WeilRep

Brandon Williams

December 31, 2020

Abstract

weilrep is a Sage program for working with Weil representations and their modular forms and Jacobi forms. In particular it computes Fourier expansions of (bases of) vector-valued modular forms and Jacobi forms. It also computes additive theta lifts and Borcherds products, and offers some features for vector-valued quasimodular forms, mock modular forms and harmonic weak Maass forms. This note gives a short explanation of what *weilrep* calculates and several examples of how to use it.

Contents

1 Weil representations

The class *WeilRep* represents the **dual Weil representation** attached to an even lattice (Λ, Q), i.e. the representation ρ of $\mathrm{Mp}_2(\mathbb{Z})$ defined on the standard generators $S = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ and $T = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ by

$$\rho(T)\mathfrak{e}_\gamma = \mathbf{e}(-Q(\gamma))\mathfrak{e}_\gamma$$

and

$$\rho(S)\mathfrak{e}_\gamma = \frac{1}{\sqrt{|\Lambda'/\Lambda|}}\mathbf{e}(\mathrm{sig}(\Lambda)/8)\sum_{\beta\in\Lambda'/\Lambda}\mathbf{e}(\langle\gamma,\beta\rangle)\mathfrak{e}_\beta.$$

Here $\mathbf{e}(x) = e^{2\pi i x}$; and $\mathfrak{e}_\gamma$ is the standard basis of $\mathbb{C}[\Lambda'/\Lambda]$; and $\langle\gamma,\beta\rangle := Q(\gamma+\beta) - Q(\gamma) - Q(\beta)$.

1.1 Construction

A *WeilRep* instance is constructed with

```
WeilRep(S)
```

where S denotes either (1) a Gram matrix; i.e. a symmetric integer matrix whose diagonal consists of even integers; or (2) a QuadraticForm defined over the integers.

1.2 Basic attributes

Let

```
w = WeilRep(S)
```

be a Weil representation. The following basic attributes can be computed.

1.2.1 Gram matrix

The underlying Gram matrix can be recovered with

```
S = w.gram_matrix()
```

1.2.2 Quadratic form

The underlying quadratic form can be recovered with

```
Q = w.quadratic_form()
```

1.2.3 Signature

The signature of the underlying discriminant form can be recovered with

```
w.signature()
```

This is an element of $\mathbb{Z}/8\mathbb{Z}$ (equal to the signature of the Gram matrix S modulo 8).

1.2.4 Discriminant

The discriminant $|\det(S)|$ is computed with

```
w.discriminant()
```

1.2.5 Symmetric weights

Call a weight $k \in \frac{1}{2}\mathbb{Z}$ *symmetric* for the Weil representation attached to S if $2k + \mathrm{sig}(S) \equiv 0 \bmod 4$. In other words, every modular form of weight k is symmetric. (Similarly, call weights k *antisymmetric* if $2k + \mathrm{sig}(S) \equiv 2 \bmod 4$.) The method

```
w.is_symmetric_weight(k)
```

outputs 1 if k is a symmetric weight; 0 if k is an antisymmetric weight; and None otherwise.

1.2.6 Discriminant group

The method

```
w.ds()
```

outputs a set of representatives of the discriminant group $S^{-1}\mathbb{Z}^N/\mathbb{Z}^N$ as a list of vectors.

1.2.7 Dual

The method

```
w.dual()
```

returns the unitary dual of this representation. (This is the Weil representation attached to the matrix $-S$.)

1.2.8 Rescaling

Let $N \in \mathbb{Z}$, $N \neq 0$. If w is the WeilRep associated to the even lattice $L = (L, Q)$ then $w(N)$ returns the WeilRep associated to the even lattice

$$L(N) = (L, N \cdot Q).$$

In particular $w(-1)$ is the dual of w.

1.2.9 Matrix representation

Let $M \in \mathrm{SL}_2(\mathbb{Z})$. Suppose w is the WeilRep associated to a discriminant form $A = (A, Q)$ and

$$\rho : \mathrm{Mp}_2(\mathbb{Z}) \to \mathrm{GL}\,\mathbb{C}[A]$$

is the corresponding Weil representation. Calling

```
w(M)
```

computes the matrix $\rho(\tilde{M})$ *over* $\mathbb{C}$ *i.e. numerically* with respect to the canonical basis $\{\mathfrak{e}_\gamma : \gamma \in A\}$ in the ordering determined by w.ds(). Here $\tilde{M} = (M, \phi) \in \mathrm{Mp}_2(\mathbb{Z})$ is the preimage whose branch ϕ of the square root of $j(M;\tau)$ satisfies

$$\mathrm{Re}[\phi(\tau)] > 0 \text{ for all } \tau \in \mathbb{H}.$$

Note that if w has odd rank then this is not multiplicative: $w(MN) \neq w(M)w(N)$ in general.

1.3 Construction of modular forms

A *modular form* of weight $k \in \frac{1}{2}\mathbb{Z}$ for the Weil representation ρ attached to a discriminant form (A, Q) is a holomorphic function

$$f : \mathbb{H} \longrightarrow \mathbb{C}[A]$$

that satisfies

$$f(M \cdot \tau) = (c\tau + d)^k \rho(M) f(\tau), \;\; (M, (c\tau + d)^{1/2}) \in \mathrm{Mp}_2(\mathbb{Z}),\; \tau \in \mathbb{H}$$

and which is bounded in the limit $\lim_{y\to\infty} f(x + iy)$ for every x.

Suppose w is a WeilRep instance. Modular forms for w can be constructed in the following ways.

1.3.1 Eisenstein series

```
w.eisenstein_series(k, prec, allow_small_weight = False)
```

This returns the Eisenstein series

$$E_{k,0} = \sum_{M \in \Gamma_\infty \backslash \Gamma} \mathfrak{e}_0 \Big|_{k,\rho} M$$

for the dual Weil representation associated to the Gram matrix S with Fourier expansion up to precision *prec*. Here $\Gamma = \mathrm{Mp}_2(\mathbb{Z})$ and Γ_∞ is the subgroup generated by $T = (\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right), 1)$ and $Z = (\left(\begin{smallmatrix} -1 & 0 \\ 0 & -1 \end{smallmatrix}\right), i)$. It only accepts weight $k \geq 1$. Warning: if $k = 3/2$ or $k = 2$ then the result may be a mock modular form or quasimodular form respectively!

The formula for the Fourier coefficients is essentially that given by Bruinier and Kuss [8]. We compute the local L-functions using Cowan–Katz–White's formula [9] for the Igusa zeta functions attached to quadratic functions (in other words, the local densities); this has the advantage that one dos not even need to compute the full Jordan decomposition of the quadratic form.

Example. The Eisenstein series of weight three for the Weil representation attached to the lattice with Gram matrix $\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$. Input:

```
w = WeilRep(matrix([[2,1],[1,2]]))
print(w.eisenstein_series(3,5))
```

Output:

```
[(0, 0), 1 + 72*q + 270*q^2 + 720*q^3 + 936*q^4 + O(q^5)]
[(2/3, 2/3), 27*q^(2/3) + 216*q^(5/3) + 459*q^(8/3) + 1080*q^(11/3) + 1350*q^(14/3) + O(q^5)]
[(1/3, 1/3), 27*q^(2/3) + 216*q^(5/3) + 459*q^(8/3) + 1080*q^(11/3) + 1350*q^(14/3) + O(q^5)]
```

1.3.2 Old Eisenstein series

Certain oldform Eisenstein series can be computed using

```
w.eisenstein_oldform(k, b, prec)
```

Here b should be a nonzero isotropic vector in the discriminant group of w. This returns the oldform Eisenstein series

$$\sum_{\lambda \in \mathbb{Z}/d_b\mathbb{Z}} E_{k,\lambda b},$$

where d_b is the denominator of b (i.e. minimal with $d_b b \in \mathbb{Z}^N$), and

$$E_{k,b} = \sum_{M \in \Gamma_\infty \backslash \Gamma} \mathfrak{e}_b \Big|_k M.$$

Example. The old Eisenstein series of weight $7/2$ for the lattice with Gram matrix (8). Input:

```
w = WeilRep(matrix([[8]]))
w.eisenstein_oldform(7/2, vector([1/2]), 5)
```

Output:

```
[(0), 1 + 126*q + 756*q^2 + 2072*q^3 + 4158*q^4 + O(q^5)]
[(1/8), O(q^5)]
[(1/4), 56*q^(3/4) + 576*q^(7/4) + 1512*q^(11/4) + 4032*q^(15/4) + 5544*q^(19/4) + O(q^5)]
[(3/8), O(q^5)]
[(1/2), 1 + 126*q + 756*q^2 + 2072*q^3 + 4158*q^4 + O(q^5)]
[(5/8), O(q^5)]
[(3/4), 56*q^(3/4) + 576*q^(7/4) + 1512*q^(11/4) + 4032*q^(15/4) + 5544*q^(19/4) + O(q^5)]
[(7/8), O(q^5)]
```

1.3.3 New Eisenstein series

Warning: this is experimental (and not at all optimized).

Certain newform Eisenstein series can be computed using

```
w.eisenstein_newform(k, b, prec)
```

Here b should be a nonzero isotropic vector in the discriminant group of w. This returns the newform Eisenstein series

$$\sum_{\lambda \in \mathbb{Z}/d_b\mathbb{Z}} \sum_{\chi} \chi(\lambda) E_{k,\lambda b},$$

where d_b is the denominator of b, and where χ runs through all primitive Dirichlet characters modulo d_b. For fixed χ we use the coefficient formula of [16] to compute the Fourier coefficients of $\sum_\lambda \chi(\lambda) E_{k,\lambda b}$ **numerically**; then the sum over all primitive χ is rational and we compute it as such by guessing denominators and rounding. (The denominator we guess is usually far too large.)

Example. Compute the Eisenstein series of weight 5/2 attached to the Gram matrix ((18)) using

```
w = WeilRep(matrix([[18]]))
w.eisenstein_newform(5/2, vector([1/3]), 5)
```

Via the theta decomposition the output corresponds to the Jacobi Eisenstein series of weight 3 and index 9.

The individual Eisenstein series $E_{k,b}$ can also be approximated as Poincaré series (of index 0), see section 1.5.

1.3.4 Theta series

Note that theta series here come from *negative-definite* Gram matrices because we use the dual Weil representation. This convention is generally more natural in the context of Jacobi forms but in this case rather unfortunate.

This method simply takes the output of PARI's qfminim() and writes it as a vector-valued theta function. If w is a WeilRep instance that comes from a *negative-definite* Gram matrix or quadratic form then one can construct theta series using

```
w.theta_series(prec, P = None, test_P = True)
```

The required parameter is the precision *prec*. If P is given then it should be a polynomial over $\mathbb{Q}$ in the appropriate number of variables which is homogeneous. *Note:* if P is not *harmonic* with respect to the quadratic form underlying w then the theta series is a quasimodular form; see section 2.6.

Example. The theta series associated to the quadratic form $Q(x,y) = x^2 + y^2$:

$$\Theta(\tau) = \sum_{a,b \in \frac{1}{2}\mathbb{Z}} q^{a^2+b^2} \mathfrak{e}_{a,b}.$$

```
w = WeilRep(diagonal_matrix([-2, -2]))
w.theta_series(5)
```

Output:

```
[(0, 0), 1 + 4*q + 4*q^2 + 4*q^4 + O(q^5)]
[(1/2, 0), 2*q^(1/4) + 4*q^(5/4) + 2*q^(9/4) + 4*q^(13/4) + 4*q^(17/4) + O(q^(21/4))]
[(0, 1/2), 2*q^(1/4) + 4*q^(5/4) + 2*q^(9/4) + 4*q^(13/4) + 4*q^(17/4) + O(q^(21/4))]
[(1/2, 1/2), 4*q^(1/2) + 8*q^(5/2) + 4*q^(9/2) + O(q^(11/2))]
```

The theta series associated to the polynomial $P(x) = x^2$:

$$\Theta(\tau; P) = \sum_{a,b \in \frac{1}{2}\mathbb{Z}} a^2 q^{a^2+b^2} \mathfrak{e}_{a,b}.$$

```
R.<x, y> = PolynomialRing(QQ)
w = WeilRep(diagonal_matrix([-2, -2]))
w.theta_series(5, P = x^2)
```

Output:

```
[(0, 0), 2*q + 4*q^2 + 8*q^4 + O(q^5)]
[(1/2, 0), 1/2*q^(1/4) + q^(5/4) + 9/2*q^(9/4) + 9*q^(13/4) + q^(17/4) + O(q^(21/4))]
[(0, 1/2), 4*q^(5/4) + 4*q^(13/4) + 16*q^(17/4) + O(q^(21/4))]
[(1/2, 1/2), q^(1/2) + 10*q^(5/2) + 9*q^(9/2) + O(q^(11/2))]
```

