WORKING WITH GALOIS GROUPS IN *MAGMA*

DAVID P. ROBERTS
UNIVERSITY OF MINNESOTA, MORRIS

These notes complement the lectures and aim to give you practical experience in working with Galois groups in *Magma*. They are aimed at people for whom working with *Magma* is something of a struggle. I myself am still in this category, although perhaps starting to leave it, having written these notes . . . .

Given the aims, the exercises corresponding to the first two lectures are intended to be easy. They are commonly just running programs and they never present obstructions to continuing. The point is to increase the comfort level of beginners to *Magma*. Another way I keep the computational context simple is that I center everything on a single datatype, polynomials in $\mathbb{Z}[x]$.

If you are truly a rank beginner, I recommend typing in the displayed code as an aid to understanding it. Some of the most basic *Magma* constructions are used repeatedly. If you are past this level and would learn nothing by your own typing, all the displayed code is available in the file `exercisesmagma`. If you are fluent at *Magma*, a meta-exercise for the first two days is to improve my code!

The exercises corresponding to the third lecture are harder and more open-ended. If the material in the first two sections becomes too easy for you at some point, feel free to move ahead.

1. Exercises for Monday's lecture: Basic practice in *Magma*

There is little in these notes on the generalities of *Magma* coding. To find out how to use say `Factorization`, type `?Factorization` and enter the large *Magma* index.

Some basics. *Magma* is very strongly typed, and we need to tell it we will be working in $\mathbb{Z}[x]$.

```
Zx<x> := PolynomialRing(Integers());
```

As examples, we will work with the following polynomials:

```
f5 := x^5-2;
f6 := x^6-x-1;
f7a := x^7-7*x-3;
f7b := x^7-7*x-4;
f8 := x^8-16*x+28;
```

Exercise 1.1. *To what extent can you see irreducibility and bad primes of these polynomials just by looking? Use* `Factorization` *to confirm the polynomials are all irreducible in* $\mathbb{Z}[x]$. *Use* `Discriminant` *to determine the bad primes belonging to each polynomial.*

Factorization patterns and guessing exponents of Galois groups. Often only degrees of factors matter. Accordingly, the following function is helpful.

```
  degrees := func<lis|
    Reverse(Sort(
        [Degree(lis[i,1]) : i in [1..#lis]]
  ))>;
```

This function is meant to be used only as a wrapper around `Factorization`. As an example, `degrees(Factorization((x^2+2*x+5)^3*(x-1)*(x-2)))` returns `[2,1,1]`, to be thought of as a partition. Note that the important multiplicity 3 is lost, but we will be using `degrees` below mainly in the context of good primes.

Now we can calculate factorization partitions by the following function:

```
  factpat := func<poly,p|
        degrees(Factorization(
        PolynomialRing(FiniteField(p))!poly
  ))>;
```

As examples, `factpat(f7a,3)` returns `[1,1,1]` corresponding to the bad factor $1^3 1^3 1$ with exponents dropped; `factpat(f7a,13)` returns the entire good factorization partition `[4,2,1]`.

The *exponent* of a group G is the smallest positive integer e with $g^e = 1$ for all $g \in G$. The least common multiple $LCM(\lambda_p)$ of a good factor partition λ_p gives the order of a corresponding Frobenius element g in the Galois group $\mathrm{Gal}(f)$. The least common multiple of all the $LCM(\lambda_p)$ is the exponent e of $\mathrm{Gal}(f)$.

Exercise 1.2. *Use* `factpat` *for primes p (arrow keys to edit!) in increasing order to get improving lower bounds on e for f_5, ..., f_8. In each case, at what prime does the lower bound appear to become exact?*

Prematurely guessing orders of Galois groups. To take data more efficiently, we can use the following command:

```
  factpats := func<poly,cut|
        [ <i,NthPrime(i),factpat(poly,NthPrime(i))> : i in [1..cut] ]
  >;
```

An extremely crude first guess at the order of $\mathrm{Gal}(f)$ is N_1, where p_{N_1} is the first prime for which the factorization partition consists of $n = \mathrm{degree}(f)$ ones. An improvement, still crude, is the nearest integer N_2 to N_1 which

- is a multiple of e;
- has no primes divisors besides those of e;
- divides $n!$.

