SUMMING A POLYNOMIAL FUNCTION OVER INTEGRAL POINTS OF A POLYGON. USER'S GUIDE.

VELLEDA BALDONI, NICOLE BERLINE, AND MICHÈLE VERGNE

Abstract. This document is a companion for the Maple program **Summing a polynomial function over integral points of a polygon**. It contains two parts. First, we see what this programs does. In the second part, we briefly recall the mathematical background.

1. Introduction

The present article is a user's guide for the Maple program **Summing a polynomial function over integral points of a polygon**, available at `http://www.math.polytechnique.fr/~berline/maple.html`. The Maple program contains two types of computation. The first computation does just what the title says. The input consists of a finite set of rational points in $\mathbb{Q}^2$, whose convex hull is a polygon $\mathfrak{p}$, and a polynomial $h(x,y)$ with rational coefficients. The output is the sum

$$\sum_{(x,y)\in\mathfrak{p}\cap\mathbb{Z}^2} h(x,y).$$

The second computation returns the function of $t\in\mathbb{N}$ which arises when the polytope $\mathfrak{p}$ is dilated by t.

$$E(t) := \sum_{(x,y)\in t\mathfrak{p}\cap\mathbb{Z}^2} h(x,y).$$

This function is a *quasi-polynomial*, meaning that is has the form

$$E(t) = \sum_{i=0}^{\deg h+2} E_i(t)\, t^i,$$

where the coefficients depend only on t mod q, where q is the smallest integer such that $q\mathfrak{p}$ has integral vertices. The function $E(t)$ is called the *weighted Ehrhart quasi-polynomial* of $\mathfrak{p}$ with respect to the weight $h(x,y)$.

Date: May 2009.

We apply two methods, the first one for a fixed polygon, the second one for the computation of the weighted Ehrhart quasi-polynomial. The first method is based directly on Brion's formula (2), [3], while the second method is based on the local Euler-Maclaurin formula of [2]. Both methods use Barvinok's decomposition into unimodular cones [1]. Although they are very similar, the first method is faster when we deal with a fixed polygon, while the second is faster when we want the Ehrhart quasi-polynomial.

The software libraries *LattE* [4] (improved version in [5]) and *Barvinok* [6] include the computation of the number of points of a rational polytope in any dimension, together with many other applications. Moreover, the weighted Ehrhart polynomials in any dimension are computed in *Barvinok*. The present program, in dimension two, is based on the same principles: Brion's formula and Barvinok's decomposition of cones. We use however some new ideas on "renormalisation" of Laurent series from [2] to speed up the computation. In the future, we will generalize it to higher dimensions.

2. Main commands

2.1. **Summing a polynomial function over the set of integral points of a polygon.** Let $P \subset \mathbb{Q}^2$ be a finite set of points. Let $\mathfrak{p} \subset \mathbb{R}^2$ be the polygon obtained as the convex hull of the set P. The program computes the sum

$$\sum_{(x,y)\in\mathfrak{p}\cap\mathbb{Z}^2} h(x,y)$$

of the values of a polynomial $h(x,y)$ over the set of integral points contained in $\mathfrak{p}$. In particular, when $h=1$, it computes the number of integral points in $\mathfrak{p}$.

For a single monomial $h(x,y)=x^{m_1}y^{m_2}$, the command is

```
>sum_monomial_polygon(P,m);
```

Here P is a set of pairs of rational numbers, and $m=[m_1,m_2]$ is a pair of non negative integers, the multidegree of the monomial $x^{m_1}y^{m_2}$.

If we want just the number of integral of integral points, we can use the command

```
>number_points_polygon(P);
```

This number can be also obtained by the command

```
>sum_monomial_polygon(polygon,[0,0]);
```

We compute the sum of a polynomial $h(x,y)$ by the command

```
>sum_polynomial_polygon(P,h);
```

Here P is a set of pairs of rational numbers, and $h = \sum_m h_m x^{m_1} y^{m_2}$ is entered as an expression in x, y.

