Modular Forms Project: Computing Expansions at Cusps

Henri Cohen,
Université de Bordeaux, Institut de Mathématiques de Bordeaux,
351 Cours de la Libération, 33405 TALENCE Cedex, FRANCE

July 8, 2018

1 Answers and Programs

Exercise 1.

(1). Let as usual $T = \left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)$. If $\gamma = \left(\begin{smallmatrix}A&B\\C&D\end{smallmatrix}\right) \in \Gamma$, we compute that

$$\gamma T^m \gamma^{-1} = \begin{pmatrix} 1-mAC & mA^2 \\ -mC^2 & 1+mAC \end{pmatrix} .$$

This belongs to $\Gamma_0(N)$ if and only if $N \mid mC^2$, hence if and only if $w_0 \mid m$ with $w_0 = N/\gcd(C^2, N)$. Since $F \in M_k(\Gamma_0(N), \chi)$, we thus have $F|_k\gamma T^{w_0 n}\gamma^{-1} = \chi(1+ACw_0n)F$, or

$$(F|_k\gamma)(\tau + w_0 n) = \chi(1+ACw_0n)(F|_k\gamma)(\tau) .$$

If we choose $n = H = w_1/w_0$, we note that

$$1 + ACw_0n = 1 + ACw_1 = 1 + A(C/\gcd(C,N))N \equiv 1 \pmod{N} ,$$

so $\chi(1+ACw_0n) = 1$ and $(F|_k\gamma)(\tau + w_1) = (F|_k\gamma)(\tau)$.

(2). It follows from the trivial equality $(\gamma T^{w_0}\gamma^{-1})^n = (\gamma T^{w_0 n}\gamma^{-1})$ that for all n we have $\chi(1+ACw_0n) = \chi(1+ACw_0)^n$. Applying this to $n = H = w_1/w_0$ shows that $\chi(1+ACw_0)$ is an Hth root of unity, hence uniquely of the form $e^{2\pi i\alpha/H}$ for some integer α such that $0 \le \alpha < H$. It is clear that if $H = 1$ we have $\alpha = 0$, but also if χ is a trivial character.

(3). If we change τ into $\tau + w_0$ in the expansion of $F|_k\gamma$ we have on the one hand

$$(F|_k\gamma)(\tau + w_0) = \sum_{n\ge 0} a_\gamma(n) e^{2\pi i n/H} q^{n/w_1} ,$$

and on the other hand

$$(F|_k\gamma)(\tau + w_0) = e^{2\pi i\alpha/H} F|_k\gamma(\tau) .$$

Identifying coefficients of q^{n/w_1} gives

$$a_\gamma(n)(e^{2\pi i n/H} - e^{2\pi i\alpha/H}) = 0 ,$$

which is equivalent to $a_\gamma(n) = 0$ when $n \not\equiv \alpha \pmod H$. It follows that we can write

$$(F|_\gamma)(\tau) = q^{\alpha/w_1} \sum_{n\ge 0} b(n) q^{n/w_0}$$

with $b(n) = a_\gamma(nH + \alpha)$.

Exercise 2.

(1). Multiplying by a common denominator M of A, B, C, and D, we can assume that $\gamma \in M_2^+(\mathbb{Z})$ (we will simply divide the triangular matrix by M at the end). We want to find $\delta = \left(\begin{smallmatrix} U & V \\ W & X \end{smallmatrix}\right) \in \Gamma$ and a, b, and d such that $\gamma = \delta \left(\begin{smallmatrix} a & b \\ 0 & d \end{smallmatrix}\right)$. This implies first that $Ua = A$ and $Wa = C$, and since U and W must be coprime, up to sign this has the only solution $a = g := \gcd(A, C)$, $U = A/g$, $W = C/g$.

Thus U and W are indeed coprime, hence by Euclid there exist (nonunique) V and X such that $UX - VW = 1$. The other condition to be verified is $\delta(b, d)^t = (B, D)^t$, and since $\delta \in \Gamma$ this gives

$$\begin{pmatrix} b \\ d \end{pmatrix} = \begin{pmatrix} X & -V \\ -W & U \end{pmatrix} \begin{pmatrix} B \\ D \end{pmatrix} ,$$

in other words $b = BX - DV$, $d = DU - BW = (AD - BC)/g$.

This last equality shows that d does not depend on the choice of V and X (which is trivial since the determinant of the triangular matrix must be equal to $AD - BC$), so only b does. If (V, X) is changed into $(V + \lambda U, X + \lambda W)$ then b is changed into $b + \lambda(BW - DU) = b - \lambda d$, hence $\left(\begin{smallmatrix} a & b \\ 0 & d \end{smallmatrix}\right)$ into $T^{-\lambda} \left(\begin{smallmatrix} a & b \\ 0 & d \end{smallmatrix}\right)$ and δ into δT^λ.

As mentioned above, we finish by dividing a, b, and d by M.

