in G$ such that $g^\rho = y$.

For an arbitrary $x \in G/R$, we first use stabilizer chain methods for $G/Pker$ (see [Ser03, p. 80]) to find $f \in G$ such that $f^\pi = x^\varpi$. As we can evaluate ρ, we calculate f^ρ and consider the quotient $x/f^\rho \in P$. By the above argument, we can determine $g \in G$ such that $g^\rho = x/f^\rho$. Thus $g \cdot f$ is a preimage of x under ρ.

3.5. Proof of the Theorem

To prove Theorem 9, we first estimate the cost of calculating images under ρ. We observe that to determine the image of $g \in G$, we conjugate the two generators for every socle factor by g. We then determine the depth of the images (one chain operation), calculate the images of the generators in the appropriate leaf (two chain operations) and finally conjugate these in $\mathrm{Aut}(T)$ to the chosen generators (one chain operation). This establishes the cost of an element image computation as $4m$ tree operations.

The initial setup and choice of the generating sets $\underline{\mathbf{x}}_i$ consists essentially of determining images g^ρ for all (at most n) generators of G. We have seen that the correction of generators for an abelian factor (Section 3.1.1) can happen at most $m(n-m)$ times and each time requires a new depth computation. The generator correction in Section 3.2 will map each new generator pair $\underline{\mathbf{x}}_{i,j}$, $(i = 1, 2, \ldots, j = 1, 2)$ under the previous ρ_z $(1 \leq z < i)$ and thus map at most

$2(1+2+\cdot+n-1) = (n-1)n$ elements. Each image computation therefore has cost at most $4m$ composition tree operations.

The initial setup thus has a worst case cost of (generator images + correction of wrong abelian depth + correction of nonabelian depth)

$$\begin{aligned} 4mn + m(n-m) \quad &+ \quad 4m(n-1)n \leq 4m(n+n-m+(n-1)n) \\ &= \quad 4m(n^2+n-m) \leq 4(mn^2 + n^2(n-m)) = 4n^3 \end{aligned}$$

composition tree operations. This completes the proof.

Having constructed ρ and the image G^ρ, we now construct normal subgroup generators for $\ker \rho$. We obtain these by evaluating relators for a presentation for G^ρ in preimages of the generators in G. Such a presentation is formed by choosing a composition series for G^ρ, for example from images of the composition tree under ρ, or by using the decomposition of G^ρ as embedded into a direct product of wreath products. For each composition factor we create a presentation using the methods of [HEO05, Chapter 6]. Finally we combine these presentations of composition factors to a presentation of G^ρ as described in [BGK$^+$97, Section 8].

As the algorithm for the radical given in the next section considers G-conjugates, such normal subgroup generators are all we need to proceed.

4. Solvable Matrix Groups

We now return to the specific case of matrix groups. The next aim is to build a data structure for handling the radical R, given its (normal subgroup) generators. For this we want to determine a normal series $R = R_0 \rhd R_1 \rhd \cdots \rhd \langle 1 \rangle$ such that $R_i \lhd G$ and R_i/R_{i+1} is elementary abelian. We also want to obtain a polycyclic generating sequence (PCGS, [LNS84]) $\underline{\mathbf{r}} = (r_1, \ldots, r_k)$ for R that exhibits this normal series, i.e. for each i there are indices $a_i < b_i$ such that $R_{i-1} = \langle R_i, r_{a_i}, r_{a_i+1}, \ldots, r_{b_i} \rangle$.

For each $x \in R$ this yields unique exponents $e_j \in \{0, \ldots, p_i - 1\}$ (setting $p_i = [\langle r_j, \ldots, r_k \rangle : \langle r_{j+1}, \ldots, r_k \rangle]$) such that $x = \prod r_j^{e_j}$. (We shall call this product expression of x the $\underline{\mathbf{r}}$-factorization of x.) These exponents will be crucial for further calculations, and it therefore is important to make their determination as efficient as possible. Thus, while the composition tree in principle allows for the determination of such exponents, we will use an alternative approach, based on [Sim90], which determines the series R_i (from the bottom up) as well as the PCGS $\underline{\mathbf{r}}$ simultaneously. This approach has been used before for handling solvable matrix groups (for example in [AE05]). The main differences here is that we form a normal closure under the larger group G, we ensure elementary abelian layers, and finally we investigate the question of orbit length and give a strategy for the choice of base points.

The main loop in the algorithm from [Sim90] (the procedure *solvable_group*) assumes that already a normal subgroup $N \lhd R$, together with a PCGS for N, is known (it is initialized with $N := \langle 1 \rangle$ at the start) and then processes a generator $x \in R$. If $x \in N$ it is discarded. Otherwise we start determining the normal closure $M := \langle N, x \rangle^R$ by adding conjugates of x to N (by transitive closure) until all conjugates under generators of R lie in the resulting subgroup M. If during this process we find two conjugates y, z of x such that the commutator $w := [y, z] \notin N$, then M/N cannot be abelian. In this case we replace x by w (we remember x for processing at a later stage) and repeat the process of extending N. (As w lies in a lower derived subgroup of R than x does, this kind of exception may happen at most a bounded number of times, a bound being given by [Dix68].)

If, on the other hand, all commutators were found to lie in N, then we determined a larger normal subgroup $M \lhd G$ with M/N abelian and use M as the next step in the normal series, extending the PCGS for N with generators representing a basis of M/N.

We modify this procedure in two places. The first is that we want to ensure that M/N is not only abelian, but elementary abelian. This is clearly the case if the order of x modulo N is prime. We thus determine $|Nx|$ in G/N and if necessary replace x by an appropriate power. (This is repeated if x originated from a commutator w. Also in such a case, the original x needs to be fed to the algorithm again afterwards.) If we assume knowledge of the prime divisors of $|R|$, we can compute $|Nx|$ using [BB99, Claim 3.5], replacing the identity test by a membership test in N.