Example 1. P is the square $\{[0,0],[1,0],[1,1],[0,1]\}$.

```
>square:= {[0,0],[1,0],[1,1],[0,1]};

>number_points_polygon(square);
                              4
```

The sum of values x^5y^5 over the 4 integral points in the square is

```
>sum_monomial_polygon(square,[5,5 ]);
                            1
```

Example 2. Here P is a randomly chosen set of 15 points.

```
> P := {[77/8,97/59], [93/44,70/29], [0,25/12], [25/32,29/48],
[92/41,57/91], [9/4,1/7], [64/43,31/75], [91/17,33/86], [12/37,77/8],
[8/5,41/27], [80/67,11/9], [16/73,11/89], [41/20,43/88],
[32/49,59/23], [77/94,65/46]}
```

The number of integral points in the convex hull is 45.

```
>number_points_polygon(P);
                            45
```

The vertices of the convex hull $\mathfrak{p}$ of P (listed in counter-clockwise order) are obtained with the command:

```
>vertices_in_counter_clock_order:=proc(polygon)

>vertices_in_counter_clock_order(P);
 [[0,25/12], [16/73,11/89], [9/4,1/7], [91/17,33/86], [77/8,97/59],
[12/37,77/8]]
```

We compute the sum of $x^{32}y^{32}$ over the set of integral points (x, y) of the convex hull $\mathfrak{p}$ of P.

```
>sum_monomial_polygon(P,[32,32]);
          987532646688766560932727042325214847653263886
```

We compute the sum of $x^{32}y^{32} + 7$ over all the integral points (x, y) of the polygon $\mathfrak{p}$. (the preceding number $+7$ times 45)

```
>h:= x^{32}y^{32}+7;
>sum_polynomial_polygon(P,h);
         987532646688766560932727042325214847653264201
```

2.2. **Weighted Ehrhart polynomial of a polygon.** Our program computes also the weighted Ehrhart quasi-polynomial of a polygon. For brevity, we treat only the case where the weight is a monomial $h(x, y) = x^{m_1}y^{m_2}$. When the polygon is dilated by a non negative integer t, and if q is a positive integer such that $q\mathfrak{p}$ has integral vertices, the function of t given by

$$t \mapsto \sum_{(x,y)\in t\mathfrak{p}\cap\mathbb{Z}^2} x^{m_1}y^{m_2}$$

is a quasi-polynomial $S(t) = \sum_{i=0}^{m_1+m_2+2} E_i(t)t^i$ of degree $m_1 + m_2 + 2$. The coefficients $E_i(t)$ are functions of t modulo q. This program computes these coefficients $E_i(t)$ in terms of the symbolic function $fmod(p * t, q)$ which stands for ($t \mapsto pt$ mod q). We can either obtain each individual coefficient $E_i(t)$ or the full weighted Ehrhart polynomial $S(t)$.

Here are the commands:

```
> coeff_t_Ehrhart_polygon(i,t,P,m);
```

The input consists of i an integer, t a letter , P a set of points and $m = [m_1, m_2]$ a pair of integers which represents the weight; the output is the coefficient $E_i(t)$.

```
>Ehrhart_polynomial_polygon(t,P,m);
```

Input is as in the previous command, except i is not needed. The output is the full Ehrhart polynomial $S(t)$.

Examples

```
>transsquare:={[-1/2,-1/2]{[1/2,-1/2],[1/2,1/2],[-1/2,1/2]};
> coeff_t_Ehrhart_polygon(0,t,square,[0,0]);
          1
> coeff_t_Ehrhart_polygon(0,t,transsquare,[0,0]);
-2*fmod(t, 2)+3/2+1/2*fmod(t, 2)^2
> Ehrhart_polynomial_polygon(t,square,[0,0]);
                          1 + 2 t + t^2
> Ehrhart_polynomial_polygon(t,transsquare,[0,0]);
       (fmod(t, 2)-1)^2+(-2*fmod(t, 2)+2)*t+t^2
```

2.3. **Experiments.** The following experiments were done with a laptop, processor 1,86 GHz, RAM 782 MHz,0,99 Go.

```
>A:={[(-567337)/102495,-1414975/95662],[1/3,1/5],[-88141/20499,12732/47831]};

>largeA:={[1000*(-567337)/102495,1000*(-1414975/95662)],
[1000*1/3,1000*1/5],[-1000*88141/20499,1000*12732/47831]};

 > number_points_polygon(A);
```

```
> number_points_polygon(largeA);
                 34922612
```

