A MAGMA SOURCE TO COMPUTE THE GENUS OF $X_0(N)/B$, WITH B GENERATED BY ATKIN-LEHNER INVOLUTIONS

FRANCESC BARS

Here we provide Magma code to compute the genus of $X_0^*(N)$ for arbitrary N, moreover we introduce different functions which may be usual for different arithmetic aspects with respect to quotients of $X_0(N)$ by subgroups generated by Atkin-Lehner involutions.

The general setting is: fix $N \geq 2$ an integer and consider the modular curve $X_0(N)$, and consider $W(N)$ the group generated by all the Atkin-Lehner involutions of $X_0(N)$, group of order 2^r where r is the number of different primes that divides N, and fix W a subgroup of $W(N)$ of order 2^s, we recall that $X_0^*(N)$ denotes $X_0(N)/W(N)$, $X_0^+(N) := X_0(N)/ < w_N >$, $X_0^W(N) := X_0(N)/W$. Denote by $g_{X_0(N)}$ the genus of $X_0(N)$ and g_W the genus of $X_0^W(N)$. Then the following formula is well-known [1],

$$2g_{X_0(N)} - 2 = 2^r(2g_{W(N)} - 2) + \sum_{1<d|N} \nu(N,d)$$

where $\nu(N,d)$ denotes the number of fixed points of the involution w_d in $X_0(N)$.

More in general, W are given by certain Atkin-Lehner involutions, and the genus formula for $X_0^W(N)$ is given by

$$2g_{X_0(N)} - 2 = 2^s(2g_W - 2) + \sum_{w_d \in W} \nu(N,d).$$

We introduce different Magma functions: the function $fixedpointsALinvsmall(m,n)$ computes $\nu(m,n)$ when $n \in \{1,2,3,4\}$, the function $fixedpointsALinvbig(m,n)$ computes $\nu(m,n)$ when $n \geq 5$, here we implement Kluit formulae in [2] to compute $\nu(m,n)$, denoted mainly in the Magma source by

```
nu_n
```

the function $generexoN(m)$ computes the genus of $X_0(m)$, and finally the provide ad-hoc Magma sentences to compute the genus of $X_0^*(m)$ with $m = 4 * 255$. The output list the different $\nu(N,n)$, the genus of $X_0(N)$ and $X_0^*(N)$ (named "genusxoNstar"). Of course with very small modifications we could obtain the genus of $X_0(N)/W$ with W a subgroup generated by certain Atkin-Lehner involutions of $X_0(N)$, following the previous formula for computing its genus introduced in the beginning of this note.

MAGMA CODE:

```
fixedpointsALinvsmall:=function(m, n)

if n eq 3 then
        q:=Factorization(Numerator(m/3));
        nu_3:=2;
        for d in [1 .. #q] do
```

