

# TENSOR DECOMPOSITION OF THE SUZUKI GROUPS

HENRIK BÄÄRNHIELM

ABSTRACT. We give a Las Vegas algorithm that, given  $X \subseteq \mathrm{GL}(d, q)$  such that  $d > 4$  and  $\langle X \rangle$  is isomorphic to a Suzuki group  $\mathrm{Sz}(q)$ , finds a tensor decomposition of the natural module of  $\langle X \rangle$ . This gives an explicit isomorphism from  $\langle X \rangle$  to  $\mathrm{Sz}(q)$ .

The algorithm has one version for large fields and another for small fields.

Implementations of the algorithms are available for the computer algebra system MAGMA.

## 1. INTRODUCTION

The problem of *constructive recognition* of  $G \leq \mathrm{GL}(d, q)$  can be defined as follows: given  $X \subseteq \mathrm{GL}(d, q)$  such that  $\langle X \rangle \cong G$ , construct an *explicit isomorphism* from  $G$  to a *standard copy*  $H$  of  $G$ . An explicit isomorphism  $\psi : G \rightarrow H$  is an isomorphism where  $\psi(g)$  and  $\psi^{-1}(h)$  can be computed efficiently for every  $g \in G$  and  $h \in H$ .

This paper will consider constructive recognition for the Suzuki groups  $\mathrm{Sz}(q)$ , where  $q = 2^{2m+1}$  for  $m > 0$ , one of the infinite families of finite simple groups. In [1] we considered constructive recognition  $\mathrm{Sz}(q)$  in any of its equivalent representations in dimension 4. Here we show how to construct explicit isomorphisms from  $G \leq \mathrm{GL}(d, q)$ , where  $G \cong \mathrm{Sz}(q)$ ,  $d > 4$  and  $q = 2^{2m+1}$  for  $m > 0$ , to the standard copy of  $\mathrm{Sz}(q)$ . The standard copy we use is defined in [1]. The motivation for the constructive recognition of  $\mathrm{Sz}(q)$  comes from the *matrix recognition project* (see [12]).

Let  $V$  be the given module of  $G$ . We will assume that  $V$  is absolutely irreducible, so that  $V$  is isomorphic to a tensor product of *twisted versions* (see Section 2) of the natural module of  $\mathrm{Sz}(q)$ . Hence, to construct an explicit isomorphism from  $G$  to a conjugate  $H$  of  $\mathrm{Sz}(q)$ , it is sufficient to find a tensor decomposition of  $V$ . This amounts to finding a basis of  $V$  that exhibits  $V$  as a tensor product. An algorithm in [1] can then be used to construct an explicit isomorphism from  $H$  to  $\mathrm{Sz}(q)$ . In Section 2 we describe the theoretical background in more detail.

Our method for finding a tensor decomposition is divided into two cases. In Section 3 we give an algorithm whose correctness and complexity rely on respective conjectures, and which only works when  $q$  is sufficiently large. It has time complexity polynomial in  $d$  and in  $\log(q)$  and relies on the tensor decomposition algorithm of [13]. Given a suitable subspace of  $V$ , that algorithm is guaranteed to be polynomial time, and we will describe how to construct such a subspace for the Suzuki groups.

In Section 4 we give another algorithm which always works, but which has time complexity polynomial in  $d$  and in  $q$ , so it is only feasible when  $q$  is small.

Implementations of the algorithms are available in MAGMA (see [4]).

We thank the anonymous referee, John Bray, Charles Leedham-Green, Eamonn O'Brien, Maud de Visscher and Robert Wilson for their comments and helpful advice.

## 2. PRELIMINARIES

We now discuss the theoretical background in more detail. If  $V$  is an  $FG$ -module for some group  $G$  and field  $F$ , with action  $f : V \times FG \rightarrow V$ , and if  $\Phi$  is an automorphism of  $G$ , denote by  $V^\Phi$  the  $FG$ -module which has the same elements as  $V$  and where the action is given by  $(v, g) \mapsto f(v, \Phi(g))$  for  $g \in G$  and  $v \in V^\Phi$ , extended to  $FG$  by linearity. We call  $V^\Phi$  a *twisted version* of  $V$ , or  $V$  *twisted by*  $\Phi$ .