1.3.5 Poincaré series

Let (A, Q) be a discriminant form of signature $\sigma \in \mathbb{Z}/8\mathbb{Z}$, and let $k \geq 5/2$ be a weight with $\kappa := k - \sigma/2 \in \mathbb{Z}$. The Poincaré series of weight $k \geq 5/2$ and index (β, m) where $\beta \in A$ and $m \in \mathbb{Z} - Q(\beta)$ is the series

$$P_{k,m,\beta}(\tau) = \sum_{M \in \Gamma_\infty \backslash \Gamma} \left(q^m \frac{\mathfrak{e}_\beta + (-1)^\kappa \mathfrak{e}_\beta}{2} \right) \Big|_{k,\rho^*} M.$$

These series have real but generally irrational Fourier coefficients, given by the following formula (cf. [6])

$$P_{k,m,\beta}(\tau) = \frac{q^m}{2}(\mathfrak{e}_\beta + (-1)^\kappa \mathfrak{e}_{-\beta}) + \sum_{n,\gamma} c(n,\gamma) q^n \mathfrak{e}_\gamma,$$

where: if $m > 0$, then

$$c(n,\gamma) = 2\pi (m/n)^{(1-k)/2} \sum_{c=1}^{\infty} c^{-1} J_{k-1}(4\pi\sqrt{mn}/c) \mathrm{Re}\Big[e^{-\pi i k/2} K_c(\beta, m, \gamma, n) \Big];$$

if $m < 0$, then

$$c(n,\gamma) = 2\pi (|m|/n)^{(1-k)/2} \sum_{c=1}^{\infty} c^{-1} I_{k-1}(4\pi\sqrt{|m|n}/c) \mathrm{Re}\Big[e^{-\pi i k/2} K_c(\beta, m, \gamma, n) \Big];$$

and if $m = 0$ then $P_{k,m,\beta}$ coincides with an Eisenstein series. Here J_k, I_k denote the Bessel J- and I-functions and $K_c(\beta, m, \gamma, n)$ is the Kloosterman sum

$$K_c(\beta, m, \gamma, n) = \sum_{d \in (\mathbb{Z}/c\mathbb{Z})^\times} e^{2\pi i (ma+nd)/c} \Big\langle \rho(M)^{-1} \mathfrak{e}_\beta, \mathfrak{e}_\gamma \Big\rangle,$$

if $M = (\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right), \phi) \in \mathrm{Mp}_2(\mathbb{Z})$ is an element of the metaplectic group with bottom row c, d which satisfies

$$\mathrm{Re}[\phi(\tau)] > 0 \text{ for all } \tau \in \mathbb{H}.$$

Note in particular that if m is negative then the result is a "nearly-holomorphic" modular form.
These series can be computed *approximately* with the method

```
w.poincare_series(k, b, m, prec, nterms)
```

Here, k is the weight; b is a rational vector (a representative of β); $m \in \mathbb{Z} - Q(\beta)$; prec is the precision; and $N =$*nterms* denotes the N at which we truncate the above series in the coefficient formula.

1.3.6 Bruinier–Bundschuh isomorphism

Bruinier and Bundschuh [7] gave an isomorphism between the spaces of modular forms for lattices of odd prime discriminant and scalar-valued modular forms of prime level with the quadratic character whose Fourier expansions are supported either entirely on quadratic residues or entirely on quadratic nonresidues.

These lifts can be constructed with the method

```
w.bb_lift(mf)
```

Here mf is a ModularFormElement of the appropriate level and character that satisfies the plus/minus-space condition on Fourier coefficients.

Example. To obtain the Eisenstein series of weight three for the Gram matrix $S = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$. Input:

```
w = WeilRep(matrix([[2,1],[1,2]]))
chi = DirichletGroup(3)[1]
mf = ModularForms(chi,3,prec=20).basis()[0]
w.bb_lift(mf)
```

Output:

```
[(0, 0), 1 + 72*q + 270*q^2 + 720*q^3 + 936*q^4 + 2160*q^5 + O(q^6)]
[(2/3, 2/3), 27*q^(2/3) + 216*q^(5/3) + 459*q^(8/3) + 1080*q^(11/3) + 1350*q^(14/3) + 2592*q^(17/3) + O(q^6)]
[(1/3, 1/3), 27*q^(2/3) + 216*q^(5/3) + 459*q^(8/3) + 1080*q^(11/3) + 1350*q^(14/3) + 2592*q^(17/3) + O(q^6)]
```

1.3.7 Poincaré square series

Let $\beta \in \Lambda'/\Lambda$ and $m \in \mathbb{Z} - Q(\beta)$ be an index and define by $Q_{k,m,\beta}$ the Poincaré square series

$$Q_{k,m,\beta} = \sum_{\lambda \in \mathbb{Z}} P_{k,\lambda^2 m,\lambda\beta}.$$

Here $P_{k,m,\beta}$ denotes the Poincaré series of exponential type (as in [6]). The Fourier coefficients of these series are rational and they can be computed in terms of Eisenstein series (associated to other lattices) [17]. (In fact these are the basic modular forms which are used here to produce cusp forms.) In this code we require $k \geq 5/2$.

These series are implemented with

```
w.pss(k, beta, m, prec)
```

The construction of modular forms of antisymmetric weights is similar [19] and implemented with

```
w.pssd(k, beta, m, prec)
```

1.3.8 Mock modular forms

Currently only mock Eisenstein series and mock Poincaré series can be constructed; see section 2.7 for details.

1.4 Bases of modular forms

1.4.1 Modular forms

The method

```
w.modular_forms_basis(k, prec)
```

returns a basis in echelon form of the space of modular forms $M_k(\rho)$ with Fourier expansions to precision $O(q^{prec})$.

For symmetric weights at least 5/2 or antisymmetric weights at least 7/2 we use linear combinations of Eisenstein series and PSS. In weights 0 and 1/2 we use an algorithm based on Ehlen–Skoruppa [10]. Otherwise we try to reduce the problem to higher weight forms, generally by writing M_k as the intersection

$$M_k(\rho) = E_4(\tau)^{-1} M_{k+4}(\rho) \cap E_6(\tau)^{-1} M_{k+6}(\rho)$$

where E_4, E_6 are the scalar Eisenstein series.

Example. Modular forms of weight 11/2 for the Weil representation attached to the Cartan matrix A_5, with Fourier expansions up to $O(q^5)$:

```
WeilRep(CartanMatrix(['A', 5])).modular_forms_basis(11/2, 4)
```

Output:

```
[(0, 0, 0, 0, 0), 1 + 180*q + 4404*q^2 + 26562*q^3 + O(q^4)]
[(5/6, 2/3, 1/2, 1/3, 1/6), 20*q^(7/12) + 1472*q^(19/12) + 14076*q^(31/12) + 57920*q^(43/12) + O(q^4)]
[(2/3, 1/3, 0, 2/3, 1/3), q^(1/3) + 693*q^(4/3) + 8680*q^(7/3) + 43252*q^(10/3) + O(q^4)]
[(1/2, 0, 1/2, 0, 1/2), 552*q^(5/4) + 7040*q^(9/4) + 39888*q^(13/4) + O(q^4)]
[(1/3, 2/3, 0, 1/3, 2/3), q^(1/3) + 693*q^(4/3) + 8680*q^(7/3) + 43252*q^(10/3) + O(q^4)]
[(1/6, 1/3, 1/2, 2/3, 5/6), 20*q^(7/12) + 1472*q^(19/12) + 14076*q^(31/12) + 57920*q^(43/12) + O(q^4)]
------------------------------------------------------------------------------
[(0, 0, 0, 0, 0), 34*q - 156*q^2 + 162*q^3 + O(q^4)]
[(5/6, 2/3, 1/2, 1/3, 1/6), -7*q^(7/12) - 13*q^(19/12) + 171*q^(31/12) - 157*q^(43/12) + O(q^4)]
[(2/3, 1/3, 0, 2/3, 1/3), q^(1/3) + 18*q^(4/3) - 14*q^(7/3) - 380*q^(10/3) + O(q^4)]
[(1/2, 0, 1/2, 0, 1/2), q^(1/4) - 30*q^(5/4) + 81*q^(9/4) - 12*q^(13/4) + O(q^4)]
[(1/3, 2/3, 0, 1/3, 2/3), q^(1/3) + 18*q^(4/3) - 14*q^(7/3) - 380*q^(10/3) + O(q^4)]
[(1/6, 1/3, 1/2, 2/3, 5/6), -7*q^(7/12) - 13*q^(19/12) + 171*q^(31/12) - 157*q^(43/12) + O(q^4)]
```

1.4.2 Cusp forms

The method

```
w.cusp_forms_basis(k, prec)
```

returns a basis in echelon form of the space of cusp forms $S_k(\rho)$ with Fourier expansions to precision $O(q^{prec})$.

Example. Cusp forms of weight 4 for the Weil representation attached to the Cartan matrix A_6, with Fourier expansions up to $O(q^4)$:

```
WeilRep(CartanMatrix(['A', 6])).modular_forms_basis(4, 4)
```

Output:

```
[(0, 0, 0, 0, 0, 0), O(q^4)]
[(6/7, 5/7, 4/7, 3/7, 2/7, 1/7), -17*q^(4/7) + 68*q^(11/7) - 135*q^(18/7) + 125*q^(25/7) + O(q^4)]
[(5/7, 3/7, 1/7, 6/7, 4/7, 2/7), 5*q^(2/7) + 27*q^(9/7) - 89*q^(16/7) + 40*q^(23/7) + O(q^4)]
[(4/7, 1/7, 5/7, 2/7, 6/7, 3/7), q^(1/7) - 45*q^(8/7) + 340*q^(22/7) + O(q^4)]
[(3/7, 6/7, 2/7, 5/7, 1/7, 4/7), -q^(1/7) + 45*q^(8/7) - 340*q^(22/7) + O(q^4)]
[(2/7, 4/7, 6/7, 1/7, 3/7, 5/7), -5*q^(2/7) - 27*q^(9/7) + 89*q^(16/7) - 40*q^(23/7) + O(q^4)]
[(1/7, 2/7, 3/7, 4/7, 5/7, 6/7), 17*q^(4/7) - 68*q^(11/7) + 135*q^(18/7) - 125*q^(25/7) + O(q^4)]
```

1.4.3 Modular forms with specified order at ∞

The method

```
w.basis_vanishing_to_order(k, N, prec, inclusive=False)
```

returns a basis in echelon form of the space of modular forms of weight k which vanish to order at least N at ∞. Setting the optional parameter *inclusive* to True forces the coefficient of q^N itself to zero.

Example. Let $M_{13}(\rho^*)$ be the space of modular forms of weight 13 for the Gram matrix $S = \left(\begin{smallmatrix} -2 & -1 \\ -1 & -2 \end{smallmatrix}\right)$. To compute the subspace of modular forms which vanish to order at least 2/3:

```
WeilRep(matrix([[-2,-1],[-1,-2]])).basis_vanishing_to_order(13, 2/3, 4)
```

Output:

```
[(0, 0), q - 18*q^2 + 108*q^3 + O(q^4)]
[(1/3, 1/3), 3*q^(4/3) - 69*q^(7/3) + 690*q^(10/3) + O(q^4)]
[(2/3, 2/3), 3*q^(4/3) - 69*q^(7/3) + 690*q^(10/3) + O(q^4)]
```

1.4.4 Nearly holomorphic modular forms

A *nearly-holomorphic modular form* is a function which transforms like a modular form and is holomorphic on $\mathbb{H}$ but may have a pole at cusps. The method

```
w.nearly_holomorphic_modular_forms_basis(k, N, prec, inclusive = False)
```

returns a basis of the space of nearly-holomorphic modular forms of weight k with a pole of order at most N at ∞ with principal parts which are as simple as possible. Setting the optional parameter *inclusive* to False does not allow forms with pole order exactly N.

Example. In [20] Zagier considered a sequence of forms $g_D(\tau) \in M^+_{3/2}(\Gamma_0(4))$ whose Fourier expansions take the form

$$g_D(\tau) = q^{-D} + \sum_{n=0}^{\infty} B(D,n)q^n, \;\; B(D,n) \in \mathbb{Z}$$

in his work on traces of singular moduli. We can realize these as modular forms for the Gram matrix (2) and compute them as follows.

```
WeilRep(matrix([[2]])).nearly_holomorphic_modular_forms_basis(3/2, 2, 3)
```

Output:

```
[(0), -2 - 492*q - 7256*q^2 - 53008*q^3 + O(q^4)]
[(1/2), q^(-1/4) + 248*q^(3/4) + 4119*q^(7/4) + 33512*q^(11/4) + 192513*q^(15/4) + O(q^4)]
------------------------------------------------------------------------------
[(0), q^-1 - 2 - 143376*q - 26124256*q^2 - 1417904008*q^3 + O(q^4)]
[(1/2), -26752*q^(3/4) - 8288256*q^(7/4) - 561346944*q^(11/4) - 18508941312*q^(15/4) + O(q^4)]
------------------------------------------------------------------------------
[(0), -565760*q - 190356480*q^2 - 16555069440*q^3 + O(q^4)]
[(1/2), q^(-5/4) + 85995*q^(3/4) + 52756480*q^(7/4) + 5874905295*q^(11/4) + 292658282496*q^(15/4) + O(q^4)]
------------------------------------------------------------------------------
[(0), q^-2 - 18473000*q - 29071392966*q^2 - 8251987131648*q^3 + O(q^4)]
[(1/2), -1707264*q^(3/4) - 5734772736*q^(7/4) - 2225561184000*q^(11/4) - 312211675238400*q^(15/4) + O(q^4)]
```

1.4.5 Borcherds obstruction space

Define the *Borcherds obstruction space* as the space of holomorphic modular forms whose constant terms are multiples of $\mathfrak{e}_0$. Following [3] (for lattices of the correct signature) elements of this space may be thought of as obstructions to the existence of Borcherds products with specificed divisor and weight. In large weight (at least 5/2) this is spanned by the cusp space $S_k(\rho)$ and the Eisenstein series $E_{k,0}$.

