GAP 4 Package RAMEGA

RAndom MEthods in Group Algebras

1.0.0

2019

Zsolt Adam Balogh

Vasyl Laver

Zsolt Adam Balogh
Email: baloghzsa@gmail.com
Address: Department of Mathematical Sciences
UAEU
Al Ain, United Arab Emirates

Vasyl Laver
Email: vasyl.laver@uzhnu.edu.ua
Address: Department of Mathematical Sciences
UAEU
Al Ain, United Arab Emirates

Copyright

Contents

Chapter 1

Introduction

RAMEGA stands for “RAndom MEthods in Group ALgebras". This package is dedicated to realization of some standard functions for group rings by the means of the random methods. This approach allows to compute the needed function in a reasonable time and with minimal memory use even for the large group rings [BR00].

1.1 Overview over this manual

Chapter 2 describes the functions for group algebras which were obtained using the random methods.

1.2 Installation

To get the newest version of this GAP 4 package download the archive file `thelma-x.x.tar.gz` and unpack it in a directory called “`pkg`”, preferably (but not necessarily) in the “`pkg`” subdirectory of your GAP 4 installation. It creates a subdirectory called “`RAMEGA`”.

As RAMEGA has no additional C libraries, there is no need in any additional installation steps.

1.3 Feedback

For bug reports, feature requests and suggestions, please use the github issue tracker/

Chapter 2

Random Methods

2.1 Basic Operations

2.1.1 BasicGroup

▷ BasicGroup(*KG*) (function)

For the group ring KG the function BasicGroup returns the basic group of KG as a subgroup of the normalized group of units.

Example

```
gap> G:=CyclicGroup(IsFpGroup,4);
<fp group of size 4 on the generators [ a ]>
gap> Elements(G);
[ <identity ...>, a, a^2, a^3 ]
gap> KG:=GroupRing(GF(2),G);
<algebra-with-one over GF(2), with 1 generators>
gap> BasicGroup(KG);
<group with 4 generators>
gap> Elements(last);
[ (Z(2)^0)*<identity ...>, (Z(2)^0)*a, (Z(2)^0)*a^2, (Z(2)^0)*a^3 ]
```

2.1.2 IsLienEngel

▷ IsLienEngel(*KG*) (function)

For the group ring KG the function IsLienEngel returns true if KG is Lie-n Engel and false otherwise.

Example

```
gap> G:=CyclicGroup(4);
<pc group of size 4 with 2 generators>
gap> KG:=GroupRing(GF(2),G);
<algebra-with-one over GF(2), with 2 generators>
gap> IsLienEngel(KG);
false
```

```
gap>
gap>
gap> G:=DihedralGroup(16);
<pc group of size 16 with 4 generators>
gap> KG:=GroupRing(GF(2),G);
<algebra-with-one over GF(2), with 4 generators>
gap> IsLienEngel(KG);
```

2.2 Random Methods for Obtaining Elements With Desired Properties

2.2.1 GetRandomUnit

▷ `GetRandomUnit(`*KG*`)` (function)

For the group ring `KG` the function `GetRandomUnit` returns an unit (i.e. an invertible element) in a random way.

Example
```
gap> G:=CyclicGroup(4);
<pc group of size 4 with 2 generators>
gap> KG:=GroupRing(GF(7),G);
<algebra-with-one over GF(7), with 2 generators>
gap> u:=GetRandomUnit(KG);;
gap> Augmentation(u);
Z(7)^4
gap> u*u^-1;
(Z(7)^0)*<identity> of ...
```

2.2.2 GetRandomNormalizedUnit

▷ `GetRandomNormalizedUnit(`*KG*`)` (function)

For the group ring `KG` the function `GetRandomNormalizedUnit` returns a normalized unit (i.e. an invertible element with augmentation 1) in a random way.

Example
```
gap> G:=DihedralGroup(IsFpGroup,16);;
gap> KG:=GroupRing(GF(2),G);;
gap> u:=GetRandomNormalizedUnit(KG);;
gap> Augmentation(u);
Z(2)^0
gap> u*u^-1;
(Z(2)^0)*<identity ...>
```

2.2.3 GetRandomNormalizedUnitaryUnit

▷ `GetRandomNormalizedUnitaryUnit(`*KG*`)` (function)

For the group ring `KG` the function `GetRandomNormalizedUnitaryUnit` returns a normalized unitary unit (i.e. such an invertible element with augmentation 1, that $u \cdot u^* = One(KG)$) in a random way. Also, there exists a two-parametrical version of this method, where the second parameter σ is an arbitrary involution.