In the next experiments, we indicate the time of computation T in seconds. The number of integral points in the rational triangle with vertices A is 36. If we dilate A by the factor 1000, we obtain the triangle $largeA$ where the number of points (34922612) is approximatively 10^6 times larger. Observe that we compute in 14 seconds the sum of the large degree monomial $h(x, y) = x^{64}y^{64}$ over the set of integral points of A, and that we compute in 16 seconds the sum of the same monomial over the integral points of $largeA$. The computation time is almost the same, although any computation by enumeration would be 10^6 times longer.

```
> T:=time(): sum_monomial_polygon(A,[32,32]);Time:=time()-T;
  1115669371408012143680968371636968254681278749400139813965 7
                              Time := 1.766

>  T:=time():sum_monomial_polygon(A,[64,64]);Time:=time()-T;
1069166274697538317169068795296300521972363937518981 4
21775607019156653055887 9\
  383651355584733489625371887946297859021 7
                             Time := 13.640

> T:=time(): sum_monomial_polygon(largeA,[64,64]): Time:=time()-T;
17831035913722066043589677840496661987989193563450057671832979767102708226068\
  90519522342895765988221612337480372436229072894493363579270305297678267123 8\
  40160119137597718403779959778986161713238013119891186401529359213636522185 2\
  95244921491613319792892241946298996049559369929767070065285383458443917290 1\
  85791611969462010599632957347801451344938373887397255088905193762020134177 1\
  82911075684183735887058845417207962477000059284528111310251783657942905087 0\
  200997036215789313590638254401223831203513017660101185561 83
                                  Time:= 15.516
```

Finally, we computed the weighted Ehrhart polynomial with weight $x^{32}y^{32}$ over the triangle with vertices[[-567337/102495, -1414975/95662], [88141, 292844676/6833], [-88141/20499, 12732/47831]]. We compute the coefficient of t^2 for example. The time of computation is 268 seconds. The result is too big to be printed here, as it involves many functions $fmod(c*t, D)$ where D runs though the denominators of the coordinates of the vertices of A (large numbers).

```
> T:=time():coeff_t_Ehrhart_polygon(2,t,[[-567337/102495,
```

```
-1414975/95662], [88141, 292844676/6833], [-88141/20499,
12732/47831]],[32,32]): Time:=time()-T;
                          Time :=  268.266
```

3. Mathematical background

The first method is for a fixed polygon, the second one for the computation of the weighted Ehrhart quasi-polynomial.

3.1. **First method: Brion's formula, Barvinok's decomposition into unimodular cones and iterated Laurent series.** Let $\mathfrak{p}$ be a convex polygon in $\mathbb{R}^2$ with rational vertices $s_i, 1 \leq i \leq n+1$. We want to compute the sum

$$\sum_{x \in \mathfrak{p} \cap \mathbb{Z}^2} x^{m_1} y^{m_2}. \tag{1}$$

We start by observing that (1) is equal to the coefficient of $\frac{\xi_1^{m_1} \xi_2^{m_2}}{m_1! m_2!}$ in

$$\sum_{x \in \mathfrak{p} \cap \mathbb{Z}^2} e^{\langle \xi, x \rangle}.$$

Our method is based on Brion's formula (2). Brion's formula is the generalization of the following formula for the sum of geometric progressions over the interval $[A, B]$ (with $A \leq B$ integers):

$$\sum_A^B e^{n\xi} = \frac{e^{A\xi}}{1 - e^{\xi}} + \frac{e^{B\xi}}{1 - e^{-\xi}}.$$

For any rational polygon $\mathfrak{q} \subset \mathbb{R}^2$ define

$$S(\mathfrak{q})(\xi) = \sum_{x \in \mathfrak{q} \cap \mathbb{Z}^2} e^{\langle \xi, x \rangle}.$$

This a meromorphic function near $\xi = 0$. Moreover the map $\mathfrak{q} \mapsto S(\mathfrak{q})(\xi)$ is a valuation on the set of rational polyhedra, and $S(\mathfrak{q}) = 0$ if $\mathfrak{q}$ contains a line. Brion's formula is the following. Let $\mathfrak{c}_i$ be the cone at vertex S_i of the polygon.

$$S(\mathfrak{p}) = \sum_{i=1}^{n+1} S(\mathfrak{c}_i). \tag{2}$$

Each term $S(\mathfrak{c}_i)(\xi) = \sum_{x \in \mathfrak{c}_i \cap \mathbb{Z}^2} e^{\langle \xi, x \rangle}$ in (2) is a meromorphic function near $\xi = 0$. The poles cancel and the sum is a holomorphic function of ξ. Thus we compute (1) as the coefficient of $\frac{\xi_1^{m_1} \xi_2^{m_2}}{m_1! m_2!}$ in the right-hand-side of (2). We actually compute the individual contribution of each cone $\mathfrak{c}_i$ (associated to the vertex s_i) to the sum. The coefficient of