This space can be computed with

```
w.borcherds_obstructions(k, prec)
```

where k is the weight and *prec* is the precision to which the Fourier series are computed.

1.4.6 Bases of quasimodular forms

See section 2.6 for the definition of quasimodular forms and their properties. Given a WeilRep w, a basis of quasimodular forms in echelon form can be constructed using

```
w.quasimodular_forms_basis(k, prec)
```

where k is the weight, and *prec* is the precision of the Fourier expansions.

1.5 Dimension formulas

We use the Riemann-Roch theorem to compute dimensions of spaces of modular forms in weight at least 2. In small weights we compute a *basis* of $M_k(\rho)$ first and then take its length. (This is slow!)

1.5.1 Modular forms

The method

```
w.modular_forms_dimension(k, eta_twist=0)
```

computes the dimension

$$\dim M_k(\rho \otimes \chi^N),$$

where χ is the multiplier system of $\eta(\tau) = q^{1/24}\prod_{n=1}^{\infty}(1-q^n)$ and N is the optional parameter *eta_twist* (by default $N = 0$).

1.5.2 Cusp forms

The method

```
w.cusp_forms_dimension(k, eta_twist=0)
```

computes the dimension

$$\dim S_k(\rho \otimes \chi^N),$$

where χ is the multiplier system of $\eta(\tau) = q^{1/24}\prod_{n=1}^{\infty}(1-q^n)$ and N is the optional parameter *eta_twist* (by default $N = 0$).

1.5.3 Hilbert series

The *Hilbert series* of the Weil representation ρ for the lattice L is the series

$$\mathrm{Hilb}_\rho(t) := \sum_{k=0}^{\infty} \dim M_{k+\varepsilon}(\rho) t^k,$$

where $\varepsilon = 0$ if L has even rank and $\varepsilon = 1/2$ if L has odd rank. It has the form

$$\mathrm{Hilb}_\rho(t) = \frac{P(t)}{(1-t^4)(1-t^6)}$$

for some polynomial $P(t)$.

The method

```
w.hilbert_series()
```

computes the Hilbert series $\mathrm{Hilb}_\rho(t)$ as a power series.

The method

```
w.hilbert_polynomial()
```

computes the polynomial $P(t)$.

2 Vector-valued modular forms

Vector-valued modular forms are instances of the *WeilRepModularForm* class. Suppose f is a vector-valued modular form.

2.1 Fourier coefficients

2.1.1 Fourier expansion

The Fourier expansion of f is recovered with

```
f.fourier_expansion()
```

The output is a list of tuples of the form $(g, N, \phi_g(q))$ where g is a vector, $N \in \mathbb{Q}$ and ϕ_g is a power series in q, meant to indicate that

$$f = \sum_g q^{-N} \phi_g(q) \mathfrak{e}_g.$$

2.1.2 Coefficients

f.coefficients() produces the Fourier coefficients of f as a dictionary. The keys are tuples $(g_1, ..., g_d, n)$ and the output is the coefficient of $q^n \mathfrak{e}_{(g_1,...,g_d)}$ in f.

Example. Let f be the (vector-valued) Cohen Eisenstein series of weight $5/2$:

$$f(\tau) = (1 - 70q - 120q^2 - 240q^3 - 550q^4 - ...)\mathfrak{e}_0 + (-10q^{1/4} - 48q^{5/4} - 250q^{9/4} - 240q^{13/4} - 480q^{17/4} - ...)\mathfrak{e}_{1/2}.$$

Construct it with

```
f = WeilRep([-2]).eisenstein_series(5/2, 5)
```

To extract the coefficient -70 use

```
f.coefficients()[(0, 1)]
```

To extract the coefficient -48 use

```
f.coefficients()[(1/2, 5/4)]
```

2.1.3 Coefficient vector

```
f.coefficient_vector()
```

sorts the Fourier coefficients of f and combines them to a vector.

Example. Let f be the Cohen Eisenstein series from the previous section. The coefficient vector

```
f.coefficient_vector()
```

is

```
(1, -10, -70, -48, -120, -250, -240, -240, -550, -480)
```

The coefficient_vector method takes a few optional arguments, most importantly:
- *starting_from*: the exponent at which we begin counting coefficients;
- *ending_with*: the exponent at which we stop counting coefficients.

2.1.4 Components

The method

```
f.components()
```

produces the components of f as a dictionary. The keys are *tuples* $(g_1, ...g_d)$, where $(g_1, ..., g_d)$ is a vector in the dual lattice, and the output is the Fourier series $f_{g_1,...,g_d}$ (i.e. the component in f) as a power series with exponents rounded up to integers.

2.2 Arithmetic operations

Addition and subtraction are defined as usual.

2.2.1 Multiplication

Multiplication of vector-valued modular forms should be understood as the tensor product. If $f \in M_{k_1}(\rho_{S_1})$ and $g \in M_{k_2}(\rho_{S_2})$ are vector-valued modular forms for Weil representations associated to S_1 and S_2 then $f \otimes g$ is a modular form for the direct sum $S_1 \oplus S_2$.

Example. Input:

```
w = WeilRep(matrix([[-2]]))
E = w.eisenstein_series(5/2, 5)
E * E
```

The output is a modular form of weight 5 for the dual Weil representation attached to $\left(\begin{smallmatrix} -2 & 0 \\ 0 & -2 \end{smallmatrix}\right)$:

```
[(0, 0), 1 - 140*q + 4660*q^2 + 16320*q^3 + 46900*q^4 + O(q^5)]
[(1/2, 0), -10*q^(1/4) + 652*q^(5/4) + 4310*q^(9/4) + 25420*q^(13/4) + 63340*q^(17/4) + O(q^5)]
[(0, 1/2), -10*q^(1/4) + 652*q^(5/4) + 4310*q^(9/4) + 25420*q^(13/4) + 63340*q^(17/4) + O(q^5)]
[(1/2, 1/2), 100*q^(1/2) + 960*q^(3/2) + 7304*q^(5/2) + 28800*q^(7/2) + 95140*q^(9/2) + O(q^5)]
```

To multiply $f \in M_\ell(\rho)$ by a scalar-valued modular form g of weight k one should convert g to a vector-valued moduar form (for the rank zero lattice) using the command

```
smf(k, g)
```

and then multiply as usual.

Example. To multiply the Cohen Eisenstein series of weight 5/2 by the discriminant Δ, use

```
E = WeilRep(matrix([[-2]])).eisenstein_series(5/2, 5)
E * smf(12, delta_qexp(5))
```

2.3 Other methods

2.3.1 Bol operator

If f is a modular form of integral weight $k \leq 1$ then its image under the *Bol operator*

$$D = \Big(\frac{1}{2\pi i}\frac{d}{d\tau}\Big)^{1-k} f(\tau)$$

is a modular form of weight $2-k$. This is implemented by the method

```
f.bol()
```

2.3.2 Conjugation

The method

```
f.conjugate(A)
```

can be used to change lattice bases. If f is a modular form for the Gram matrix S then *f.conjugate(A)* is a modular form for the Gram matrix A^TSA. (Note: A does not need to be invertible.)

2.3.3 Derivative and Serre derivative

The derivative of a modular form is generally a *quasimodular form*, not a modular form. This is implemented with the method

```
f.derivative()
```

See section 2.6 for more details.

If f is a modular form of weight k then its *Serre derivative*

$$Sf(\tau) = \frac{1}{2\pi i}f'(\tau) - \frac{k}{12}f(\tau)E_2(\tau)$$

is a modular form of weight $k+2$. This is implemented by the method

```
f.serre_derivative()
```

2.3.4 Hecke operators

The method

```
f.hecke_T(N)
```

applies the Hecke operator T_N to f using the formula of [1]. The lattice does not have to be positive-definite. (Warning: this is only implemented when N is coprime to the level of the lattice!)

Similarly the methods

```
f.hecke_U(N)
```

and

```
f.hecke_V(N)
```

apply the index-raising Hecke operators U_N and V_N (cf. [4], [15]). These are maps

$$U_N : M_k(\rho) \longrightarrow M_k(\rho(N^2)), \text{ and } V_N : M_k(\rho) \longrightarrow M_k(\rho(N)),$$

where $\rho(N)$ denotes the Weil representation for the lattice $L(N)$ with quadratic form rescaled by N, i.e. $L(N) = (L, N \cdot Q)$. They specialize to the Eichler–Zagier Hecke operators on Jacobi forms when L is positive-definite.

Finally the method

```
f.hecke_P(N)
```

implements the index-*lowering* Hecke operator P_N of [4]. This is a map

$$P_N : M_k(\rho(N^2)) \longrightarrow M_k(\rho)$$

with the property $P_N \circ U_N = \mathrm{id}$.

2.3.5 Lattice reduction

Suppose L is an isotropic lattice over $\mathbb{Z}$ of signature (b^+, b^-) with a norm zero vector z. The method

```
f.reduce_lattice()
```

implements the lattice-reduction map from L to the signature $(b^+ - 1, b^- - 1)$ lattice $z^\perp/z$. In the notation of Borcherds ([2], especially sections 5, 6) this takes the form F_M as input and yields the form F_K. The norm-zero vector z can be provided; if no z is given then we try to find one using PARI qfsolve().

In this setup note that $|L| = N^2 \cdot |z^\perp/z|$ for some N. In particular if L is isotropic but has squarefree discriminant (or if L has odd rank, and $|L|/2$ is squarefree) then this method always yields a modular form on a smaller lattice whose discriminant form is equivalent to that of L.

2.3.6 Principal part

Suppose F is a nearly holomorphic modular form.

```
F.principal_part()
```

outputs its principal part (its terms with negative exponent, and the $\mathfrak{e}_0$-component of its constant term) as a WeilRepPrincipalPart instance. This is useful in the Borcherds lift as it determines the weight and divisor of the output.

2.3.7 Rankin–Cohen brackets

If f_1 and f_2 are modular forms of weight k_1 and k_2 attached to the Gram matrices $\mathbf{S}_1$ and $\mathbf{S}_2$, and $N \in \mathbb{N}_0$, then we obtain modular forms of weight $k_1 + k_2 + 2N$ as Rankin–Cohen brackets:

$$[f_1, f_2]_N = \sum_{r=0}^{N} (-1)^r \binom{k_1 + N - 1}{N - r} \binom{k_2 + N - 1}{r} \frac{d^r}{d\tau^r} f_1(\tau) \otimes \frac{d^{N-r}}{d\tau^{N-r}} f_2(\tau) \in M_{k_1+k_2+2N}(\mathbf{S}_1 \oplus \mathbf{S}_2).$$

These are cusp forms if $N \geq 1$. The Rankin–Cohen brackets are implemented as the function *rankin_cohen(N, f_1, f_2)*.

Note that certain expressions which are trivial for scalar modular forms are generally nonzero for vector-valued forms. For example, if f has weight k then its first Rankin–Cohen bracket with itself,

$$[f, f]_1 = k\Big(f \otimes f' - f' \otimes f\Big)$$

is generally nonzero.

Example. The first Rankin–Cohen bracket of the standard unary theta function with itself is nonzero. Compute it with

```
w = WeilRep(matrix([[-2]]))
theta = w.theta_series(10)
rankin_cohen(1, theta, theta)
```

When $N = 0$ the Rankin–Cohen bracket reduces to the (tensor) product of two modular forms.

2.3.8 Theta contraction

Suppose f is a modular form of weight k for a Gram matrix $\tilde{S}$ which can be written as a block matrix

$$\tilde{S} = \begin{pmatrix} S & S\beta \\ (S\beta)^T & 2m + \beta^T S \beta \end{pmatrix}$$

where $m \in \mathbb{Q}$, $m > 0$. The *theta contraction* Θf of f is a twisted product of f with a rank one theta function. The result is a modular form of weight $k + 1/2$ for the Gram matrix S. This is useful in the context of theta lifts as it corresponds to the pullbacks of modular forms on orthogonal groups.

Example. Compute the theta contraction of the weight three Eisenstein series E attached to the Gram matrix $\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$. Input:

```
S = matrix([[2, 1], [1, 2]])
E = WeilRep(S).eisenstein_series(3,5)
print(E.theta_contraction())
```

Output:

```
[(0), 1 + 126*q + 756*q^2 + 2072*q^3 + 4158*q^4 + O(q^5)]
[(1/2), 56*q^(3/4) + 576*q^(7/4) + 1512*q^(11/4) + 4032*q^(15/4) + 5544*q^(19/4) + O(q^5)]
```

i.e. ΘE is the Eisenstein series of weight $7/2$ attached to the Gram matrix (2).