Here we assume that the previous guess of e is correct and if a tie occurred we would break it by taking the smaller option. The correct order $|\mathrm{Gal}(f)|$ definitely satisfies the three bulleted conditions.

Exercise 1.3. *Compute the approximations N_1 and N_2 to $|\mathrm{Gal}(f_i)|$. Which N_2 do you think are right?*

Confidently guessing orders of Galois groups. The output of `factpats` keeps track of which primes give rise to which factpats. This is certainly fundamental information in many contexts, such as the construction of Dedekind zeta functions.

But one can also forget the source primes and just keep track of what factpats arise and with what multiplicity. First, as a data management function,

```
tally := function(seq)
   set := Seqset(seq);
   return Sort([<s,Multiplicity(seq,s)> : s in set]);
end function;
```

This function tallies, with e.g. `tally(["a","b","a"])` returning `[<"a",2>,<"b",1>]`. Now as the main function,

```
factpatstats:=func<poly,cutoff|
     tally([factpat(poly,NthPrime(j)) : j in [1..cutoff]])
>;
```

The Chebotarev density theorem says that the all-ones factorization partition occurs with asymptotic frequency $1/|\mathrm{Gal}(f)|$. Using this fact at a heuristic level, and keeping in mind that $|\mathrm{Gal}(f)|$ satisfies the bulleted conditions above, one can make confident guesses at $|\mathrm{Gal}(f)|$.

Exercise 1.4. *Use* `factpatstats` *with moderately large cutoffs to heuristically determine* $|\mathrm{Gal}(f_i)|$.

Corresponding group theory. Now we switch gears and consider permutation groups. The number of conjugacy classes of transitive degree n permutation groups is given in *Magma* by the command `NumberOfTransitiveGroups`. There are e.g. 50 such groups in degree 8, and `TransitiveGroup(8,35)` gives you the 35th on the standard list, partially sorted by increasing order. One of many group-theoretic commands is `Order` with e.g. `Order(TransitiveGroup(8,35))` returning 128.

For each group, one has its corresponding distribution of cycle types. To get it in *Magma*, we again first use a data management command

```
untally := func<tallied|
   [tallied[i,1] : j in [1..tallied[i,2]], i in [1..#tallied]]
>;
```

This function turns `[<"a",2>,<"b",1>]` back into `["a","a","b"]` which is more readable in our context. The cycle distribution command is then

```
grpstats := func<n,i|
   tally([untally(CycleStructure(g)) : g in TransitiveGroup(n,i)])
>;
```

(Two notes: First, the prettification commands tally and untally are applied in different contexts and do not undo each other. Second, the command `grpstats` is intended to be run only for small groups, as we are looping over group elements; for larger groups, one would modify `grpstats` to loop only over representatives of conjugacy classes).

Exercise 1.5. *It can happen that different transitive groups nTi have the same distribution of cycle types, e.g. $8T10$ and $8T11$. Find all such multiplets among octic transitive groups.*

Comparing factorization statistics with cycle structure statistics. Now we can compare our functions for polynomials with our functions for groups:

Exercise 1.6. *For each f_j find the only (n,i) with the output of* `factpatstats` *matching* `grpstats`*, thereby determing $Gal(f_j)$ with great confidence.*

Some Galois-related *Magma* commands. So far, we have not used any Galois-theory specific *Magma* commands. *Magma*—as opposed to say *Mathematica*—has such commands, and the main one is `GaloisGroup`. This command operates at a deeper level than our viewpoint today, and so we surround it with two more *Magma* commands to get a function matching today's viewpoint:

```
galgrp := function(f)
     G := GaloisGroup(f);
     return <Degree(f),TransitiveGroupIdentification(G),
               TransitiveGroupDescription(G)>;
end function;
```

Typing e.g. `galgrp(f6)` returns the degree $n = 6$, the "T-number" $i = 16$, and the more descriptive name S_6. (Feel free to look at `GaloisGroup` itself now; we will be discussing it on Wednesday!)