Now assume that  $G \leq \text{GL}(d, q)$  where  $G \cong \text{Sz}(q)$ ,  $d > 4$  and  $q = 2^{2m+1}$  for some  $m > 0$ . Then  $\text{Aut } \mathbb{F}_q = \langle \phi \rangle$ , where  $\phi$  is the Frobenius automorphism. Let  $W$  be the given module of  $G$  and let  $V$  be the natural module of the standard copy of  $\text{Sz}(q)$ , so that  $\dim W = d$  and  $\dim V = 4$ . If  $W$  is absolutely irreducible, then from [17] we know that

$$(2.1) \quad W \cong V^{\phi^{i_0}} \otimes V^{\phi^{i_1}} \otimes \dots \otimes V^{\phi^{i_{n-1}}}$$

for some integers  $0 \leq i_0 < i_1 < \dots < i_{n-1} \leq 2m$ . In fact, we may assume that  $i_0 = 0$  and clearly  $d = \dim W = (\dim V)^n = 4^n$ . We want to find the isomorphism between  $W$  and this tensor product, so we want to find a change of basis  $c \in \text{GL}(d, q)$  which exhibits  $W$  as the tensor product of (2.1). When  $W$  is such an explicit tensor product, it is straightforward to find the image of an element of  $W$  in one of the tensor factors.

More specifically, it is sufficient to find an isomorphism  $W \cong U_1 \otimes U_2$  where  $\dim U_1 = 4$  and  $\dim U_2 = 4^{n-1}$ . Then  $U_1$  is a twisted version of  $V$ , and hence the image of the corresponding representation is a conjugate of  $\text{Sz}(q)$ . The tensor decomposition algorithm of [13] can be used to find such an isomorphism of  $W$ . By [14] it is sufficient to find a *flat* in a projective geometry corresponding to the decomposition. This flat contains a *point*, which in this case is a subspace of  $W$  of the form  $\langle v \rangle \otimes U_2$  for some  $v \in U_1$ . If we can provide a flat to the algorithm of [13], then it will find a tensor decomposition in polynomial time.

Thus the essential problem is to find a flat of  $W$ . Since  $\dim U_1 = 4$ , by [14] the only possible flats are points or lines, where a line is a subspace of  $W$  of the form  $V' \otimes U_2$ , where  $V' \leq U_1$  and  $\dim V' = 2$ .

If  $\phi^{i_0}, \dots, \phi^{i_{n-1}}$  are the elements of  $\text{Gal}(\mathbb{F}_q, \mathbb{F}_s)$  where  $\mathbb{F}_s \leq \mathbb{F}_q$ , then  $G$  is conjugate in  $\text{GL}(d, q)$  to a subgroup of  $\text{GL}(d, s)$ . Therefore,  $W$  may be given to our algorithms as an  $\mathbb{F}_s G$ -module, but the recognition algorithm of [3] will determine  $q$  such that  $G \cong \text{Sz}(q)$ . In the case where  $W$  is not over  $\mathbb{F}_q$ , we can embed it canonically into an  $\mathbb{F}_q G$ -module, so henceforth assume that  $W$  is an  $\mathbb{F}_q G$ -module.

**2.1. Complexity.** We shall be concerned with the time complexity of the algorithms involved, where the basic operations are the field operations, and not the bit operations. All simple arithmetic with matrices can be done using  $O(d^3)$  field operations, and raising a matrix to the  $O(q)$  power can be done using  $O(d^3 \log q)$  field operations using the standard method of repeated squaring.

We will need to find an element of order  $q - 1$ . The order can be computed using the algorithm of [6]. To obtain the *precise* order, this algorithm requires a factorisation of  $q - 1$ , otherwise it might return a multiple of the correct order. However, it is sufficient for our purposes to learn the *pseudo-order* of the element, which is a multiple of its order, since it is sufficient to find a nontrivial element of order dividing  $q - 1$ . Hence we avoid the requirement to factorise  $q - 1$ . The algorithm of [6] can also be used to obtain the pseudo-order, and for this it has time complexity  $O(d^3 \log(q) \log \log(q^d))$  field operations.

**2.2. Random elements.** Our analysis assumes that we can construct uniformly distributed random elements of a group  $G$  defined by a generating set  $X$ . The polynomial time algorithm of [2] produces nearly uniformly distributed random

elements; an alternative polynomial time algorithm is the *product replacement* algorithm of [6]. We will assume that we have a random element oracle, which can produce a uniformly random element using  $O(\xi(d))$  field operations, where  $\xi : \mathbb{N} \rightarrow \mathbb{N}$ .

**2.3. Las Vegas algorithms.** Most of the algorithms we consider are probabilistic of the type known as *Las Vegas algorithms*. This type of algorithm is discussed in [18, Section 25.8], [16, Section 1.3] and [8, Section 3.2.1]. In short it is a probabilistic algorithm with an input parameter  $\varepsilon$  that either returns **failure**, with probability at most  $\varepsilon$ , or otherwise returns a correct result. The time complexity naturally depends on  $\varepsilon$ .