2.3.9 Trace map

Suppose $f \in M_k(\rho)$ and $g \in M_\ell(-\rho)$ are modular forms whose Gram matrices are negatives of one another. The operation

```
f & g
```

computes the trace-map

$$\langle f, g \rangle := \sum_\gamma f_\gamma(\tau) g_\gamma(\tau) \in M_{k+\ell},$$

which is a (scalar) modular form of weight $k + \ell$.

2.4 Operations on lists of modular forms

The output of most functions that yield lists of vector-valued modular forms is actually an instance of the WeilRepModularFormsBasis class. In addition to the usual commands that apply to lists, WeilRepModularFormsBasis has a few extra methods.

Suppose X is a WeilRepModularFormsBasis instance.

2.4.1 Coordinates

Suppose f is a vector-valued modular form for the same lattice and of the same weight as X. Write $X = (x_1, ..., x_n)$. Then

```
X.coordinates(f)
```

computes a vector $v = (v_1, ..., v_n)$ such that

$$f = v_1 x_1 + ... + v_n x_n$$

(and raises a ValueError if this does not exist).

Warning: we only compute v using the known Fourier coefficients. If an insufficient number of coefficients are known then the result is likely to be wrong.

2.4.2 Principal parts

X.principal_parts() enumerates the principal parts of all elements of X as a string separated by newlines.

2.4.3 Theta contraction

X.theta() computes the theta contraction of all elements of X simultaneously. The result is again a WeilRepModularFormsBasis instance.

2.5 Automorphisms

Let $A = (A, Q)$ be a discriminant form. By an *automorphism* of A we mean a $\mathbb{Z}$-linear map $g : A \to A$ for which $Q \circ g = Q$. (In other words the group of automorphisms is the orthogonal group $\mathrm{O}(A)$.)

2.5.1 Automorphism group

Let

```
w = WeilRep(S)
```

be a WeilRep instance. The automorphism group of w can be constructed with

```
G = w.automorphism_group()
```

(Warning: this can be slow, particularly if G is reasonably big!)

G acts as a list of *WeilRepAutomorphisms* g. To view G explicitly it is better to type *list(G)*. Each g can be used as follows:
(i) if f is a WeilRepModularForm, with Fourier expansion

$$f(\tau) = \sum_{x \in A} f_x(\tau) e_x,$$

then $g(f)$ is the WeilRepModularForm

$$g(f)(\tau) = \sum_{x \in A} f_x(\tau) e_{g \cdot x}.$$

(ii) if $x \in A$ then $g(x) \in A$ is the image of x under g.
Additionally WeilRepAutomorphisms can be multiplied (corresponding to composition) and inverted using the syntax

```
~g
```

Some additional methods for the WeilRepAutomorphismGroup G are:
– Generators: *G.gens()*. This produces a list of generators of G.
– Character group: *G.characters()*. This produces a list of all the characters, or homomorphisms $\chi : G \to \mathbb{C}^\times$. Here a character is represented as a list

$$\chi = [\chi_1, ..., \chi_N],$$

where: if $G = [g_1, ..., g_N]$ then $\chi_i = \chi(g_i)$.

2.5.2 Invariant modular forms

Let w be a WeilRep instance with automorphism group G, and let $\chi : G \to \mathbb{C}^\times$ be a character.

The method

```
w.invariant_forms_dimension(k, chi = chi)
```

computes the dimension

$$\dim M_{k,\chi}(\rho),$$

where $M_{k,\chi}(\rho)$ is the subspace of forms $f \in M_k(\rho)$ for which $g \cdot f = \chi(g) f$ for all $g \in G$. If $k \geq 5/2$ then this is reasonably fast (up to determining G) as we can use the Riemann–Roch formula. If χ is not given then we assume that χ is the trivial character. (Warning: in this case the weight k must satisfy $2k + \sigma(A, Q) \equiv 0 \bmod 4$.)

Note: if G is not abelian then $M_k(\rho)$ generally does not decompose into $\bigoplus_\chi M_{k,\chi}(\rho)$.
Similarly the method

```
w.invariant_cusp_forms_dimension(k, chi = chi)
```

computes the dimension $\dim S_{k,\chi}(\rho)$ where $S_{k,\chi}(\rho) = M_{k,\chi}(\rho) \cap S_k(\rho)$.

The method

```
w.invariant_forms_basis(k, prec, chi = chi)
```

computes a WeilRepModularFormsBasis of $M_{k,\chi}(\rho)$ up to precision *prec*. Similarly the method

```
w.invariant_cusp_forms_basis(k, prec, chi = chi)
```

computes a WeilRepModularFormsBasis of $S_{k,\chi}(\rho)$ up to precision *prec*.

2.6 Quasimodular forms

See e.g. section 5.3 of [21] for an introduction to quasimodular forms. This extends immediately to vector-valued forms.

Let $A = (A, Q)$ be a discriminant form. A *quasimodular form* for the Weil representation of A is a holomorphic vector-valued function $f = f_0 : \mathbb{H} \to \mathbb{C}[A]$ with a Fourier expansion of the form

$$f(\tau) = \sum_{\gamma \in A} \sum_{n \in \mathbb{Z} - Q(\gamma)} c(n, \gamma) q^n \mathfrak{e}_\gamma$$

with the property that there are finitely many holomorphic functions $f_1, ..., f_d : \mathbb{H} \to \mathbb{C}[A]$ such that the *completion* of f_0,

$$F(\tau) = \sum_{r=0}^{d} f_r(\tau)(4\pi y)^{-r}$$

transforms like a modular form. The forms $f_i(\tau)$ are then also quasimodular forms.

Typical examples of quasimodular forms are theta series attached to non-harmonic polynomials; Eisenstein series of weight two; and derivatives (of any order) of holomorphic modular forms.

2.6.1 Completion

Suppose f is a quasimodular form. The completion of f to an almost-holomorphic modular form can be computed with the method

```
f.completion()
```

The result is a *WeilRepAlmostHolomorphicModularForm* instance that prints the nonholomorphic modular form $F(\tau)$ as in the following example:

```
w = WeilRep(matrix([[2, 0], [0, -2]]))
w.eisenstein_series(2, 5).completion()
```

yields

```
Almost holomorphic modular form f_0 + f_1 * (4 pi y)^(-1), where:
f_0 =
[(0, 0), 1 - 16*q - 24*q^2 - 64*q^3 - 72*q^4 + O(q^5)]
[(1/2, 0), -16*q^(3/4) - 32*q^(7/4) - 48*q^(11/4) - 96*q^(15/4) - 80*q^(19/4) + O(q^(23/4))]
[(0, 1/2), -4*q^(1/4) - 24*q^(5/4) - 52*q^(9/4) - 56*q^(13/4) - 72*q^(17/4) + O(q^(21/4))]
[(1/2, 1/2), -8*q - 48*q^2 - 32*q^3 - 96*q^4 + O(q^5)]
-------------------------------------------------------------------------------
```

```
f_1 =
[(0, 0), -6 + O(q^5)]
[(1/2, 0), O(q^(23/4))]
[(0, 1/2), O(q^(21/4))]
[(1/2, 1/2), -6 + O(q^5)]
```

Additionally, the completion of f acts as a list of quasimodular forms: in the above example, letting

```
X = w.eisenstein_series(2, 5).completion()
```

the terms f_0, f_1 can be accessed as *X[0], X[1]*. The first term f_0 can also be accessed as the holomorphic part:

```
X.holomorphic_part()
```

2.6.2 Other operations

Many (but not all) of the operations that apply to modular forms also apply to quasimodular forms. In particular, quasimodular forms can be added, multiplied (tensored), etc. The following additional features are useful. Suppose f is a quasimodular form.

```
f.depth()
```

computes the depth of f, i.e. the minimal $d \in \mathbb{N}$ such that $F(\tau) = \sum_{r=0}^{d} f_r(\tau)(4\pi y)^{-r}$ for some quasimodular forms f_r.

```
f.shift()
```

applies the map δ in section 5.3 of [21] i.e. it sends f to the term f_1 in its completion.

2.7 Mock modular forms

There is some support for vector-valued mock modular forms and harmonic Maass forms. For background see [5].

A harmonic weak Maass form of weight k is a real-analytic function $f : \mathbb{H} \to \mathbb{C}[A]$ satisfying the transformation rules

$$f(M \cdot \tau) = (c\tau + d)^k \rho(M) f(\tau), \quad M = (\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right), (c\tau + d)^{1/2}) \in \mathrm{Mp}_2(\mathbb{Z})$$

and a growth condition at ∞ and which solves the differential equation

$$\Delta_k f = 0,$$

where

$$\Delta_k = -4y^2 \frac{\partial^2}{\partial\tau\partial\overline{\tau}} + 2iky\frac{\partial}{\partial\tau}.$$

Such a function is represented by its Fourier series which takes the form

$$f(\tau) = \sum_{n \geq -N} a_f(n) q^n + b_f(0) y^{1-k} + \sum_{n<0} b_f(n)\Gamma(1-k, 4\pi|n|y)q^n, \quad a_f(n), b_f(n) \in \mathbb{C},\ q = e^{2\pi i\tau}.$$

When $k = 1$ the term y^{1-k} must be replaced by $\log(y)$. Here $\Gamma(s,x) = \int_x^\infty e^{-t}t^{s-1}\,\mathrm{d}t$ is the upper incomplete Gamma function; when computing numerically it may be better to write

$$\Gamma(1-k, 4\pi|n|y)q^n = U(k,k;4\pi|n|y)e^{2\pi i|n|(-\overline{\tau})}$$

where U is the confluent hypergeometric U-function, as this is less likely to cause roundoff or overflow errors.

The *holomorphic part* of f,

$$\sum_{n \geq -N} a_f(n)$$

is called a *mock modular form* of weight k. The remaining coefficients $b_f(n)$ can be expressed conveniently through a modular form of weight $2-k$ called the *shadow*:

$$\xi_k f(\tau) := 2i(-4\pi)^{k-1} y^k \overline{\partial_{\overline{\tau}} f(\tau)} = (1-k)\overline{b_f(0)} + \sum_{n>0} \overline{b_f(-n)} n^{1-k} q^n.$$

(Warning: the factor $(-4\pi)^{k-1}$ is not always present in the references.) (Warning: when $k=1$ the factor in front of $\overline{b_f(0)}$ is incorrect.)

The following constructions of modular forms are supported.

2.7.1 Zagier Eisenstein series

Suppose $L=(L,Q)$ is an even lattice whose signature is 1 mod 4 with WeilRep w. The value at $s=0$ of the spectrally deformed Eisenstein series

$$E_{3/2}(\tau; s) = \sum_{M \in \Gamma_\infty \backslash \Gamma} (y^s \mathfrak{e}_0)\Big|_{3/2,\rho} M = \frac{1}{2} \sum_{\substack{c,d \in \mathbb{Z} \\ \gcd(c,d)=1}} \frac{y^s}{|c\tau+d|^{2s}(c\tau+d)^{3/2}} \rho(M)^{-1} \mathfrak{e}_0,$$

(where $M \in \mathrm{Mp}_2(\mathbb{Z})$ may be any element whose bottom row is c, d and whose square root branch agrees with the branch used in $(c\tau+d)^{3/2}$) is a harmonic Maass form (since Δ_k is self adjoint) but may fail to be holomorphic. In general the *Zagier Eisenstein series* is the holomorphic part of $E_{3/2}(\tau; 0)$.

The holomorphic part is the formal result of the Bruinier–Kuss formula [8] and the shadow can be computed as in [18].

Example. The original Zagier Eisenstein series appears when $L=\mathbb{Z}$ with norm $Q(x)=x^2$. We compute its completion to a Maass form with

```
WeilRep([[2]]).eisenstein_series(3/2, 10).completion()
```

which outputs

```
Harmonic Maass form with holomorphic part
[(0), 1 - 6*q - 12*q^2 - 16*q^3 - 18*q^4 - 24*q^5 - 24*q^6 - 24*q^7 - 36*q^8 - 3
[(1/2), -4*q^(3/4) - 12*q^(7/4) - 12*q^(11/4) - 24*q^(15/4) - 12*q^(19/4) - 36*q
and shadow 1/pi times
[(0), 3/2 + 3*q + 3*q^4 + 3*q^9 + O(q^10)]
[(1/2), 3*q^(1/4) + 3*q^(9/4) + 3*q^(25/4) + O(q^(41/4))]
```

From this one can read off that the nonholomorphic Fourier coefficients $b_f(n)$ are

$$b_f(0) = -\frac{3}{\pi}, \;\; b_f(n^2) = \frac{3|n|}{\pi}, \; n<0, \; b_f(n) = 0 \text{ otherwise.}$$

The coefficients of the holomorphic part are the Hurwitz class numbers $H(4n)$. Altogether

$$E_{3/2}(\tau;0) = \sum_{n=0}^{\infty} H(n) q^{n/4} \mathfrak{e}_{n/2} - \frac{3}{\pi\sqrt{y}} \mathfrak{e}_0 - \frac{3}{2\pi\sqrt{y}} \sum_{\substack{n \in \mathbb{Z} \\ n \neq 0}} \Big(\int_1^\infty t^{-3/2} e^{-\pi n^2 y t} \, \mathrm{d}t \Big) q^{-n^2/4} \mathfrak{e}_{n/2}.$$

2.7.2 Maass Eisenstein series

Warning: this needs further testing when the weight is half-integral.

