PSS examples

1 Introduction

This note gives a short explanation of what the program PSS calculates as well as a few examples to show how to use it.

To any Gram matrix S (a symmetric integer matrix whose diagonal consists of even integers), we are interested in modular forms for the dual Weil representation ρ^* of the lattice $\Lambda = \mathbb{Z}^n$ with quadratic form $q(v) = \frac{1}{2}v^T S v$. These modular forms are the obstructions to constructing input functions for the Borcherds lift. If (b^+, b^-) is the signature of S, then nonzero modular forms exist only in weights k for which $2k+b^+-b^-$ is an even integer; and for many applications, only those k for which $2k + b^+ - b^- \equiv 0\,(4)$ are interesting. PSS only works when $2k + b^+ - b^- \equiv 0\,(4)$. (For example, classical (scalar-valued) modular forms come from unimodular lattices, which necessarily have $b^+ - b^- \equiv 0\,(4)$, so PSS cannot be used to try to construct classical modular forms of odd weight.)

For large enough weight ($k > 2$), the basic modular forms one can construct are the Poincaré series

$$P_{k,m,\beta} = \frac{1}{2}\sum_{c,d}\left(\mathbf{e}(m\tau)\mathfrak{e}_\beta\right)\Big|_{k,\rho^*}\begin{pmatrix} a & b \\ c & d\end{pmatrix}, \quad \beta \in \Lambda'/\Lambda, \quad m \in -q(\beta)+\mathbb{Z}, \quad m \geq 0,$$

where $\mathbf{e}(m\tau) = e^{2\pi i m\tau}$, and (c,d) runs through all pairs of coprime integers. For $m = 0$ (and $q(\beta) \in \mathbb{Z}$), we get the Eisenstein series

$$E_{k,\beta} = P_{k,0,\beta} = \frac{1}{2}\sum_{c,d}\mathfrak{e}_\beta\Big|_{k,\rho^*}\begin{pmatrix} a & b \\ c & d\end{pmatrix}.$$

These series were studied by Bruinier and Kuss who found a computable formula for their Fourier coefficients, which are always rational.

The coefficients of the Poincaré series $P_{k,m,\beta}$, $m > 0$ are by nature much more complicated than those of Eisenstein series. Instead, PSS calculates the coefficients of the series

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

which turn out to be rational numbers that can be evaluated exactly. The space spanned by $Q_{k,m,\beta}$ as β runs through Λ'/Λ and m runs through $(-q(\beta)+\mathbb{Z}) \cap \mathbb{Q}_{>0}$ always contains all cusp forms.

2 The functions

Fix a Gram matrix S with signature (b^+, b^-) and a weight k with $2k + b^+ - b^- \equiv 0\,(4)$.

The function

```
DimensionFormula(k,S)
```

computes the dimension of $M_k(\rho^*)$ (all modular forms) and $S_k(\rho^*)$ (cusp forms) and outputs these as a list $[\dim M_k(\rho^*), \dim S_k(\rho^*)]$.

The function

```
DiscriminantGroup(S)
```

outputs a list of vectors that form a system of representatives of Λ'/Λ. The function

```
ReducedDiscriminantGroup(S)
```

is similar but only outputs one representative from each pair $\pm\gamma$. ReducedDiscriminantGroup is usually more useful because $Q_{k,m,\beta} = Q_{k,m,-\beta}$ and because the coefficient $c(n,\gamma)$ of $q^n\mathfrak{e}_\gamma$ in any modular form equals the coefficient $c(n,-\gamma)$ of $q^n\mathfrak{e}_{-\gamma}$.

The function

```
EisensteinCoefficient(g,n,k,S)
```

returns the coefficient $c(n,g)$ of $q^n\mathfrak{e}_g$ in the Eisenstein series $E_k = E_{k,0}$. We use the formula of Bruinier and Kuss.

The function

```
PSS(g,b,m,n,k,S)
```

returns the coefficient $c(n,g)$ of $q^n\mathfrak{e}_g$ in the Poincaré square series $Q_{k,m,b}$.

The function

```
CuspSpan(k,S)
```

returns a list of lists $[m,\beta]$ such that the differences $E_k - Q_{k,m,\beta}$ are a basis of $S_k(\rho^*)$. Depending on the lattice, this can take a long time.