We present Las Vegas algorithms as probabilistic algorithms that either return a correct result, with probability bounded below by  $1/p(n)$  for some polynomial  $p(n)$  in the size  $n$  of the input, or otherwise return **failure**. By enclosing such an algorithm in a loop that iterates  $\lceil \log \varepsilon / \log(1 - 1/p(n)) \rceil$  times, we obtain an algorithm that returns **failure** with probability at most  $\varepsilon$ , and hence is a Las Vegas algorithm in the above sense. Clearly if the enclosed algorithm is polynomial time, the Las Vegas algorithm is polynomial time.

One can also enclose the algorithm in a loop that iterates until the algorithm returns a correct result, thus obtaining a probabilistic time complexity, and the expected number of iterations is then  $O(p(n))$ .

### 3. THE MAIN ALGORITHM

We now describe our main algorithm that finds a tensor decomposition of  $W$  when  $q$  is large. Using the notation from Section 2, it is sufficient to find a flat in  $W$ . For  $k = 0, \dots, n-1$ , let  $H_k \leq \text{GL}(4, q)$  be the image of the representation corresponding to  $V^{\phi^k}$ , and let  $\psi_k : G \rightarrow H_k$  be an isomorphism. Our goal is then to find  $\psi_k$  explicitly for some  $k$ .

We begin with an overview of the method. Our approach for finding a flat in  $W$  is to consider eigenspaces of an element of  $G$  of order dividing  $q-1$ . From [1, Lemma 2.5] we know that such elements are easy to find by random search.

Let  $g \in G$  where  $|g| = q-1$ , and let  $t = 2^{m+1}$ . From [10, Chapter 11] we know that for  $k = 0, \dots, n-1$ ,  $\psi_k(g)$  has four distinct eigenvalues  $\lambda_k^{\pm 1}$  and  $\lambda_k^{\pm(t+1)}$  for some  $\lambda_k \in \mathbb{F}_q^\times$ . Also, the eigenspaces of  $\psi_k(g)$  are of dimension 1. Our method for finding a flat in  $W$  is to construct a line as a suitable sum of eigenspaces of  $g$ .

Let  $E$  be the multiset of eigenvalues of  $g$ , so that  $|E| = d$  and every element of  $E$  has the form

$$(3.1) \quad \lambda_0^{j_0} \lambda_1^{j_1} \dots \lambda_{n-1}^{j_{n-1}}$$

where each  $\lambda_k \in \mathbb{F}_q^\times$  and each  $j_k \in \{\pm 1, \pm(t+1)\}$ . A set  $E'$  that satisfies

- $|E'| = n$
- $\lambda_k \in E'$  or  $\lambda_k^{-1} \in E'$  for each  $k = 0, \dots, n-1$

is called a set of *base values* for  $E$ . Clearly  $E$  is easily calculated from  $E'$ .

Moreover  $\lambda_k = \lambda_0^{2^k}$ , and since  $|g| = q-1$  we must have  $|\lambda_k| = q-1$  for  $k = 0, \dots, n-1$ . Hence all  $\lambda_k$  are distinct. First we try to find a set of base values for  $E$ .

Consider the multiset  $S$  of products  $ef^{-1}$  where  $e, f \in E$ . We see from (3.1) that  $\lambda_k^{\pm 2} \in S$  for each  $k = 0, \dots, n-1$  and also that it must occur with multiplicity at least  $4^{n-1}$ . The multiplicity might be greater than this, because of identities in  $\mathbb{F}_q$ , e.g.  $x^q = x$  for  $x \in \mathbb{F}_q$ , and in particular some eigenvalues of  $g$  might have multiplicity greater than 1.

Furthermore, it is easy to see that the values of  $\{\lambda_k^{\pm(2t+2)}, \lambda_k^{\pm t}, \lambda_k^{\pm(t+2)}\} \subseteq S$  occur with multiplicity at least  $4^{n-1}$ , and for each  $j \neq k$ , the values in  $\{\lambda_k^{\pm a} \lambda_j^{\pm b} \mid a, b \in \{t, t+2\}\} \subseteq S$  also occur with multiplicity at least  $4^{n-1}$ .

Let  $K \subseteq S$  be the set of values of multiplicity at least  $4^{n-1}$ . A necessary condition on  $x \in K$  to be a base value for  $E$  is that  $\{x, x^{2t+2}, x^t, x^{t+2}\} \subseteq K$  and that there exist  $n-1$  elements  $\mu_1, \dots, \mu_{n-1}$  of  $K$  and integers  $\alpha_1, \dots, \alpha_{n-1}$  such that

- $x^{2\alpha_i} = \mu_i$  and  $0 \leq \alpha_i \leq 2m$  for  $i = 1, \dots, n-1$ .
- $\{x^{\pm a} \mu_i^{\pm b} \mid a, b \in \{t, t+2\}\} \subseteq K$  for  $i = 1, \dots, n-1$ .