Secondly, we modify the procedure by determining normal closure (i.e using element conjugation) not just under R, but under G. This guarantees that the subgroups for each elementary abelian layer are G-normal. It also enables us to feed only a set of G-normal subgroup generators for R to the algorithm. The only issue in this modification is termination of the process.

Lemma 10. *When replacing R-closure by G-closure, a commutator can lie outside N only finitely many times.*

Proof. Without loss of generality, assume that $N = \langle 1 \rangle$ and that $A = \langle x \rangle^R$ is elementary abelian (because otherwise there will be commutators with R-conjugates that generate a smaller nontrivial subgroup). Suppose for $g, h \in G$ that $[x^g, x^h] \neq 1$. Then $B = \langle x^g \rangle^R \neq \langle x^h \rangle^R = C$ are elementary abelian normal subgroups of R such that $1 \neq [x^g, x^h] \in B \cap C$. Thus $B \cap C \lhd R$ is a smaller, nontrivial, elementary abelian normal subgroup of R. As R is finite, every strictly descending chain of normal subgroups is finite. $\square$

To allow for an element test in N, we will represent N by a stabilizer chain. As discussed in Section 4.1, this is a feasible endeavor. Once all generators of R have been processed, we end up with a stabilizer chain for R.

As mentioned in [HEO05, p.102], sifting $x \in R$ along this chain allows us to determine the first nonzero exponent in the $\underline{\mathbf{r}}$-factorization of x. In fact, we can determine the exponents for a whole abelian layer R_{i-1}/R_i simultaneously. The stabilizer chain will yields an expression of x as a product in $\underline{\mathbf{r}}$. The restriction to a single abelian layer becomes necessary, because this product will not be in the canonical order of a $\underline{\mathbf{r}}$-factorization. If the multiplication of elements of R is comparatively expensive (and the iterated decompositions for the multiple elementary abelian layers thus is costly), it may be sensible to determine a PC-presentation for R in $\underline{\mathbf{r}}$ for once, and afterwards use this PC-presentation to collect a non-canonical factorization afforded by the stabilizer chain to a proper $\underline{\mathbf{r}}$-factorization.

Furthermore (assuming no rebalancing of the Schreier tree takes place), since the stabilizer chain is built by adding normalizing elements and since each element of $\underline{\mathbf{r}}$ only extends one layer of the stabilizer chain, the position of the base image in its orbit determines the exponent of an element (e.g. if the element r_i extends an orbit of length a to length $a \cdot p$, any element of the form $\prod_{j=i}^{k} r_j^{e_j}$ maps the base point to an image at position $a \cdot e_i \leq q < a \cdot (e_i + 1)$.) That means that exponents can be read off immediately from the positions of the base images without need of group element division.

By also using powers $r_i^{e_i}$, $1 \leq e_i < p_i$, of the PCGS elements as Schreier generators, the sifting process along the stabilizer chain will require exactly k steps, rendering the issue of balancing Schreier trees less relevant.

Remark 11. *The algorithm of [Sim90] selects base points ad-hoc when a new one is needed. In the case of permutation groups, all orbits are bounded by the permutation degree. In the case of matrix groups more care is needed, as a good base point for a normal subgroup R_i might extend to a long orbit under R.*

We therefore first need to determine a suitable base for R, which requires knowledge of a generating set for R. Note that the initial composition tree gave us $|G|$. As we also know $|G/R|$ we thus know $|R|$. The following method thus can certify that a set $\underline{\mathbf{q}} \subset R$ generates R:

- *Set $\hat{R} = \langle \underline{\mathbf{q}} \rangle$.*
- *In a Random-Schreier-Sims algorithm (with a small number of random stabilizer generators on each level) compute a probabilistic stabilizer chain for $\hat{R}$. Section 4.1 below describes how we choose suitable base points for this chain.*

  Let a be the group order obtained from this stabilizer chain.
- *As a failure in the Random-Schreier-Sims algorithm produces an order that is too small, a result $a = |R|$ thus certifies that $R = \langle \underline{\mathbf{q}} \rangle$.*

We perform this test, setting $\underline{\mathbf{q}}$ in first approximation to be the list of normal subgroup generators for R. If the test fails, then we extend $\underline{\mathbf{q}}$ by a few random conjugates and repeat this process until we have a proven generating set for R.

The base points obtained from this stabilizer chain for R then will be used for building the data structure as described above.

4.1. Choice of base points

To make use of a stabilizer chain for solvable matrix groups feasible, we need to show the existence of comparatively short orbits, and give a way of finding them. An indication of this is already given by the solvable groups in the experiments in [MO95]. The actions we will be considering are the natural actions of matrix groups, i.e. the actions on vectors, on subspaces and on cosets of submodules (that is, if M is a submodule, we act on cosets $M+\overrightarrow{\mathbf{v}}$, represented by considering representatives $\overrightarrow{\mathbf{v}}$ sifted through an echelonized basis for M). We will consider these actions not just over the defining field, but also over algebraic extensions, if this provides shorter orbits.

We shall assume in this section that for a finite field $K = \mathbb{F}_q$ the group $R \leq \mathrm{GL}_n(K)$ is a finite solvable group. (If R is defined in characteristic zero, we can first reduce modulo a prime and determine a short orbit there.) We denote the natural module by $V = K^n$.

The condition for "short" we shall use is that the orbit length is bounded by $\max(12, n) \cdot (q^{\frac{n}{2}} + 1)$, i.e. roughly $\sqrt{q^n}$. This is motivated by the following theorem.

Theorem 12 ([Ber76]). *Assume that $|R|$ and q are odd, n even, and that R preserves a nonsingular symplectic form on V. Then R has an orbit on $V \setminus \{\overrightarrow{\mathbf{0}}\}$ of length bounded by $(q^{\frac{n}{2}} + 1)/2$.*

As the following observation shows, in general only the trivial bound $2^n - 1$ holds.

Remark 13. *Let $p = 2^n - 1$ a Mersenne prime. Then $H := GL_n(2)$ has a Singer cycle $\langle s \rangle \leq H$ which is a Sylow p-group of G. Since p is prime, $\langle s \rangle$ will have only regular nontrivial orbits of length $2^n - 1$ which is essentially the full natural module.*

The core of this problem however is the large prime factor of $|G|$. Such large prime factors in general seem to be an obstacle for matrix group calculations, for example [BBS09] needs to assume a discrete logarithm oracle.