Magma has a small database consisting of one polynomial for each nTi with $n \leq 15$. After loading it via `load galpols` you can access the polynomials via e.g. `PolynomialWithGaloisGroup(13,7)`. Since we might want to use this command a lot, we abbreviate

```
examp := PolynomialWithGaloisGroup;
```

Exercise 1.7. *Use* `galgrp` *to confirm that your answer to Exercise 1.6 is correct. Then check that the small database gives a polynomial g_i defining a different field (either by finding a disagreeing good factpat or checking that discriminants do not agree modulo squares).*

Supplementary commands. The next two commands let one use some of *Magma*'s number field commands without leaving the gentler environment of univariate polynomials.

```
cleanup := func<f|Zx!DefiningPolynomial(
                 OptimizedRepresentation(NumberField(f)))>;

fielddisc := func<f|Discriminant(MaximalOrder(NumberField(f)))>;
```

As a simple example, `cleanup(x^5-486)` returns the polynomial $f_5 = x^5 - 2$ defining the same field. While f_5 has polynomial and field discriminant $D = d = 2^4\,5^5$, the original polynomial $x^5 - 486$ has $D = 2^4\,3^{20}\,5^5$ and $d = 2^4\,5^5$.

(Note: `OptimizedRepresentation` aims to minimize D while *Pari*'s comparable *polredabs* aims to minimize the sum of the absolute squares of the roots. An important problem is to improve the implementation of `OptimizedRepresentation` so that it successfully works in the larger range that *polredabs* currently does.)

One often has two polynomials simultaneously under consideration with coordinated factpats. As an aid to seeing the coordination quickly:

```
factpatstats2:=function(poly1,poly2,cutoff)
   ugly := tally([[factpat(poly1,NthPrime(j)),
           factpat(poly2,NthPrime(j))] : j in [1..cutoff]]);
   return [<u[1,1],u[1,2],u[2]> : u in ugly];
end function;
```

For example, `factpatstats2(x^3+2,x^2+3,100)` reproduces one of the tables in the lecture.

For those wanting a modest programming exercise:

Exercise 1.8. *a) The programs* `cleanup` *and* `fielddisc` *work only for irreducible polynomials. Write better versions which accept general separable polynomials (so that e.g.* `fielddisc` *would return the algebra discriminant).*

b) Similarly, one often has more than two polynomials simultaneously under consideration. Extend `factpatstats2` *accordingly.*

2. Exercises for Wednesday's lecture: Galois theory at three levels

In the lecture, we talked about two levels at which you can do Galois theory over $\mathbb{Z}$. Here we'll introduce an intermediate level as well:

Coarse level. One works purely algebraically in $\mathbb{Z}[x]$, never actually seeing roots anywhere.

Intermediate level. One works numerically with roots but is still content to compute Galois groups only up to conjugation.

Fine level. One works numerically with roots, and wants to identify the Galois group as an actual permutation group on the roots.

The exercises take place at these levels in order. Our goal is to get to the fine level, which *Magma* supports very well. The two earlier levels present some of the ideas in easier contexts.

Coarse-level exercises. The `discpoly` resolvent from the lecture can be clunkily implemented in *Magma* as follows.

```
  Zxy<x1,y> := PolynomialRing(Integers(),2);

  discpoly := function(poly)
       return Zx ! Evaluate(
               Resultant(Evaluate(poly,y),Evaluate(poly,x1+y),y)/
               x1^Degree(poly),[x,0]);
  end function;

  discpolypat := function(f)
      dp := discpoly(f);
      return [<Degree(s[1]),s[2]> : s in Factorization(dp)];
  end function;
```

The first line complements our earlier and still-needed definition of `Zx`. The middle chunk is the main program. The last chunk suppresses information and outputs the numerical invariants that we care most about.