An algorithm to find a set of candidates for base values of  $E$  is given as Algorithm 1. If the algorithm returns a set of size  $n$ , then clearly that is a set of base values. Also note that the method described above for finding base values may still work if  $|g|$  is a proper divisor of  $q-1$ , and therefore the algorithm only requires  $|g| \mid q-1$ . The time complexity of our algorithm for finding a flat relies on the following conjecture about Algorithm 1.

**Conjecture 3.1.** *Let  $d = 4^n$  with  $n > 1$  be fixed. If  $q = 2^{2m+1}$  and  $m \geq n+1$ , then for every absolutely irreducible  $G \leq \text{GL}(d, q)$  with  $G \cong \text{Sz}(q)$  and every  $g \in G$  with  $|g| = q-1$ , Algorithm 1 returns a set  $N$  with  $|N| \leq 2n-1$ .*

Let  $S_i$  denote the sum of eigenspaces of  $g$  corresponding to the eigenvalues  $\lambda_0^{j_0} \cdots \lambda_i^{j_i} \cdots \lambda_{n-1}^{j_{n-1}}$  and  $\lambda_0^{j_0} \cdots \lambda_i^{-1} \cdots \lambda_{n-1}^{j_{n-1}}$ , where each  $j_k$  ranges over  $\{\pm 1, \pm(t+1)\}$ .

**Lemma 3.2.** *If  $\dim S_k = 2 \cdot 4^{n-1}$  for some  $0 \leq k \leq n-1$ , then  $S_k$  is a line in  $W$ .*

*Proof.* Clearly  $\psi_k(g)$  has eigenvalues  $\lambda_k^{\pm 1}$  with corresponding eigenspaces  $L_{\pm 1}$  of dimension 1. If  $V' = L_1 + L_{-1} \leq V^{\phi_{i_k}}$  then  $\dim V' = 2$ . Since  $\dim S_k = 2 \cdot 4^{n-1}$ , it follows from (2.1) that  $S_k \cong V' \otimes U$  where  $U \leq W$  and  $\dim U = 4^{n-1}$ . Thus  $S_k$  is a line in  $W$ .  $\square$

The correctness of the algorithm for finding a flat relies on the following conjecture.

**Conjecture 3.3.** *Let  $d = 4^n$  with  $n > 1$  be fixed. If  $q = 2^{2m+1}$  and  $m \geq n$ , then for every absolutely irreducible  $G \leq \text{GL}(d, q)$  with  $G \cong \text{Sz}(q)$  and every  $g \in G$  with  $|g| = q-1$ , we have  $\dim S_i = 2 \cdot 4^{n-1}$  for some  $0 \leq i \leq n-1$ .*

We comment on the validity of Conjectures 3.1 and 3.3 in Section 5. The algorithm for finding a flat is shown as Algorithm 2.

**Theorem 3.4.** *Assuming Conjectures 3.1 and 3.3 and given a random element oracle for  $G$ , if  $q = 2^{2m+1}$  and  $m > n$  then Algorithm 2 is a Las Vegas algorithm. The probability  $s$  of success satisfies*

$$(3.2) \quad s \geq \frac{\varphi(q-1)}{2(q-1)} > \frac{1}{12 \log \log(q)}.$$

*The algorithm has time complexity*

$$O(\xi(d) + d^3 \log(q) \log \log(q^d))$$

*field operations.*

*Proof.* By [1, Theorem 2.1], the proportion of elements of order  $q-1$  in  $G$  is  $\phi(q-1)/(2(q-1))$ , which gives a lower bound for the probability that line 3 succeeds. If  $|g| = q-1$  then Conjecture 3.1 asserts that line 6 will succeed, and Conjecture 3.3 asserts that lines 8 and 12 will succeed. If  $|g|$  is a proper divisor of  $q-1$  then these lines might still succeed, and the first inequality of (3.2) follows. The second inequality follows from [15, Section II.8].

---

**Algorithm 1: FindBaseValues**

---

**Data:** Multiset  $E$  of eigenvalues of  $g \in G \leq \text{GL}(d, q)$  where  $G \cong \text{Sz}(q)$  and  $|g| \mid q - 1$ .

**Result:**  $N \subseteq E$  that contains all base values of  $E$ .

```

1 begin
    /*E and S are a multisets */
2   S := {ef^{-1} | e, f \in E}
3   K := {x \in S | Multiplicity(x) \geq 4^{n-1}}
4   B := \emptyset
5   for x \in K do
6       if {x, x^{2t+2}, x^t, x^{t+2}} \subseteq K then
7           L := K
8           C := {x}
9           for i := 1 to n - 1 do
10              for z \in L do
11                  if x^{2^j} = z some 0 \leq j \leq 2m then
12                      if {x^{\pm a} z^{\pm b} | a, b \in \{t, t+2\}} \subseteq K then
13                          C := C \cup \{z\}
14                          L := L \setminus \{z\}
15                          break
16                      end
17                  end
18              end
19          end
20          if |C| = n then
21              B := B \cup \{x\}
22          end
23      end
24  end
25  N := \emptyset
26  for x \in B do
27      if x^{-1} \notin N then
28          N := N \cup \{x\}
29      end
30  end
31  return N
32 end
    
```

---

If line 13 is reached, then it follows from Lemma 3.2 that the algorithm returns a correct result and hence it is Las Vegas.