(2) The matrix $\left(\begin{smallmatrix} a & b \\ 0 & d \end{smallmatrix}\right)$ corresponds to the Möbius transformation $(a\tau + b)/d = (a/d)\tau + (b/d)$. Thus, if $G := F|_k\delta = \sum_{n\ge 0} a_\delta(n) q^{n/w_1}$ we have

$$G(\tau + b/d) = \sum_{n\ge 0} a_\delta(n) e^{2\pi i (b/d) n/w_1} q^{n/w_1} ,$$

hence

$$G((a/d)\tau + b/d) = \sum_{n\ge 0} a_\delta(n) e^{2\pi i (b/d) n/w_1} q^{an/(dw_1)} .$$

Exercise 3.

(1). Note that `Pari/GP` already has a built-in command `mfslashexpansion` to compute $F|_k\gamma$. Of course we want to implement Collins's algorithm, so we do not want to use `Pari/GP`'s command. However, handling general Dirichlet characters, although not difficult is slightly tedious, so we will simply use `mfslashexpansion` to compute the integer α such that $0 \le \alpha < H$ which is needed in the expansions, which is not really cheating. We could of course also restrict the algorithm to the known cases where $\alpha = 0$, i.e., χ trivial or $H = \gcd(C^2, N)/\gcd(C, N) = 1$.

A possible program is as follows:

```
/* Collins's algorithm for cusp expansions. */
collins(mf,F,ga,K0,G0=1)=
```

```
{
  my(N,k,A,B,C,D,E,K,G,L,w0,w1,H,al,T,im);
  my(Y,S,Q,R,M,MS,X);
  [N,k]=mfparams(mf); [A,C]=ga[,1]; [B,D]=ga[,2];
  E=bitprecision(1.)*0.69315;
  if (K0*G0<E,G=G0;K=ceil(E/G),K=K0;G=E/K);
  L=2*K; w0=N/gcd(C*C,N); w1=N/gcd(C,N); H=w1/w0;
  mfslashexpansion(mf,F,ga,0,0,&params); al=params[1]*w1;
/* only used to compute alpha. */
  T=vector(L); /* vector of random tau's. */
  im=G*w0*I/(2*Pi); /* desired imaginary part */
  for (i=1,L,T[i]=(random(1.)-1/2)*w0+im-D/C);
/* random real parts in the interval [-D/C-w0/2,-D/C+w0/2]. */
  Y=vectorv(L,i,(C*T[i]+D)^(-k)*mfeval(mf,F,(A*T[i]+B)/(C*T[i]+D)));
  S=vector(L,i,exp(2*Pi*I*T[i]/w1));
  Q=vector(L,i,S[i]^H); R=vector(L,i,S[i]^al);
  M=matrix(L,K,i,j,R[i]*Q[i]^(j-1));
  MS=conj(M~); X=matsolve(MS*M,MS*Y);
  X[1..K0];
}
```

In the above program there is an additional cheat: we use the `Pari/GP` function `mfeval` instead of computing sufficiently many terms of the Fourier expansion of F at infinity and summing the corresponding series.

(2). For E_4 we can simply write

```
E4=mfEk(4); mf=mfinit(E4);
C=collins(mf,E4,[0,-1;1,0],20)
CI=bestappr(C,10^4)
CI~==mfcoefs(E4,19)
norml2(C-CI)
```

We first obtain some complex numbers which are clearly close to rational numbers (in fact to integers here). We then compute the best rational approximations to these complex numbers with denominator less than 10^4, and we recognize the coefficients of E_4, which we can check without reading all coefficients by the command `CI~==mfcoefs(E4,19)` (note that `CI` is a column vector, so that it must be transposed, and that `mfcoefs(F,L)` gives coefficients $a(0)$ up to $a(L)$, so to obtain 20 coefficients we must choose $L = 19$).

Finally, to check how close the least squares approximation was, we compute the square of the L^2 norm of the difference of `C` and `CI` (note that contrary to what its name could imply, the `norml2` command gives the *square* of the L^2 norm). We obtain $1.076\cdots E - 45$, so the values of the coefficients have approximately 22 correct decimal digits, the computation being done with the default accuracy of 38.

A similar program (replacing the first line by `D=mfDelta(); mf=mfinit(D,1);`) also gives correct results, but since the weight is much larger, we only obtain 20 correct decimal digits.

(3). (a). To initialize the computation, as required we first write:

```
mf=mfinit([6,4],0); F=mfeigenbasis(mf)[1];
ga=[1,-1;3,-2]; CF=mfcoefs(F,19)
C=collins(mf,F,ga,20);
CI=bestappr(C,10^4)
CI~==CF/4
norml2(C-CI)
```

which first outputs

[0, 1, -2, -3, 4, 6, 6, -16, -8, 9, -12, 12, -12, 38, 32, -18, 16, -126, -18, 20]

as the first 20 Fourier coefficients of F, then the complex numbers computed by the program are approximated by

[0, 1/4, -1/2, -3/4, 1, 3/2, 3/2, -4, -2, 9/4, -3, 3, -3, 19/2, 8, -9/2, 4, -63/2, -9/2, 5]

The next command checks that this is indeed $1/4$ times the coefficients of F, and the last command (which outputs $7.088E-58$) shows that the algorithm gives values which are correct to 29 decimal digits.