F.Bars supported by MTM2016-75980-P.

```
          if q[d][1] eq 2 then
              if q[d][2] gt 2 then
               nu_3:=0;
              else
              end if;
          else
            nu_3:=nu_3*(1+LegendreSymbol(-n,q[d][1]));
          end if;
        end for;
        return nu_3;
else
        if n eq 1 then
        else
            if n eq 2 then
              q:=PrimeDivisors(Numerator(m/n));
              nu_2factorminus1:=1;
              nu_2factorminus2:=1;
              for d in [1 .. #q] do
                 nu_2factorminus1:=nu_2factorminus1*(1+LegendreSymbol(-1,q[d]));
                 nu_2factorminus2:=nu_2factorminus2*(1+LegendreSymbol(-2,q[d]));
              end for;
              return nu_2factorminus1+nu_2factorminus2;
            else
               if n eq 4 then
                 q:=PrimeDivisors(Numerator(m/n));
                 l:=Divisors(Numerator(m/n));
                 nu_4factorminus1:=1; nu_4sumeuler:=0;
                 for d in [1..#q] do
                   nu_4factorminus1:=nu_4factorminus1*(1+LegendreSymbol(-1,q[d]));
                 end for;
                 for dd in [1..#l] do
                    ss:=Numerator(m/(n*l[dd]));
                    ssh:=GCD(l[dd],ss);
                    nu_4sumeuler:=nu_4sumeuler+EulerPhi(ssh);
                 end for;
                 return nu_4factorminus1+nu_4sumeuler;
               end if;
            end if;
         end if;
end if; end function;


fixedpointsALinvbig:=function(m, n)

PD:=PrimeDivisors(Numerator(m/n)); D:=Divisors(Numerator(m/n));

n_mod2:=n mod 2; n_mod4:=n mod 4;
```

```
if n_mod2 eq 1 then
     if n_mod4 eq 3 then
          if 2 in PD then
              if 8 in D then
                  nu_nodd3mod4_8:=2*(ClassNumber(-n)+ClassNumber(-4*n))*(1+KroneckerSymbol(-n,2));
                    if #PD eq 1 then
                    else
                        for p1 in PD do
                         p1_mod2:=p1 mod 2;
                         if p1_mod2 eq 1 then
                           nu_nodd3mod4_8:=nu_nodd3mod4_8*(1+LegendreSymbol(-n,p1));
                         end if;
                        end for;
                    end if;
                    return nu_nodd3mod4_8;
              else
                  if 4 in D then
                     nu_nodd3mod4_4:=
                       (2*ClassNumber(-4*n)+2*(1+KroneckerSymbol(-n,2))*ClassNumber(-n));
                     if #PD eq 1 then
                       return nu_nodd3mod4_4;
                     else
                       for p2 in PD do
                         p2_mod2:=p2 mod 2;
                           if p2_mod2 eq 1 then
                            nu_nodd3mod4_4:=nu_nodd3mod4_4*(1+LegendreSymbol(-n,p2));
                           end if;
                        end for;
                        return nu_nodd3mod4_4;
                     end if;
                  else
                        nu_nodd3mod4_2:=ClassNumber(-4*n)+3*ClassNumber(-n);
                        if #PD eq 1 then
                           return nu_nodd3mod4_2;
                        else
                           for p3 in PD do
                              p3_mod2:=p3 mod 2;
                              if p3_mod2 eq 1 then
                                nu_nodd3mod4_2:=nu_nodd3mod4_2*(1+LegendreSymbol(-n,p3));
                              end if;
                            end for;
                            return nu_nodd3mod4_2;
                        end if;
                  end if;
              end if;
          else
              if #PD eq 0 then
                nu_nodd3mod4_no2:=ClassNumber(-n)+ClassNumber(-4*n);
                return nu_nodd3mod4_no2;
```

```
            else
              nu_nodd3mod4_no2:=ClassNumber(-n)+ClassNumber(-4*n);
               for j in [1..#PD] do
                 nu_nodd3mod4_no2:=nu_nodd3mod4_no2*(1+LegendreSymbol(-n,PD[j]));
               end for;
               return nu_nodd3mod4_no2;
            end if;
        end if;
   else
        if #PD eq 0 then
           nu_nodd1mod4:=ClassNumber(-4*n);
           return nu_nodd1mod4;
        else
           if 2 in PD then
                if 4 in D then
                  return 0;
                else
                   nu_nodd1mod4_2:=ClassNumber(-4*n);
                   PD2:=PrimeDivisors(Numerator(m/(2*n)));
                    for k in [1..#PD2] do
                     nu_nodd1mod4_2:=nu_nodd1mod4_2*(1+LegendreSymbol(-n,PD2[k]));
                    end for;
                 return nu_nodd1mod4_2;
                end if;
           else
               nu_nodd1mod4_no2:=ClassNumber(-4*n);
                  for i in [1 .. #PD] do
                   nu_nodd1mod4_no2:=nu_nodd1mod4_no2*(1+LegendreSymbol(-n,PD[i]));
                  end for;
                return nu_nodd1mod4_no2;
           end if;
        end if;
   end if;
 else
    nu_neven:=ClassNumber(-4*n);
    if #PD eq 0 then
      return nu_neven;
    else
      for u in [1 .. #PD] do
        nu_neven:=nu_neven*(1+LegendreSymbol(-n,PD[u]));
      end for;
      return nu_neven;
    end if;
 end if;
end function;
```

```
generexoN:=function(b)

  m:=PrimeDivisors(b);
  l:=Divisors(b);
  factor_b:=Factorization(b);
 psiEulerindex:=b;
 order4elliptic:=1;
 order3elliptic:=1;
 cusps:=0;

 for x in [1 .. #m] do
 psiEulerindex:=psiEulerindex*(1+1/m[x]);
 end for;

   for y in [1..#m] do
     if factor_b[y][1] eq 2 then
         order3elliptic:=0;
        if factor_b[y][2] gt 1 then
          order4elliptic:=0;
        end if;
     else
        if factor_b[y][1] eq 3 then
            if factor_b[y][2] gt 1 then
              order3elliptic:=0;
              order4elliptic:=order4elliptic*(1+LegendreSymbol(-1,factor_b[y][1]));
            else
              order3elliptic:=order3elliptic*(1+LegendreSymbol(-3,factor_b[y][1]));
              order4elliptic:=order4elliptic*(1+LegendreSymbol(-1,factor_b[y][1]));
            end if;
        else
           order4elliptic:=order4elliptic*(1+LegendreSymbol(-1,factor_b[y][1]));
           order3elliptic:=order3elliptic*(1+LegendreSymbol(-3,factor_b[y][1]));
        end if;
     end if;
   end for;

 for a in [1 .. #l] do
   n1:=Numerator(b/l[a]);
   t1:=GCD(l[a],n1);
   cusps:=cusps+EulerPhi(t1);
  end for;

genus:=1+(psiEulerindex/12)-(order4elliptic/4)-(order3elliptic/3)-(cusps/2);
return genus;
 end function;
```