Let `f7a := x^7-7*x-3` and `f7b := x^7-7*x-4` be as before. The next exercise shows that carelessly going into quite high degrees is actually ok in some circumstances:

Exercise 2.1. *Verify the following statements: a)* `discpolypat(   )` *gives the same answer on* f_{7a} *and* f_{7b}*; b)* `discpolypat(discpoly(  ))` *gives different answers on* f_{7a} *and* f_{7b}*, but these answers correspond to inseparable polynomials.*

Because of the repeated factors, the output of `discpolypat(discpoly(  ))` *leaves many possibilities for the corrected factorization pattern (e.g.* `<21,2>` *could be corrected to either* `[21, 21]` *or* `[42]`*). Is the ambiguous information given just by* `discpolypat(discpoly(  ))` *enough to say that* f_{7a} *and* f_{7b} *have different Galois groups?*

To avoid inseparability issues, one can replace a polynomial by a field-equivalent polynomial at any step. The next function clunkily does this (assuming separability of the output!):

```
messup := function(f)
    compmat := CompanionMatrix(f);
    return CharacteristicPolynomial(compmat + compmat*compmat);
end function;
```

The program `messup` typically makes coefficients bigger. Again this is not desirable but often not a problem:

Exercise 2.2. *Use* `messup` *to conceptually simplify the last computation, thus cleanly showing that* f_{7a} *and* f_{7b} *have different Galois groups without entering into inseparability issues.*

In general, of course, one wants to keep degrees as small as possible. A simple modification in the present case is to note that if $D_f(x)$ is the discpoly of $f(x)$ then one can write $D_f(x) = SD_f(x^2)$. We'll call $SD_f(x)$ the small discpoly.

```
half := function(f)
   coeffs := Eltseq(f);
   return &+[coeffs[i]*x^((i-1) div 2) : i in [1..#coeffs by 2]];
end function;

smalldiscpoly := func<f | half(discpoly(f))>;

smalldiscpolypat := function(f)
    sdp := smalldiscpoly(f);
    return [<Degree(s[1]),s[2]>:s in Factorization(sdp)];
end function;
```

(Pretty obvious room for coding improvement on the last page and a half, but maybe a little repetition is occasionally good...)

Exercise 2.3. *Always inserting* `messup` *as necessary to stay away from inseparability,*
a) What is the partition of $\{1, \ldots, 16\}$ *corresponding to the ability of* `smalldiscpolypat(  )` *by itself to distinguish different* $6Ti$*?*
b) Same question for `smalldiscpolypat(smalldiscpoly( ))`

Intermediate-level exercises. The program just discussed passes from a polynomial $f(x)$ to its discriminant polynomial $D_f(x)$ purely algebraically. As a consequence there are universal formulas. One such formula is that for $f(x) = x^3 + ax^2 + bx + c$ the corresponding discpoly is

$$D_f(x) = x^6 + (6b - 2a^2)x^4 + (a^4 - 6a^2b + 9b^2)x^2 + (4a^3c - a^2b^2 - 18abc + 4b^3 + 27c^2).$$

The resolvent construction D corresponds to the subgroup S_{n-2} of S_n while the smaller resolvent construction SD corresponds to the larger subgroup $S_{n-2} \times S_2$. In general, for any $H \subset S_n$ one can make a similar construction $f \mapsto f^H$. Here the degree of f^H is the index $[S_n : H]$. These other constructions also have universal formulas, but they can be very unwieldy. Instead, one can carry them out non-algebraically, using say complex roots.

