

HANDBOOK FOR MAGMA SUZUKI PACKAGE

HENRIK BÄÄRNHIELM

The MAGMA Suzuki package primarily provides functionality for constructive recognition and constructive membership testing of the Suzuki groups $Sz(q)$, with $q = 2^{2m+1}$ for some $m > 0$. This family of finite simple groups is already available in MAGMA via the `Sz` intrinsic. It also provides routines for finding and conjugating Sylow and maximal subgroups of $Sz(q)$.

The main intrinsics of the package are `RecogniseSz(GrpMat G)` which performs constructive recognition of $G \cong Sz(q)$, `SzElementToWord(GrpMat G, GrpMatElt g)` which returns a `GrpSLPElt` for g in the generators of G , and `SuzukiRecognition(GrpMat G)` which is a non-constructive test for isomorphism between G and $Sz(q)$.

Sylow p -subgroups can be found using the `SuzukiSylow(GrpMat G, RngIntElt p)` intrinsic. To find conjugating elements between Sylow p -subgroups, an intrinsic `SuzukiSylowConjugacy` is provided. A list of representatives of the conjugacy classes of maximal subgroups can be found using the `SuzukiMaximalSubgroups(GrpMat G)` intrinsic, and the intrinsic `SuzukiMaximalSubgroupsConjugacy` can be used to find conjugating elements between maximal subgroups.

Presentations for $Sz(q)$ can be handled via the intrinsics `SzPresentation` and `SatisfiesSzPresentation`.

There are a few verbose flags used in the package.

- `SuzukiGeneral`, for the general routines.
- `SuzukiStandard`, for the routines related to the standard copy.
- `SuzukiConjugate`, for the routines related to conjugation.
- `SuzukiTensor`, for the routines related to tensor decomposition.
- `SuzukiMembership`, for the routines related to membership testing.
- `SuzukiCrossChar`, for the routines related to cross-characteristic representations.
- `SuzukiSylow`, for the routines related to Sylow subgroups.
- `SuzukiMaximals`, for the routines related to maximal subgroups.
- `SuzukiTrick`, for the routines related to the double coset trick.
- `SuzukiNewTrick`, for the routines related to the stabiliser trick.

All the flags can be set to values up to 10, with higher values resulting in more output.

1. INTRINSICS

`RecogniseSz(G : parameters) : GrpMat -> BoolElt, Map, Map, Map, Map`

Parameters:

`Verify, BOOLELT, Default : true`
`FieldSize, RNGINTELT`

G is absolutely irreducible and defined over minimal field. Constructively recognise G as a Suzuki group. If G is isomorphic to $Sz(q)$ where q is the size of the defining field of G , then return:

Date: 2007-04-21.

- (1) Isomorphism from G to $Sz(q)$.
- (2) Isomorphism from $Sz(q)$ to G .
- (3) Map from G to the word group of G .
- (4) Map from the word group of G to G .

The isomorphisms are composed of maps that are defined by rules, so **Function** can be used on each component, hence avoiding unnecessary built-in membership testing.

The word group is the **GrpSLP** which is the parent of the elements returned by **SzElementToWord**. In general this is not the same as **WordGroup(G)**, but is created from it using **AddRedundantGenerators**.

If **Verify** is true, then it is checked that G is isomorphic to $Sz(q)$, using **SuzukiRecognition**, otherwise this is not checked. In that case, **FieldSize** must be set to the correct value of q .

Constructive recognition of $2.Sz(8)$ is also handled.

The algorithms for constructive recognition are those of [2] and [1].

SzElementToWord(G, g) : GrpMat, GrpMatElt -> BoolElt, GrpSLPElt

If G has been constructively recognised as a Suzuki group, and if g is an element of G , then return **true** a **GrpSLPElt** from the word group of G which evaluates to g , else return **false**.

This facilitates membership testing in G .

SzPresentation(q) : RngIntElt -> GrpFP, HomGrp

If $q = 2^{2m+1}$ for some $m > 0$, return a short presentation of $Sz(q)$ on the MAGMA standard generators, *i.e.* the generators returned by the **Sz** intrinsic.