The ad-hoc subroutine for computing the genus of $X_0^*(N)$ with $N = 4 * 255$

```
N:=4*255; FixedpointsALinvolutions:=[* *]; Dd:=Divisors(N);

for i in [1..#Dd] do
   u:=GCD(Dd[i],Numerator(N/Dd[i]));
   if u eq 1 then
      if Dd[i] eq 1 then
      else
         if Dd[i] gt 4 then
            nu_Ddi:=fixedpointsALinvbig(N,Dd[i]);
            FixedpointsALinvolutions:=Append(FixedpointsALinvolutions,nu_Ddi);
         else
            nu_Ddi:=fixedpointsALinvsmall(N,Dd[i]);
            FixedpointsALinvolutions:=Append(FixedpointsALinvolutions,nu_Ddi);
         end if;
      end if;
    end if;
end for;

CountAllFixedPointsALinvolutions:=0;

for u in FixedpointsALinvolutions do
  CountAllFixedPointsALinvolutions:=CountAllFixedPointsALinvolutions+u;
end for;

 genusxoN:=generexoN(N);
 genusxoNstar:=1+2^(-#PrimeDivisors(N))*(genusxoN-1-(CountAllFixedPointsALinvolutions/2));
 genusxoNstar;
```

The Output of running the above source with $N = 4 * 255$ is 8.

Acknowledgements. We thank referees for their comments, especially those that have contributed to improve the presentation of the Magma source presented here. I am very grateful to Josep González for his constant enthusiasm in the study of modular curves and their computational aspects.

References

[1] J. González and J.-C. Lario. Rational and elliptic parametrizations of **Q**-curves. *J. Number Theory*, 72(1):13–31, 1998.
[2] P. G. Kluit. On the normalizer of $\Gamma_0(N)$. pages 239–246. Lecture Notes in Math., Vol. 601, 1977.

• Francesc Bars Cortina

Departament Matemàtiques, Edif. C, Universitat Autònoma de Barcelona, 08193 Bellaterra, Catalonia
E-mail address: francesc@mat.uab.cat