We now aim to show that large prime factors are indeed the only obstacle.

For this we first consider the case of irreducible abelian groups:

Definition 14. *For a finite field K and an integer n, let F be the degree n extension of K and α a generator of F^*. A* Singer subgroup *of $G = GL_n(K)$ is a subgroup that is G-conjugate to the multiplication action of α^e on $F \cong K^n$, where $K(\alpha^e) = F$.*

Lemma 15. *Let K be a finite field with q elements and $A \leq GL_n(K)$ an irreducible abelian group with natural module $V = K^n$. Then:*
a) [Hup67, Satz II.3.10] A is a Singer subgroup and thus is cyclic.
b) In this situation A has a nontrivial orbit on subspaces (of L^n for an algebraic extension $K \leq L$) of length dividing $\frac{q^n-1}{q^a-1}$ for every $a \mid n$, $a < n$.

Proof. Let L be a minimal normal extension of K that contains all eigenvalues of all elements of A. Over this field we can bring every element of A into upper triangular form. Since commuting elements preserve each others eigenspaces, this can be done *simultaneously* for all elements. Thus A preserves a 1-dimensional eigenspace W over L. For $\sigma \in \mathrm{Gal}(L/K)$ the Galois conjugate W^σ is a module for $A^\sigma = A$. Now consider $U := \langle W^\sigma \mid \sigma \in \mathrm{Gal}(L/K)\rangle$. Then U is a K-invariant A-module, and thus a KA-submodule of V. Since V is irreducible we have that $U = V$.

By construction, U is generated from 1-dimensional submodules for A, the action of A on each of these being determined by its action on W. Thus the action of A on $U = V$ is diagonalizable and is completely determined by the action of A on W. Thus A is isomorphic to a subgroup of L^* and therefore cyclic. Let $\alpha \in L^*$ be a generator of this cyclic image and $g \in A$ the corresponding group element. The minimality of L implies that $L = K(\alpha)$, thus A is a Singer subgroup and $|L| = q^n$.

Now suppose that there is a subfield $K \le M \le L$ with $|M| = q^a$ with $a \mid n$. Let e be minimal such that $\alpha^e \in M$, thus e is a divisor of $\frac{|L|-1}{|M|-1} = \frac{q^n-1}{q^a-1}$. Then the minimal polynomial of α^e and of g^e has degree at most $[M:K] < [L:K]$. This means that the action of $\langle g^e\rangle$ on V cannot be irreducible, it must preserve a submodule $S \le V$. Then this subspace S has an orbit of length e under A, i.e. A has a nontrivial orbit on subspaces of L^n of length e. $\square$

If n is even, we can simply chose $a = n/2$ and obtain orbit length $\frac{q^n-1}{q^a-1} = q^{\frac{n}{2}} + 1$ as desired.

For other cases (indeed this is without conditions on n) we observe that in general $\frac{q^n-1}{q^a-1}$ has large prime divisors that are unlikely to arise in the order of the groups R that are obtained as radical. Since e must be a divisor of the group order this will limit its magnitude substantially.

Definition 16. *Let q be a prime power, n be an integer and a the largest proper divisor of n. Let $m = \frac{q^n-1}{q^a-1} = \prod_i p_i^{e_i}$ with p_i prime. A prime $p|m$ is called q,n-*fat*, if $\prod_{p_i \le p} p_i^{e_i} > q^{\frac{n}{2}} + 1$.*

As the orbit length of Lemma 15 must also divide the group order, this yields the trivial consequence:

Corollary 17. *Let $A \le GL_n(q)$ be abelian irreducible. If $|A|$ is not divisible by any q,n-fat prime, then A has an orbit of length bounded by $q^{\frac{n}{2}} + 1$ on subspaces.*

Factorizations of $q^n - 1$ indicate that fat primes are usually very large:

Lemma 18. *For q,n as given in the following table*

q	2	3	4	5	7	8	9	11	13	16	17	19
$n \le$	822	522	226	204	268	226	138	226	192	226	172	162

a q,n-fat prime is larger than $\frac{n^3}{24}$, with the exception $q = 2$, $n = 21$ (for which the orbit length is bounded by 16513). Furthermore, all fat primes in this range are primitive prime divisors for $q^n - 1$, i.e. groups divisible by fat primes in this range are large *in the sense of [GPPS99].*

Proof. Explicit calculation in GAP [GAP13], using the "Cunningham" tables [BLS$^+$88]. $\square$

The limits on n given are purely due to limitations of factorization algorithms. We conjecture that this bound $\frac{n^3}{24}$ for the magnitude of fat primes holds for arbitrary n.

Now we consider the general case.

Theorem 19. *Let $R \leq GL_n(q)$ be a solvable group whose order is not divisible by any q,m-fat prime for $n/2 < m \leq n$. Then R has an orbit on vectors, subspaces or submodule cosets of L^n (where $\mathbb{F}_q \leq L \leq \overline{\mathbb{F}_q}$) of length at most* $\max(12, n) \cdot (q^{\frac{n}{2}} + 1)$

Proof. Assume first that R acts reducibly, i.e. it affords a proper submodule $M \leq V$ of dimension $d < n$. If the action of R on M is nontrivial we recursively find an orbit on vectors, subspaces or submodule cosets within M of length bounded by $\max(12, d) \cdot (q^{\frac{d}{2}} + 1) \leq \max(12, n) \cdot (q^{\frac{n}{2}} + 1)$.

Similarly, if the action of R on V/M is nontrivial, we recursively consider this action and find an orbit of the desired bounded length. By replacing vectors of V/M with M-cosets, subspaces of V/M with their full preimages in V, and subspace cosets with cosets for the subspace preimage, this also provides an action for R with the same orbit length.