The element  $g$  is found using  $O(\xi(d))$  field operations, and by [6] we compute its pseudo-order using  $O(d^3 \log(q) \log \log(q^d))$  field operations. To find the eigenvalues, we calculate the characteristic polynomial of  $g$  using  $O(d^3)$  field operations, and by [18, Corollary 14.16] we find its roots using  $O(d(\log d)^2 \log \log(d) \log(dq))$  field operations. The set  $N$  is found by Algorithm 1 using  $O(d^2 n \log(q))$  field operations.

**Algorithm 2:** TensorDecomposeSz

---

**Data:** Generating set  $X$  for  $G \cong \text{Sz}(q)$  with natural module  $W$ , where  $\dim W = 4^n$ ,  $n > 1$  and  $W$  is absolutely irreducible and over  $\mathbb{F}_q$ .

**Result:** A line  $S$  in  $W$ .

```

1 begin
  /*Find random element  $g$  of pseudo-order  $q - 1$  */
2   $g := \text{Random}(G)$ 
3  if  $|g| \mid q - 1$  then
4     $E := \text{Eigenvalues}(g)$ 
5     $N := \text{FindBaseValues}(E)$ 
6    if  $n \leq |N| \leq 2n - 1$  then
7      foreach  $vals \in \{w \subseteq N \mid |w| = n\}$  do
8        if  $vals = \{\lambda_0, \dots, \lambda_{n-1}\}$  then
9          for  $i := 0$  to  $n - 1$  do
10              $E_i := \left\{ \lambda_0^{j_0} \dots \lambda_i \dots \lambda_{n-1}^{j_{n-1}} \mid j_k \in \{\pm 1, \pm(t+1)\} \right\} \cup$ 
                 $\left\{ \lambda_0^{j_0} \dots \lambda_i^{-1} \dots \lambda_{n-1}^{j_{n-1}} \mid j_k \in \{\pm 1, \pm(t+1)\} \right\}$ 
11              $S_i := \sum_{e \in E_i} \text{Eigenspace}(g, e)$ 
12             if  $\dim S_i = 2 \cdot 4^{n-1}$  then
13               return  $S_i$ 
14             else
15               return fail
16             end
17           end
18         end
19       end
20     end
21   end
22   return fail
23 end

```

---

Conjecture 3.1 asserts that the set considered at line 7 has size at most  $\binom{2n-1}{n} < 2^{2n-1} = d/2$ , so the rest of the algorithm uses  $O(d^2 n \log(q))$  field operations. Thus the theorem follows.  $\square$

**Corollary 3.5.** *Assuming Conjectures 3.1 and 3.3, if  $q = 2^{2m+1}$ ,  $d = 4^n$ ,  $n > 1$  and  $m \geq n$ , and given a random element oracle for subgroups of  $\text{GL}(d, q)$ , there exists a Las Vegas algorithm that, given  $X \subseteq \text{GL}(d, q)$  with  $\langle X \rangle \cong \text{Sz}(q)$ , finds an explicit isomorphism  $\Psi : \langle X \rangle \rightarrow \text{Sz}(q)$ . The algorithm has time complexity  $O(\xi(d) + d^3 \log(q) \log \log(q^d) + |X|)$  field operations.*

*Proof.* Let  $W$  be the given module of  $G = \langle X \rangle$ . The algorithm proceeds as follows:

- (1) Use Algorithm 2 to find a flat  $L \leq W$ .
- (2) Use [13] with  $L$  to get  $x \in \text{GL}(d, q)$  such the change of basis determined by  $x$  exhibits  $W$  as a tensor product  $V \otimes U$  with  $\dim V = 4$ . Let  $G_V$  and  $G_U$  be the images of the corresponding representations.
- (3) Define  $\rho_V : G_V \otimes G_V \rightarrow G_U$  as  $g_v \otimes g_u \mapsto g_v$  and let  $Y = \{\rho_V(g^x) \mid g \in X\}$ . Then  $\langle Y \rangle$  is conjugate to  $\text{Sz}(q)$ .
- (4) Use [1, Theorem 7.5] to get  $y \in \text{GL}(4, q)$  such that  $\langle Y \rangle^y = \text{Sz}(q)$ .

- (5) An explicit isomorphism  $\Psi : G \rightarrow \text{Sz}(q)$  is given by  $g \mapsto \xi_V(g^x)^y$ .

