The GAP package SingerAlg – using the GAP-Julia integration

Thomas Breuer

Lehrstuhl für Algebra und Zahlentheorie, RWTH Aachen University, Germany

GAP Days Spring 2021, February 15, 2021

SingerAlg – what is it about?

- Fix coprime integers $q, z > 1$ and a field F,
- take n such that z divides $q^n - 1$, and set $e = (q^n - 1)/z$.
- Consider the F-algebra

$$A[q, z, F] = \bigoplus_{i=1}^{z+1} FB_i$$

with multiplication defined by

$$B_i \cdot B_j = B_{i+j-1}$$

if the q-adic expansions of $(i-1) \cdot e$ and $(j-1) \cdot e$ can be added without carry, and

$$B_i \cdot B_j = 0$$

otherwise.

Example: $z = 7, q = 8$

$n = 1$,
$e = (8^1 - 1)/7 = 1$

$B_1: \quad 0 \cdot e: \quad [0]$
$B_2: \quad 1 \cdot e: \quad [1]$
$B_3: \quad 2 \cdot e: \quad [2]$
$B_4: \quad 3 \cdot e: \quad [3]$
$B_5: \quad 4 \cdot e: \quad [4]$
$B_6: \quad 5 \cdot e: \quad [5]$
$B_7: \quad 6 \cdot e: \quad [6]$
$B_8: \quad 7 \cdot e: \quad [7]$

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	3	4	5	6	7	8	
3	3	4	5	6	7	8		
4	4	5	6	7	8			
5	5	6	7	8				
6	6	7	8					
7	7	8						
8	8							

Example: $z = 7, q = 2$

$n = 3$,
$e = (2^3 - 1)/7 = 1$

$B_1 : \quad 0 \cdot e : \quad [0,0,0]$
$B_2 : \quad 1 \cdot e : \quad [1,0,0]$
$B_3 : \quad 2 \cdot e : \quad [0,1,0]$
$B_4 : \quad 3 \cdot e : \quad [1,1,0]$
$B_5 : \quad 4 \cdot e : \quad [0,0,1]$
$B_6 : \quad 5 \cdot e : \quad [1,0,1]$
$B_7 : \quad 6 \cdot e : \quad [0,1,1]$
$B_8 : \quad 7 \cdot e : \quad [1,1,1]$

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2		4		6		8	
3	3	4			7	8		
4	4				8			
5	5	6	7	8				
6	6		8					
7	7	8						
8	8							

Example: $z = 7, q = 3$

$n = 6$,
$e = (3^6 - 1)/7 = 104$

$$
\begin{array}{lll}
B_1: & 0 \cdot e: & [0,0,0,0,0,0] \\
B_2: & 1 \cdot e: & [2,1,2,0,1,0] \\
B_3: & 2 \cdot e: & [1,0,2,1,2,0] \\
B_4: & 3 \cdot e: & [0,2,1,2,0,1] \\
B_5: & 4 \cdot e: & [2,0,1,0,2,1] \\
B_6: & 5 \cdot e: & [1,2,0,1,0,2] \\
B_7: & 6 \cdot e: & [0,1,0,2,1,2] \\
B_8: & 7 \cdot e: & [2,2,2,2,2,2]
\end{array}
$$

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2						8	
3	3					8		
4	4				8			
5	5			8				
6	6		8					
7	7	8						
8	8							

Example: $z = 7, q = 6$

$n = 2$,
$e = (6^2 - 1)/7 = 5$

$$
\begin{array}{lll}
B_1: & 0 \cdot e: & [0,0] \\
B_2: & 1 \cdot e: & [5,0] \\
B_3: & 2 \cdot e: & [4,1] \\
B_4: & 3 \cdot e: & [3,2] \\
B_5: & 4 \cdot e: & [2,3] \\
B_6: & 5 \cdot e: & [1,4] \\
B_7: & 6 \cdot e: & [0,5] \\
B_8: & 7 \cdot e: & [5,5]
\end{array}
$$

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2						8	
3	3					8		
4	4				8			
5	5			8				
6	6		8					
7	7	8						
8	8							

What are we interested in?