Example

```
gap> G:=CyclicGroup(4);;
gap> KG:=GroupRing(GF(2),G);;
gap> u:=GetRandomNormalizedUnitaryUnit(KG);;
gap> u*Involution(u);
(Z(2)^0)*<identity> of ...
gap> Augmentation(u);
Z(2)^0
```

2.2.4 GetRandomCentralNormalizedUnit

▷ `GetRandomCentralNormalizedUnit(`*KG*`)` (function)

For the group ring `KG` the function `GetRandomCentralNormalizedUnit` returns a central normalized unit (i.e. such an invertible element with augmentation 1, that $u \cdot x = x \cdot u$, $\forall x \in KG$) in a random way.

Example

```
gap> G:=CyclicGroup(IsFpGroup,4);;
gap> KG:=GroupRing(GF(2),G);;
gap> u:=GetRandomCentralNormalizedUnit(KG);;
gap> Augmentation(u);
Z(2)^0
gap> bool:=true;
true
gap> for x in Elements(KG) do
> if x*u<>u*x then bool:=false; break; fi;
> od;
gap> bool;
true
```

2.2.5 GetRandomElementFromAugmentationIdeal

▷ `GetRandomElementFromAugmentationIdeal(`*KG*`)` (function)

For the group ring `KG` the function `GetRandomElementFromAugmentationIdeal` returns an element from augmentation ideal of KG.

Example

```
gap> G:=QuaternionGroup(16);
<pc group of size 16 with 4 generators>
gap> KG:=GroupRing(GF(2),G);
<algebra-with-one over GF(2), with 4 generators>
gap> u:=GetRandomElementFromAugmentationIdeal(KG);;
gap> Augmentation(u);
0*Z(2)
```

2.3 Random Methods for Group Rings

2.3.1 RandomLienEngelLength

▷ `RandomLienEngelLength(KG, num)` (function)

Let `KG` be a group ring and let $[x,y,y,\dots,y]=0$ for all $x,y\in KG$. Then the number of y's in the last equation is called the Lie n-Engel length.

For the group ring `KG` and the maximal number of iterations `num` the function `RandomLienEngelLength` returns the Lie n-Engel length of KG by a random way.

Example

```
gap> G:=DihedralGroup(16);;
gap> KG:=GroupRing(GF(2),G);;
gap> RandomLienEngelLength(KG,100);
4
```

2.3.2 RandomExponent

▷ `RandomExponent(KG, num)` (function)

For the group ring `KG` and the maximal number of iterations `num` the function `RandomExponent` returns the exponent of the group of normalized units of `KG` by a random way.

Example

```
gap> G:=DihedralGroup(16);;
gap> KG:=GroupRing(GF(2),G);;
gap> RandomExponent(KG,100);
8
```

2.3.3 RandomExponentOfNormalizedUnitsCenter

▷ `RandomExponentOfNormalizedUnitsCenter(KG, num)` (function)

For the group ring `KG` and the maximal number of iterations `num` the function `RandomExponentOfNormalizedUnitsCenter` returns the exponent of the center of the group of normalized units of `KG` by a random way.

Example
```
gap> G:=DihedralGroup(16);;
gap> KG:=GroupRing(GF(2),G);;
gap> RandomExponentOfNormalizedUnitsCenter(KG,100);
4
```

2.3.4 RandomNilpotencyClass

▷ `RandomNilpotencyClass(`*KG, num*`)` (function)

For the group ring `KG` and the maximal number of iterations `num` the function `RandomNilpotencyClass` returns the nilpotency class of the group of normalized units of `KG` by a random way.

Example
```
gap> G:=DihedralGroup(16);;
gap> KG:=GroupRing(GF(2),G);;
gap> RandomNilpotencyClass(KG,100);
4
```

2.3.5 RandomDerivedLength

▷ `RandomDerivedLength(`*KG, n*`)` (function)

FG is called *`Lie solvable`*, if some of the terms of the Lie derived series $\delta^{[n]}(FG) = [\delta^{[n-1]}(FG), \delta^{[n-1]}(FG)]$ with $\delta^{[0]}(FG) = FG$ are equal to zero.

Denote by $_L(FG)$ the minimal element of the set $\{m \in \mathbb{N} \mid \delta^{[m]}(FG) = 0\}$, which is said to be the *`Lie derived length`* of FG.

For the group ring `KG` and a positive integer `n` the function `RandomDerivedLength` returns the Lie derived length by a random way.