Note that it is an open problem whether or not this is a presentation of $Sz(q)$ for every such q .

SatisfiesSzPresentation(G) : GrpMat -> BoolElt

G is constructively recognised as $Sz(q)$ for some q . Verify that it satisfies a presentation for this group.

SuzukiRecognition(G) : GrpMat -> BoolElt, RngIntElt

Determine (non-constructively) if G is isomorphic to $Sz(q)$. The corresponding q is also returned.

If G is over a field of odd characteristic or has degree greater than 4, the Monte Carlo algorithm of **IdentifyLieType** is used. If G has degree 4 and is over a field of characteristic 2, then a fast Las Vegas algorithm is used, described in [2].

SuzukiIrreducibleRepresentation(F, twists : parameters) : FldFin, SeqEnum[RngIntElt] -> GrpMat

Parameters:

CheckInput, **BOOLELT**, *Default* : *true*

F must have size $q = 2^{2m+1}$ for some $m > 0$, and *twists* should be a sequence of n distinct integers in the range $[0 \dots 2m]$.

Return an absolutely irreducible representation of $Sz(q)$ of dimension 4^n , a tensor product of twisted powers of the copy returned by the **Sz** intrinsic, where the twists are given by the input sequence.

If **CheckInput** is true, then it is verified that F and *twists* satisfy the above requirements. Otherwise this is not checked.

SuzukiSylow(G, p) : GrpMat, RngIntElt -> GrpMat, SeqEnum

If G has been constructively recognised as a Suzuki group, and if p is a prime number, return a random Sylow p -subgroup S of G .

Also returns a list of GrpSLPElt from the word group of G , of the generators of S . If p does not divide $|G|$, then the trivial subgroup is returned.

SuzukiSylowConjugacy(G, R, S, p) : GrpMat, GrpMat, GrpMat, RngIntElt -> GrpMatElt, GrpSLPElt

If G has been constructively recognised as a Suzuki group, if p is a prime number and if R and S are Sylow p -subgroups of G , then return an element g of G that conjugates R to S . A GrpSLPElt from the word group of G , that evaluates to g , is also returned.

SuzukiMaximalSubgroups(G) : GrpMat -> SeqEnum, SeqEnum

If G has been constructively recognised as a Suzuki group, find a list of representatives of the maximal subgroups of G .

Also returns lists of GrpSLPElt of the generators of the subgroups, from the word group of G .

SuzukiMaximalSubgroupsConjugacy(G, R, S) : GrpMat, GrpMat, GrpMat -> GrpMatElt, GrpSLPElt

If G has been constructively recognised as a Suzuki group and if R and S are conjugate maximal subgroups of G , then return an element g of G that conjugates R to S . A GrpSLPElt from the word group of G , that evaluates to g , is also returned.