- Classify isomorphism types?
- Loewy structure?
- Substructures of $A = A[q, z, F]$:
  - radical series $A > J(A) > J(A)^2 > \cdots$,
  - socle series $0 < S(A) < S_2(A) < \cdots$,
  - in char. p: p-th powers $\{x^p; x \in U\}$ for $U \leq A$,
  - in char. p: p-th roots $\{x \in A; x^p \in U\}$ for $U \leq A$,
  - and their intersections, sums, annihilators, ...
- Compute derivations of subquotients of A (dimension)
- Count solutions of (nonlinear) equation systems.

Implementation

- GAP's generic `AlgebraByStructureConstants`?
- We can do better ("combinatorial approach"):

  The interesting substructures are generated by subsets of the basis $(B_1, B_2, \ldots, B_{z+1})$.

  Provide an independent, much simpler implementation.
  (We think of $z \leq 10000$.)

- Why Julia: Expect speedup.
  And expect that this can be achieved easier than via C code.

Package overview

- `SingerAlgebra( q, z )`: domains, methods for them, elements; store attributes; besides that, use only for testing
- `LoewyStructureInfo( q, z )`: store combinatorial data
- database of all $A[q, z]$ for $z \leq 10\,000$
- compute the abovementioned substructures/invariants
- compute permutation isomorphisms, using the GraPe package
- compute $m(q, e)$ (a number theoretic function, see exercises)
- decide when the upper bound

$$\lfloor n(q-1)/m(q,e) \rfloor + 1 \geq \mathrm{LL}(A[q,z])$$

is attained (for example for fixed e)

Package overview, continued

- compute $\dim(\mathrm{Der}(U(A)/S(A)))$ for a certain $\mathbb{F}_2$-subalgebra $U(A)$;

  if $U(A)/S(A)$ has dimension d then this means to compute the rank of a $(d^2 \times d^3)$-matrix over $\mathbb{F}_2$

- count the number of solutions of a polynomial equation system of degree two over $\mathbb{F}_2$,

  by iterating over the elements in a $\mathbb{F}_2$-vector space

Package structure

- surface/administration: GAP
- data structures: GAP record and Julia dictionary,

  `SingerAlg.MultTable( gapdata )`
  `Julia.SingerAlg.MultTable( juliadata )`

  store as needed q-adic expansions, multiplication table, basis subsets representing J^i, S_i, ...
- worker functions: GAP and Julia implementations

  `LoewyStructureInfoGAP( q, z )`
  `LoewyStructureInfoJulia( q, z )`

  (in particular: separate caching on both sides)

Example 1: q-adic expansion of $k \cdot e$, in GAP

```
gap> MyCoefficientsQadic:= function( ke, q, n )
>      local v, i;
>      v:= ListWithIdenticalEntries( n, 0 );
>      for i in [ 1 .. n ] do
>        v[i]:= RemInt( ke, q );   ke:= QuoInt( ke, q );
>      od;
>      return v;
>    end;;

gap> q:= 6;;  z:= 41;;  n:= OrderMod( q, z );
40
gap> e:= ( q^n - 1 ) / z;
326036452166920343118020633575
gap> MyCoefficientsQadic( e, q, n );
[ 1, 1, 2, 3, 5, 2, 2, 5, 1, 1, 3, 4, 1, 0, 2, 2, 4, 0, 5, 5,
  4, 4, 3, 2, 0, 3, 3, 0, 4, 4, 2, 1, 4, 5, 3, 3, 1, 5, 0, 0 ]
```

Example 1: q-adic expansion of $k \cdot e$, in Julia

```
julia> function my_coefficients_qadic(ke::Int, q::Int, n::Int)
         v = zeros(Int, n)
         for i in 1:n
           ke, r = divrem(ke, q);   v[i] = r
         end
         return v
       end;

julia> q = 6;  z = 41;  n = 40;

julia> e = div( q^n - 1, z )
5607619252820377 5
```

