Groups and GAP

George McNinch

2021-07-06

Note: In class, I promised I'd include these in the notes… you'll have to read the docs a bit to figure out how to get Character Tables. (The notes her might help you get started).

A overview of GAP

GAP is

> GAP - Groups, Algorithms, Programming - a System for Computational Discrete Algebra

You can find lots of information at the preceding link, including a tutorial and a variety of learning resources.

My basic goal here is to try to give you some idea of what you might be able to do with GAP (something of a trailer for the tutorial, perhaps?)

Installation

The installation instructions for GAP can be found here.

Personally, my computer's OS is Debian GNU/Linux, and I installed GAP by running the command

```
$ apt install gap
```

But while that is handy, I don't think the install process should be **too** difficult on any machine.

Starting GAP

After installing GAP, open a terminal on the computer and run GAP; you should see something like the following:

```
$ gap
        GAP 4.11.0 of 29-Feb-2020
   GAP     https://www.gap-system.org
        Architecture: x86_64-pc-linux-gnu-default64-kv7
 Configuration:  gmp 6.2.1, GASMAN, readline
 Loading the library and packages ...
 Packages:   Alnuth 3.1.2, AtlasRep 2.1.0, AutPGrp 1.10.2, CTblLib 1.3.1,
             FactInt 1.6.3, GAPDoc 1.6.3, IO 4.7.0, Polycyclic 2.15.1,
             PrimGrp 3.4.0, SmallGrp 1.4.1, TomLib 1.2.9, TransGrp 2.0.6
 Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap>
```

The bit at the end – i.e. gap> – is a *prompt*, and it indicates that GAP is waiting for you to type a command. Once you do so and press Enter, GAP will (try to) execute the command you entered.

It will print any output (or errors) from the command on the screen, and then it will display another prompt and again wait for input.

Permutation Groups

In GAP, probably the most straightforward way of representing (finite) groups is as permutation groups – i.e. subgroups of S_n for some $n \geq 1$.

For example, one knows that the symmetric group S_n is generated by an n-cycle and a transposition. In GAP, we may check this for some chosen n – say, $n = 8$ – as follows:

```
gap> G := Group((1,2),(1,2,3,4,5,6,7,8));
Group([ (1,2), (1,2,3,4,5,6,7,8) ])
gap> Order(G) = Factorial(8);
true
```

Note that elements are represented as *products of disjoint cycles*. If you enter them otherwise, GAP rewrites your representation as a product of disjoint cycles:

```
gap> (1,2,3,4)*(1,2,5);
(1,5)(2,3,4)
```

We can ask about the derived subgroup of G:

```
gap> H := DerivedSubgroup(G);
Group([ (1,2,3), (2,4,3), (2,4,5), (2,5,6,3,4), (3,7,4), (2,6)(4,7,8,5) ])
gap> Order(H) = Factorial(8)/2;
true
```

And we can ask about *simplicity* of groups:

```
gap> IsSimple(H);
true
gap> IsSimple(G);
false
```

We can ask about Sylow subgroups:

```
gap> f:=Factors(Order(G));
[ 2, 2, 2, 2, 2, 2, 2, 3, 3, 5, 7 ]
gap> Set(f);
[ 2, 3, 5, 7 ]
gap> ps:=List(Set(f),p->[p,SylowSubgroup(G,p)]);
[ [ 2, Group([ (1,2), (3,4), (1,3)(2,4), (5,6), (7,8), (5,7)(6,8), (1,5)(2,6)
          (3,7)(4,8) ]) ], [ 3, Group([ (1,2,3), (4,5,6) ]) ],
  [ 5, Group([ (1,2,3,4,5) ]) ], [ 7, Group([ (1,2,3,4,5,6,7) ]) ] ]
gap> for x in ps do
  Print("p=",x[1]," Sylow order=",Order(x[2]),
        " Sylow abelian? ",IsAbelian(x[2]),"\n");
od;
p=2 Sylow order=128 Sylow abelian? false
p=3 Sylow order=9 Sylow abelian? true
p=5 Sylow order=5 Sylow abelian? true
p=7 Sylow order=7 Sylow abelian? true
```

Let's define a *function* that takes a (finite) group as an argument and prints the "sylow information".

```
SylowInfo:=function(G)
  ps:=List(Set(Factors(Order(G))),p->[p,SylowSubgroup(G,p)]);
  for x in ps do
    Print("p=",x[1]," Sylow order=",Order(x[2]),
        " Sylow abelian? ",IsAbelian(x[2]),"\n");
  od;
  return ps;
end
```

