

Modular forms, modular symbols

(PARI-GP version 2.15.5)

Modular Forms

Dirichlet characters

Characters are encoded in three different ways:

- a `t_INT` $D \equiv 0, 1 \bmod 4$: 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 $[G, \text{chi}]$, where $G = \text{znstar}(q, 1)$ encodes $(\mathbf{Z}/q\mathbf{Z})^* = \sum_{j \leq k} (\mathbf{Z}/d_j\mathbf{Z}) \cdot g_j$ and the vector $\text{chi} = [c_1, \dots, c_k]$ encodes the character such that $\chi(g_j) = e(c_j/d_j)$.

initialize $G = (\mathbf{Z}/q\mathbf{Z})^*$ `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 $S_k^{\text{new}}(\Gamma_0(N), \chi)$ `mfinit([N, k, \chi], 0)`
initialize $S_k(\Gamma_0(N), \chi)$ `mfinit([N, k, \chi], 1)`
initialize $S_k^{\text{old}}(\Gamma_0(N), \chi)$ `mfinit([N, k, \chi], 2)`
initialize $E_k(\Gamma_0(N), \chi)$ `mfinit([N, k, \chi], 3)`
initialize $M_k(\Gamma_0(N), \chi)$ `mfinit([N, k, \chi])`
find eigenforms `mfsplit(M)`
statistics on self-growing caches `getcache()`

We let $M = \text{mfinit}(\dots)$ denote a modular space.
describe the space M `mfdescribe(M)`
recover (N, k, χ) `mfparams(M)`
... the space identifier (0 to 4) `mfspace(M)`
... the dimension of M over \mathbf{C} `mfdim(M)`
... a \mathbf{C} -basis (f_i) of M `mfbasis(M)`
... a basis (F_j) of eigenforms `mfeigenbasis(M)`
... polynomials defining $\mathbf{Q}(\chi)/(F_j)/\mathbf{Q}(\chi)$ `mffields(M)`

matrix of Hecke operator T_n on (f_i) `mfheckemat(M, n)`
eigenvalues of w_Q `mfatkineigenvalues(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 $S_k^+(4N, \chi) \rightarrow S_{2k-1}(N, \chi^2)$ `mfkohnenbijection(M)`

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

dimension of $S_k^{\text{new}}(\Gamma_0(N), \chi)$ `mfdim([N, k, \chi])`
dimension of $S_k(\Gamma_0(N), \chi)$ `mfdim([N, k, \chi], 1)`
dimension of $S_k^{\text{old}}(\Gamma_0(N), \chi)$ `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**
list of right $\Gamma_0(N)$ cosets `mfcosets(N)`
identify coset a matrix belongs to `mftocoset`

Cusps

a cusp is given by a rational number or ∞ .
lists of cusps of $\Gamma_0(N)$ `mfcusps(N)`
number of cusps of $\Gamma_0(N)$ `mfnumcusps(N)`
width of cusp c of $\Gamma_0(N)$ `mfcuspswidth(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 $Sl_2(\mathbf{Z})$ `mfEk(k)`
Eisenstein-Hurwitz series on $\Gamma_0(4)$ `mfEH(k)`
unary θ function (for character ψ) `mfTheta({\psi})`
Ramanujan's Δ `mfDelta()`
 $E_k(\chi)$ `mfeisenstein(k, \chi)`
 $E_k(\chi_1, \chi_2)$ `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/\mathbf{Q} `mffromell(E)`
identify an L -function as a eigenform `mffromlfun(L)`
 θ function attached to $Q > 0$ `mffromqt(Q)`
trace form in $S_k^{\text{new}}(\Gamma_0(N), \chi)$ `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
 $f \times g$ `mfmul(f, g)`
 f/g `mfddiv(f, g)`
 f^n `mfpow(f, n)`
 $f(q)/q^v$ `mfshift(f, v)`
 $\sum_{i \leq k} \lambda_i F[i]$, $L = [\lambda_1, \dots, \lambda_k]$ `mflinear(F, L)`
 $f = g?$ `mfisequal(f, g)`
expanding operator $B_d(f)$ `mfbd(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. $q \cdot d/dq$ `mfderiv(f)`
see f over an absolute field `mfreltoabs(f)`
Serre derivative $\left(q \cdot \frac{d}{dq} - \frac{k}{12} E_2\right) f$ `mfderivE2(f)`
Rankin-Cohen bracket $[f, g]_n$ `mfbracket(f, g, n)`
Shimura lift of f for discriminant D `mfshimura(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, χ .
describe the form f `mfdescribe(f)`
 (N, k, χ) for form f `mfparams(f)`
the space identifier (0 to 4) for f `mfspace(mf, f)`
 $[f_0, \dots, f_n]$ `mfcoefs(f, n)`
 f_n `mfcoef(f, n)`
is f a CM form? `mfisCM(f)`
is f an eta quotient? `mfisetaquo(f)`

Galois rep. attached to all $(1, \chi)$ eigenforms `mfgaloistype(M)`
... single eigenform `mfgaloistype(M, F)`
... as a polynomial fixed by $\text{Ker } \rho_F$ `mfgaloisprojrep(M, F)`
decompose f on `mfbasis(M)` `mftobasis(M, f)`
smallest level on which f is defined `mfconductor(M, f)`
decompose f on $\oplus S_k^{\text{new}}(\Gamma_0(d))$, $d \mid N$ `mftonew(M, f)`
valuation of f at cusp c `mfcuspsval(M, f, c)`
expansion at ∞ of $f \mid_k \gamma$ `mfslashexpansion(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 \mathbf{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 $\mathbf{C}[[q]]$; 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 $\tau \in \mathcal{H}$ `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 `mfssymboleval(fs, p)`
Petersson product of f and g `mfpetersson(fs, gs)`
period polynomial of form f `mferiodpol(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 $M = \mathbf{M}_k(\Gamma_0(N))^\varepsilon$ `msinit(N, k, {\varepsilon = 0})`
the level M `msgetlevel(M)`
the weight k `msgetweight(M)`
the sign ε `msgetsign(M)`
Farey symbol attached to G `mspolygon(M)`
... attached to $H < G$ `msfarey(F, inH)`
 $H \backslash G$ and right G -action `mscosets(genG, inH)`

$\mathbf{Z}[G]$ -generators (g_i) and relations for Δ `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/\mathbf{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)`
Petersson product of s and t `mspetersson(M, s, t)`

Operators on subspaces

An operator is given by a matrix of a fixed \mathbf{Q} -basis. H , if given, is a stable \mathbf{Q} -subspace of $\mathbf{M}_k(G)$: 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 fist component is a matrix with integer coefficients whose columns for 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 μ 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, ϕ)</code>
eval overconvergent symbol Φ on path p	<code>msomseval(Mp, Φ, p)</code>
μ for p -adic L -functions	<code>mspadicmoments(Mp, S, {$D = 1$})</code>
$L_p^{(r)}(\chi^s)$, $s = [s_1, s_2]$	<code>mspadicL(μ, {$s = 0$}, {$r = 0$})</code>
$\hat{L}_p(\tau^i)(x)$	<code>mspadicseries(μ, {$i = 0$})</code>