The *Maass Eisenstein series* $\tilde{E}_k(\tau)$ of weight $k < 0$ is the value at $s = 1-k$ of $E_k(\tau; s) = \sum_{M \in \Gamma_\infty \backslash \Gamma} (y^s \mathfrak{e}_0)|_k M$. Since

$$\Delta_k E_k(\tau; s) = s(1-k-s)E_k(\tau; s)$$

it follows that $\tilde{E}_k(\tau)$ is annihilated by Δ_k and therefore defines a harmonic Maass form. The *mock Eisenstein series* is its holomorphic part.

When k is an integer, (and in particular the underlying lattice $L = (L, Q)$ has even rank) the mock Eisenstein series has a Fourier expansion of the form

$$\pi^{k-1}\Big(C + \sum_{n>0} c(n)q^n\Big), \;\; C \in \mathbb{C}[A], \;\; c(n) \in \mathbb{Q}[A].$$

The constant term C involves non-critical values of quadratic L functions (or the Riemann zeta function) e.g. $\zeta(3)$ which probably cannot be simplified.

Non-critical values of quadratic L-functions are represented formally by

$$L(s, D) = \sum_{n=1}^{\infty} \chi_D(n) n^{-s}, \;\; \chi_D(n) := \left(\frac{D}{n}\right).$$

More precisely these are instances of the QuadraticLFunction class from the file *weilrep_misc.py*, which is not available by default but can be imported manually. They can be evaluated numerically (using Pari/GP) with the usual $.n()$. These values are expressed in terms of the Riemann zeta function if possible (i.e. if D is a square).

The mock Eisenstein series of weight k to precision *prec* can be computed using

```
w.mock_eisenstein_series(k, prec)
```

Its completion to a harmonic Maass form can be computed using

```
w.maass_eisenstein_series(k, prec)
```

Example. The scalar-valued Maass Eisenstein series of weight (-2) for the full group $\mathrm{SL}_2(\mathbb{Z})$ can be computed with

```
WeilRep([]).maass_eisenstein_series(-2, 5)
```

which yields

```
Harmonic Maass form with holomorphic part pi^(-3) times
-45/2*zeta(3) - 45/2*q - 405/16*q^2 - 70/3*q^3 - 3285/128*q^4 + O(q^5)
and shadow
3 + 720*q + 6480*q^2 + 20160*q^3 + 52560*q^4 + O(q^5)
```

Compare this with the explicit formula

$$\tilde{E}_{-2}(\tau) = y^3 - \frac{45}{2\pi^3}\Big[\zeta(3) + \sum_{n=1}^{\infty} \sigma_{-3}(n)\Big(q^n + \overline{q}^n(1 + 4\pi n y + 8\pi^2 n^2 y^2)\Big)\Big].$$

(Here $q = e^{2\pi i \tau}$ and $\overline{q} = e^{2\pi i(-\overline{\tau})}$, and $\tau = x + iy$.)

Example. Let (L, Q) be the lattice $\mathbb{Z}^2$ with Gram matrix $\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$. The Maass Eisenstein series of weight (-1) is computed by typing

```
WeilRep([[2, 1], [1, 2]]).maass_eisenstein_series(-1, 5)
```

which outputs

```
Harmonic Maass form with holomorphic part pi^(-2) times
[(0, 0), -81/8*L(2, -3) - 9*q - 135/16*q^2 - 10*q^3 + O(q^4)]
[(2/3, 2/3), -243/32*q^(2/3) - 243/25*q^(5/3) - 4131/512*q^(8/3) - 1215/121*q^(11/3) + O(q^(14/3))]
[(1/3, 1/3), -243/32*q^(2/3) - 243/25*q^(5/3) - 4131/512*q^(8/3) - 1215/121*q^(11/3) + O(q^(14/3))]
and shadow
[(0, 0), -2 + 180*q + 432*q^2 + 1476*q^3 + O(q^4)]
[(1/3, 1/3), 18*q^(1/3) + 234*q^(4/3) + 900*q^(7/3) + 1296*q^(10/3) + O(q^(13/3))]
[(2/3, 2/3), 18*q^(1/3) + 234*q^(4/3) + 900*q^(7/3) + 1296*q^(10/3) + O(q^(13/3))]
```

Here the constant term of the holomorphic part involves the L-value

$$L(2,-3) = \sum_{n=1}^{\infty} \chi_{-3}(n) n^{-2} \approx 0.781302412896486.$$

To get a numerical approximation use the method $.n()$, i.e.

```
WeilRep([[2, 1], [1, 2]]).maass_eisenstein_series(-1, 2).n()
```

for which the output is:

```
Harmonic Maass form with holomorphic part
[(0, 0), -0.801520163230027 - 0.911890652781040*q + O(q^2)]
[(2/3, 2/3), 0.000000000000000*q^(-1/3) - 0.769407738284002*q^(2/3) - 0.984841905003523*q^(5/3) + O(q^(8/3))]
[(1/3, 1/3), 0.000000000000000*q^(-1/3) - 0.769407738284002*q^(2/3) - 0.984841905003523*q^(5/3) + O(q^(8/3))]
and shadow
[(0, 0), -2.00000000000000 + 180.000000000000*q + O(q^2)]
[(1/3, 1/3), 0.000000000000000*q^(-2/3) + 18.0000000000000*q^(1/3) + 234.000000000000*q^(4/3) + O(q^(7/3))]
[(2/3, 2/3), 0.000000000000000*q^(-2/3) + 18.0000000000000*q^(1/3) + 234.000000000000*q^(4/3) + O(q^(7/3))]
```

(Note: the multiplier π^{-2} is applied automatically!)

2.7.3 Maass Poincaré series

Let (L, Q) be an even lattice and $\beta \in L'/L$. Let $m \in \mathbb{Z} - Q(\beta)$ be an index (similarly to the holomorphic Poincaré series) with $m < 0$. The **Maass Poincaré series** of weight $k \in \frac{1}{2}\mathbb{Z}$ (where k is integral if L has even rank; and k is half-integral otherwise) is defined by Poincaré averaging:

$$F_{k,m,\beta}(\tau) := \sum_{M \in \Gamma_\infty \backslash \Gamma} \left(\phi_{k,m}(\tau) \mathfrak{e}_\beta \right) \Big|_{k,\rho} M,$$

where

$$\phi_{k,m}(\tau) := (1-k) \cdot q^m \gamma(1-k, 4\pi|m|y), \quad \tau = x + iy$$

and γ is the *lower* incomplete Gamma function:

$$\gamma(s,z) := \int_0^z t^{s-1} e^t t^{s-1} \, \mathrm{d}t = \Gamma(s) e^{-z} \sum_{n=0}^{\infty} \frac{z^{n+s}}{\Gamma(n+s+1)}, \ \mathrm{Re}[s] > 0.$$

This defines a harmonic weak Maass form whose shadow is (a simple multiple of) the Poincaré series $P_{k,-m,\beta}$ for the Weil representation associated to the lattice $(L, -Q)$.

Similarly to the method *poincare_series()* it can be computed by calling

```
w.maass_poincare_series(k, b, m, prec, nterms = 50)
```

where $b = \beta$ is the appropriate element of L'/L; m is the index; *prec* is the precision (the highest power of q in the Fourier expansion); and *nterms* is the number of terms used in the coefficient formula (an infinite sum over values of Bessel functions and Kloosterman sums, cf. Proposition 1.9 of [6]); by default we compute using the first 50 terms in this series. The holomorphic part can be called with

```
w.mock_poincare_series(k, b, m, prec, nterms = 50)
```

The first Fourier coefficients are generally more accurate.

Example. The scalar-valued Maass Poincaré series of weight (-2) and index (-1), using the first 50 terms in the coefficient formula. Input:

```
WeilRep([]).maass_poincare_series(-2, vector([]), -1, 2)
```

Output:

```
Harmonic weak Maass form with holomorphic part
1.00000000000000*q^-1 - 240.000000000000 - 141443.999817774*q - 8.52927999981621e6*q^2 + O(q^3)
and shadow
0.000000000000000 + 0.318058316204658*q - 3.75224055055394*q^2 + O(q^3)
```

In fact, the Maass Poincaré series is

$$E_{10}(\tau)/\Delta(\tau) = q^{-1} - 240 - 141444q - 8529280q^2 - ...$$

and the shadow is zero.

2.8 Visualization

The *plot()* and *plot_q()* methods provide phase plots of vector-valued modular forms. This is inspired by the paper [13] which considers plotting modular forms in much more detail.

Suppose f is a vector-valued modular form (or quasi-modular form; almost-holomorphic modular form; mock modular form; etc.) The method

```
f.plot()
```

by default creates a list of complex plots for each of the components of $f(\tau)$ on the domain

$$\{\tau = x + iy : \ -1 \leq x \leq 1, \ 0 < y \leq 2\}.$$

This accepts all optional arguments for Sage's *complex_plot()* (e.g. plot_points, show_axes, etc.) as well as the following optional arguments:
x_range: the range for x (by default $[-1, 1]$);
y_range: the range for y (by default $[0.01, 2]$);
show: a Boolean (default True). If True, then we attempt to show all plots as they are created, with a description of the vector component, before returning the list of plots.
isotherm: a Boolean (default True). If True then the magnitude of f is indicated by isotherm lines rather than brightness.
function: if this is given, it should be a function that accepts complex vectors and outputs complex numbers. We apply this function first and then plot. For example to sum the components together use

```
f.plot(function = sum)
```

Most forms can be plotted to reasonable accuracy with very few Fourier coefficients. To get better pictures it is usually more important to adjust the plot_points parameter.

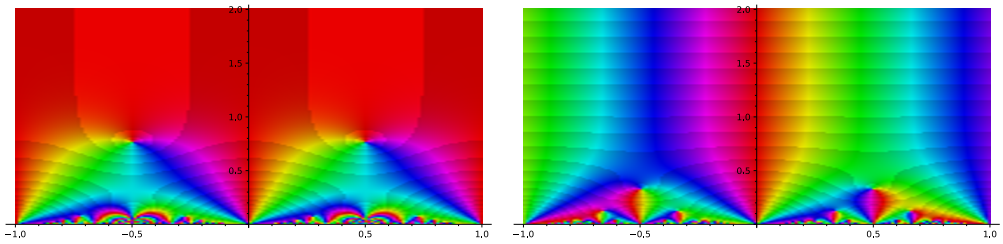
Example. The Cohen Eisenstein series of weight $7/2$ can be constructed as a vector-valued modular form to precision $O(q^5)$, i.e.

$$(1+126q+756q^2+2072q^3+4158q^4+...)\mathfrak{e}_0+(56q^{3/4}+576q^{7/4}+1512q^{11/4}+4032q^{15/4}+5544q^{19/4}+...)\mathfrak{e}_{1/2}$$

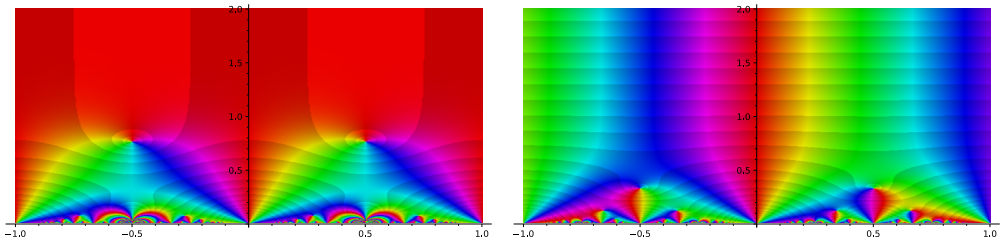
by entering

```
f = WeilRep([2]).eisenstein_series(7/2, 5)
```

Plotting with default settings (i.e. *f.plot()*) yields the pictures



for the components of $\mathfrak{e}_0$ and $\mathfrak{e}_{1/2}$, respectively. Raising the parameter *plot_points* to 400 yields the better pictures



Both examples use only the first five Fourier coefficients.

The method

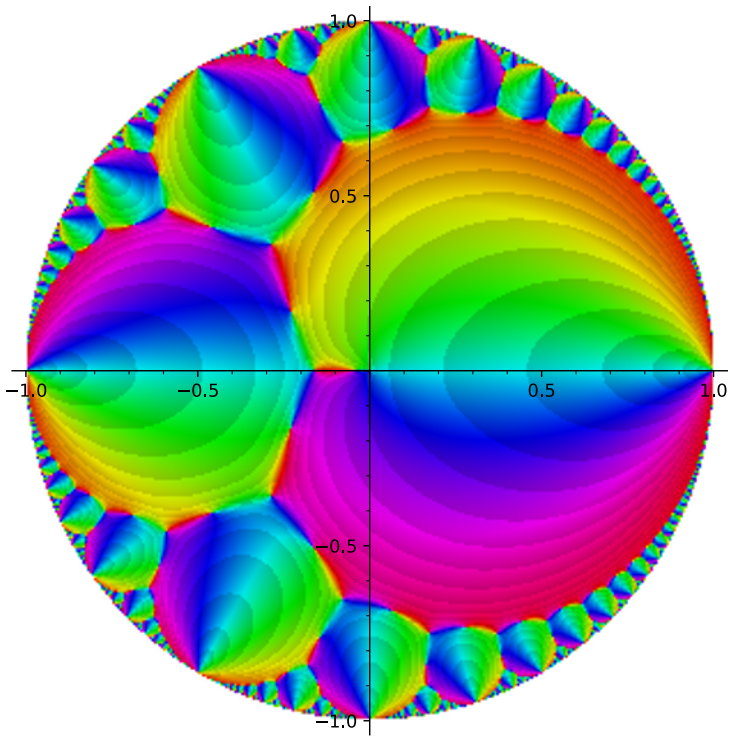
```
f.plot_q()
```

creates a similar list of complex plots of the components of f; this time as functions of q on the unit disc. This accepts the same optional parameters as *f.plot()*, with the exception of *x_range* and *y_range* (which are both fixed to $[-1,1]$).