The map  $\rho_V$  is straightforward to compute, since given  $g \in \text{GL}(d, q)$  it only involves dividing  $g$  into submatrices of degree  $4^{n-1}$ , checking that they are scalar multiples of each other and returning the 4 by 4 matrix consisting of these scalars. Hence by Theorem 3.4, [13] and [1, Theorem 7.5], the algorithm is Las Vegas.

By Theorem 3.4, finding  $L$  uses  $O(d^3 \log(q) \log \log(q^d))$  field operations. From [13], finding  $x$  uses  $O(d^3 \log(q))$  field operations when a flat  $L$  is given. From [1, Theorem 7.5], finding  $y$  uses  $O(\log(q) \log \log(q) + |Y|)$  field operations, given a random element oracle for  $\langle Y \rangle$  that finds a random element using  $O(1)$  field operations. In this case we can construct random elements for  $\langle Y \rangle$  using the random element oracle for  $\langle X \rangle$ , but we will not find them in  $O(1)$  field operations, but in  $O(\xi(d))$  field operations. However, this is not the dominating term in the complexity, and the theorem follows.  $\square$

#### 4. SMALL FIELD APPROACH

Even if Conjectures 3.1 and 3.3 are true, they only guarantee the feasibility of Algorithm 2 when  $q$  is large. In other cases we need another algorithm, and then  $q$  is polynomial in  $d$ , so we are content with an algorithm that has time complexity polynomial in  $q$ . We use the notation of Section 2. The goal is now to find the tensor decomposition, not just compute an explicit isomorphism to a standard copy, and the approach is not to use the tensor decomposition algorithm of [13], since in this case we have no efficient method of finding a flat. Instead we find an explicit isomorphism from  $G$  to  $\text{Sz}(q)$  using permutation group techniques, then enumerate all tensor products of the form (2.1) and for each one we determine if it is isomorphic to  $W$ .

More specifically, the algorithm is given as Algorithm 3. It finds a permutation representation of  $G \cong \text{Sz}(q)$ , which is done using the following result.

**Theorem 4.1.** *Given a random element oracle for subgroups of  $\text{GL}(d, q)$ , there exists a Las Vegas algorithm that, given  $X \subseteq \text{GL}(d, q)$  such that  $q = 2^{2m+1}$  with  $m > 0$  and  $\langle X \rangle \cong \text{Sz}(q)$ , finds an explicit injective homomorphism  $\Pi : \langle X \rangle \rightarrow \text{Sym}(O)$  where  $|O| = q^2 + 1$ . The algorithm has time complexity  $O(\xi(d) + |X| d^2(d + q^2 + 1))$  field operations.*

*Proof.* By [1, Theorem 2.1],  $\text{Sz}(q)$  acts doubly transitively on a set of size  $q^2 + 1$ . Hence  $G = \langle X \rangle$  also acts doubly transitively on  $O$ , where  $|O| = q^2 + 1$ , and we can find the permutation representation of  $G$  if we can find a point  $P \in O$ . The set  $O$  is a set of projective points of  $\mathbb{F}_q^d$ , and the algorithm proceeds as follows.

- (1) Choose random  $g \in G$ . Determine if  $|g| \mid q - 1$  and return with failure if not.
- (2) Choose random  $x \in G$ . Let  $h = g^x$ . Determine if  $[g, h]^4 = 1$  and return with failure if not.
- (3) Let  $M$  be the natural module of  $\langle g, h \rangle$ . Find a composition series for  $M$  and let  $P \subseteq M$  be the submodule of dimension 1 in the series.
- (4) Find the orbit  $O = P^G$  and compute permutation group  $S \leq \text{Sym}(O)$  of  $G$  on  $O$ , together with an explicit isomorphism  $\Pi : G \rightarrow S$ .

By [1, Theorem 2.1], elements in  $G$  of order dividing  $q - 1$  fix two points of  $O$ , and hence  $\langle g, h \rangle \leq G_P$  for some  $P \in O$  if and only if  $g$  and  $h$  have a common fixed point. From [10, Chapter 11] we know that  $[G_P, G_P]$  has exponent 4, and all elements of even order are in the derived group of a stabiliser of some point. Hence if  $[g, h]^4 = 1$  then  $\langle g, h \rangle$  fixes a point.

From [10, Chapter 11], all composition factors of  $M$  have dimension 1, so a composition series of  $M$  must contain a submodule  $P$  of dimension 1. This submodule

is a fixed point for  $\langle g, h \rangle$  and its orbit must have size  $q^2 + 1$ , since [1, Theorem 2.1] implies that  $|G| = q^2(q^2 + 1)(q - 1)$  and  $|G_P| = q^2(q - 1)$ . It follows that  $P \in O$ .