If the actions both on M (of dimension d) and on V/M (of dimension $\bar{d} = n - d$) are trivial, then the elements of R (in a basis adapted to M) are matrices of the form $\begin{pmatrix} 1_d & 0 \\ A & 1_{\bar{d}} \end{pmatrix}$ and multiply by addition of the A-parts. In particular R is abelian.

Furthermore, in this basis, the subspaces L_i, spanned by the first i basis vectors ($1 \leq i < n$) all are submodules. If for some i the action on L_i or the action on V/L_i is nontrivial, the above recursive argument again holds.

If the action on all the L_i and all the V/L_i is trivial, then all elements of R have 1 on the diagonal and the only other nonzero entry in position $n, 1$, thus $|R| = q$. The result follows trivially. This concludes the case of a reducible action.

If R acts irreducibly, but preserves a system of imprimitivity $V = V_1 \oplus \cdots \oplus V_k$, we can act on the subspaces of this system, resulting in an orbit of length at most $k \leq n$.

Otherwise, by [Luk92, Theorem 6.1], R has an abelian subgroup of index $\leq \max(12, n)$. If this subgroup A is reducible, we take the R-orbit of an A-submodule, which therefore must have length $\leq \max(12, n)$.

If A is irreducible, then Lemma 15, together with the exclusion of fat primes, shows that A has an orbit on vectors of length at most $q^{\frac{n}{2}} - 1$. Now R will extend this orbit to length at most $\max(12, n) \cdot (q^{\frac{n}{2}} - 1)$ as claimed. $\square$

Remark 20. *For a unimodular R, the recursion to vectors in submodules and to cosets of submodules guarantees an orbit of length $\leq q$.*

The proof of Theorem 19 indicates a strategy to find short orbits for a matrix group R. (In practice these orbits often are much shorter than promised by the theorem).

After testing a few vectors using the default strategy of [MO95], test whether R acts reducibly, using the MeatAxe [Par84]. If this is the case, consider the action on vectors in the submodule or cosets of the submodule and try to find short orbits there. This forms the first class of candidates for short orbits.

Instead of determining imprimitivity, we observe that an imprimitive R can be embedded in a wreath product of the form $B \wr C$ with C solvable. Some subgroup in the derived series of R will then be contained in the base subgroup $B^m \cap R$ and thus will act reducibly, exposing submodules. Therefore, if R acts irreducibly, we look for reducible subgroups in the derived series of R.

To approximate the derived series, we determine iteratively subgroups $R_i \leq R$ by setting $R_0 = R$ and R_i being generated by random commutators of elements of R_{i-1}. (In practice, forming 10 random commutators seems to work well.) We construct these subgroups for increasing i until, for some i, R_i acts reducibly on the natural module; let M be a minimal proper submodule for this R_i. As a subspace, M has an orbit under G of length $\leq \left[G{:}R_i\right]$. If R acts imprimitively and $R_i \leq B^m$, then M is a candidate for the imprimitivity decomposition of the regular module. In this case the length of the orbit of M divides the dimension of the natural module.

If neither of these strategies succeeds, R (and thus G) acts irreducibly and primitively. Using the classification of [GPPS99], in these cases either G has only small solvable normal subgroups, in which case orbit length of R is not a concern; or G is solvable, in which case it is probably most efficient to work in G using a polycyclic presentation, following [AE05].

5. Representation of Subgroups

We now consider how to represent arbitrary subgroups of G in a way that provides a constructive element test.

Thanks to the PCGS, the radical R, as well as its subgroups, can be handled efficiently. We also already assumed that the radical factor G/R is represented in a suitable way so that we can compute preimages of elements of G/R under ρ. The process described in Section 3 can be used to determine element images under ρ, but this calculation is expensive, possibly requiring multiple conjugacy tests in $\mathrm{Aut}(T)$ to determine the image of a single element. It therefore is desirable to minimize the number of evaluations of ρ. This suggests the following data structure to represent subgroups:

Definition 21. *Let $U \leq G$. A* generalized PCGS *(GPCGS) for U is the following triple of elements:*

a) *A PCGS $\{u_1, \ldots, u_k\}$ for $U \cap R$, induced with respect to the PCGS for R.*
b) *A set of group elements $\{g_1, \ldots, g_m\}$, such that*

$$U = \langle u_1, \ldots, u_k, g_1, \ldots, g_m \rangle.$$

c) *Images $a_i := g_i^\rho$ of these extra generators under ρ. (This means that $U^\rho = \langle a_1, \ldots, a_m \rangle$.)*

Clearly such a data structure is obtained for G. If $U \leq G$ is given by generators, it can be obtained by determining $U^\rho \leq G/R$ (by calculating generator images) and by evaluating relators for a presentation for U^ρ to obtain normal subgroup generators for $U \cap R$. We can then form an induced PCGS for the U-closure of the subgroup generated by these using standard techniques for solvable groups [LNS84].

We observe that a GPCGS lets us solve the three basic tasks for subgroups:

Subgroup Order $|U| = |U^\rho| \cdot |U \cap R|$. The first of these two factors can be computed in G/R from the images $\{a_i\}$, the second is the product of the relative orders of the PCGS for $U \cap R$.

Membership test To decide if $x \in U$, first test membership $x^\rho \in U^\rho = \langle a_1, \ldots \rangle$. If this test fails, $x \notin U$. Otherwise write $x^\rho = \prod_j a_{i_j}^{e_j}$ and consider $y = x/\left(\prod_j g_{i_j}^{e_j}\right) \in R$. Then $x \in U$ if and only if $y \in U \cap R$, which can be tested using the PCGS.

Evaluating Homomorphisms The membership test implicitly decomposes $x \in U$ as a product of the elements in the GPCGS. Thus, if a homomorphism is given by prescribing images of a GPCGS, we can evaluate the image.

5.1. Generalized PCGS in the Trivial-Fitting model

To work with such a GPCGS and group elements x, we will in general need to know the image $x^\rho \in G/R$. To avoid evaluation of ρ, we will therefore maintain for every $x \in G$ that arises in an algorithm its image x^ρ as a "shadow". (These images are known by definition for all generators in a GPCGS.) When forming a product $x \cdot y$, we form the shadow $x^\rho \cdot y^\rho$ at the same time.