Example 1: q-adic expansion of $k \cdot e$, in Julia, 2nd attempt

```
julia> function my_coefficients_qadic(ke::T, q::Int,
                                      n::Int) where T<:Integer
         v = zeros(Int, n)
         for i in 1:n
           ke, r = divrem(ke, q);   v[i] = r
         end
         return v
       end;

julia> q = 6;  z = 41;  n = 40;

julia> e = div( big(q)^n - 1, z )
326036452166920343118020633575

julia> println(my_coefficients_qadic(e, q, n))
[1, 1, 2, 3, 5, 2, 2, 5, 1, 1, 3, 4, 1, 0, 2, 2, 4, 0, 5, 5,
 4, 4, 3, 2, 0, 3, 3, 0, 4, 4, 2, 1, 4, 5, 3, 3, 1, 5, 0, 0]
```

Example 1: q-adic expansion of $k \cdot e$, in Julia, 3rd attempt

- Be careful when translating GAP code to Julia.
- Do not expect speedup when dealing with Julia's "big integers".
- What is the solution?

  Use theory:
  q, z, n, k are small, but e is usually not.

  Compute the q-adic expansion of $k \cdot e$,
  without ever writing down e:

  `CoefficientsQadicReversed( k, z, q, n )`

Example 2:
Some functions from elementary number theory

In GAP:

`Factors`,
`IsPrimeInt`,
`IsPrimePowerInt`,
`NextPrimeInt`,
`OrderMod`,
`Phi`.

Currently, let Julia call the GAP functions:

```
factors(n::T) where {T<:Integer} =
    Vector{Int}(GAP.Globals.Factors(GAP.julia_to_gap(n)))
```

Example 3: Compute derivations

Given an algebra of dimension n,
compute (the dimension of) the Lie algebra of its (right) derivations.

In GAP:

`RightDerivations( B )`,
solve a linear equation system ($n^2 \times n^3$).

The matrix of this system is in general sparse.
In our situation, it is often very sparse.

We work over $\mathbb{F}_2$:
Represent each equation by the set of positions of nonzero cofficients.
In Julia, use objects of type `BitArray`.
(Implement only one new method for that type, which "flips one bit".)

Example 4:
Count solutions of a quadratic polynomial equation system

Translate the problem into a matrix equation $v \cdot A \cdot w^{tr} = 0$.

We work over $\mathbb{F}_2$:
Run over the vectors in a $\mathbb{F}_2$-vector space,
such that subsequent vectors differ in exactly one position (Gray code),
and compute the rank of a (small) matrix in each step.

Implement a suitable iterator, both in GAP and in Julia.

Conclusions

- We get speedup by using Julia.
  In the above examples, the factor is 5 to 10,
  but it depends also on the hardware.
  (And I did not run really large examples with the GAP functions.)

- One does not get this speedup for free,
  one has to invest work.

- It is true that this can be done also by people
  who cannot write C programs.

- "Translating" existing GAP code to Julia
  is often not a good point of view.

- Sooner or later one notices
  that the amount of Julia code is growing
  for which one has no GAP counterpart.

Thank you for your attention.

References


T. Breuer.
SingerAlg, Loewy lengths of certain algebras, Version 1.0.1.
http://www.math.rwth-aachen.de/~Thomas.Breuer/singeralg,
Jan 2021.


T. Breuer, L. Héthelyi, E. Horváth, and B. Külshammer.
The Loewy structure of certain fixpoint algebras, Part I.
J. Algebra 558:199–220, 2020.


T. Breuer, L. Héthelyi, E. Horváth, and B. Külshammer.
The Loewy structure of certain fixpoint algebras, Part II.
http://arxiv.org/abs/1912.03065, 2021.
To appear in International Electronic Journal of Algebra.


T. Breuer, S. Gutsche and M. Horn.
JuliaInterface, GAP interface to Julia, Version 0.5.2.
https://github.com/oscar-system/GAP.jl, Feb 2021.