$\frac{\xi_1^{m_1}\xi_2^{m_2}}{m_1!m_2!}$ in the meromorphic function $S(\mathfrak{c}_i)$ of two variables ξ_1, ξ_2 has no intrinsic meaning. Our method consists in applying **iterated Laurent series** expansions to $S(\mathfrak{c}_i)(\xi)$ with respect to the variables ξ_1 then ξ_2. We obtain a Laurent series $L(\mathfrak{c}_i)$ in the ring $\mathbb{Q}[\xi_1, \xi_1^{-1}, \xi_2, \xi_2^{-1}]$ and we compute the coefficient $\frac{\xi_1^{m_1}\xi_2^{m_2}}{m_1!m_2!}$ in $L(\mathfrak{c}_i)$.

Thus, in order to compute the contribution of a vertex s to the sum (2), we need to compute $S(\mathfrak{c})(\xi)$ for the supporting cone $\mathfrak{c}$. The crucial tool here is Barvinok's decomposition into unimodular cones. Actually, we use the following variant of Barvinok's decomposition, (procedure `signed_decomp`).

Let $\mathfrak{c}$ be a simplicial cone in $\mathbb{R}^d$. Let V_i, for $i = 1, \dots, d$, be the generators of $\mathfrak{c}$. Let V be a vector in $\mathbb{R}^d$. We write $V = \sum_i u_i V_i$. We split $[V_1, \dots, V_d]$ into three parts, as follows.

$$L_+ := [X_1, \dots, X_k]$$

formed by the V_i such that $u_i > 0$,

$$L_- := [Y_1, \dots, Y_m]$$

formed by the V_i such that $u_i < 0$,

$$L_0 := \{Z_1, \dots, Z_b\}$$

formed by the V_i such that $u_i = 0$.

Then we have the equality of characteristic functions modulo characteristic functions of cones containing lines.

$$(-1)^{(k+1)}[\mathfrak{c}] = \sum_{i=1}^{k} (-1)^{i+1}[\mathfrak{c}(X_1, \dots, X_{i-1}, -X_{i+1}, \dots, -X_k, V, L_-, L_0)] +$$
$$\sum_{j=1}^{m} (-1)^{j+k}[\mathfrak{c}(L_+, -V, -Y_1, \dots, -Y_{j-1}, Y_{j+1}, \dots, Y_m, L_0)].$$

Remark. This decomposition is not the stellar decomposition. It involves only cones of maximal dimension d. It avoids the dualizing trick of Brion.

Example. $\mathfrak{c} = \mathbb{R}^+ e_1 \oplus \mathbb{R}^+ e_2$, $V = e_1 + e_2$, so that L_- and L_0 are empty and $k = 2$. Then

$$-[\mathfrak{c}] = \mathfrak{c}(V, -e_2) - \mathfrak{c}(e_1, V) - \mathfrak{c}[e_2, -e_2, e_1].$$

Indeed $[\mathfrak{c}(e_2, -e_2, e_1)] - [\mathfrak{c}]$ is equal to the characteristic function of the quadrant $(e_1, -e_2)$ minus that of the half-line $\mathbb{R}^+ e_1$. This is also the case for $[\mathfrak{c}(V, -e_2)] - [\mathfrak{c}(e_1, V)]$.

If we use a lattice vector V with sufficiently small coordinates in the basis (V_i), the cones appearing in this decomposition have indices smaller than $\mathfrak{c}$. One obtains such a *short* vector V by the Lenstra-Lenstra-Lovasz algorithm. By a repeated application of this decomposition, one obtains a decomposition of $\mathfrak{c}$ in a signed sum of unimodular cones $\mathfrak{c}_z$ (modulo cones containing lines). As $S(\mathfrak{a}) = 0$ for a cone $\mathfrak{a}$ which contains a line, we can use this decomposition to compute $S(\mathfrak{c})$.