Doing so might seem complicated, but using the Trivial-Fitting model of computation, the extra cost is limited.

These algorithms first solve the problem in G/R. This is typically the part of the calculation that is more complicated, having to deal with the simple nonabelian composition factors. However all of these calculations will take place only in G/R.

The second stage of these algorithms then takes preimages under ρ. Suppose U is the preimage of a subgroup $V \leq G/R$. A GPCGS for U then consists of a PCGS for R, together with preimages of generators for V, as well as these generators themselves (as images of the preimages).

The algorithms now iteratively lift over the elementary abelian layers of R. In each lifting step the result is assumed to be known modulo R_i and to be computed modulo R_{i+1}. This computation typically involves an orbit/stabilizer calculation for an action on R_i/R_{i+1} (or a derived entity, such as a cohomology group).

For $U \leq G$ given by a GPCGS, an orbit/stabilizer computation for $\omega \in \Omega$ now proceeds in two stages: For $N = U \cap R$, calculate (using solvable groups methods [LNS84]) the orbit ω^N and stabilizer $\mathrm{Stab}_N(\omega) = R \cap \mathrm{Stab}_U(\omega)$. As $N \lhd U$, ω^N is a block in a system of imprimitivity for U. Now U will act on such blocks and we can do an orbit/stabilizer calculation under the action of the extra generators $\{u_i\}$. (To test whether two set images are the same, it is sufficient to test containment of a single point. If "minimal" (in any definition) orbit elements under $N \leq R$ can be determined efficiently, every image $(\omega^N)^u$ needs to be represented by its minimal element only.) Whenever a Schreier generators for the stabilizer arises, form the corresponding product generator in the generator images $\{a_i\} \subset G/R$; this will be the image of the Schreier generator under ρ.

As $N \leq \mathrm{Stab}_U(\omega^N)$, we can test redundancy of Schreier generators via these images in G/R.

The elements of U obtained this way will stabilize ω^N, they need to be corrected by an element of N (using the known orbit ω^N) such that they will fix ω.

As we already know a PCGS for $\mathrm{Stab}_N(\omega) = R \cap \mathrm{Stab}_U(\omega)$, we can extend it with these corrected Schreier generators (and their images) to obtain a GPCGS for $\mathrm{Stab}_U(\omega)$.

6. Element Conjugacy

The earliest practical Trivial-Fitting algorithm is that for conjugacy classes of elements [CS97]. This section will briefly describe how to adapt this algorithm to the data structures described so far. For this problem, both the lifting step (essentially duplicating the calculation for solvable groups from [MN89]), as well as the calculations for the radical factor, exploiting its structure as a subdirect product of subgroups of wreath products [Hul00, CH06], are well understood. This section therefore assumes knowledge of the algorithm for permutation groups and will only describe the adaptations that were necessary. We concentrate on the case of element centralizer whose description is easiest; element conjugacy and determination of class representatives can be done analogously.

Assume that $g \in G$ is given and we want to determine a GPCGS for $C_G(g)$. Let $x = g^\rho \in G/R$. The first part of the calculation is to determine $C_{G/R}(x)$.

If we represent the radical factor G/R as a permutation group, we can use standard backtrack algorithms for this centralizer computation.

Otherwise we can use the structure of the radical factor to perform such a calculation.

To centralize $x \in G/R$, let $S = S^*/R$ be the socle of G/R and let $C = C_{G/R}(x)$. Clearly it is sufficient to determine $C \cap S$ together with coset representatives for (generators of) $C/C \cap S$.

To determine $C \cap S$ consider $c \in S$, formed from n components $c_1, \ldots, c_n$. We want to find all such c such that $cx = xc$. Assume first that component

i is left fixed by x. Then c_i must centralize the projection of x onto the i-th component, and the set of all possibilities for c_i can be obtained by a centralizer computation in the i-th component T_i of S.

If x maps the i-th component to the j-th component, then $c_j = c_i^x$ as both are the j-th projection of $c = c^x$. Furthermore c also must centralize any power of x, thus c_i must centralize x^k where k is minimal such that $i^{x^k} = i$. (If this holds for one component in the x-orbit, it automatically holds for all others once $c_j = c_i^x$ is fulfilled.)

It is easily seen that these conditions on the c_i are also sufficient to centralize x. This completes the description of $C \cap S$.

To find representatives of $C/C \cap S$, first centralize x modulo S. (This can be done for example by working in the permutation image $G/Pker$ and then lifting over the solvable factor $Pker/S^*$.) Let $S \leq D \leq G/R$ such that $D/S = C_{G/S}(Sx)$. For $d \in D$ we can test by iterated conjugacy tests in the components of S whether there exists $s = s_d \in S$ such that $x^d = x^s$. We now perform a backtrack search over D/S (which typically is a small group), finding the largest subgroup such that all its elements d have such an associated s_d. The set of these s_d then generates C modulo $C \cap S$.

We now form a GPCGS for the full preimage of C in G. This is formed by taking preimages for generators of C, together with the PCGS for R. The second part of the calculation now consists of lifting steps over the elementary abelian layers of R.

In each step of this lifting (as described in [MN89]) we have a subgroup, given by a GPCGS, that represents the centralizer in a factor group. This group acts on the next elementary abelian layer and a vector stabilizer is computed in an orbit/stabilizer algorithm as described in Section 5.1. (When calculating centralizers, only the stabilizer is of interest; when determining conjugacy or listing class representatives, orbit membership or representatives are also used.)

6.1. Large Modules

This orbit-stabilizer calculation (a group G acting on the module N) can become problematic if N is too large to enumerate all elements. This is of particular concern when determining conjugacy classes for which the representative h (using the notation of [MN89]) is central, as in this case the commutator $[h, N]$ is trivial, and no reduction to a space smaller than N is possible.