To find the orbit  $O = P^G$  we can compute a Schreier tree on the generators in  $X$  with  $P$  as root, using  $O(|X||O|d^2)$  field operations. Then  $\Pi(g)$  can be computed for any  $g \in \langle X \rangle$  using  $O(|O|d^2)$  field operations, by computing the permutation on  $O$  induced by  $g$ . Hence  $\Pi$  is explicit, and  $S$  is found by computing the image of each element of  $X$ . Therefore the algorithm is correct and it is clearly Las Vegas.

Finding  $g$  and  $h$  uses  $O(\xi(d))$  field operations. Finding a composition series for  $M$  using [9] and [11] uses  $O(|X|d^3)$  field operations. Thus the theorem follows.  $\square$

---

**Algorithm 3: SmallFieldTensorDecompose**

---

**Data:** Generating set  $X$  for  $G \cong \text{Sz}(q)$  with natural module  $W$ , where  $\dim W = 4^n$ ,  $n > 1$  and  $W$  is absolutely irreducible and over  $\mathbb{F}_q$ .

**Result:** A change of basis matrix  $c$  which exhibits  $W$  as (2.1).

```

1 begin
  /*Find permutation representations, i.e. permutation groups and
   corresponding isomorphisms */
2  (flag,  $\alpha, P_G$ ) := SuzukiPermRepresentation( $G$ )
3  if flag = true then
4    (flag,  $\beta, P_{\text{Sz}}$ ) := SuzukiPermRepresentation( $\text{Sz}(q)$ )
5    if flag = true then
6      /*Find isomorphism between permutation groups */
7       $\theta := \text{PermGroupIsomorphism}(P_G, P_{\text{Sz}})$ 
8      /*Get iso from  $G$  to  $\text{Sz}(q)$  */
9       $\gamma := \beta^{-1} \circ \theta \circ \alpha$ 
10      $H := \gamma(G)$ 
11     twists :=  $\{1, \dots, \log_2 q\}^n$ 
12     /*Let  $V$  be the natural module for  $H$  */
13     foreach  $(i_1, \dots, i_n) \in \text{twists}$  do
14        $U := V^{\phi^{i_1}} \otimes \dots \otimes V^{\phi^{i_n}}$ 
15       /*Find isomorphism between modules */
16       (flag,  $c$ ) := ModuleIsomorphism( $U, W$ )
17       if flag = true then
18         return  $c$ 
19       end
20     end
21   end
22 end

```

---

**Corollary 4.2.** *Algorithm 3 is a Las Vegas algorithm with time complexity  $O(\xi(d) + |X|d^2(dm^n + q^2 + 1))$  field operations.*

*Proof.* The permutation representations  $\alpha$  and  $\beta$  can be found using Theorem 4.1, and  $\theta$  can be found using [5]. Testing if modules are isomorphic can be done using [9] and [11].

TABLE 1. Benchmark of tensor decomposition

$m$	$n = 2$ [s]	$n = 3$ [s]	$n = 4$ [s]
1	0.59	4.8	–
2	0.39	41	560
3	0.56	20	
4	0.70	20	
5	0.92	15	
6	1.3	23	
7	1.9	26	
8	2.7		
9	3.0		
10	8.4		

If the algorithm returns an element  $c$  then the change of basis determined by  $c$  exhibits  $W$  as a tensor product, so by Theorem 4.1 the algorithm is Las Vegas.

The set `twists` has size  $(\log q)^n$  and  $q = 2^{2m+1}$ . Module isomorphism testing uses  $O(|X|d^3)$  field operations. Hence by [5] and Theorem 4.1 the time complexity of the algorithm is  $O(\xi(d) + |X|d^2(d + q^2 + 1) + |X|d^3m^n)$  field operations, and the result follows.  $\square$

### 5. IMPLEMENTATION AND PERFORMANCE

Implementations of the algorithms are available in MAGMA. The implementations use the existing MAGMA implementations of the algorithms described in [1], [3], [5], [6], [7], [9] and [13].

We have benchmarked the tensor decomposition for various field sizes, and degrees 16, 64 and 256, as shown in Table 1. When possible, the algorithm described in Corollary 3.5 was used, but in the cases given by Conjecture 3.3 where this algorithm fails, Algorithm 3 was used.

The benchmark was carried out on a PC, with an Intel Xeon CPU running at 2.8 GHz and with 1 GB of RAM, using MAGMA V2.12-16 EM64T. The largest values of  $m$  used in the table was the largest possible, in the sense that larger field sizes required too much memory. The largest dimension is 256 for the same reason. The largest possible  $d$  for a given  $q$  is  $q^2$ , so  $d = 256$  is not possible when  $q = 8$ , hence the dash in the table.