(3). (b). To check the eta product without too much using `Pari/GP`'s functionalities, the simplest is to do as follows:

```
G=mffrometaquo([1,2;2,2;3,2;6,2]); par=mfparams(G)[1..2]
st=mfsturm(par)
mfcoefs(F,st)==mfcoefs(G,st)
```

The first line outputs `[6, 4]`, which tells us that $G \in M_4(\Gamma_0(6))$, the second line gives the *sturm bound*, here 5, and the last command checks that the first `st+1` (counting the constant term) coefficients of F and G are equal, thus *proving* that $F = G$. The shorter `Pari/GP` command `mfisequal(F,G)` does exactly that.

(3). (c). The builtin `mfslashexpansion(mf,F,ga,19,1)` command outputs

[0, 1/4, -1/2, -3/4, 1, 3/2, 3/2, -4, -2, 9/4, -3, 3, -3, 19/2, 8, -9/2, 4, -63/2, -9/2, 5]

equal to what we found above.

(4). We write the following simple programs:

```
tstg(G0)=my(C=collins(mf,F,ga,20,G0)); norml2(C-bestappr(C,10^4));
LG=[1/2,1,2,3,4];
for(i=1,5,\
  gettime(); er=tstg(LG[i]); r=gettime();\
  print(LG[i],": ",er," (time = ",r,")"));
```

This outputs:

```
1/2: 4.732342396833962897 E-64 (time = 6757)
1: 2.763570679490571905 E-57 (time = 945)
2: 2.1521231746487435604 E-44 (time = 232)
3: 2.135926501725837551147532356175982323 1 E-28 (time = 132)
4: 1.267609544466537838980443691211134876 8 E-12 (time = 64)
```

This output shows the (logical) fact that as G_0 increases the time becomes considerably shorter, but the accuracy considerably worse (only 6 correct decimal digits for $G_0 = 4$).

(5). (a). As required, we write

```
mf=mfinit([27,4],0); F=mfeigenbasis(mf)[2];
ga=[1,-1;3,-2]; CF=mfcoefs(F,19)
C=collins(mf,F,ga,20);
```

The Fourier coefficients of F are thus
[0, 1, -3, 0, 1, -15, 0, -25, 21, 0, 45, 15, 0, 20, 75, 0, -71, -72, 0, 2]

We do not print the output of the `collins` command since they are complicated complex numbers. Instead, as suggested we write the following programs which generalises the `bestappr` function to cyclotomic fields by using `lindep` instead:

```
/* Find z in $\Q(e^{2\pi i/N})$. */
cycloapprox(z,N)=
{
  my(pn=eulerphi(N)-1,P=powers(exp(2*I*Pi/N),pn),T);
  T=powers(t,pn);
  L=lindep(concat(-z,P));
  return (T*(L[2..#L]/L[1]));
}
cyclovecapprox(V,N)=apply(z->cycloapprox(z,N),V);
```

If we execute `cyclovecapprox(C,9)`, we obtain

```
[0, -1/9*t^4, -1/3*t^5 - 1/3*t^2, 0, 1/9*t^4 + 1/9*t, 5/3*t^2,\
 0, 25/9*t, -7/3*t^5, 0, -5*t^4, 5/3*t^5 + 5/3*t^2, 0,\
 20/9*t^4 + 20/9*t, -25/3*t^2, 0, 71/9*t, 8*t^5, 0, -2/9*t^4]
```

(output edited for clarity), where in the above t is to be interpreted as $e^{2\pi i/9}$. In fact, since $t^3 + 1 = -t^6$, it is easy to see that all the coefficients are rational multiples of 9th roots of unity.

(5). (b). Here we must write a generalization of the program given in the previous question. For this, we simply replace the `tstg` program by:

```
tstg(G0)=
{
  my(C=collins(mf,F,ga,20,G0));
  CI=cyclovecapprox(C,9);
  norml2(C-subst(CI,t,exp(2*Pi*I/9)));
}
```

and the rest is the same. The output will now be

```
1/2: 8.111206046459227532 E-64 (time = 10292)
1: 1.7354873000414302779 E-57 (time = 2689)
2: 3.3933846210407921667 E-46 (time = 828)
3: 1.67042101689085238950655129082995977 48 E-46 (time = 585)
4: 1.46313427958427692047045571098035527 15 E-45 (time = 617)
```

Here there is much less loss of accuracy when G_0 increases.

(5). (c). Typing `CJ=mfslashexpansion(mf,F,ga,19,1)` of course gives the same expansion, except that the coefficients are expressed as `POLMODS` in the field defined by $t^6 + t^3 + 1 = 0$, i.e., the ninth cyclotomic field, so to obtain exactly the same output one simply writes `lift(CJ)`.