If S is a negative definite matrix, then the function

```
BorcherdsPrincipalPart(k,S)
```

returns a list of principal parts of all input functions that produce Borcherds products of weight k for the lattice $\Lambda \oplus II_{1,1} \oplus II_{1,1}$, where Λ is the lattice attached to S. (The restriction to lattices that contain two unimodular hyperbolic planes could probably be removed; if there is interest then I will look into this further.) The principal parts in this list can be entered into the functions

```
WeylVector(inputs,S)
```

and

```
InputFunction(inputs,prec,S)
```

to calculate the Weyl vector and exponents of the Borcherds lift of that input function.

3 Example 1: Unimodular lattices

When S is any unimodular lattice, this algorithm produces classical modular forms of level 1. The algorithm is faster for lattices of small dimension. The fastest input is the empty matrix S.

```
S = matrix([])
g = vector([])
b = vector([])
k = 8
for n in range(10):
    print [n,EisensteinCoefficient(g,n,k,S), PSS(g,b,1,n,k,S)]
```

The output is

```
[0, 1, 1]
[1, 480, 480]
[2, 61920, 61920]
[3, 1050240, 1050240]
[4, 7926240, 7926240]
[5, 37500480, 37500480]
[6, 135480960, 135480960]
[7, 395301120, 395301120]
[8, 1014559200, 1014559200]
[9, 2296875360, 2296875360]
```

Of course, these coefficients must be the same because the space M_8 is one-dimensional. We can double-check that with the command

```
DimensionFormula(8,S)
```

which produces

```
[1,0]
```

The first weight where the dimension is greater than 1 is $k = 12$.

```
S = matrix([])
g = vector([])
b = vector([])
k = 12
for n in range(10):
    print [n,EisensteinCoefficient(g,n,k,S), PSS(g,b,1,n,k,S)]
```

produces the output

```
[0, 1, 1]
[1, 65520/691, 7806960/77683]
[2, 134250480/691, 15082003440/77683]
[3, 11606736960/691, 1304953634880/77683]
[4, 274945048560/691, 30908983376880/77683]
[5, 3199218815520/691, 359661921161760/77683]
[6, 23782204031040/691, 2673619554553920/77683]
[7, 129554448266880/691, 14564649928832640/77683]
[8, 563087459516400/691, 63302965077769200/77683]
[9, 2056098632318640/691, 231148879035948720/77683]
```

So $E_{12} = 1 + \frac{65520}{691}q + \frac{134250480}{691}q^2 + ...$ and

$$Q_{12,1,0} = 1 + \frac{7806960}{77683}q + \frac{15082003440}{77683}q^2 + ... = E_{12} + \frac{304819200}{53678953}\Delta.$$

(The denominators 691 and 77683 come from the numerators of zeta values $\zeta(-11)$ resp. $\zeta(-21)$. Letting m grow will usually lead to larger denominators.)

```
S = matrix([])
k = 12
CuspSpan(k,S)
```

produces the output

```
[[1, ()]]
```

which implies that $E_{12} - Q_{12,1,0}$ spans the cusp space S_{12}.

4 Example 2: The Kohnen plus space

Modular forms of level $4N$ with N squarefree and weight $k = 2n - 1/2$ that satisfy Kohnen's plus space condition are equivalent to vector-valued modular forms of weight k for the Gram matrix $S = (2N)$.

For $k = 7/2$ and $S = (2)$, the input

```
S = matrix([2])
b = vector([0])
k = 7/2
for g in ReducedDiscriminantGroup(S):
    offset = frac(g*S*g/2)
    for n in range(10):
        N = n - offset
        print [N,EisensteinCoefficient(g,N,k,S),PSS(g,b,1,N,k,S)]
```

produces

```
[0, 1, 1]
[1, 126, 126]
[2, 756, 756]
[3, 2072, 2072]
[4, 4158, 4158]
[5, 7560, 7560]
[6, 11592, 11592]
[7, 16704, 16704]
[8, 24948, 24948]
[9, 31878, 31878]
[-1/4, 0, 0]
[3/4, 56, 56]
[7/4, 576, 576]
[11/4, 1512, 1512]
[15/4, 4032, 4032]
[19/4, 5544, 5544]
[23/4, 12096, 12096]
[27/4, 13664, 13664]
[31/4, 24192, 24192]
[35/4, 27216, 27216]
```

so

$$E_{7/2} = Q_{7/2,1,0} = \Big(1 + 126q + 756q^2 + ...\Big)\mathfrak{e}_0 + \Big(56q^{3/4} + 576q^{7/4} + ...\Big)\mathfrak{e}_{1/2}.$$