For matrix groups this situation occurs naturally for reducible groups. If the elements of G have the form $\begin{pmatrix} A & 0 \\ B & C \end{pmatrix}$ then the A-parts and the C-parts describe factor groups. The intersection of the kernels for these two factor groups consists of matrices of the form $\begin{pmatrix} 1 & 0 \\ B & 1 \end{pmatrix}$ but these matrices form an elementary abelian normal subgroup N in defining characteristic. Often this yields a minimal normal subgroup of large order.

However there is a normal subgroup $M \lhd G$ (e.g. the kernel when projecting on the A-part or on the C part) that acts reducibly on N. By Clifford theory, N is the direct sum of conjugate M-modules.

In such a situation, we first enumerate M-orbits on one such M-submodule. This also determines the orbits of M on the conjugate M-submodules. The M-orbits on N can then be parameterized as cartesian products of these submodule orbits without the need to enumerate N. As $M \lhd G$, these M-orbits form blocks for the action of G. We can thus fuse the M-orbits to G-orbits, and obtain G-stabilizers, by considering the action of G on single block representatives, similar to the process described in Section 5.1.

In the examples below, groups such as $11^9.(\mathrm{SL}_3(11) \times \mathrm{SL}_3(11))$ are of this type and special treatment of this case substantially aided performance.

6.2. Implementation

The existing algorithm in GAP [GAP13] for conjugacy classes in permutation groups has been adapted by the author to use only the interface of the radical factor homomorphism ρ and GPCGS of subgroups. These data structures are provided for permutation and for matrix groups. For permutation groups this is done using existing code (essentially [LS97] for ρ and [Sim90] for the PCGS). For matrix groups this is done as described in the preceding sections, utilizing the recog package [NS$^+$] to obtain the composition tree. The resulting algorithm will apply independent of the representation of the group and will be made available in a future release of GAP.

The following table gives timings for a pool of examples, many of them taken from maximal subgroups of sporadic groups as provided by the ATLAS web pages [WWT$^+$10]. Many of the larger examples do not have faithful orbits on vectors (or related geometric structures) of reasonable length and thus are beyond the scope of existing stabilizer-chain based methods. Timings are in seconds on a 2.66GHz Mac Pro with 12GB of memory.

While the number of direct factors of $\mathrm{Soc}(G/R)$ in the examples is typically small, this is purely to avoid an unfeasibly large number of conjugacy classes; construction of ρ has been successful in larger cases.

In one case, the same group is given in different representations to indicate the effect of element size – comparing the timings show they scale essentially as element arithmetic does.

Each group's name will either identify it in ATLAS notation as a subgroup of a simple group, or (in particular if no larger parent group is given) give a construction from smaller constituents. In this case semidirect products (":") with elementary abelian groups were formed as affine groups. Direct products are formed by acting on direct sums of submodules. Subdirect products (indicated by the symbol "$\curlywedge_n$", where n is the order of the shared factor group) are constructed as subgroups of the corresponding direct product. Wreath products stabilize a decomposition as direct sum of subspaces (Aschbacher class $\mathcal{C}_2$).

All of these constructions are easy cases for finding a composition tree. The runtimes given thus reflect primarily the algorithms described in this paper, not the composition tree construction. A group that is a harder case for matrix group recognition of course would show a larger absolute runtime for determination of its conjugacy classes. This in itself however would not make it a better test example here, as we considered the composition tree as a black-box structure.

For each group, "**Parent**" indicates in which group the example was constructed. An reverse arrow over this parent group name $\overleftarrow{G}$ means that the representation of the subgroup obtained this way was reduced to a smaller degree. If no parent is given, the group is constructed generically from imprimitive wreath products and reducible direct products.

As all dependency on choice of basis vectors is hidden in the construction of the composition tree, no attempt to hide the group construction by conjugation with a random matrix was done.

The group's order is given by "**Order**"; "**Classes**" is the number of conjugacy classes determined.

"q" indicates the field over which the matrices are written. If no q is given the group is a permutation group. "**deg**" is the degree of the representation.

In the current implementation the factor group $F = G/R$ is always represented as a permutation group, "$\mathbf{deg}_F$" indicates the permutation degree used.

"$\mathbf{deg}_R$" is the maximal orbit length used in the stabilizer chain for R. The existing implementation used some heuristic bounds to accept orbit lengths as "good enough" without attempting further improvements.

"$t_{\mathbf{Setup}}$" is the time for constructing the composition tree and for setting up the initial data structures (the homomorphism ρ and the PCGS for $R = \ker \rho$). "$t_{\mathbf{Class}}$" is the (additional) time for determining representatives for the conjugacy classes of elements of the group as well as GPCGS for their centralizers.

The implementation in GAP performs better for permutation groups than the existing library method, as the choice of a GPCGS to represent subgroups naturally limits the number of subgroup generators.

Acknowledgement

The author would like to thank an unnamed referee for pointing out the need for generator correction (Section 3.1.1) as well as many other corrections, James Wilson for helpful comments on a draft version, and Eamonn O'Brien for suggestions on exposition. Part of the work was done during the author's visits to the University of St. Andrews and to the University of Auckland. The author is grateful for the hospitality experienced during these visits. The author's work has been supported in part by NSF Grant DUE-0633333 and by Simons Foundation Collaboration Grant 244502.

