

Modular forms, modular symbols

(PARI-GP version 2.15.1)

Modular Forms

Dirichlet characters

Characters are encoded in three different ways:

- a `t_INT` $D \equiv 0, 1 \pmod 4$: the quadratic character (D/\cdot) ;
- a `t_INTMOD` $\text{Mod}(m, q)$, $m \in (\mathbf{Z}/q)^*$ using a canonical bijection with the dual group (the Conrey character $\chi_q(m, \cdot)$);
- a pair $[G, chi]$, where $G = znstar(q, 1)$ encodes $(\mathbf{Z}/q\mathbf{Z})^* = \sum_{j \leq k} (\mathbf{Z}/d_j\mathbf{Z}) \cdot g_j$ and the vector $chi = [c_1, \dots, c_k]$ encodes the character such that $\chi(g_j) = e(c_j/d_j)$.

```

initialize G = (Z/qZ)*          G = znstar(q, 1)
convert datum D to [G, chi]    znchar(D)
Galois orbits of Dirichlet characters  chargalois(G)

```

Spaces of modular forms

Arguments of the form $[N, k, \chi]$ give the level weight and nebentypus χ ; χ can be omitted: $[N, k]$ means trivial χ .

```

initialize S_k^new(Gamma_0(N), chi)  mfininit([N, k, chi], 0)
initialize S_k(Gamma_0(N), chi)      mfininit([N, k, chi], 1)
initialize S_k^old(Gamma_0(N), chi)  mfininit([N, k, chi], 2)
initialize E_k(Gamma_0(N), chi)      mfininit([N, k, chi], 3)
initialize M_k(Gamma_0(N), chi)      mfininit([N, k, chi])
find eigenforms                      mfsplit(M)
statistics on self-growing caches     getcache()

```

We let $M = \text{mfininit}(\dots)$ denote a modular space.

```

describe the space M              mfdescribe(M)
recover (N, k, chi)              mfparams(M)
... the space identifier (0 to 4)  mfspace(M)
... the dimension of M over C     mfdim(M)
... a C-basis (f_i) of M         mfbasis(M)
... a basis (F_j) of eigenforms  mfeigenbasis(M)
... polynomials defining Q(chi)(F_j)/Q(chi)  mffields(M)

```

```

matrix of Hecke operator T_n on (f_i)  mfheckemat(M, n)
eigenvalues of w_Q                   mfkineigenvalues(M, Q)
basis of period polynomials for weight k  mferiodpolbasis(k)
basis of the Kohnen +-space          mfkohnenbasis(M)
... new space and eigenforms         mfkohneneigenbasis(M, b)
isomorphism S_k^+(4N, chi) -> S_{2k-1}(N, chi^2)  mfkohnenbijection(M)

```

Useful data can also be obtained a priori, without computing a complete modular space:

```

dimension of S_k^new(Gamma_0(N), chi)  mfdim([N, k, chi])
dimension of S_k(Gamma_0(N), chi)      mfdim([N, k, chi], 1)
dimension of S_k^old(Gamma_0(N), chi)  mfdim([N, k, chi], 2)
dimension of M_k(Gamma_0(N), chi)      mfdim([N, k, chi], 3)
dimension of E_k(Gamma_0(N), chi)      mfdim([N, k, chi], 4)
Sturm's bound for M_k(Gamma_0(N), chi) mfsturm(N, k)
Gamma_0(N) cosets                    mfcosets(N)
list of right Gamma_0(N) cosets
identify coset a matrix belongs to

```

Cusps

```

a cusp is given by a rational number or oo.
lists of cusps of Gamma_0(N)          mfcusps(N)
number of cusps of Gamma_0(N)         mfnnumcusps(N)
width of cusp c of Gamma_0(N)         mfcuspwidth(N, c)
is cusp c regular for M_k(Gamma_0(N), chi)?  mfcuspisregular([N, k, chi], c)

```

Create an individual modular form

Besides `mfbasis` and `mfeigenbasis`, an individual modular form can be identified by a few coefficients.

```

modular form from coefficients        mftobasis(mf, vec)

There are also many predefined ones:
Eisenstein series E_k on Sl_2(Z)     mfEk(k)
Eisenstein-Hurwitz series on Gamma_0(4)  mfEH(k)
unary theta function (for character psi)  mfTheta({psi})
Ramanujan's Delta                    mfDelta()
E_k(x)                                mfeisenstein(k, chi)
E_k(x_1, x_2)                         mfeisenstein(k, chi_1, chi_2)
eta quotient \prod_i eta(a_{i,1} \cdot z)^{a_{i,2}}  mffrometaquo(a)
newform attached to ell. curve E/Q    mffromell(E)
identify an L-function as an eigenform  mffromlfun(L)
theta function attached to Q > 0      mffromqt(Q)
trace form in S_k^new(Gamma_0(N), chi)  mftraceform([N, k, chi])
trace form in S_k(Gamma_0(N), chi)     mfttraceform([N, k, chi], 1)