If we enter the above function in GAP, we can *call* it for different groups:

```
gap> SylowInfo(G);
p=2 Sylow order=128 Sylow abelian? false
p=3 Sylow order=9 Sylow abelian? true
p=5 Sylow order=5 Sylow abelian? true
p=7 Sylow order=7 Sylow abelian? true
[ [ 2, Group([ (1,2), (3,4), (1,3)(2,4), (5,6), (7,8), (5,7)(6,8), (1,5)(2,6)
          (3,7)(4,8) ]) ], [ 3, Group([ (1,2,3), (4,5,6) ]) ],
  [ 5, Group([ (1,2,3,4,5) ]) ], [ 7, Group([ (1,2,3,4,5,6,7) ]) ] ]


gap> SylowInfo(H);
p=2 Sylow order=64 Sylow abelian? false
p=3 Sylow order=9 Sylow abelian? true
p=5 Sylow order=5 Sylow abelian? true
p=7 Sylow order=7 Sylow abelian? true
[ [ 2, Group([ (1,2)(3,4), (1,3)(2,4), (1,2)(5,6), (1,2)(7,8), (5,7)(6,8),
          (1,5)(2,6)(3,7)(4,8) ]) ], [ 3, Group([ (1,2,3), (4,5,6) ]) ],
  [ 5, Group([ (1,2,3,4,5) ]) ], [ 7, Group([ (1,2,3,4,5,6,7) ]) ] ]

gap> SylowInfo(SymmetricGroup(13));
p=2 Sylow order=1024 Sylow abelian? false
p=3 Sylow order=243 Sylow abelian? false
p=5 Sylow order=25 Sylow abelian? true
p=7 Sylow order=7 Sylow abelian? true
p=11 Sylow order=11 Sylow abelian? true
p=13 Sylow order=13 Sylow abelian? true
[ [ 2, <permutation group of size 1024 with 10 generators> ],
  [ 3, Group([ (1,2,3), (4,5,6), (7,8,9), (1,4,7)(2,5,8)(3,6,9), (10,11,12)
         ]) ], [ 5, Group([ (1,2,3,4,5), (6,7,8,9,10) ]) ],
  [ 7, Group([ (1,2,3,4,5,6,7) ]) ],
  [ 11, Group([ (1,2,3,4,5,6,7,8,9,10,11) ]) ],
  [ 13, Group([ (1,2,3,4,5,6,7,8,9,10,11,12,13) ]) ] ]

gap> SylowInfo(AlternatingGroup(14));
p=2 Sylow order=1024 Sylow abelian? false
p=3 Sylow order=243 Sylow abelian? false
p=5 Sylow order=25 Sylow abelian? true
p=7 Sylow order=49 Sylow abelian? true
p=11 Sylow order=11 Sylow abelian? true
p=13 Sylow order=13 Sylow abelian? true
[ [ 2, <permutation group of size 1024 with 10 generators> ],
  [ 3, Group([ (1,2,3), (4,5,6), (7,8,9), (1,4,7)(2,5,8)(3,6,9), (10,11,12)
         ]) ], [ 5, Group([ (1,2,3,4,5), (6,7,8,9,10) ]) ],
  [ 7, Group([ (1,2,3,4,5,6,7), (8,9,10,11,12,13,14) ]) ],
  [ 11, Group([ (1,2,3,4,5,6,7,8,9,10,11) ]) ],
  [ 13, Group([ (1,2,3,4,5,6,7,8,9,10,11,12,13) ]) ] ]
```

Finally, I'll point out that we can ask about group *isomorphisms*.

Let's look at a dihedral group:

```
gap> sigma:=function(n)
  return CycleFromList([1..n])*CycleFromList([n+1..2*n]);
end;
function( n ) ... end
```

```
gap> sigma(9);
(1,2,3,4,5,6,7,8,9)(10,11,12,13,14,15,16,17,18)
  # element of S_18 of order 9

gap> tau:= function(n)
  return Product(List([1..n],i->(i,2*n-i+1)));
function( n ) ... end

gap> tau(9);
(1,18)(2,17)(3,16)(4,15)(5,14)(6,13)(7,12)(8,11)(9,10)
  # element of S_18 of order 2
```

Note that "`tau * sigma * tau = sigma^-1`". e.g.

```
gap> tau(9)*sigma(9)*tau(9)=sigma(9)^-1;
true
```

```
gap> MyDihedralGroup:=function(n)
  return Group(sigma(n),tau(n));
end;
function( n ) ... end
```

We've now defined the function `MyDihedralGroup`, and `MyDihedralGroup(n)` returns the subgroup of S_{2n} generated by `sigma(n)` and `tau(n)`.