Group	Parent	Order	deg	q	Classes	$\deg_F$	$\deg_R$	t_{Setup}	t_{Class}
$2^2.\mathrm{U}_6(2).2$	Fi_{23}	$2^{18}3^6 5{\cdot}7{\cdot}11$	253	3	146	672	4	148	103
$2^{11}.\mathrm{M}_{24}$	J4	$2^{21}3^3 5{\cdot}7{\cdot}11{\cdot}23$	112	2	72	24	2048	7	4
3.Suz.2	Co_1	$2^{14}3^8 5^2 7{\cdot}11{\cdot}13$	98280	–	106	32760	98280	67	436
3.Suz.2	Co_1	$2^{14}3^8 5^2 7{\cdot}11{\cdot}13$	24	2	106	1782	3	12	33
$3.2^{15}.\mathrm{M}_{24} \times \mathrm{A}_5$		$2^{27}3^5 5^2 7{\cdot}11{\cdot}23$	14	4	1520	281	4	3	97
$\mathrm{GL}_2(5) \wr \mathrm{S}_5$	$\mathrm{GL}_{10}(5)$	$2^{28}3^6 5^6$	10	5	176256	35	256	12	765
$2^{1+20} : \mathrm{U}_6(2)$	${}^2\mathrm{E}_6(2)$	$2^{36}3^6 5{\cdot}7{\cdot}11$	78	2	286	672	2048	82	54
$2^{2+10+20}.(\mathrm{M}_{22} : 2 \times \mathrm{S}_3)$	$\overleftarrow{\mathrm{B}}$	$2^{41}3^3 5{\cdot}7{\cdot}11$	813	2	1226	22	16384	657	11708
$2^{1+20} : \mathrm{L}_6(2)$	$\mathrm{E}_6(2)$	$2^{36}3^4 5{\cdot}7^2 31$	27	2	340	63	2048	2	38
$(4^2 : \mathrm{SL}_4(2)) \wr \mathrm{S}_5$		$2^{33}3^6 5^6$	15	4	3753	35	16	5	105
$(\mathrm{GL}_2(5) \wr \mathrm{S}_3) \curlywedge_6 (\mathrm{L}_2(11) \wr \mathrm{S}_3)$		$2^{22}3^7 5^6 11^3$	21	5	1235200	216	64	17	22886
$(\mathrm{GL}_2(5) \wr \mathrm{S}_3) \curlywedge_2 (\mathrm{L}_2(11) \wr \mathrm{S}_3)$		$2^{22}3^8 5^6 11^3$	21	5	503808	216	64	22	9078
$2^{9+16}.\mathrm{S}_8(2)$	$\overleftarrow{\mathrm{B}}$	$2^{41}3^5 5^2 7{\cdot}17$	394	2	703	255	8192	455	998
$3^{1+12}.2\mathrm{Suz}.2$	$\overleftarrow{\mathrm{M}}$	$2^{15}3^{20}5^2 7{\cdot}11{\cdot}13$	78	3	253	1782	6	427	76
$5^9{:}(\mathrm{GL}_3(5) \times \mathrm{GL}_3(5))$		$2^{14}3^2 5^{15}31^2$	6	5	18464	62	125	3	361
$(6.\mathrm{A}_5) \wr \mathrm{S}_5$	$\mathrm{GL}_6(25) \wr \mathrm{S}_5$	$2^{23}3^{16}5^6$	30	25	526473	80	7776	27	2863
$3^{15} : (\mathrm{M}_{11} \wr \mathrm{S}_3)$	$\mathrm{AGL}_{15}(3)$	$2^{13}3^{22}5^3 11^3$	16	3	3200	33	6561	14	129
$2^{1+22}.\mathrm{Co}_2$	$\overleftarrow{\mathrm{B}}$	$2^{41}3^6 5^3 7{\cdot}11{\cdot}23$	1045	2	448	2300	8192	654	4790
$((2^2 \times 3).\mathrm{U}_6(2)) \wr \mathrm{S}_2$		$2^{35}3^{14}5^2 7^2 11^2$	54	4	77814	1386	4	41	1281
$11^9{:}(\mathrm{SL}_3(11) \times \mathrm{SL}_3(11))$		$2^9 3^4 5^{24}7{\cdot}31$	6	11	20759	266	1331	5	2819
$(5^3 \cdot \mathrm{L}_3(5)) \wr \mathrm{S}_3$	$\overleftarrow{\mathrm{Ly}} \wr \mathrm{S}_3$	$2^{16}3^4 5^{18}31^3$	75	5	12200	93	125	97	1270
$2^{44}{:}\big(\mathrm{M}_{11} \times \big(2^4{:}(\mathrm{S}_3 \times \mathrm{S}_3)\ \big)\big)$		$2^{60}3^5 5{\cdot}7{\cdot}11{\cdot}23$	15	2	10759	276	16	5	4167
$\big(3^{10}{:}(\mathrm{M}_{11} \wr 2)\big) \curlywedge_2 (\mathrm{J}_2 \wr 2)$		$2^{23}3^{20}5^6 7^2 11^2$	83	3	127764	310	6561	44	13729
$7^{12}{:}(\mathrm{SL}_3(7) \times \mathrm{SP}_4(7))$		$2^{14}3^5 5^2 7^{19}19$	7	7	7701	457	2401	12	2670

References

[AE05] Björn Assmann and Bettina Eick. Computing polycyclic presentations for polycyclic rational matrix groups. *J. Symbolic Comput.*, 40(6):1269–1284, 2005.

[AG84] M. Aschbacher and R. Guralnick. Some applications of the first cohomology group. *J. Algebra*, 90(2):446–460, 1984.

[Atk84] Michael D. Atkinson, editor. *Computational group theory*. Academic press, 1984.

[BB99] László Babai and Robert Beals. A polynomial-time theory of black box groups. I. In C. M. Campbell, E. F. Robertson, N. Ruskuc, and G. C. Smith, editors, *Groups St Andrews 1997 in Bath*, volume 260/261 of *London Mathematical Society Lecture Note Series*, pages 30–64. Cambridge University Press, 1999.

[BBS09] László Babai, Robert Beals, and Ákos Seress. Polynomial-time theory of matrix groups. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA*, pages 55–64. ACM Press, 2009.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The MAGMA algebra system I: The user language. *J. Symbolic Comput.*, 24(3/4):235–265, 1997.

[Ber76] T. R. Berger. Characters and derived length in groups of odd order. *J. Algebra*, 39(1):199–207, 1976.

[BGK$^+$97] László Babai, Albert J. Goodman, William M. Kantor, Eugene M. Luks, and Péter P. Pálfy. Short presentations for finite groups. *J. Algebra*, 194:97–112, 1997.

[BHLGO] Henrik Bäärnhielm, Derek Holt, C.R Leedham-Green, and E.A. O'Brien. A practical model for computation with matrix groups. Preprint.