This corresponds to the Cohen Eisenstein series

$$H(\tau) = 1 + 56q^3 + 126q^4 + 576q^7 + 756q^8 + ...$$

of weight $7/2$ and level 4, which spans the Kohnen plus space. These coefficients also occur in the Jacobi Eisenstein series $E_{4,1}$. For modular forms of weight $2n + 1/2$ you can instead use the Gram matrix $(-2N)$.

5 Example 3: $Q_{k,m,\beta}$ at nonzero β

Let S be the matrix $\begin{pmatrix} -4 & -2 \\ -2 & -4 \end{pmatrix}$. For weight $k = 3$, the dimensions of $M_k(\rho^*)$ and $S_k(\rho^*)$ are 3 and 2, respectively, as we see by computing

```
S = matrix([[-4,-2],[-2,-4]])
k = 3
DimensionFormula(k,S)
```

However, when we calculate

```
S = matrix([[-4,-2],[-2,-4]])
b = vector([0,0])
k = 3
for g in ReducedDiscriminantGroup(S):
    offset = frac(g*S*g/2)
    for n in range(5):
        N = n - offset
        print [g,N,EisensteinCoefficient(g,N,k,S),PSS(g,b,1,N,k,S),PSS(g,b,2,N,k,S)]
```

we do not see any cusp forms: in fact, $Q_{3,m,0} = E_3$ for all m. The reason for this is that the entire cusp space is supported on components other than $\mathfrak{e}_0$. To find a basis of cusp forms, we use

```
CuspSpan(3,S)
```

which outputs

```
[[1/6, (2/3, 1/6)], [1/2, (1/2, 0)]]
```

In other words, $E_3 - Q_{3,1/6,(2/3,1/6)}$ and $E_3 - Q_{3,1/2,(1/2,0)}$ is a basis of $S_3(\rho^*)$. Entering

```
S = matrix([[-4,-2],[-2,-4]])
k = 3
for g in ReducedDiscriminantGroup(S):
    offset = frac(g*S*g/2)
    for n in range(5):
        N = n - offset
        E = EisensteinCoefficient(g,N,k,S)
        print [g,N,E-PSS(g,vector([2/3,1/6]),1/6,N,k,S),E - PSS(g,vector([1/2,0]),1/2,N,k,S)]
```

will list the first few of their Fourier coefficients.

6 Example 4: Small weights

In weight 1 this method does not produce cusp forms; however, the corrected Eisenstein series $E_1^*(\tau, 0)$ always defines a modular form (which does not necessarily have constant term $\mathfrak{e}_0$). EisensteinCoefficient(g,n,1,S) now produces the corrected Eisenstein series. (Previously it did not correct the constant term.) For example, the theta series of $x^2 + y^2$ is half of the output from the following.

```
S = matrix([[-2,0],[0,-2]])
g = vector([0,0])
for n in range(10):
    print [n,EisensteinCoefficient(g,n,1,S)]
```

However the function PSS will raise an error in weight 1.

In weight 3/2 the formula for the Eisenstein series usually does not produce a modular form. However, the function PSS will always produce a modular form (which may be identically 0). The functions $Q_{3/2,m,\beta}$ are sometimes enough to span all modular forms but I do not think they are enough in general. The Gram matrix $S = (74)$ is probably a counterexample.

```
S = matrix([[-2,-1,0],[-1,-4,-1],[0,-1,-2]])
g = vector([0,0,0])
b = vector([5/6,1/3,5/6])
for n in range(10):
    print [n,EisensteinCoefficient(g,n,3/2,S),PSS(g,b,1/6,n,3/2,S)]
```

In weight 2, the Eisenstein series and PSS both produce modular forms whenever the determinant of S is not square. When $\det(S)$ is square, they may be quasimodular forms (but their difference is always a cusp form). For nonsquare discriminants this program uses computations based on Pell-type equations that can be very slow for large $\det(S)$ and large denominators of g and b.