Now, GAP has a built-in function `DihedralGroup`; here `DihedralGroup(n)` returns a copy of the dihedral group of order `n` (so `n` needs to be even…).

Let's check that we get isomorphic groups:

```
gap> DihedralGroup(18);
<pc group of size 18 with 3 generators>

gap> MyDihedralGroup(9);
Group([ (1,2,3,4,5,6,7,8,9)(10,11,12,13,14,15,16,17,18), (1,18)(2,17)(3,16)
  (4,15)(5,14)(6,13)(7,12)(8,11)(9,10) ])

gap> IsomorphismGroups(DihedralGroup(18),MyDihedralGroup(9));
[ f1, f2, f3 ] -> [ (1,18)(2,17)(3,16)(4,15)(5,14)(6,13)(7,12)(8,11)(9,10),
  (1,2,3,4,5,6,7,8,9)(10,11,12,13,14,15,16,17,18),
  (1,4,7)(2,5,8)(3,6,9)(10,13,16)(11,14,17)(12,15,18) ]
```

The function `IsomorphismGroups` takes two groups as arguments, and returns an isomorphism (if one exists) or "fails". In the preceding case, GAP found an explicit isomorphism between GAP's `DihedralGroup(18)` and `MyDihedralGroup(9)`.

You can check it for other values of `n`…

```
gap> IsomorphismGroups(DihedralGroup(32),MyDihedralGroup(16));
[ f1, f2, f3, f4, f5 ] ->
[ (1,32)(2,31)(3,30)(4,29)(5,28)(6,27)(7,26)(8,25)(9,24)(10,23)(11,22)(12,
    21)(13,20)(14,19)(15,18)(16,17), (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
    16)(17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32),
  (1,3,5,7,9,11,13,15)(2,4,6,8,10,12,14,16)(17,19,21,23,25,27,29,31)(18,20,22,
    24,26,28,30,32), (1,5,9,13)(2,6,10,14)(3,7,11,15)(4,8,12,16)(17,21,25,
    29)(18,22,26,30)(19,23,27,31)(20,24,28,32),
  (1,9)(2,10)(3,11)(4,12)(5,13)(6,14)(7,15)(8,16)(17,25)(18,26)(19,27)(20,
    28)(21,29)(22,30)(23,31)(24,32) ]
```

And in the other order:

```
gap> IsomorphismGroups(MyDihedralGroup(16),DihedralGroup(32));
[ (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)(17,18,19,20,21,22,23,24,25,26,27,
```

```
    28,29,30,31,32), (1,32)(2,31)(3,30)(4,29)(5,28)(6,27)(7,26)(8,25)(9,
    24)(10,23)(11,22)(12,21)(13,20)(14,19)(15,18)(16,17) ] -> [ f2, f1 ]
```

And you can observe that it *fails* on nonsense:

```
gap> IsomorphismGroups(DihedralGroup(30),MyDihedralGroup(16));
fail
```

Matrix groups over finite fields

Finite fields

Recall the following:

- a finite field $\mathbb{F} = \mathbb{F}_q$ has order $q = p^n$ for some prime p and some $n \geq 1$
- the multiplicative group $\mathbb{F}_q^\times$ is cyclic of order $q - 1$.

GAP writes `Z(p,n)` or `Z(q)` for a *generator* of the multiplicative group $\mathbb{F}_q^\times$.

Now you can perform field operations with these elements:

```
gap> Z(9) + 5*Z(3) + 2*Z(9)*Z(3);
Z(3^2)^3

gap> Z(9) + Z(27);  # note F_9 is not contained in F_27;
                    # the smallest field containing both F_9 and
                    # F_27 is F_3^6=F_729
Z(3^6)^297
```

You can get the elements `0` or `1` viewed as members of a finite field using `0*Z(p,n)` for `0` and `Z(p,n)^0` for `1`.

Matrices

A *row vector* is just a list; e.g.

```
gap> [ 1, 2, 3, 4 ];
[ 1, 2, 3, 4 ]
```

A matrix is a list of row vectors:

```
gap> [ [ 1, 1 ], [ 0, 1 ] ];   # integer entries...
[ [ 1, 1 ], [ 0, 1 ] ]

gap> [ [ 1, 1 ], [ 0, 1 ] ]*Z(9)^0;   # entries in the field F_9.
[ [ Z(3)^0, Z(3)^0 ], [ 0*Z(3), Z(3)^0 ] ]
```

Matrix Groups

In GAP, we can create the *general linear group* $\mathrm{GL}_n(\mathbb{F}_q)$ of invertible $n \times n$ matrices over the field of order q using a command of the form `G:=GL(n,q);`; for example:

```
gap> G:=GL(3,17);
GL(3,17)
gap> Order(G);
111203278848
```

Similarly, we can construct the *special linear group* – the kernel of

$$\det : \mathrm{GL}_n(\mathbb{F}_q) \to \mathbb{F}_q^\times$$

– using `SL(n,q)`.