[BLS$^+$88] John Brillhart, D. H. Lehmer, J. L. Selfridge, Bryant Tuckerman, and S. S. Wagstaff, Jr. *Factorizations of $b^n \pm 1$*, volume 22 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI, second edition, 1988. $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers.

[CCH01] John Cannon, Bruce Cox, and Derek Holt. Computing the subgroup lattice of a permutation group. *J. Symbolic Comput.*, 31(1/2):149–161, 2001.

[CCN$^+$85] J[ohn] H. Conway, R[obert] T. Curtis, S[imon] P. Norton, R[ichard] A. Parker, and R[obert] A. Wilson. *ATLAS of finite groups*. Oxford University Press, 1985.

[CH03] John Cannon and Derek Holt. Automorphism group computation and isomorphism testing in finite groups. *J. Symbolic Comput.*, 35(3):241–267, 2003.

[CH04] John Cannon and Derek Holt. Computing maximal subgroups of finite groups. *J. Symbolic Comput.*, 37(5):589–609, 2004.

[CH06] John J. Cannon and Derek F. Holt. Computing conjugacy class representatives in permutation groups. *J. Algebra*, 300(1):213–222, 2006.

[CS97] John Cannon and Bernd Souvignier. On the computation of conjugacy classes in permutation groups. In Wolfgang Küchlin, editor, *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 392–399. The Association for Computing Machinery, ACM Press, 1997.

[Dix68] John D. Dixon. The solvable length of a solvable linear group. *Math. Z.*, 107:151–158, 1968.

[EH01] Bettina Eick and Alexander Hulpke. Computing the maximal subgroups of a permutation group I. In William M. Kantor and Ákos Seress, editors, *Proceedings of the International Conference at The Ohio State University, June 15–19, 1999*, volume 8 of *Ohio State University Mathematical Research Institute Publications*, pages 155–168, Berlin, 2001. de Gruyter.

[Fei80] Walter Feit. Some consequences of the classification of finite simple groups. In *The Santa Cruz Conference on Finite Groups (Univ. California, Santa Cruz, Calif., 1979)*, volume 37 of *Proc. Sympos. Pure Math.*, pages 175–181. Amer. Math. Soc., Providence, R.I., 1980.

[GAP13] The GAP Group, `http://www.gap-system.org`. *GAP – Groups, Algorithms, and Programming, Version 4.6.3*, 2013.

[GPPS99] Robert Guralnick, Tim Penttila, Cheryl E. Praeger, and Jan Saxl. Linear groups with orders having certain large prime divisors. *Proc. London Math. Soc. (3)*, 78(1):167–214, 1999.

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of Computational Group Theory*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, FL, 2005.

[Hol97] Derek F. Holt. Representing quotients of permutation groups. *Quart. J. Math. Oxford Ser. (2)*, 48(191):347–350, 1997.

[HS08] Derek F. Holt and Mark J. Stather. Computing a chief series and the soluble radical of a matrix group over a finite field. *LMS J. Comput. Math.*, 11:223–251, 2008.

[Hul98] Alexander Hulpke. Computing normal subgroups. In Oliver Gloor, editor, *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, pages 194–198. The Association for Computing Machinery, ACM Press, 1998.

[Hul00] Alexander Hulpke. Conjugacy classes in finite permutation groups via homomorphic images. *Math. Comp.*, 69(232):1633–1651, 2000.

[Hup67] Bertram Huppert. *Endliche Gruppen I*, volume 134 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1967.

[LNS84] Reinhard Laue, Joachim Neubüser, and Ulrich Schoenwaelder. Algorithms for finite soluble groups and the SOGOS system. In Atkinson [Atk84], pages 105–135.

[LS97] Eugene M. Luks and Ákos Seress. Computing the fitting subgroup and solvable radical for small-base permutation groups in nearly linear time. In Larry Finkelstein and William M. Kantor, editors, *Proceedings of the 2nd DIMACS Workshop held at Rutgers University, New Brunswick, NJ, June 7–10, 1995*, volume 28 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 169–181, Providence, RI, 1997. American Mathematical Society.

[Luk92] E. M. Luks. Computing in solvable matrix groups. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 111–120. IEEE Computer Society, 1992.

[MN89] M. Mecky and J. Neubüser. Some remarks on the computation of conjugacy classes of soluble groups. *Bull. Austral. Math. Soc.*, 40(2):281–292, 1989.

[MO95] Scott H. Murray and E. A. O'Brien. Selecting base points for the Schreier-Sims algorithm for matrix groups. *J. Symbolic Comput.*, 19(6):577–584, 1995.

[NS$^+$] Max Neunhöffer, Ákos Seress, et al. recog. `http://www.gap-system.org/Packages/recog.html`. A GAP package.

[NS06] Max Neunhöffer and Ákos Seress. A data structure for a uniform approach to computations with finite groups. In *ISSAC 2006*, pages 254–261. ACM, New York, 2006.

[O'B11] E A. O'Brien. Algorithms for matrix groups. In C. M. Campbell, M. Quick, E. F. Robertson, C. Roney-Dougal, and G. C. Smith, editors, *Groups St Andrews 2009 in Bath*, London Mathematical Society Lecture Note Series, pages 83–90. Cambridge University Press, 2011.

[Par84] Richard Parker. The Computer Calculation of Modular Characters (the MeatAxe). In Atkinson [Atk84], pages 267–274.

[Ser03] Ákos Seress. *Permutation Group Algorithms*. Cambridge University Press, 2003.

[Sim90] Charles C. Sims. Computing the order of a solvable permutation group. *J. Symbolic Comput.*, 9:699–705, 1990.

[WWT+10] R.A. Wilson, P. Walsh, J. Tripp, I. Suleiman, S. Rogers, R.A. Parker, S. Norton, S. Nickerson, S. Linton, J. Bray, and R. Abbott. ATLAS of finite group representations. `http://brauer.maths.qmul.ac.uk/Atlas/v3/`, 2010.