```

Operations on modular forms

```

In this section, f, g and the F[i] are modular forms
f x g                                  mfmul(f, g)
f/g                                    mfdiv(f, g)
f^n                                    mfpow(f, n)
f(q)/q^v                               mfmshift(f, v)
\sum_{i \leq k} lambda_i F[i], L = [lambda_1, ..., lambda_k]  mflinear(F, L)
f = g?                                 mfishequal(f, g)
expanding operator B_d(f)             mfbfd(f, d)
Hecke operator T_n f                  mfhecke(mf, f, n)
initialize Atkin-Lehner operator w_Q  mfatkininit(mf, Q)
... apply w_Q to f                    mfatkin(w_Q, f)
twist by the quadratic char (D/\cdot)  mftwist(f, D)
derivative wrt. q \cdot d/dq          mfdderiv(f)
see f over an absolute field          mfreoltoabs(f)
Serre derivative (q \cdot d/dq - k/12) E_2 f  mfdderivE2(f)
Rankin-Cohen bracket [f, g]_n        mfbracket(f, g, n)
Shimura lift of f for discriminant D  mfmshimura(mf, f, D)

```

Properties of modular forms

```

In this section, f = \sum_n f_n q^n is a modular form in some space M
with parameters N, k, chi.
describe the form f                    mfdescribe(f)
(N, k, chi) for form f                 mfparams(f)
the space identifier (0 to 4) for f     mfspace(mf, f)
[f_0, ..., f_n]                        mfcocoeffs(f, n)
f_n                                     mfcoef(f, n)
is f a CM form?                        mfishCM(f)
is f an eta quotient?                  mfishetaquo(f)
Galois rep. attached to all (1, chi) eigenforms  mfgaloistype(M)
... single eigenform                   mfgaloistype(M, F)
... as a polynomial fixed by Ker rho_F  mfgaloisprojrep(M, F)
decompose f on mfbasis(M)              mftobasis(M, f)
smallest level on which f is defined   mfconductor(M, f)
decompose f on \oplus S_k^new(Gamma_0(d)), d | N  mftnew(M, f)
valuation of f at cusp c                mfcuspval(M, f, c)
expansion at \infty of f |_k gamma     mflashexpansion(M, f, gamma, n)
n-Taylor expansion of f at i           mftaylor(f, n)
all rational eigenforms matching criteria  mfeigensearch
... forms matching criteria            mfsearch

```

Forms embedded into C

Given a modular form f in $M_k(\Gamma_0(N), \chi)$ its field of definition $Q(f)$ has $n = [Q(f) : Q(\chi)]$ embeddings into the complex numbers. If $n = 1$, the following functions return a single answer, attached to the canonical embedding of f in $\mathbb{C}[[q]]$; else a vector of n results, corresponding to the n conjugates of f .

```

complex embeddings of Q(f)             mfembed(f)
... embed coefs of f                  mfembed(f, v)
evaluate f at tau in H                 mfeval(f, tau)
L-function attached to f               lfunmf(mf, f)
... eigenforms of new space M          lfunmf(M)

```

Periods and symbols

```

The functions in this section depend on [Q(f) : Q(chi)] as above.
initialize symbol fs attached to f      mfsymbol(M, f)
evaluate symbol fs on path p            mfsymboleval(fs, p)
Pettersson product of f and g          mfpetersson(fs, gs)
period polynomial of form f            mfperiodpol(M, fs)
period polynomials for eigensymbol FS   mfmanin(FS)

```

Modular Symbols

Let $G = \Gamma_0(N)$, $V_k = \mathbf{Q}[X, Y]_{k-2}$, $L_k = \mathbf{Z}[X, Y]_{k-2}$ and $\Delta = \text{Div}^0(\mathbf{P}^1(\mathbf{Q}))$. An element of Δ is a *path* between cusps of $X_0(N)$ via the identification $[b] - [a] \rightarrow$ path from a to b , coded by the pair $[a, b]$ where a, b are rationals or $\infty = (1 : 0)$.