For example, the space $S_2(\rho^*)$ for the lattice with Gram matrix $S = \begin{pmatrix} 4 & 9 \\ 9 & 4 \end{pmatrix}$ is two-dimensional, spanned by $E_2 - Q_{2,1,0}$ and $E_2 - Q_{2,2/65,(61/65,9/65)}$. Computing $Q_{2,2/65,(61/65,9/65)}$ is much slower than computing $Q_{2,1,0}$.

```
S = matrix([[4,9],[9,4]])
for g in ReducedDiscriminantGroup(S):
    offset = frac(g*S*g/2)
    for n in range(10):
        print [g,n-offset,EisensteinCoefficient(g,n-offset,2,S) - PSS(g,vector([0,0]),1,n-offset,2,S),
        EisensteinCoefficient(g,n-offset,2,S) - PSS(g,vector([61/65,9/65]),2/65,n-offset,2,S)]
```

7 Example 5: Borcherds products

In this section S must be a negative-definite matrix. The function BorcherdsPrincipalPart(k,S) finds the principal parts of all input functions that lead to holomorphic Borcherds products of weight k for the lattice given by appending two hyperbolic planes to S. From these principal parts one can read off the divisors of these automorphic products.

Example. Axel Marschner's thesis (`http://publications.rwth-aachen.de/record/59634/files/05_064.pdf`, in particular page 90) works out some Borcherds products of small weights for paramodular forms of level 5 (which are equivalent to certain orthogonal modular forms of type $O(2,3)$). These arise from input functions which are modular forms for the Weil representation attached to the Gram matrix $S = (-10)$. We can verify his computation with the command

```
S = matrix([[-10]])
BorcherdsPrincipalPart(k,S)
```

for various values of k.

This command does not make any use of parallelization and uses a slower algorithm for lattice point enumeration than for example LattE. Its runtime increases quickly with the determinant of S. For large lattices it is better to do things differently.

The Weyl vectors can be calculated with the function WeylVector. For the lattice $(-2t) \oplus II_{1,1} \oplus II_{1,1}$ this produces a vector $(\rho_1, \rho_2, \rho_3) \in \mathbb{Q}^3$. The corresponding Weyl vector in the picture of paramodular forms of level t will be $\lambda = \begin{pmatrix} \rho_1 & t\rho_2 \\ t\rho_2 & t\rho_3 \end{pmatrix}$. For example to compute the Weyl vector for the paramodular Borcherds product of weight 4 and level 5 we use

```
S = matrix([[-10]])
inputs = BorcherdsPrincipalPart(4,S)[0]
WeylVector(inputs,S)
```

to find the result $(1/2, 3/20, 1/2)$; so the Weyl vector for the paramodular product is $\lambda = \begin{pmatrix} 1/2 & 3/4 \\ 3/4 & 5/2 \end{pmatrix}$.

Finally the function InputFunction computes the input function F with given principal part up to specified precision, by identifying $F \cdot \Delta^m$ as a cusp form for large enough $m \in \mathbb{N}$. (This may take a while.) **Example:** BorcherdsPrincipalPart(10,S) finds that there are two holomorphic products of weight 10 for the lattice above. The first one is the square of the weight 5 product. We compute the input function of the second one up to q^5:

```
S = matrix([[-10]])
inputs = BorcherdsPrincipalPart(10,S)[1]
for L in InputFunction(inputs,5,S):
    print L
```

The result is

```
[(0), 0, 20 + 1200*q + 20720*q^2 + 204160*q^3 + 1481200*q^4 + O(q^5)]
[(9/10), -1/20, 1 - 320*q - 5633*q^2 - 56768*q^3 - 416767*q^4 + O(q^5)]
[(4/5), -1/5, -492*q - 10044*q^2 - 107616*q^3 - 821648*q^4 + O(q^5)]
[(7/10), -9/20, 1 + 192*q + 5118*q^2 + 61824*q^3 + 506372*q^4 + O(q^5)]
[(3/5), -4/5, 20*q + 688*q^2 + 10480*q^3 + 95636*q^4 + O(q^5)]
[(1/2), -1/4, -510*q - 10880*q^2 - 119300*q^3 - 922880*q^4 + O(q^5)]
```

The list $[\gamma, \nu, f]$ in the above output implies that the components of $\mathfrak{e}_\gamma$ and $\mathfrak{e}_{-\gamma}$ in F are both $q^\nu f(\tau)$. Compare this output to the computation on page 92 of Marschner's thesis.