Important cases correspond to the subgroups S_{n-k}, $S_k \times S_{n-k}$, and, in the case that $n = 2k$, the wreath product $S_k \wr 2 = (S_k \times S_k).2$. We denote the resolvents of $f(x)$ under these constructions by $f^{\underline{k}}(x)$, $f^{[k]}(x)$, and $f^{\mathrm{mid}}(x)$. If f has roots α_i then we define e.g. $f^{\underline{3}}(x)$ to have roots $\alpha_{i_1} + 2\alpha_{i_2} + 3\alpha_{i_3}$ with i_1, i_2, i_3 distinct. The smaller resolvents $f^{[k]}$ are similar, with $f^{[3]}$ having roots $\alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3}$ with $i_1 < i_2 < i_3$. The special case f^{mid} is modeled after the classical $k = 2$ case, where the roots are $\alpha_1\alpha_2 + \alpha_3\alpha_4$, $\alpha_1\alpha_3 + \alpha_2\alpha_4$, and $\alpha_1\alpha_4 + \alpha_2\alpha_3$. The somewhat arbitrary explicit formulas at the level of polynomials correspond to canonical operations at the level of algebras. At the level of algebras, $F \mapsto F^{\underline{2}}$ and $F \mapsto F^{[2]}$ agree with the discpoly and the small discpoly.

The resolvent $f \mapsto f^{[k]}$ can be implemented using complex roots as follows:

```
coercedown := function(f)
   coeffs := Eltseq(f);
   return &+[Round(Real(coeffs[i]))*x^(i-1) : i in [1..#coeffs]];
end function;

resk := function(f,k,prec)
   Cprec := ComplexField(prec);
   Cprecz<z> := PolynomialRing(Cprec);
   fprecz := Cprecz ! f;
   rootpairs := Roots(fprecz);
   roots := {rp[1] : rp in rootpairs};
   newroots := {&+s : s in Subsets(roots,k)};
   resz := &*{z - n : n in newroots};
   return coercedown(resz);
end function;
```

The input to `coercedown` is a polynomial with coefficients which are typed by *Magma* as complex, but mathematically should be integral except for round-off errors. The function `resk` then essentially has input (f, k) and output $f^{[k]}$. However a third argument, the decimal digits of precision of the numerical steps, is needed. It may need to be considerably more than *Magma*'s standard 30 for the output to be the desired algebraically-correct resolvent.

As a variant,

```
resmid := func<f,prec |half(resk(f,Degree(f) div 2,prec))>;
```

does $f \mapsto f^{\mathrm{mid}}$, where f is required to be traceless: $f(x) = x^n + 0x^{n-1} + \cdots$.

Exercise 2.4. *How does $f^{\mathrm{mid}}(x)$ factor for polynomials with Galois group S_4, S_6, $8T48 = E(8){:}L(7) = AL(8)$, and $12T295 = M_{12}$? What precision is necessary for the program to work on the M_{12} polynomial*

```
f12 := x^12 - 6*x^10 - 10*x^9 - 90*x^8 - 150*x^7 + 430*x^6
       - 720*x^5 + 900*x^4 + 1350*x^3 - 1350*x^2 + 4050*x + 675;
```

of the lecture? From the M_{12} factorization and some group theory, what must the factorization for an A_{12} polynomial be? Confirm this in an example.

Fine-level calculations. The command `GaloisGroup` has been highlighted in the lectures. It is the core command in a bigger package. Here we illustrate how the supplementary (and strangely named) command `GaloisSubgroup` can be used to compute general resolvents.

The group S_6 has a transitive subgroup $PGL_2(5)$ isomorphic to the standard intransitive subgroup $S_5 \times S_1$. The corresponding operator $f \mapsto f^{PGL_2(5)}$ is involutory on the level of algebras. This particular operator really takes place at the previously discussed intermediate level, but we implement it using `GaloisGroup` to illustrate the general fine-level approach:

```
sextictwin := function(f)
   G,r,S := GaloisGroup(f);
   H := Subgroups(G:IndexEqual:=6,IsTransitive:=true)[1]`subgroup;
   return GaloisSubgroup(S,H);
end function;
```

Because of the quick-and-dirty method to pick out the right H, this program doesn't work on all sextics; it does work for sextics with Galois group A_6 and S_6.