GAP has constructors for other matrix groups (projective special linear group, symplectic group, unitary group , orthogonal group, …) which I'm going to ignore today.

It is a fact that unless `(n,q)` `[(2,2),(2,3)]` one knows that `SL(n,q)` is a simple group $\iff$ $\gcd(n, q-1) = 1$, which we can confirm in some cases in GAP:

```
gap> IsSimple(SL(5,7));
true
gap> IsSimple(SL(3,7));
false
```

In fact, the quotient of `SL(n,q)` by its center *is* simple (at least when n and q aren't "tiny"):

```
gap> ZZ:=Center(SL(3,7)); Order(ZZ);
<group of 3x3 matrices over GF(7)>
3
gap> P:=SL(3,7)/ZZ; Order(P); IsSimple(P);
<permutation group with 2 generators>
1876896
true
```

GAP can compute the *conjugacy classes* of a finite group:

```
gap> ConjugacyClasses(SL(2,7));
[ [ [ Z(7)^0, 0*Z(7) ], [ 0*Z(7), Z(7)^0 ] ]^G,
  [ [ 0*Z(7), Z(7)^3 ], [ Z(7)^0, Z(7)^5 ] ]^G,
  [ [ 0*Z(7), Z(7)^4 ], [ Z(7)^5, Z(7)^5 ] ]^G,
  [ [ Z(7)^3, 0*Z(7) ], [ 0*Z(7), Z(7)^3 ] ]^G,
  [ [ 0*Z(7), Z(7)^3 ], [ Z(7)^0, Z(7)^2 ] ]^G,
  [ [ 0*Z(7), Z(7)^4 ], [ Z(7)^5, Z(7)^2 ] ]^G,
  [ [ 0*Z(7), Z(7)^3 ], [ Z(7)^0, 0*Z(7) ] ]^G,
  [ [ 0*Z(7), Z(7)^3 ], [ Z(7)^0, Z(7)^4 ] ]^G,
  [ [ 0*Z(7), Z(7)^3 ], [ Z(7)^0, Z(7) ] ]^G,
  [ [ Z(7)^4, 0*Z(7) ], [ 0*Z(7), Z(7)^2 ] ]^G,
  [ [ Z(7)^5, 0*Z(7) ], [ 0*Z(7), Z(7) ] ]^G ]
```

For each of these conjugacy classes, one may use the function `Representative` to choose a representative element.

Let's write some code that takes as input a whole number `n`, a prime power `q=p^m` and a matrix `M` and finds the conjugacy class in `GL(n,q)` containing `M*Z(q)`.

```
LocateClass:=function(n,q,M)
  G:=GL(n,q);
  cls:=ConjugacyClasses(G);
  return First(cls,c->IsConjugate(G,Representative(c),M*Z(q)^0));
end;
```

This code uses the boolean-valued function `IsConjugate`. Note that `First(list,f)` applies the boolean-valued function `f` to successive elements of `list`, and returns the first element of `list` for which `f` evaluates to `true`.

So for example:

```
gap> LocateClass(2,17,[[1,1],[0,1]]);
[ [ 0*Z(17), Z(17)^8 ], [ Z(17)^0, Z(17)^14 ] ]^G
gap> LocateClass(2,17,[[1,5],[5,1]]);
[ [ Z(17)^15, 0*Z(17) ], [ 0*Z(17), Z(17)^4 ] ]^G
gap> LocateClass(2,17,[[1,5],[5,1]]*Z(17^2));
fail
```

A bit more on group actions

Let's consider a finite field $k = \mathbb{F}_q$ and a 2 dimensional vector space $V \simeq k^2$ over k.

The group `G=GL(4,q)` acts on V by matrix multiplication.