Example. A plot of the scalar Maass Eisenstein series of weight -2. Entering

```
f = WeilRep([]).maass_eisenstein_series(-2, 5)
f.plot_q(plot_points = 300)
```

produces the image



3 Jacobi forms

Let $M = (M, Q)$ be a positive-definite even lattice. A *Jacobi form* of weight $k \in \mathbb{Z}$ and index m is a holomorphic function

$$\phi = \phi(\tau, z) : \mathbb{H} \times (m \otimes \mathbb{C}) \longrightarrow \mathbb{C}$$

that satisfies

$$\phi\Big(\frac{a\tau + b}{c\tau + d}, \frac{z}{c\tau + d}\Big) = (c\tau + d)^k \exp\Big(2\pi i Q(z)c/(c\tau + d)\Big)\phi(\tau, z)$$

for all $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$, and

$$\phi(\tau, z + \tau\lambda) = \exp\Big(-2\pi i\tau Q(\lambda) - \langle\lambda, z\rangle\Big)\phi(\tau, z), \quad \phi(\tau, z + \lambda) = \phi(\tau, z)$$

for all $\lambda \in M$, as well as a vanishing condition on Fourier coefficients: the expansion of ϕ takes the form

$$\phi(\tau, z) = \sum_{n=0}^{\infty} \sum_{\lambda \in M'} c(\lambda, n) q^n \zeta^\lambda, \quad q := e^{2\pi i\tau}, \ \zeta^\lambda := e^{2\pi i\langle\lambda, z\rangle},$$

and $c(\lambda, n) = 0$ if $Q(\lambda) > n$.

Jacobi forms of scalar index $m \in \mathbb{N}$ (as in [11]) are the same as Jacobi forms for the lattice $M = \mathbb{Z}$ with quadratic form $Q(x) = mx^2$.

Construct the module of Jacobi forms of index m with

```
JacobiForms(m)
```

where m is either a positive-definite Gram matrix or a positive integer.

3.1 Basic attributes

Let

```
j = JacobiForms(m)
```

be a JacobiForms instance. The following basic attributes can be computed.

3.1.1 Index

```
j.index()
```

returns the index (as a matrix, or as a number if the matrix is one-dimensional).

```
j.index_matrix()
```

always returns the index as a matrix.

3.1.2 Weil representation

```
j.theta_decomposition()
```

returns the Weil representation corresonding to this module of Jacobi forms.

3.2 Construction of Jacobi forms

Suppose j is a JacobiForms instance: $j = JacobiForms(m)$.

3.2.1 Jacobi Eisenstein series

The Jacobi Eisenstein series is the sum

$$\sum_{M \in \mathcal{J}_\infty \backslash \mathcal{J}} 1\Big|_{k,m} M,$$

where $\mathcal{J}$ is the Jacobi group and $\mathcal{J}_\infty$ the stabilizer of ∞, and $|_{k,m}$ is the Petersson slash operator. cf. chapter 2 of [11].

```
j.eisenstein_series(k, prec)
```

computes the Jacobi Eisenstein series of weight k to precision *prec* (with respect to the $q = e^{2\pi i \tau}$ variable). For example the Jacobi Eisenstein series $E_{4,1}$ of weight 4 and index 1 can be computed to precision $O(q^5)$ with

```
JacobiForms(1).eisenstein_series(4, 5)
```

More directly,

```
jacobi_eisenstein_series(k, m, prec)
```

constructs the Jacobi Eisenstein series of weight k and index m.

Similarly the commands

```
j.eisenstein_oldform()
```

and

```
j.eisenstein_newform()
```

can be used to compute other Eisenstein series.

3.2.2 Theta blocks

Theta blocks were introduced in [12]. For $\mathbf{a} = (a_1, ..., a_r) \in \mathbb{Z}\backslash\{0\}$ and $n \in \mathbb{Z}$ define

$$\vartheta_{\mathbf{a},n} := \eta^n(\tau) \prod_{i=1}^{r} \vartheta(\tau, a_i z)$$

where

$$\eta(\tau) = q^{1/24} \prod_{n=1}^{\infty} (1 - q^n) = \sum_{n=1}^{\infty} \binom{12}{n} q^{n^2/24}$$

and

$$\vartheta(\tau, z) = q^{1/8} \zeta^{1/2} \prod_{n=1}^{\infty} (1 - q^n)(1 - q^n\zeta)(1 - q^{n-1}\zeta^{-1}) = \sum_{n \in \mathbb{Z}} \binom{-4}{n} q^{n^2/8} \zeta^{n/2}.$$

When $\sum_{i=1} a_i$ is even and $r/8 + n/24 \in \mathbb{Z}$ this transforms like a Jacobi form. (Whether it is holomorphic in the cusps is a trickier point.)

Theta blocks are implemented with the command *theta_block(a,n,prec).*

Example (from [12]). The (unique, up to scalar) Jacobi form of weight 2 and index 25 is the theta block with $\mathbf{a} = [1, 1, 1, 1, 2, 2, 2, 3, 3, 4]$ and $n = -6$. To compute it (here only to precision $O(q^2)$): Input:

```
print(theta_block([1,1,1,1,2,2,2,3,3,4],-6,2))
```

Output:

```
(w_0^-10 - 4*w_0^-9 + 3*w_0^-8 + 6*w_0^-7 - 7*w_0^-6 - 2*w_0^-5 - 4*w_0^-4 + 10*w_0^-3 + 6*w_0^-2
- 10*w_0^-1 + 2 - 10*w_0 + 6*w_0^2 + 10*w_0^3 - 4*w_0^4 - 2*w_0^5 - 7*w_0^6 + 6*w_0^7 + 3*w_0^8
- 4*w_0^9 + w_0^10)*q + O(q^2)
```

3.2.3 Jacobi Poincaré series

```
j.poincare_series(k, n, r, prec, nterms)
```

computes the Jacobi Poincaré series:

$$f(\tau, z) = \sum_{M \in \mathcal{J}_\infty \backslash \mathcal{J}} (q^n \zeta^r)\Big|_{k,m} M$$

of index (n, r). Here $n \in \mathbb{Z}$ should be an integer and $r \in \mathbb{Z}^N$ should be a vector (or an integer, if $N = 1$), where N is the index rank i.e. number of abelian variables. The procedure is the same as the vector-valued Poincaré series: the Fourier coefficients are series over Bessel function values and Kloosterman sums, and we compute approximations by truncating these series at *nterms.*

3.2.4 Jacobi forms from vector-valued modular forms

If f is a vector-valued modular form for the Weil representation attached to a positive-definite Gram matrix m, then f comes with the method

```
f.jacobi_form()
```

which produces the Jacobi form of index m whose theta decomposition is f itself.

3.3 Spaces of Jacobi forms

Again let

```
j = JacobiForms(m)
```

be a JacobiForms instance.

3.3.1 Jacobi forms

The method

```
j.basis(k, prec)
```

computes a basis of (holomorphic) Jacobi forms of weight k for the given index with Fourier expansions to precision *prec* (with respect to the $q = e^{2\pi i\tau}$ variable).

Generally we use whatever methods are avaiable to compute bases of spaces of vector-valued modular forms and then pass to the associated Jacobi forms. For scalar-index Jacobi forms of low weights (2 and 3) we sometimes look through families of theta blocks first.

The method

```
j.dimension(k)
```

computes the dimension of the space of (holomorphic) Jacobi forms of weight k using the Riemann-Roch formula.

Jacobi forms of all weights form a finite free $\mathbb{C}[E_4, E_6]$-module so the Hilbert series of dimensions has the form

$$\text{Hilb} = \sum_{k\geq 0} \dim J_k \, t^k = \frac{P(t)}{(1-t^4)(1-t^6)}$$

for some polynomial $P \in \mathbb{C}[t]$. The method

```
j.hilbert_series()
```

computes the Hilbert series as a power series. The method

```
j.hilbert_polynomial()
```

computes the polynomial P.

3.3.2 Cusp forms

The method

```
j.cusp_forms_basis(k, prec)
```

computes a basis of Jacobi cusp forms of weight k for the given index with Fourier expansions to precision *prec* (with respect to the $q = e^{2\pi i\tau}$ variable).

Generally we use whatever methods are avaiable to compute bases of spaces of vector-valued cusp forms and then pass to the associated Jacobi forms. For scalar-index Jacobi forms of low weights (2 and 3) we sometimes look through families of theta blocks first.

The method

```
j.cusp_forms_dimension(k)
```

computes the dimension of the space of Jacobi cusp forms of weight k using the Riemann-Roch formula.

3.3.3 Weak Jacobi forms

A weak Jacobi form is a holomorphic function that transforms like a Jacobi form and has Fourier expansion of the form

$$\phi(\tau, z) = \sum_{n=0}^{\infty} \sum_{r} c(n, r) q^n \zeta^r$$

but without restrictions on r.

The method

```
j.weak_forms_basis(k, prec)
```

computes a basis of weak Jacobi forms of weight k for the given index with Fourier expansions to precision *prec*. (Here the weight k may be negative!)

The method

```
j.weak_forms_dimension(k)
```

computes the dimension J_k^w of the space of weak Jacobi forms of weight k.

Weak Jacobi forms of all weights form a finite free $\mathbb{C}[E_4, E_6]$-module, so the Hilbert series of dimensions has the form

$$\mathrm{Hilb}^w = \sum_{k \gg -\infty} \dim J_k^w \, t^k = \frac{P(t)}{(1-t^4)(1-t^6)}$$

for some Laurent polynomial $P \in \mathbb{C}[t, t^{-1}]$. The method

```
j.weak_hilbert_series()
```

computes the weak Hilbert series Hilb^w as a power series. The method

```
j.weak_hilbert_polynomial()
```

computes the Laurent polynomial P.

3.4 Operations on Jacobi forms

3.4.1 Arithmetic operations

Jacobi forms of the same weight and index can be added and subtracted. Jacobi forms whose indices have the same rank can be multiplied in the usual way. Also Jacobi forms can be multiplied by modular forms (i.e. ModularFormElements).

3.4.2 Direct product

Call the *direct product* of two Jacobi forms f (of index m_1) and g (of index m_2) the Jacobi form in $\mathrm{rank}(m_1)+\mathrm{rank}(m_2)$ elliptic variables obtained by inserting distinct elliptic variables in f and g and multiplying i.e.

$$(f \times g)(\tau, (z_1, z_2)) = f(\tau, z_1) \cdot g(\tau, z_2).$$

This is a Jacobi form of index $m_1 \oplus m_2$. It is implemented as the operation $**$.

Example. Let $E_{4,1}$ be the Jacobi Eisenstein series of weight 4 and index 1. We will take its direct product with itself. Input:

```
E41 = jacobi_eisenstein_series(4,1,2)
E41 ** E41
```

Output:

```
1 + (w_0^2 + w_1^2 + 56*w_0 + 56*w_1 + 252 + 56*w_1^-1 + 56*w_0^-1 + w_1^-2 + w_0^-2)*q + O(q^2)
```

3.4.3 Hecke operators

The Hecke operators T_N are currently only implemented for values N that are coprime to the level of the lattice. (Warning: when the index m is a scalar then the level is $4m$!) This is the method *hecke_T(N)*. The formula is taken from section 2.6 of A. Ajouz's thesis [1].

Example. The image of $E_{4,1}$ under T_3. Input:

```
E41 = jacobi_eisenstein_series(4, 1, 40)
print(E41.hecke_T(3))
```

The Hecke U-operators are defined by

$$\Big(\phi|U_N\Big)(\tau,z) = \phi(\tau,Nz).$$

If ϕ has index m then $\phi|U_N$ has index N^2m. This is implemented with the method *hecke_U(N)*.

Example. The image of $E_{4,1}$ under U_2. Input:

```
E41 = jacobi_eisenstein_series(4,1,3)
print(E41.hecke_U(2))
```

Output:

```
1 + (w_0^-4 + 56*w_0^-2 + 126 + 56*w_0^2 + w_0^4)*q
+ (126*w_0^-4 + 576*w_0^-2 + 756 + 576*w_0^2 + 126*w_0^4)*q^2 + O(q^3)
```

The Hecke V-operators are defined by

$$\Big(\phi|V_N\Big)(\tau,z) = N^{k-1}\sum_M (c\tau+d)^{-k} e^{-2\pi i N c Q(z)/(c\tau+d)}\phi\Big(\frac{a\tau+b}{c\tau+d}, \frac{Nz}{c\tau+d}\Big)$$

where $M = \left(\begin{smallmatrix} a & b \\ c & d\end{smallmatrix}\right)$ runs through $\mathrm{SL}_2(\mathbb{Z})$-cosets of matrices with integer entries and determinant N. If ϕ has index m then $\phi|V_N$ has index Nm. This is implemented with the method *hecke_V(N)*.