Exercise 2.5. *How do the factpats of* `f6:=x^6-x-1` *and its twin compare? Confirm, using* `cleanup`*, that the twinning operator is involutory at the level of algebras in this case. Does this one instance really suffice to demonstrate that it is involutory in general?*

The command `sextictwin` models what can be done in general.

Exercise 2.6. *Make tiny changes to* `sextictwin`*, the largest being dropping the* `IsTransitive` *part, to turn it into a much more general program* `res`*. This program should take (f, i, j) as input, and return the j^{th} degree i resolvent of f in* Magma *ordering.*

(The program `res` as described is not ideal when $\text{Gal}(f)$ has more than one conjugacy class of subgroups of index i, because we don't have control over the ordering; this is related to the unsatisfactory aspect of `sextictwin`).

Exercise 2.7. *Take a cleaned-up $11T6 = M_{11}$ polynomial, turn it via your program* `res` *into a cleaned-up $12T272 = M_{11}^t$ polynomial, and then use* `res` *again to go back into the original polynomial.*

The general program `res` essentially supercedes the previous `resk` and `resmid`. For example, Part a of the next exercise gets the interesting output of `resmid` and Part b goes beyond the range of intermediate-level techniques.

Exercise 2.8. *a) Get the smaller degree factor of the degree 462 resolvent $f_{12}^{\text{mid}}(x)$ directly. b) Get the degree 12 twin f_{12}^t of f_{12} (*`cleanup` *does not work quickly here). c) Run* `factpatstats2` *on the pair (f_{12}, f_{12}^t). From the output, do you expect field discriminants to be necessarily the same? d) Compute the two field discriminants.*

3. Exercises for Friday's lecture

Rapidly recognizing fullness. Let $f(x) \in \mathbb{Z}[x]$ be a polynomial with Galois group S_n. How many Frobenius partitions λ_p can one expect to have to compute to prove that the Galois group is S_n?

Proving that the Galois group is indeed S_n by Jordan's method is not optimal. As described in the lecture, λ_p contains a Jordan prime j with probability $\sum_{j \in (n/2, n-2]} \frac{1}{j} \approx \frac{\log 2}{\log n}$. One has to simultaneously use the collected λ_p to ensure that $f(x)$ is irreducible and also that the Galois group is not in A_n. Improvements like Manning's method mentioned in the lecture are also suboptimal.

There are canonical numbers associated with this situation as follows. Let $\mathcal{M}_n$ be the set of conjugacy classes of maximal subgroups in S_n, so that e.g. $\mathcal{M}_5 = \{S_4 \times S_1, S_3 \times S_2, F_5, A_5\}$. Each $M \in \mathcal{M}_n$ has an "imitation factor" $I(M)$, namely the chance that a single λ_p allows M to continue as a possible Galois group. The chance $c(k)$ that computation of k Frobenius elements suffices to prove the Galois group is all of S_n then satisfies

$$c(k) \geq 1 - \sum_{M \in \mathcal{M}_n} I(M)^k.$$

For k large, one can expect near-equality because the remaining terms in the exact inclusion-exclusion expression for $c(k)$ are small.

Exercise 3.1. *Use* Magma *to compute $I(M)$ for all $M \in \mathcal{M}_n$ for say $n \leq 100$. Pursue this situation further. Natural questions include identifying what what type of M gives the largest $I(M)$ and finding good asymptotic expressions for the corresponding $I(M)$. Also it is natural to consider the analog for ambient groups A_n.*

Note that *Magma* gives maximal subgroups as a bare list. The O'Nan-Scott theorem divides this list into six sublists, and thereby adds considerable structure to this situation. (Search on `O'Nan-Scott Cameron` for a good brief introduction).