For a particular q, we can ask `GAP` for the list of `G` orbits on elements of `V` as follows:

```
gap> G:=GL(2,13);
gap> k:=GF(13);      # "Galois field" = finite field of order 13
gap> V:=k^2;
gap> oo:=OrbitsDomain(G,V);
[ [ [ 0*Z(13), 0*Z(13) ] ],
  [ [ 0*Z(13), Z(13)^0 ], [ Z(13)^6, 0*Z(13) ], [ Z(13)^7, 0*Z(13) ],
      [ Z(13)^0, Z(13)^6 ], [ Z(13)^8, 0*Z(13) ], [ Z(13), Z(13)^7 ],
      [ Z(13), Z(13)^6 ], [ Z(13)^9, 0*Z(13) ], [ Z(13)^2, Z(13)^8 ],
      [ Z(13)^2, Z(13)^7 ], [ 0*Z(13), Z(13) ], [ Z(13)^2, Z(13)^6 ],
      ...

      ## there are two orbits...
gap> oo[1];
[ [ 0*Z(13), 0*Z(13) ] ]
gap> Length(oo[2]);
168
```

Now, the stabilizer of a 1-dimensional subspace is the "upper triangular subgroup", which we can construct as the normalizer of a 13-sylow subgroup.

A 13-sylow subgroup is equal to

$$\left\{ \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \mid x \in \mathbb{F}_{13}. \right\}$$

and the normalizer this sylow is

$$B = \left\{ \begin{bmatrix} a & x \\ 0 & b \end{bmatrix} \mid a, b, x \in \mathbb{F}_{13}, ab \neq 0. \right\}$$

Let's describe B using `GAP` in two ways:

As normalizer:

```
gap> u:=[[1,1],[0,1]]*Z(13)^0;
gap> U:=Subgroup(G,[u]);
ActGroup([ [ [ Z(13)^0, Z(13)^0 ], [ 0*Z(13), Z(13)^0 ] ] ])
gap> B:=Normalizer(G,U);
<group of 2x2 matrices over GF(13)>
```

And as stabilizer:

```
gap> BB:=Basis(V);
CanonicalBasis( ( GF(13)^2 ) )
gap> H:=Stabilizer(G,BB[2],OnLines);
Group([ [ [ Z(13), 0*Z(13) ], [ 0*Z(13), Z(13)^0 ] ],
  [ [ Z(13)^0, 0*Z(13) ], [ Z(13)^6, Z(13) ] ] ])

gap> H=B;
true
```

Now, since the action of G on lines in V is *transitive*, the set of cosets G/B identifies with the set of such lines.

```
gap> Index(G,B);
14
```

Thus there are 14 lines in V. Since G acts on lines, we get a (transitive) action of G on a set of 14 elements, which we are going to view as the cosets of B in G:

```
gap> P1:=RightCosets(G,B);
[ RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^0, 0*Z(13) ], [ 0*Z(13), Z(13)^0 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^6, Z(13)^0 ], [ Z(13)^6, 0*Z(13) ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ 0*Z(13), Z(13)^6 ], [ Z(13)^0, Z(13)^6 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^2, 0*Z(13) ], [ Z(13)^8, Z(13) ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^10, 0*Z(13) ], [ Z(13)^11, Z(13)^11 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^2, Z(13)^7 ], [ Z(13)^3, Z(13)^7 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^5, Z(13)^5 ], [ Z(13)^4, Z(13)^5 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^8, Z(13)^2 ], [ Z(13), Z(13)^8 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ 0*Z(13), Z(13)^3 ], [ Z(13)^11, Z(13)^8 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^8, Z(13)^2 ], [ Z(13)^0, Z(13)^3 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^7, Z(13)^8 ], [ Z(13)^0, Z(13)^8 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^0, Z(13)^5 ], [ Z(13)^2, Z(13)^4 ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ 0*Z(13), Z(13)^8 ], [ Z(13)^2, Z(13) ] ]),
  RightCoset(<group of size 1872 with 2 generators>,
   [ [ Z(13)^0, Z(13) ], [ Z(13)^9, Z(13)^7 ] ]) ]
```

This action corresponds to a homomorphism

$$\mathrm{GL}_2(\mathbb{F}_{13}) \to S_{14}$$

which we can create in GAP as follows:

```
gap> hom:=ActionHomomorphism(G,P1,OnRight);
gap> Image(hom);
Group([ (3,4,6,10,12,7,5,13,14,9,11,8), (1,2,3)(4,8,5)(7,12,9)(10,14,13) ])

gap> Kernel(hom);
Group([ [ [ Z(13)^3, 0*Z(13) ], [ 0*Z(13), Z(13)^3 ] ],
  [ [ Z(13)^4, 0*Z(13) ], [ 0*Z(13), Z(13)^4 ] ],
  [ [ Z(13)^8, 0*Z(13) ], [ 0*Z(13), Z(13)^8 ] ],
  [ [ Z(13)^9, 0*Z(13) ], [ 0*Z(13), Z(13)^9 ] ] ])

gap> Order(Kernel(hom));    # should be just the scalar matrices!
12
```

Bibliography