Let $\mathbf{M}_k(G) = \text{Hom}_G(\Delta, V_k) \simeq H_c^1(X_0(N), V_k)$; an element of $\mathbf{M}_k(G)$ is a V_k -valued *modular symbol*. There is a natural decomposition $\mathbf{M}_k(G) = \mathbf{M}_k(G)^+ \oplus \mathbf{M}_k(G)^-$ under the action of the $*$ involution, induced by complex conjugation. The `msinit` function computes either \mathbf{M}_k ($\varepsilon = 0$) or its \pm -parts ($\varepsilon = \pm 1$) and fixes a minimal set of $\mathbf{Z}[G]$ -generators (g_i) of Δ .

```

initialize M = M_k(Gamma_0(N))^epsilon  msinit(N, k, {epsilon = 0})
the level M                             msgetlevel(M)
the weight k                             msgetweight(M)
the sign epsilon                         msgetsign(M)
Farey symbol attached to G              mspolygon(M)
... attached to H < G                   msfarey(F, inH)
H \ G and right G-action                 mscosets(genG, inH)

```

```

Z[G]-generators (g_i) and relations for Delta  mspathgens(M)
decompose p = [a, b] on the (g_i)           mspathlog(M, p)

```

Create a symbol

```

Eisenstein symbol attached to cusp c       msfromcusp(M, c)
cuspidal symbol attached to E/Q           msfromell(E)
symbol having given Hecke eigenvalues     msfromhecke(M, v, {H})
is s a symbol?                            msissymbol(M, s)

```

Operations on symbols

```

the list of all s(g_i)                   mseval(M, s)
evaluate symbol s on path p = [a, b]      mseval(M, s, p)
Pettersson product of s and t            mspetersson(M, s, t)

```

Operators on subspaces

```

An operator is given by a matrix of a fixed Q-basis. H, if given, is
a stable Q-subspace of M_k(G): operator is restricted to H.
matrix of Hecke operator T_p or U_p      mshecke(M, p, {H})
matrix of Atkin-Lehner w_Q               msatkinlehner(M, Q{H})
matrix of the * involution               msstar(M, {H})

```

Subspaces

A subspace is given by a structure allowing quick projection and restriction of linear operators. Its first component is a matrix with integer coefficients whose columns form a \mathbf{Q} -basis. If H is a Hecke-stable subspace of $M_k(G)^+$ or $M_k(G)^-$, it can be split into a direct sum of Hecke-simple subspaces. To a simple subspace corresponds a single normalized newform $\sum_n a_n q^n$.

cuspidal subspace $S_k(G)^\varepsilon$	<code>mscuspidal(M)</code>
Eisenstein subspace $E_k(G)^\varepsilon$	<code>mseisenstein(M)</code>
new part of $S_k(G)^\varepsilon$	<code>msnew(M)</code>
split H into simple subspaces (of $\dim \leq d$)	<code>mssplit(M, H, {d})</code>
dimension of a subspace	<code>msdim(M)</code>
(a_1, \dots, a_B) for attached newform	<code>msqexpansion(M, H, {B})</code>
\mathbf{Z} -structure from $H^1(G, L_k)$ on subspace A	<code>mslattice(M, A)</code>

Overconvergent symbols and p -adic L functions

Let M be a full modular symbol space given by `msinit` and p be a prime. To a classical modular symbol ϕ of level N ($v_p(N) \leq 1$), which is an eigenvector for T_p with nonzero eigenvalue a_p , we can attach a p -adic L -function L_p . The function L_p is defined on continuous characters of $\text{Gal}(\mathbf{Q}(\mu_{p^\infty})/\mathbf{Q})$; in GP we allow characters $\langle \chi \rangle^{s_1} \tau^{s_2}$, where (s_1, s_2) are integers, τ is the Teichmüller character and χ is the cyclotomic character.

The symbol ϕ can be lifted to an *overconvergent* symbol Φ , taking values in spaces of p -adic distributions (represented in GP by a list of moments modulo p^n).

`mspadicinit` precomputes data used to lift symbols. If *flag* is given, it speeds up the computation by assuming that $v_p(a_p) = 0$ if *flag* = 0 (fastest), and that $v_p(a_p) \geq \textit{flag}$ otherwise (faster as *flag* increases).

`mspadicmoments` computes distributions mu attached to Φ allowing to compute L_p to high accuracy.

initialize Mp to lift symbols	<code>mspadicinit(M, p, n, {flag})</code>
lift symbol ϕ	<code>mstooms(Mp, phi)</code>
eval overconvergent symbol Φ on path p	<code>msomseval(Mp, Phi, p)</code>
mu for p -adic L -functions	<code>mspadicmoments(Mp, S, {D = 1})</code>
$L_p^{(r)}(\chi^s)$, $s = [s_1, s_2]$	<code>mspadicL(mu, {s = 0}, {r = 0})</code>
$\hat{L}_p(\tau^i)(x)$	<code>mspadicseries(mu, {i = 0})</code>

Based on an earlier version by Joseph H. Silverman

August 2022 v2.38. Copyright © 2022 K. Belabas

Permission is granted to make and distribute copies of this card provided the copyright and this permission notice are preserved on all copies.

Send comments and corrections to Karim.Belabas@math.u-bordeaux.fr