Computing discriminants of splitting fields. Let Q be say $\mathbb{Q}$ or one of the $\mathbb{Q}_p$. Let $f(x) \in Q[x]$ be a separable polynomial with a splitting field $L = F^{\mathrm{gal}}$. For each G-set X one has a corresponding algebra F_X characterized by the equality $\mathrm{Hom}(F_X, F^{\mathrm{gal}}) = X$ of G-sets. The corresponding discriminants behave multiplicatively: $d(F_{X_1 \coprod X_2}) = d(F_{X_1}) d(F_{X_2})$. The general case reduces to the case of transitive X and a choice of $x \in X$ with stablizer H then identifies X with G/H. In this case, $F_{G/H}$ is the classical fixed field L^H. This formalism lets one compute discriminants of large-degree fields in terms of discriminants of smaller-degree fields.

Exercise 3.2. *For $G = S_3$ and then some other G of your choice, find small index subgroups $H_1, \ldots H_k$ of G and then rational numbers $\alpha_1, \ldots, \alpha_k$ such that $d(L) = d(F_{H_1})^{\alpha_1} \cdots d(F_{H_k})^{\alpha_k}$.*

The character table of G should be very useful.

Specializing three-point covers. Consider Malle's one-parameter family $f_{22}(t, x)$ from the lecture, with discriminant $-2^{284} 11^{253} (t-1)^7 t^{15}$ and generic Galois group M_{22}. Interpreting $t \in \mathbb{Q} - \{0, 1\}$, one has a number algebra $F(t) = \mathbb{Q}[x]/f_{22}(t, x)$. When considering the p-adic behavior of these number algebras it is natural to let t

vary over all of $\mathbb{Q}_p - \{0,1\}$. Both the global and the local aspects of specialization have many points of interest, and the exercise alternates between local and global subparts.

To always stay in the univariate polynomial ring $\mathbb{Z}[x]$ we implement $f_{22}(t,x)$ as a function:

```
f22 := func<t|
  (-4194304*Numerator(t)*(3 - x + x^2)^11
   + Denominator(t)*(32 + 28*x - 12*x^2 + 19*x^3)^2
      *(-164 + 212*x - 119*x^2 + 34*x^3 + 5*x^4)^4)
>;
```

(Other implementations would work too, but need to deal with things such as *Magma*'s refusal to give meaning to `Factorization(1/6)` or even `Factorization(6)` when 6 is typed as a rational number.)

Exercise 3.3. *a) As a simple illustration of local constancy, the number r of real roots of $f_{22}(t,x)$ depends only on the interval $(-\infty,0)$, $(0,1)$, $(1,\infty)$ containing t. What is r in each case?*

b) Do a computer search to find what seems to be all rational t keeping the polynomial discriminant of the form $\pm 2^a 11^b$.

c) The lecture indicated how $\mathrm{ord}_p(d(F(t))$ is an explicit continuous function of the p-adic variable $t \in \mathbb{Q}_p - \{0,1\}$ for $p \not\in \{2,11\}$. Write a formula for this function.

d) Using the results of c, do a longer computer search to find what seems to be all rational t keeping the field discriminant of the form $\pm 2^a 11^b$.

e) Interpolate specializations to make a guess at $\mathrm{ord}_p(d(F(t)))$ as continuous functions on $\mathbb{Q}_p - \{0,1\}$ for the wild primes $p = 2$ and $p = 11$.

f) Find specialization points giving groups besides M_{22} (the example $t = 2401/192$ with Galois group $PGL_2(11)$ and field discriminant $-2^{20}3^{20}11^{19}$ is a good model).

Note: Parts *a*-*f* are interesting things to do for any three-point cover. For many such covers, there is no non-generic specialization. Malle's covers was chosen from among many because the answer to f is more interesting than usual.

Transferring knowledge from G_p to G_∞. Knowledge of any single G_v for a given $f(x) \in \mathbb{Z}[x]$ can be used to facilitate computation of factors of resolvents $f^H(x) \in \mathbb{Z}[x]$. In turn, these factors can be used to compute G_w for other places w.

Exercise 3.4. *For Bosman's polynomial* `f24` *considered in the lecture with complex roots, compute G_∞ as a subgroup of S_{24}.*

To do this exercise, one needs to use root-manipulation commands in *Magma*'s Galois group package beyond what we have talked about.