

# Modular forms, modular symbols

(PARI-GP version 2.17.3)

## 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 $G = (\mathbf{Z}/q\mathbf{Z})^*$	<code>G = znstar(q, 1)</code>
convert datum $D$ to $[G, \chi]$	<code>znchar(D)</code>
Galois orbits of Dirichlet characters	<code>chargalois(G)</code>

### Spaces of modular forms

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

initialize $S_k^{\text{new}}(\Gamma_0(N), \chi)$	<code>mfinit([N, k, \chi], 0)</code>
initialize $S_k(\Gamma_0(N), \chi)$	<code>mfinit([N, k, \chi], 1)</code>
initialize $S_k^{\text{old}}(\Gamma_0(N), \chi)$	<code>mfinit([N, k, \chi], 2)</code>
initialize $E_k(\Gamma_0(N), \chi)$	<code>mfinit([N, k, \chi], 3)</code>
initialize $M_k(\Gamma_0(N), \chi)$	<code>mfinit([N, k, \chi])</code>
find eigenforms	<code>mfsplit(M)</code>
statistics on self-growing caches	<code>getcache()</code>

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

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

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

dimension of $S_k^{\text{new}}(\Gamma_0(N), \chi)$	<code>mfdim([N, k, \chi])</code>
dimension of $S_k(\Gamma_0(N), \chi)$	<code>mfdim([N, k, \chi], 1)</code>
dimension of $S_k^{\text{old}}(\Gamma_0(N), \chi)$	<code>mfdim([N, k, \chi], 2)</code>
dimension of $M_k(\Gamma_0(N), \chi)$	<code>mfdim([N, k, \chi], 3)</code>
dimension of $E_k(\Gamma_0(N), \chi)$	<code>mfdim([N, k, \chi], 4)</code>
Sturm's bound for $M_k(\Gamma_0(N), \chi)$	<code>mfsturm(N, k)</code>
$\Gamma_0(N)$ <b>cosets</b>	
list of right $\Gamma_0(N)$ cosets	<code>mfcosets(N)</code>
identify coset a matrix belongs to	<code>mftocoset</code>

### Cusps

a cusp is given by a rational number or $\infty$ .	
lists of cusps of $\Gamma_0(N)$	<code>mfcusps(N)</code>
number of cusps of $\Gamma_0(N)$	<code>mfnumcusps(N)</code>
width of cusp $c$ of $\Gamma_0(N)$	<code>mfcuspswidth(N, c)</code>
is cusp $c$ regular for $M_k(\Gamma_0(N), \chi)$ ?	<code>mfcuspisregular([N, k, \chi], c)</code>

### Create an individual modular form

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

modular form from coefficients	<code>mftobasis(mf, vec)</code>
--------------------------------	---------------------------------

There are also many predefined ones:

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

### Operations on modular forms

In this section,  $f, g$  and the  $F[i]$  are modular forms

$f \times g$	<code>mfmul(f, g)</code>
$f/g$	<code>mfddiv(f, g)</code>
$f^n$	<code>mfpow(f, n)</code>
$f(q)/q^v$	<code>mfshift(f, v)</code>
$\sum_{i \leq k} \lambda_i F[i]$ , $L = [\lambda_1, \dots, \lambda_k]$	<code>mflinear(F, L)</code>
$f = g?$	<code>mfisequal(f, g)</code>
expanding operator $B_d(f)$	<code>mfbd(f, d)</code>
Hecke operator $T_n f$	<code>mfhecke(mf, f, n)</code>
initialize Atkin-Lehner operator $w_Q$	<code>mfatkininit(mf, Q)</code>
... apply $w_Q$ to $f$	<code>mfatkin(w_Q, f)</code>
twist by the quadratic char $(D/\cdot)$	<code>mftwist(f, D)</code>
derivative wrt. $q \cdot d/dq$	<code>mfderiv(f)</code>
see $f$ over an absolute field	<code>mfreltoabs(f)</code>
Serre derivative $\left(q \cdot \frac{d}{dq} - \frac{k}{12} E_2\right) f$	<code>mfderivE2(f)</code>
Rankin-Cohen bracket $[f, g]_n$	<code>mfbracket(f, g, n)</code>
Shimura lift of $f$ for discriminant $D$	<code>mfshimura(mf, f, D)</code>