The values in Table 1 were computed as follows: for each dimension  $d = 4^n$  and field size  $q = 2^{2m+1}$ , each module of the form (2.1) was created and 10 random conjugates of it was tensor decomposed, so that  $10 \cdot \binom{2m+1}{n}$  tensor decompositions were carried out for each  $m$  and  $n$ . The average time for each tensor decomposition is listed in the table.

It was necessary to initialise the product replacement algorithm for each tensor decomposition, which in our case involved 20 calls to the MAGMA implementation of [7]. The values in the table include the time for these calls.

During the benchmark Conjectures 3.1 and 3.3 always held. In fact these conjectures were motivated partly by the experimental data from the benchmark. Algorithm 2 failed to find a line precisely in the cases  $(m, n) \in \{(1, 2), (1, 3), (2, 3), (2, 4)\}$  and Algorithm 1 failed to find a small set of base values precisely in the cases  $(m, n) \in \{(1, 2), (1, 3), (2, 2), (2, 3), (2, 4), (3, 3)\}$ . The total number of tensor decompositions during the benchmark, and hence the number of times the conjectures

were checked, was

$$10 \left[ \sum_{m=1}^{10} \binom{2m+1}{2} + \sum_{m=1}^7 \binom{2m+1}{3} + \sum_{m=2}^2 \binom{2m+1}{4} \right] = 18660.$$

The fact that the conjectures always held in these cases gives strong evidence to support them.

#### REFERENCES

1. H. Bäärnhjelm, *Recognising the Suzuki groups in their natural representations*, J. Algebra (2006), to appear.
2. L. Babai, *Local expansion of vertex-transitive graphs and random generation in groups*, Proc. 23rd ACM Symp. Theory of Computing (Los Angeles), Association for Computing Machinery, 1991, pp. 164–174.
3. L. Babai, W. M. Kantor, P. P. Pálffy, and Á. Seress, *Black-box recognition of finite simple groups of Lie type by statistics of element orders*, J. Group Theory **5** (2002), 383–401.
4. W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system I: The user language*, J. Symbolic Comput. **24** (1997), 235–265.
5. J. J. Cannon and D. F. Holt, *Automorphism group computation and isomorphism testing in finite groups*, J. Symbolic Comput. **35** (2003), no. 3, 241–267.
6. F. Celler and C. R. Leedham-Green, *Calculating the order of an invertible matrix*, Groups and Computation II (Larry Finkelstein and William M. Kantor, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 28, American Mathematical Society, 1997, pp. 55–60.
7. F. Celler, C. R. Leedham-Green, S. H. Murray, A. C. Niemeyer, and E. A. O’Brien, *Generating random elements of a finite group*, Comm. Algebra (1995), no. 23, 4931–4948.
8. D. F. Holt, B. Eick, and E. A. O’Brien, *Handbook of Computational Group Theory*, Chapman & Hall/CRC, January 2005.
9. D. F. Holt and S. Rees, *Testing modules for irreducibility*, J. Austral. Math. Soc. Series A **57** (1994), 1–16.
10. B. Huppert and N. Blackburn, *Finite groups III*, Grundlehren Math. Wiss., vol. 243, Springer-Verlag, Berlin, Heidelberg, New York, 1982.
11. G. Ivanyos and K. Lux, *Treating the exceptional cases of the MeatAxe*, Experiment. Math. **9** (2000), 373–381.
12. C. R. Leedham-Green, *The computational matrix group project*, Groups and Computation III, Ohio State Univ. Math. Res. Inst. Publ., vol. 8, de Gruyter, 2001, pp. 113–121.
13. C. R. Leedham-Green and E. A. O’Brien, *Recognising tensor products of matrix groups*, Internat. J. Algebra Comput. **7** (1997), 541–559.
14. ———, *Tensor products are projective geometries*, J. Algebra **189** (1997), 514–528.
15. D. S. Mitrinovic, J. Sándor, and B. Crstici, *Handbook of number theory, mathematics and its applications*, vol. 351, Kluwer Academic Publishers, 1996.
16. Á. Seress, *Permutation group algorithms*, Cambridge Tracts in Mathematics, vol. 152, Cambridge University Press, 2003.
17. R. Steinberg, *Representations of algebraic groups*, Nagoya Math. J. **22** (1963), 33–56.
18. J. von zur Gathen and J. Gerhard, *Modern computer algebra*, 2nd ed., Cambridge University Press, Cambridge, 2003.

SCHOOL OF MATHEMATICAL SCIENCES, QUEEN MARY, UNIVERSITY OF LONDON, MILE END ROAD, LONDON E1 4NS, UNITED KINGDOM

URL: <http://www.maths.qmul.ac.uk/~hb/>

E-mail address: [h.baarnhielm@qmul.ac.uk](mailto:h.baarnhielm@qmul.ac.uk)