2. EXAMPLES

Example showing the basic features.

```
> // Let's try a conjugate of the the standard copy
> G := Sz(32);
> G ^:= Random(Generic(G));
> // perform non-constructive recognition
> flag, q := SuzukiRecognition(G);
> print flag, q eq 32;
true true
> // perform constructive recognition
> flag, iso, inv, g2slp, slp2g := RecognizeSz(G);
> print flag;
true
> // the explicit isomorphisms are defined by rules
> print iso, inv;
Mapping from: GrpMat: G to MatrixGroup(4, GF(2^5)) of order 32537600 = 2^10 *
5^2 * 31 * 41 given by a rule [no inverse]
Mapping from: MatrixGroup(4, GF(2^5)) of order 32537600 = 2^10 * 5^2 * 31 * 41
to GrpMat: G given by a rule [no inverse]
> // so we can use Function to avoid Magma built-in membership testing
> // we might not obtain the shortest possible SLP
> w := Function(g2slp)(G.1);
> print #w;
300
> // and the algorithm is probabilistic, so different executions will most
> // likely give different results
```

```

> ww := Function(g2slp)(G.1);
> print w eq ww;
false
> // the resulting SLPs are from another word group
> W := WordGroup(G);
> print NumberOfGenerators(Parent(w)), NumberOfGenerators(W);
7 3
> // but can be coerced into W
> flag, ww := IsCoercible(W, w);
> print flag;
true
> // so there are two ways to get the element back
> print slp2g(w) eq Evaluate(ww, UserGenerators(G));
true
> // an alternative is this intrinsic, which is better if the elements are not
> // known to lie in the group
> flag, ww := SzElementToWord(G, G.1);
> print flag, slp2g(w) eq slp2g(ww);
true true
> // let's try something just outside the group
> H := Sp(4, 32);
> flag, ww := SzElementToWord(G, H.1);
> print flag;
false
> // in this case we will not get an SLP
> ww := Function(g2slp)(H.1);
> print ww;
false
> // we do indeed have a Suzuki group
> print SatisfiesSzPresentation(G);
true
> // finally, let's try 2.Sz(8)
> A := ATLASGroup("2Sz8");
> reps := MatRepKeys(A);
> G := MatrixGroup(reps[3]);
> print Degree(G), CoefficientRing(G);
40 Finite field of size 7
> // we can handle constructive recognition, although it takes some time
> time flag, iso, inv, g2slp, slp2g := RecognizeSz(G);
Time: 415.660
> print flag;
true
> // and constructive membership testing
> R := RandomProcess(G);
> g := Random(R);
> w := Function(g2slp)(g);
> // in this case the elements is determined up to a scalar
> print IsScalar(slp2g(w) * g^(-1));
true

```

Example showing a case in large dimension.

```

> // let's try a 64-dimensional Suzuki group
> F := GF(2, 9);

```

```

> twists := [0, 3, 6];
> G := SuzukiIrreducibleRepresentation(F, twists);
> print Degree(G), IsAbsolutelyIrreducible(G);
64 true
> // conjugate it to hide the tensor product
> G ^:= Random(Generic(G));
> // and write it over a smaller field to make things difficult
> flag, GG := IsOverSmallerField(G);
> print flag, CoefficientRing(GG);
true Finite field of size 2^3
> // non-constructive recognition is harder in this case
> // and will give us the defining field size
> time print SuzukiRecognition(GG);
true 512
Time: 1.030
> // constructive recognition will decompose the tensor product
> time flag, iso, inv, g2slp, slp2g := RecogniseSz(GG);
Time: 2.170
> print iso;
Mapping from: GrpMat: GG to MatrixGroup(4, GF(2^9)) of order 2^18 * 5 * 7 * 13 *
37 * 73 * 109 given by a rule [no inverse]
> // constructive membership is again easy
> R := RandomProcess(GG);
> g := Random(R);
> time w := Function(g2slp)(g);
Time: 0.010
> // but SLP evaluation is harder in large dimensions
> time print slp2g(w) eq g;
true
Time: 0.040
> // we still have a Suzuki group
> time print SatisfiesSzPresentation(GG);
true
Time: 1.750

```

Example showing a case in cross characteristic.

```

> // Let's try a Suzuki group in cross characteristic
> G := Sz(8);
> _, P := SuzukiPermutationRepresentation(G);
> // for example over GF(9)
> M := PermutationModule(P, GF(3, 2));
> factors := CompositionFactors(M);
> H := ActionGroup(factors[2]);
> // we actually end up with a group over GF(3)
> print IsAbsolutelyIrreducible(H);
true
> flag, G := IsOverSmallerField(H);
> print Degree(G), CoefficientRing(G);
64 Finite field of size 3
> // constructive recognition is easy in this case
> time flag, iso, inv, g2slp, slp2g := RecogniseSz(G);
Time: 0.650
> print iso;

```

Mapping from: GrpMat: G to MatrixGroup(4, GF(2³)) of order 29120 = 2⁶ * 5 * 7 * 13 given by a rule [no inverse]

```
> // as well as constructive membership
> R := RandomProcess(G);
> g := Random(R);
> time w := Function(g2slp)(g);
Time: 0.000
> time print slp2g(w) eq g;
true
Time: 0.020
> // we still have a Suzuki group
> time print SatisfiesSzPresentation(G);
true
Time: 0.100
```

