Readme

weilrep contains some Sage code 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. This note gives a short explanation of what the program *weilrep* calculates and a few examples of how to use it.

This program is built around an algorithm for vector-valued Eisenstein series using Cowan–Katz–White's formula [4] for the Igusa zeta functions attached to quadratic functions (in other words, the local densities). In particular it works best in weights at least 5/2.

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 $-S$.)

1.3 Construction of modular forms

Modular forms for a given Weil representation 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 $E_{k,0}$ for the dual Weil representation associated to the Gram matrix S with Fourier expansion up to precision *prec*. It only accepts weight $k \geq 5/2$. Setting the optional parameter *allow_small_weight* to true allows k to be arbitrary (but the result may not be a modular form)!

Example. 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, allow_small_weight = False)
```

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 $d_b b \in \mathbb{Z}^N$).

Example. 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, allow_small_weight = False)
```

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 [6] 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.

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 turns it into 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 in the appropriate number of variables which is homogeneous and harmonic with respect to the quadratic form. (If the optional parameter test_P is set to false then we do not test these properties of P - use at your own risk.)

1.3.5 Bruinier–Bundschuh isomorphism

Bruinier and Bundschuh [3] 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)]
```

1.3.6 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 [2]). The Fourier coefficients of these series are rational and they can be computed in terms of Eisenstein series (associated to other lattices) [7]. (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 [8] and implemented with

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

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 $O(q^{prec})$.

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 [9] 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 [1] (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.

1.5 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 $N = 0$).

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.

2.1.3 Coefficient vector

```
f.coefficient_vector()
```

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

2.1.4 Components

```
f.components()
```

produces the components of f as a dictionary. The keys are tuples $(g_1, ...g_d)$ 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 $f \in M_k(\rho)$ by a scalar-valued modular form g 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 operations

2.3.1 Bol operator

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

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

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 then *f.conjugate(A)* is a modular form for the Gram matrix $A^T S A$. (Note: A does not need to be invertible.)

2.3.3 Serre derivative

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 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).

3 Jacobi forms

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

3.2.1 Jacobi Eisenstein series

```
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 [5]. 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 [5]). 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 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.

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!)

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 rank(m_1)+ 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 Index-raising Hecke operators

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)
```

The Hecke T-operators T_N are not (yet?) implemented.

3.4.4 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.

3.4.5 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.

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.

References

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

[2] 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.

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

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

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

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

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

[8] Brandon Williams. A construction of antisymmetric modular forms for Weil representations. *Math. Z*, 2019. In press.

[9] 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.