### 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 $f$	<code>mfdescribe(f)</code>
$(N, k, \chi)$ for form $f$	<code>mfparams(f)</code>
the space identifier (0 to 4) for $f$	<code>mfspace(mf, f)</code>
$[f_0, \dots, f_n]$	<code>mfcoefs(f, n)</code>
$f_n$	<code>mfcoef(f, n)</code>
is $f$ a CM form?	<code>mfisCM(f)</code>
is $f$ an eta quotient?	<code>mfisetaquo(f)</code>
Galois rep. attached to all $(1, \chi)$ eigenforms	<code>mfgaloisotype(M)</code>
... single eigenform	<code>mfgalois(M, F)</code>
... as a polynomial fixed by $\text{Ker } \rho_F$	<code>mfgaloisprojrep(M, F)</code>
decompose $f$ on <code>mfbasis(M)</code>	<code>mftobasis(M, f)</code>
smallest level on which $f$ is defined	<code>mfconductor(M, f)</code>
decompose $f$ on $\oplus S_k^{\text{new}}(\Gamma_0(d))$ , $d \mid N$	<code>mftonew(M, f)</code>
valuation of $f$ at cusp $c$	<code>mfcuspsval(M, f, c)</code>
expansion at $\infty$ of $f \mid_k \gamma$	<code>mfslashexpansion(M, f, \gamma, n)</code>
$n$ -Taylor expansion of $f$ at $i$	<code>mftaylor(f, n)</code>
all rational eigenforms matching criteria	<code>mfeigensearch</code>
... forms matching criteria	<code>mfsearch</code>

### 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 $Q(f)$	<code>mfembed(f)</code>
... embed coefs of $f$	<code>mfembed(f, v)</code>
evaluate $f$ at $\tau \in \mathcal{H}$	<code>mfeval(f, \tau)</code>
$L$ -function attached to $f$	<code>lfunmf(mf, f)</code>
... eigenforms of new space $M$	<code>lfunmf(M)</code>

### Periods and symbols

The functions in this section depend on  $[Q(f) : Q(\chi)]$  as above.

initialize symbol $fs$ attached to $f$	<code>mfsymbol1(M, f)</code>
evaluate symbol $fs$ on path $p$	<code>mfsymbolval(fs, p)</code>
Petersson product of $f$ and $g$	<code>mfpetersson(fs, gs)</code>
period polynomial of form $f$	<code>mferiodpol(M, f)</code>
period polynomials for eigensymbol $FS$	<code>mfmanin(FS)</code>

## Modular Symbols

Let  $G = \Gamma_0(N)$ ,  $V_k = \mathbf{Q}[X, Y]_{k-2}$  and  $L_k = \mathbf{Z}[X, Y]_{k-2}$ . Let  $\Delta = \text{Div}^0(\mathbf{P}^1(\mathbf{Q}))$ , generated by *paths* between cusps of  $X_0(N)$ , via the identification  $[b] - [a] \rightarrow$  path from  $a$  to  $b$ . In GP, the latter is 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  $\Delta$ .

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

$\mathbf{Z}[G]$ -generators $(g_i)$ and relations for $\Delta$	<code>mspathgens(M)</code>
decompose $p = [a, b]$ on the $(g_i)$	<code>mspathlog(M, p)</code>

### Create a symbol

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

### Operations on symbols

the list of all $s(g_i)$	<code>mseval(M, s)</code>
evaluate symbol $s$ on path $p = [a, b]$	<code>mseval(M, s, p)</code>
Petersson product of $s$ and $t$	<code>mspetersson(M, s, t)</code>

### 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 $U_p$	<code>mshecke(M, p, {H})</code>
matrix of Atkin-Lehner $w_Q$	<code>msatkinlehner(M, Q{H})</code>
matrix of the $*$ involution	<code>msstar(M, {H})</code>

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  $\phi$  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,  $\tau$  is the Teichmüller character and  $\chi$  is the cyclotomic character.

The symbol  $\phi$  can be lifted to an *overconvergent* symbol  $\Phi$ , 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  $\Phi$  allowing to compute  $L_p$  to high accuracy.

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