Example showing a larger field case.

```
> // let's try a larger field
> q := 2121;
> G := Sz(q);
> // let's try a conjugate
> G~ := Random(Generic(G));
> // also move to another generating set
> G := DerivedGroupMonteCarlo(G);
> print NumberOfGenerators(G);
19
> // non-constructive recognition is now a bit harder
> time SuzukiRecognition(G);
true 2658455991569831745807614120560689152
Time: 0.090
> // and is your machine up for this?
> time flag, iso, inv, g2slp, slp2g := RecogniseSz(G);
Time: 291.160
> // each call to constructive membership is then easy
> R := RandomProcess(G);
> g := Random(R);
> time w := Function(g2slp)(g);
Time: 0.060
> // evaluating SLPs always takes some time
> time print slp2g(w) eq g;
true
Time: 1.470
```

Example about Sylow subgroups.

```
> // let's try a small field
> q := 29;
> G := Sz(q);
> // let's try a conjugate
> G~ := Random(Generic(G));
> // also move to another generating set
> G := DerivedGroupMonteCarlo(G);
> print NumberOfGenerators(G);
19
> // first we must recognise the group
> time flag, iso, inv, g2slp, slp2g := RecogniseSz(G);
```

```

Time: 0.100
> // try creating some Sylow subgroups
> p := Random([x[1] : x in Factorization(q - 1)]);
> print p;
73
> time R := SuzukiSylow(G, p);
Time: 0.020
> time S := SuzukiSylow(G, p);
Time: 0.000
> // that was easy, as is conjugating them
> time g, slp := SuzukiSylowConjugacy(G, R, S, p);
Time: 0.000
> time print R^g eq S;
true
Time: 0.000
> // in this case we also automatically get an SLP of the conjugating element
> print slp2g(slp) eq g;
true
> // those Sylow subgroups are cyclic, and we know the order
> print #R, NumberOfGenerators(R);
73 1
> // creating Sylow 2-subgroups is harder, since there are lots of generators
> time R := SuzukiSylow(G, 2);
Time: 0.040
> time S := SuzukiSylow(G, 2);
Time: 0.050
> print NumberOfGenerators(R), #R;
9 262144
> // but conjugation is easy
> time g, slp := SuzukiSylowConjugacy(G, R, S, 2);
Time: 0.000
> // verifying the conjugating element can be hard
> time print R^g eq S;
true
Time: 201.470

```

Example about maximal subgroups.

```

> // let's try a small field
> m := 4;
> F := GF(2, 2 * m + 1);
> q := #F;
> print q;
512
> G := Sz(F);
> // let's try a conjugate
> G^ := Random(Generic(G));
> // also move to another generating set
> G := DerivedGroupMonteCarlo(G);
> print NumberOfGenerators(G);
19
> // first we must recognise the group
> time flag, iso, inv, g2slp, slp2g := RecogniseSz(G);
Time: 0.140

```

```

> // try finding the maximal subgroups
> time l, slps := SuzukiMaximalSubgroups(G);
Time: 0.070
> // should have a parabolic, a torus normaliser, two Frobenius groups
> // and one smaller Sz
> print #l;
5
> // we can conjugate all maximals around
> r := Random(G'RandomProcess);
> for H in l do
for>   time g, slp := SuzukiMaximalSubgroupsConjugacy(G, H, H^r);
for>   time print H^g eq H^r;
for> end for;
Time: 0.010
true
Time: 193.440
Time: 0.000
true
Time: 0.030
Time: 0.020
true
Time: 0.040
Time: 0.020
true
Time: 0.020
Time: 0.140
true
Time: 0.040

```

REFERENCES

1. H. Bäärnhielm, *Tensor decomposition of the Suzuki groups*, (2005), submitted.
2. ———, *Recognising the Suzuki groups in their natural representations*, J. Algebra **300** (2006), no. 1, 171–198.

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