Example. The image of $E_{4,1}$ under V_2. Input:

```
E41 = jacobi_eisenstein_series(4,1,5)
print(E41.hecke_V(2))
```

Output:

```
9 + (126*w_0^-2 + 576*w_0^-1 + 756 + 576*w_0 + 126*w_0^2)*q
+ (9*w_0^-4 + 576*w_0^-3 + 2520*w_0^-2 + 4032*w_0^-1 + 5166 + 4032*w_0 + 2520*w_0^2 + 576*w_0^3 + 9*w_0^4)*q^2
+ O(q^3)
```

3.4.4 is_cusp_form, is_holomorphic

Suppose f is a (weak, weakly holomorphic) Jacobi form. The methods *f.is_cusp_form()* and *f.is_holomorphic()* attempt to determine from the Fourier expansion of f whether f is a cusp form resp. a holomorphic Jacobi form.

Note: unlike regular modular forms, these properties cannot be determined from the q-expansion up to exponent 0. The methods *f.is_cusp_form()* and *f.is_holomorphic()* are rigorous - we do not guess! If we cannot determine whether the form is holomorphic (or cuspidal) from the known coefficients then we raise a ValueError (and suggest a better precision to use).

3.4.5 Pullback

Let $f(\tau, \mathbf{z})$ be a Jacobi form with $\mathbf{z} \in \mathbb{C}^d$ and let $a \in \mathbb{Z}^{c\times d}$ be a matrix of rank c. The method *pullback(a)* computes the Jacobi form $f(\tau, a\mathbf{z})$. If f has index m then its pullback along a has index ama^T (and the same weight).

Example. The pullback of the weight 4 Jacobi Eisenstein series of index $\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ along the matrix $a = (1, 2)$ is a Jacobi form of index 7. Input:

```
f = jacobi_eisenstein_series(4,matrix([[2,1],[1,2]]),2)
f.pullback(matrix([[1,2]]))
```

Output:

```
1 + (w_0^-5 + w_0^-4 + 27*w_0^-3 + 27*w_0^-2 + 28*w_0^-1 + 72
+ 28*w_0 + 27*w_0^2 + 27*w_0^3 + w_0^4 + w_0^5)*q + O(q^2)
```

3.4.6 Theta decomposition

The method

```
f.theta_decomposition()
```

returns the Theta decomposition of a Jacobi form f as a vector-valued modular form, cf. chapter 5 of [11].

3.4.7 Zero-values

Let $f(\tau, z_0, ..., z_{d-1})$ be a Jacobi form in d elliptic variables. The method *substitute_zero(indices)* returns the Jacobi form obtained by setting some subset of the z_i to zero. (In other words setting $w_i = e^{2\pi i z_i}$ to 1.) *indices* should be a list containing distinct numbers between 0 and $d - 1$.

Example. The zero-value of the Jacobi Eisenstein series of weight four and index 1. Input:

```
E41 = jacobi_eisenstein_series(4,1,5)
E41.substitute_zero([0])
```

Output:

```
1 + 240*q + 2160*q^2 + 6720*q^3 + 17520*q^4 + O(q^5)
```

(This is a JacobiForm instance of index (empty matrix).)

The higher Taylor coefficients about zero are not (yet?) implemented.

3.5 Recovering Fourier coefficients from Jacobi forms

Let f be a JacobiForm instance.

The method

```
f.coefficient_vector()
```

outputs the Fourier coefficients $c(n,r)$ of f without redundancy as a vector sorted by increasing value of $n - r^2/4m$ (and its generalization to lattice index Jacobi forms).

The method

```
f.q_coefficients()
```

outputs the Fourier coefficients of f with respect to q *only* as a list of multivariate Laurent polynomials in variables $w_i = e^{2\pi i z_i}$.

The method

```
f.fourier_expansion()
```

outputs the Fourier expansion of f as a power series in q.

4 Modular forms on orthogonal groups

The class *OrthogonalModularForms* represents spaces of modular forms on the Type IV domain attached to an even lattice (L, Q). For now we only allow lattices that are split by a hyperbolic plane over $\mathbb{Z}$; i.e. those of the form

$$L \oplus H$$

where H is the hyperbolic plane ($\mathbb{Z}^2$ with quadratic form $(x,y) \mapsto xy$) and where L is Lorentzian. In this section we only consider the case where L is itself of the form $K \oplus H$ with K positive-definite. (For general L see the next chapter.)

An *OrthogonalModularForms* instance can be constructed by calling

```
OrthogonalModularForms(S)
```

where S is a Gram matrix for the *positive-definite* part L, or

```
OrthogonalModularForms(w)
```

where w is a WeilRep instance for a positive-definite quadratic form.

Orthogonal modular forms for the Gram matrix S (of rank n) have Fourier expansions of the form

$$f(Z) = f(\tau, z, w) = \sum_{a,c=0}^{\infty} \sum_{b \in L} \alpha(a,b,c) q^a r^b s^c,$$

where $\tau, w \in \mathbb{H}$ and $z \in \mathbb{C}^n$ with $Q(\mathrm{im}(z)) < \mathrm{im}(\tau) \cdot \mathrm{im}(w)$, and where $q = e^{2\pi i \tau}$, $r = e^{2\pi i b^T z}$ and $s = e^{2\pi i w}$.

Note:

```
ParamodularForms(N)
```

is a shortcut for the OrthogonalModularForms instance with Gram matrix $S = ((2N))$.

4.1 Construction of modular forms

Let

```
m = OrthogonalModularForms(S)
```

be an orthogonal modular forms instance.

4.1.1 Eisenstein series

The method

```
m.eisenstein_series(k, prec)
```

computes the weight k Eisenstein series

$$E_k(Z) = \sum_{M \in \Gamma_{S,\infty} \backslash \Gamma_S} j(M; Z)^{-k},$$

where Γ_S is the orthogonal modular group; $j(M; Z)$ is the cocycle; and $\Gamma_{S,\infty}$ is the subgroup of Γ_S that leaves the constant 1 invariant. (This is actually computed as a theta lift.)

Here k should be an even integer (sufficiently large) and *prec* denotes the precision with respect to both q and s.

Example. The Siegel Eisenstein series of degree 2 and weight 4 up to precision 5. Input:

```
ParamodularForms(1).eisenstein_series(4, 5)
```

4.1.2 Additive theta lift

Let L be a positive-definite even lattice of rank n and let $M_k(\Gamma_L)$ denote the space of modular forms of weight k for the Type IV domain attached to $L \oplus \mathrm{II}_{2,2}$. The additive theta lift is a linear map

$$\Phi : M_{k-n/2}(\rho_L) \longrightarrow M_k(\Gamma_L).$$

Also Φ takes cusp forms to cusp forms. This is implemented with the method *theta_lift()*.

Example. The theta lift of the weight 8 cusp form attached to the A_2 root lattice. Input:

```
w = WeilRep(matrix([[2, 1], [1, 2]]))
f = w.cusp_forms_basis(8, 5)[0]
f.theta_lift()
```

4.1.3 Borcherds lift

Let L be a positive-definite even lattice of rank n. The Borcherds lift [2] is a multiplicative map

$$\Phi : M^!_{-n/2}(\rho_L) \longrightarrow M_*(\Gamma_L)$$

where $M^!_{-n/2}$ denotes nearly-holomorphic modular forms. The result may transform with a character. This is implemented with the method *borcherds_lift()*.

Example. The product of ten theta-constants is a Siegel modular form of degree two with weight 5 with a character of order two. Construction as a Borcherds product: input

```
w = WeilRep(matrix([[2]]))
f = w.nearly_holomorphic_modular_forms_basis(-1/2, 1/4, 5)[0]
f.borcherds_lift()
```

4.1.4 Gritsenko lift

If f is a Jacobi form then its Gritsenko lift

$$\Phi(f) = \sum_{n=0}^{\infty} (f|V_n)s^n$$

is an orthogonal modular form of the same weight. This is closely related to the additive theta lift. It is implemented with the method *gritsenko_lift()*.

Example. The Gritsenko lift of the Jacobi cusp form of index 2 and weight 11. Input:

```
f = JacobiForms(2).cusp_forms_basis(11, 5)[0]
f.gritsenko_lift()
```

4.1.5 Spezialschar

The method *spezialschar(k, prec)* computes a basis of the Maass Spezialschar of weight k to precision *prec* (i.e. cusp forms which are additive lifts).

Example. The Spezialschar of Siegel modular forms of degree two and weight 10. Input:

```
ParamodularForms(1).spezialschar(10, 5)
```

Calling *spezialschar* with no arguments produces the Maass Spezialschar as an abstract object. This can be used to test whether a modular form is a lift. For example, construct the Siegel modular form Borcherds product of weight 24 as follows:

```
f = WeilRep([2]).nearly_holomorphic_modular_forms_basis(-1/2, 5/4, 10)[2]
f = f.borcherds_lift()
```

and test whether it is a Maass lift (it is not) using

```
f in ParamodularForms(1).spezialschar()
```

4.2 Representations of modular forms

4.2.1 Fourier expansion

The Fourier expansion of the modular form can be recovered with *fourier_expansion()*. The result is a power series in variables q, s over a ring of Laurent polynomials in the variables $r_0, ..., r_{n-1}$ where n is the rank of the lattice.

4.2.2 Fourier–Jacobi expansion

The Fourier–Jacobi expansion of the modular form f is the representation

$$f(\tau, z, w) = \sum_{n=0}^{\infty} \phi_n(\tau, z)s^n, \quad s = e^{2\pi i w}$$

where each ϕ_n is a Jacobi form of the same weight. This is implemented with *fourier_jacobi()*. The result is a list of JacobiForm instances.

Example. The Fourier–Jacobi expansion of the Siegel Eisenstein series of degree 2 and weight 4. Input:

```
E4 = ParamodularForms(1).eisenstein_series(4, 5)
E4.fourier_jacobi()
```

4.2.3 Coefficient dictionary

The method *coefficients()* produces a dictionary of the modular form f's Fourier coefficients.

4.3 Inputs for Borcherds products

Let

```
m = OrthogonalModularForms(S)
```

be an OrthogonalModularForms instance.

4.3.1 Finding all holomorphic products of a given weight

The method *borcherds_input_by_weight(k, prec)* outputs a list of all nearly-holomorphic vector-valued modular forms (to precision *prec*) whose Borcherds lifts are holomorphic and have weight k.

Example. To find the two Siegel modular forms of degree 2 and weight 35 that are Borcherds products: input

```
ParamodularForms(1).borcherds_input_by_weight(35, 5)
```

which produces the input functions to precision $O(q^5)$.

4.3.2 Bases of holomorphic products

Let $D \in \mathbb{R}_{>0}$ be a bound. The method *borcherds_input_basis(D, prec)* outputs a list $(F_1, ..., F_N)$ of nearly-holomorphic vector-valued modular forms (to precision *prec*) with a pole at ∞ of order at most D, and which is minimal with the following property: the input functions F with pole order at most D whose Borcherds lifts are holomorphic are exactly the linear combinations

$$F = k_1 F_1 + ... + k_N F_N,\ k_i \in \mathbb{N}_0.$$

Example. To compute the semigroup of holomorphic Borcherds products for the paramodular group of level $N = 5$ with zeros on Humbert surfaces of discriminant at most 10. Input:

```
ParamodularForms(5).borcherds_input_basis(1/2, 5)
```

Taking the Borcherds lifts of these forms yields 13 semigroup generators. These generators have weights $4, 5, 10, 11, 17, 18, 23, 24, 29, 30, 36, 42, 48$.

The method *borcherds_input_Qbasis(D, prec)* is similar, but the list $(F_1, ..., F_N)$ it outputs is minimal with the following weaker property: every input function F with pole order at most D whose Borcherds lift is holomorphic is a linear combination

$$F = \lambda_1 F_1 + ... + \lambda_N F_N$$

where $\lambda_i \in \mathbb{Q}_{\geq 0}$. Computing a $\mathbb{Q}$-basis in this sense is often much faster than computing a true Hilbert basis.

4.4 Other methods

4.4.1 Jacobian

Suppose L has rank n and a list $F_0, ..., F_n$ of $(n+1)$ orthogonal modular forms of weights $k_0, ..., k_n$ is given. The function

```
jacobian([F_0,...,F_n])
```

computes the modular Jacobian, or Rankin–Cohen–Ibukiyama operator:

$$J(F_0, ..., F_n) = \det \begin{pmatrix} k_0 F_0 & ... & k_n F_n \\ \nabla F_0 & ... & \nabla F_n \end{pmatrix}.$$

Example. The Jacobian of the Siegel Eisenstein series E_4, E_6, E_{10}, E_{12} is the Siegel cusp form of weight 35. Check this with

```
m = ParamodularForms(1)
C = -589927441461779261030400000/2354734631251
jacobian([m.eisenstein_series(k, 7) for k in [4, 6, 10, 12]]) / C
```

(We divide at the end to remove the funny multiple from the Fourier coefficients.)