Example
```
gap> D:=DihedralGroup(IsFpGroup,8);;
gap> KG:=GroupRing(GF(2),D);;
gap> RandomDerivedLength(KG,100);
2
```

2.3.6 RandomCommutatorSubgroup

▷ `RandomCommutatorSubgroup(`*KG, n*`)` (function)

For the group ring KG and a positive integer n the function RandomCommutatorSubgroup returns the commutator subgroup by a random way.

```
Example
gap> G:=DihedralGroup(IsFpGroup,8);;
gap> KG:=GroupRing(GF(2),G);;
gap> SG:=RandomCommutatorSubgroup(KG,100);
gap> StructureDescription(SG);
"C2 x C2 x C2"
gap> G:=CyclicGroup(8);;
gap> KG:=GroupRing(GF(3),G);;
gap> SG:=RandomCommutatorSubgroup(KG,100);;
gap> Elements(SG);
[ (Z(3)^0)*<identity> of ... ]
```

2.3.7 RandomCommutatorSubgroupOfNormalizedUnits

▷ RandomCommutatorSubgroupOfNormalizedUnits(*KG*, *n*) (function)

For the group ring KG and a positive integer n the function RandomCommutatorSubgroupOfNormalizedUnits returns the commutator subgroup of normalized units by a random way.

```
Example
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(3),G);;
gap> SG:=RandomCommutatorSubgroup(KG,100);;
gap> u:=Random(Elements(SG));;
gap> Augmentation(u);
Z(3)^0
```

2.3.8 RandomCenterOfCommutatorSubgroup

▷ RandomCenterOfCommutatorSubgroup(*KG*, *n*) (function)

For the group ring KG and a positive integer n the function RandomCenterOfCommutatorSubgroup returns the center of the commutator subgroup by a random way.

```
Example
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(3),G);;
gap> SG:=RandomCenterOfCommutatorSubgroup(KG,100);;
gap> x1:=Random(Elements(SG));; x2:=Random(Elements(SG));;
gap> x1*x2=x2*x1;
true
```

2.3.9 RandomNormalizedUnitGroup

▷ `RandomNormalizedUnitGroup(KG, n)` (function)

For the group ring `KG` and a positive integer `n` the function `RandomNormalizedUnitGroup` returns the normalized unit group by a random way.

Example

```
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(2),G);;
gap> SG:=RandomNormalizedUnitGroup(KG);
<group with 4 generators>
gap> Size(SG);
128
gap> u:=Random(Elements(SG));;
gap> Augmentation(u);
Z(2)^0
```

2.3.10 RandomUnitarySubgroup

▷ `RandomUnitarySubgroup(KG, n)` (function)

For the group ring `KG` and a positive integer `n` the function `RandomUnitarySubgroup` returns the unitary subgroup by a random way.

Example

```
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(3),G);;
gap> SG:=RandomUnitarySubgroup(KG,100);;
gap> u:=Random(Elements(SG));;
gap> Augmentation(u);
Z(3)^0
gap> u*u^-1;
(Z(3)^0)*<identity> of ...
```

2.3.11 RandomCommutatorSeries

▷ `RandomCommutatorSeries(KG, n)` (function)

For the group ring `KG` and a positive integer `n` the function `RandomCommutatorSeries` returns the commutator series by a random way.

Example

```
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(2),G);;
gap> CS:=RandomCommutatorSeries(KG,100);
[ <group of size 128 with 4 generators>, <group of size 8 with 8 generators>,
  <group of size 1 with 1 generators> ]
```

2.3.12 RandomLowerCentralSeries

▷ `RandomLowerCentralSeries(KG, n)` (function)

For the group ring `KG` and a positive integer `n` the function `RandomLowerCentralSeries` returns the lower central series by a random way.

Example

```
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(2),G);;
gap> CS:=RandomLowerCentralSeries(KG,100);
[ <group of size 128 with 4 generators>, <group of size 8 with 8 generators>,
  <group of size 1 with 1 generators> ]
```

2.3.13 RandomUnitaryOrder

▷ `RandomUnitaryOrder(KG, n)` (function)

For the group ring `KG` and a positive integer `n` the function `RandomUnitaryOrder` returns the unitary order of *KG* by a random way. Also, there exists a three-parametrical version of this method, where the third parameter σ is an arbitrary involution.

Example

```
gap> G:=DihedralGroup(8);;
gap> KG:=GroupRing(GF(3),G);;
gap> ord:=RandomUnitaryOrder(KG,100);
243
```

2.3.14 RandomDihedralDepth

▷ `RandomDihedralDepth(KG, n)` (function)

For the group ring `KG` and a positive integer `n` the function `RandomCentralUnitaryOrder` returns the depth of *KG* by a random way.

Example

```
gap> G:=DihedralGroup(16);;
gap> UD:=PcNormalizedUnitGroup(KG);
<pc group of size 32768 with 15 generators>
gap> DihedralDepth(UD);
3
gap> time;
4211
gap> RandomDihedralDepth(UD,100);
2
gap> RandomDihedralDepth(UD,300);
0
gap> RandomDihedralDepth(UD,500);
```

```
2
gap> RandomDihedralDepth(UD,1000);
0
gap> RandomDihedralDepth(UD,1000);
```

References

[BR00] V. Bovdi and A. Rosa. On the order of the unitary subgroup of a modular group algebra. *Comm. Algebra*, (28(4)):1897–1905, 2000. 4

Index