4.4.2 Linear relations

The method *omf_rank(X)* determines the dimension of the space spanned by the list of orthogonal modular forms X. (Warning: we only check the rank using the known coefficients! This does not use any sort of Sturm bound so it cannot prove that a given set of modular forms does not have full rank!)

The method *omf_relations(X)* determines the linear relations satisfied by the list of orthogonal modular forms X. (A similar warning applies here; we only check that the relations hold among all known coefficients.)

Example. We check that the square of the Siegel Eisenstein series of weight 4 equals the Siegel Eisenstein series of weight 8.

```
m = ParamodularForms(1)
e4 = m.eisenstein_series(4, 5)
e8 = m.eisenstein_series(8, 5)
omf_relations([e4 * e4, e8])
```

4.4.3 Maass relations

The method *f.is_lift()* tests whether the given orthogonal modular form f satisfies the Maass relations, i.e. whether it is a lift.

Example. Test whether the Borcherds product of weight 9 associated to the Gram matrix $\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ satisfies the Maass relations:

```
m = OrthogonalModularForms(matrix([[2, 1], [1, 2]]))
X = m.borcherds_input_Qbasis(1/3, 15)[0]
X.borcherds_lift().is_lift()
```

4.4.4 Siegel Phi operator

The Siegel Phi operator is a linear map

$$\Phi : M_k(\Gamma_L) \to M_k, \;\; \Phi f(\tau) := \lim_{w \to i\infty} f(\tau, 0, w).$$

(In other words Φf is the constant term of f's Fourier–Jacobi expansion.) This is given by the method *f.phi()*.

Note: the result is implemented as an orthogonal modular form of weight $k/2$ for the Lorentzian lattice $\mathbb{Z}$ with generator of norm -1 (see the next chapter). This corresponds to weight k for $\mathrm{Mp}_2(\mathbb{Z})$.

4.4.5 Witt operator

The Wiitt operator is a linear map

$$W : M_k(\Gamma_L) \to M_k(\mathrm{Mp}_2(\mathbb{Z}) \times \mathrm{Mp}_2(\mathbb{Z})), \;\; Wf(\tau, w) := f(\tau, 0, w).$$

(In other words we evaluate all terms in f's Fourier–Jacobi expansion at $\mathfrak{z} = 0$.) This is given by the method *f.witt()*.

Note: the result is implemented as an orthogonal modular form for the Lorentzian lattice $\mathbb{Z}^2$ with quadratic form $Q(x, y) = x(y - x)$.

5 Modular forms on orthogonal groups II

In this chapter we compute with orthogonal modular forms attached to more general lattices of the form $L + \mathrm{II}_{1,1}(N)$, where $\mathrm{II}_{1,1}(N)$ is the hyperbolic plane rescaled by N (i.e. the plane $\mathbb{Z}^2$ with quadratic form $(x, y) \mapsto Nxy$) and where L is a Lorentzian lattice i.e. of signature $(l - 1, 1)$. Construct an OrthogonalModularForms instance as follows:

(1) If $N = 1$, then

```
OrthogonalModularForms(S)
```

where S is a Gram matrix for the Lorentzian part L, or

```
OrthogonalModularForms(w)
```

where w is a WeilRep instance for a Lorentzian quadratic form.

(2) For general N, call

```
OrthogonalModularForms(w + II(N))
```

where w is the WeilRep for the Lorentzian part L.

Warning: for most applications we need to fix a generator of the negative cone attached to L. The program always tries to choose the vector $(1, 0..., 0)$. In other words *the Gram matrix S must always have a strictly negative upper-left entry*; otherwise many things do not work! (In earlier versions we used the vector (0, ..., 0, 1) and considered the bottom-right entry!)

5.1 Construction of modular forms

Suppose m = *OrthogonalModularForms(w)* is an orthogonal modular forms instance where w is Lorentzian or of the form $w_0 + II(N)$ where w_0 is Lorentzian.

5.1.1 Eisenstein series

The method

```
m.eisenstein_series(k, prec)
```

computes the Fourier series of the Eisenstein series of weight k to precision *prec*.

5.1.2 Additive theta lifts

The additive theta lift of a modular form f of weight $1 + k - n/2$ is implemented with the method

```
f.theta_lift()
```

The result is an orthogonal modular form of weight k.

A basis of the space of cuspidal theta lifts can be computed using the method

```
m.lifts_basis(k, prec)
```

where k is the desired weight and *prec* is the precision to which the Fourier series is computed.

5.1.3 Borcherds lift

Suppose f is a nearly-holomorphic modular form of weight $1 - n/2$ with integral Fourier coefficients. The Borcherds product constructed from f is implemented with the method

```
f.borcherds_lift()
```

To compute Hilbert bases of input functions into the Borcherds lift, use the commands

```
m.borcherds_input_basis(pole_order, prec)
```

and

```
m.borcherds_input_Qbasis(pole_order, prec)
```

exactly as in section 4.3.2. (Generally the Qbasis is much faster.) Finding all products of a given weight is not (yet?) implemented.

Example. Suppose $L = \mathrm{II}_{1,1}$. The lift of $j(\tau) - 744$ is the modular function $j(\tau_1) - j(\tau_2)$. To check this, input:

```
j = OrthogonalModularForms(II(1)).borcherds_input_basis(1, 10)[1]
j.borcherds_lift()
```

Example. The lattice $A_1(2) + 2U(2)$ admits 10 products of singular weight 1 (cf. [14]). We can compute them as follows. (Warning: this computation is slow; just under 40 seconds on my computer)

```
w = WeilRep(matrix([[4]]))
m = OrthogonalModularForms(w + II(2) + II(2))
X = m.borcherds_input_basis(1/8, 15)
for x in X:
    print(x.principal_part())
    print(x.borcherds_lift())
    print('-' * 80)
```

This yields the 10 genus two theta constants.

5.2 Other functions

The functions *jacobian()*, *omf_rank()*, *omf_relations()* from section 4 can also be applied to the more general orthogonal modular forms considered here. The Siegel Phi and Witt operators can be applied to orthogonal modular forms for lattices constructed from WeilReps of the form $w + \mathrm{II}(n_1) + \mathrm{II}(n_2)$ with w positive-definite.

5.3 Special modular forms

5.3.1 Hilbert modular forms

Hilbert modular forms for real-quadratic number fields can be identified with certain orthogonal modular forms for subgroups of $\mathrm{O}(2,2)$.

If K is a real-quadratic number field then *HMF(K)* represents orthogonal modular forms for the (lorentzian) lattice of integers of K, which are essentially the same as Hilbert modular forms. Modular forms attached to *HMF* are represented as power series of the form

$$f(\tau_1, \tau_2) = \sum_{\nu} c(\nu) q_1^{\nu} q_2^{\nu'},$$

where $q_n = e^{2\pi i \tau_n}$ and where ν runs through totally-nonnegative elements of the dual lattice $\mathcal{O}_K^{\#}$ i.e. $\nu, \nu' \geq 0$ where ν' is the conjugate. As an example, for Hilbert modular forms over $\mathbb{Q}(\sqrt{5})$:

```
x = var('x')
K.<sqrt5> = NumberField(x^2 - 5)
h = HMF(K)
```

Some congruence subgroups are implemented using

```
h = HMF(K, level)
```

where level is a natural number.

Given a HilbertModularForm f on the full modular group you can compute its character by

```
f.character()
```

Fourier coefficients of f can be extracted using

```
f[a]
```

where $a \in \mathcal{O}_K^{\#}$ (or more generally in K, for Hilbert modular forms with characters)

The constructions of orthogonal modular forms from section 5 all apply here.

5.3.2 Siegel modular forms and Paramodular forms

For $N \in \mathbb{N}$, *ParamodularForms(N)* represents orthogonal modular forms on $A_1(N) + \mathrm{II}_{2,2}$. These have an interpretation as paramodular forms.

These are represented as orthogonal modular forms in the usual sense. The only extra feature is that their coefficients of f can be extracted using

```
f[A]
```

where, if f has trivial character then A is a symmetric half-integral matrix i.e. it has integral diagonal and half-integral off-diagonal entries. (If f has a character then the exponents A may be more general.)

5.3.3 Hermitian modular forms

Hermitian modular forms of degree two can be identified with certain orthogonal modular forms for subgroups of $\mathrm{O}(4,2)$.

Suppose K is an imaginary-quadratic number field. *HermitianModularForms(K)* represents orthogonal modular forms for the (positive-definite) lattice of integers $\mathcal{O}_K$ of K. More generally

```
HermitianModularForms(K, level = N)
```

represents orthogonal modular forms for the lattice $\mathcal{O}_K \oplus U(N) \oplus U$ where $U = \mathrm{II}_{1,1}$. These are represented as power series of the form

$$f\Big(\left(\begin{smallmatrix} \tau & z_1 \\ z_2 & w \end{smallmatrix}\right) \Big) = \sum_A c(A) q^a r_1^b r_2^{\bar{b}} s^c,$$

where $q = e^{2\pi i \tau}$, $r_i = e^{2\pi i z_i}$, $s = e^{2\pi i w}$, and where

$$A = \left(\begin{smallmatrix} a & b \\ \bar{b} & c \end{smallmatrix}\right)$$

runs through Hermitian half-integral matrices with values in K. The Fourier coefficients of f can be extracted with

```
f[A]
```

where A is a matrix.

References

[1] Ali Ajouz. Hecke operators on Jacobi forms of lattice index and the relation to elliptic modular forms. Dissertation (adviser N. Skoruppa, 2015.

[2] Richard Borcherds. Automorphic forms with singularities on Grassmannians. *Invent. Math.*, 132(3): 491–562, 1998.

[3] Richard Borcherds. The Gross-Kohnen-Zagier theorem in higher dimensions. *Duke Math. J.*, 97(2): 219–233, 1999.

[4] Vincent Bouchard, Thomas Creutzig, and Aniket Joshi. Hecke operators on vector-valued modular forms. *SIGMA Symmetry Integrability Geom. Methods Appl.*, 15:Paper 041, 31, 2019.

[5] Kathrin Bringmann, Amanda Folsom, Ken Ono, and Larry Rolen. *Harmonic Maass forms and mock modular forms: theory and applications*, volume 64 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, 2017.

[6] Jan Bruinier. *Borcherds products on O(2, l) and Chern classes of Heegner divisors*, volume 1780 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2002.

[7] Jan Bruinier and Michael Bundschuh. On Borcherds products associated with lattices of prime discriminant. *Ramanujan J.*, 7(1-3):49–61, 2003.

[8] Jan Bruinier and Michael Kuss. Eisenstein series attached to lattices and modular forms on orthogonal groups. *Manuscripta Math.*, 106(4):443–459, 2001.

[9] Raemeon Cowan, Daniel Katz, and Lauren White. A new generating function for calculating the Igusa local zeta function. *Adv. Math.*, 304:355–420, 2017.

[10] Stephan Ehlen and Nils-Peter Skoruppa. Computing invariants of the Weil representation. In *L-functions and automorphic forms*, volume 10 of *Contrib. Math. Comput. Sci.*, pages 81–96. Springer, Cham, 2017.

[11] Martin Eichler and Don Zagier. *The theory of Jacobi forms*, volume 55 of *Progress in Mathematics*. Birkhäuser Boston, Inc., Boston, MA, 1985.

[12] Valery Gritsenko, Nils-Peter Skoruppa, and Don Zagier. Theta blocks. URL `https://arxiv.org/pdf/1907.00188.pdf`.

[13] David Lowry-Duda. Visualizing modular forms. URL `https://arxiv.org/pdf/2002.05234`.

[14] Sebastian Opitz and Markus Schwagenscheidt. Holomorphic Borcherds products of singular weight for simple lattices of arbitrary level. *Proc. Amer. Math. Soc.*, 147(11):4639–4653, 2019.

[15] Martin Raum. Computing genus 1 Jacobi forms. *Math. Comp.*, 85(298):931–960, 2016.

[16] Markus Schwagenscheidt. Eisenstein series for the Weil representation. *J. Number Theory*, 193:74–90, 2018.

[17] Brandon Williams. Poincaré square series for the Weil representation. *Ramanujan J.*, 47(3):605–650, 2018.

[18] Brandon Williams. Vector-valued Eisenstein series of small weight. *Int. J. Number Theory*, 15(2): 265–287, 2019.

[19] Brandon Williams. A construction of antisymmetric modular forms for Weil representations. *Math. Z*, 296:391–408, 2020.

[20] Don Zagier. Traces of singular moduli. In *Motives, polylogarithms and Hodge theory, Part I (Irvine, CA, 1998)*, volume 3 of *Int. Press Lect. Ser.*, pages 211–244. Int. Press, Somerville, MA, 2002.

[21] Don Zagier. Elliptic modular forms and their applications. In *The 1-2-3 of modular forms*, Universitext, pages 1–103. Springer, Berlin, 2008.