For a unimodular cone $\mathfrak{c}$, the sum $S(\mathfrak{c})$ has a simple closed expression. Let (V_1, V_2) be primitive generators of the edges of $\mathfrak{c}$ and let s be its vertex. Let $\tilde{s}$ be the unique integral point contained in the *semi-closed box*

$$\{s + t_1V_1 + t_2V_2, 0 \leq t_i < 1\}$$

If $s = s_1V_1 + s_2V_2$, then $\tilde{s} = \tilde{s}_1V_1 + \tilde{s}_2V_2$ with $\tilde{s}_i = ceil(s_i)$. Then

$$S(\mathfrak{c})(\xi) = \frac{e^{\langle \xi, \tilde{s} \rangle}}{(1 - e^{\langle \xi, V_1 \rangle})(1 - e^{\langle \xi, V_2 \rangle})}. \tag{3}$$

In order to simplify the computation of iterated Laurent series, we introduce the analytic function

$$B(X, u) = \frac{e^{uX}}{1 - e^X} + \frac{1}{X} = -\sum_{n=0}^{\infty} \frac{b(n+1, u)}{(n+1)!} X^n$$

where $b(n, u)$ are the Bernoulli polynomials. Writing

$$\frac{e^{uX}}{1 - e^X} = B(X, u) - \frac{1}{X}, \tag{4}$$

we obtain

$$S(\mathfrak{c})(\xi) = A + G + R,$$

where

$$A = B(\langle \xi, V_1 \rangle, \tilde{s}_1) B(\langle \xi, V_2 \rangle, \tilde{s}_2)$$

is an analytic function of ξ,

$$G := -\frac{1}{\langle \xi, V_1 \rangle} B(\langle \xi, V_2 \rangle, \tilde{s}_2) - \frac{1}{\langle \xi, V_2 \rangle} B(\langle \xi, V_1 \rangle, \tilde{s}_1),$$

$$R := \frac{1}{\langle \xi, V_1 \rangle \langle \xi, V_2 \rangle}.$$

We replace $\frac{1}{\langle \xi, V_1 \rangle}$ and $\frac{1}{\langle \xi, V_2 \rangle}$ by their iterated Laurent series expansion in the ring $R[\xi_1, \xi_1^{-1}, \xi_2, \xi_2^{-1}]$. For example, if $V_1 = [2, 1]$, we write

$$\frac{1}{2\xi_1 + \xi_2} = \frac{1}{\xi_2} \frac{1}{(1 + 2\xi_1/\xi_2)} = \frac{1}{\xi_2} \sum_{k=0}^{\infty} (-1)^k 2^k (\xi_1/\xi_2)^k.$$

We then replace $S(\mathfrak{c})$ by the corresponding element in $\mathbb{Q}[[\xi_1, \xi_2, \xi^{-1}, \xi^{-2}]]$ and we take the coefficient of $\xi_1^{m_1}\xi_2^{m_2}$.

Remark The weighted Ehrhart polynomial can also be computed by this method. We did not write the corresponding algorithm in the Maple file, because we observed that a faster algorithm is given by the second method which we describe in the next section. However, let us explain what one should do. When the polytope $\mathfrak{p}$ is dilated in $t\mathfrak{p}$, its vertices are dilated by t, while the edges of the cones at vertices do not change. Thus we have to compute

$$S(\mathfrak{c}_t)(\xi) = \frac{e^{\langle \xi, \tilde{s_t} \rangle}}{(1 - e^{\langle \xi, V_1 \rangle})(1 - e^{\langle \xi, V_2 \rangle})} \tag{5}$$

where now s_t is the unique point with integral coordinates in the box

$$\{ts + u_1V_1 + u_2V_2, 0 \leq u_i < 1\}.$$

If $s = s_1V_1 + s_2V_2$, with $s_i = p_i/q_i$, we see that

$$s_t = [ts_1 + mod(-tp_1, q_1)/q_1, ts_2 - mod(-tp_2, q_2)/q_2].$$

The iterated Laurent series in ξ_1, ξ_2 has coefficients which are polynomials in t and the periodic functions $mod(tp_i, q_i)$. We extract the coefficient of $t^j\xi_1^{m_1}\xi_2^{m_2}$.

3.2. **Second method. Weighted Ehrhart quasi-polynomial using local Euler-Maclaurin formula.** We now recall the results of [2] and explain how they can be applied to the computation of the weighted Ehrhart quasi-polynomials. Let $\mathfrak{p}$ be a convex polytope in $\mathbb{R}^d$, with rational vertices. Let $h(x)$ be a polynomial function of degree r on $\mathbb{R}^d$. We want to compute the sum $\sum_{x\in\mathfrak{p}\cap\Lambda} h(x)$ of values $h(x)$ over the set of integral points of the polytope $\mathfrak{p}$.

The local Euler-Maclaurin formula has the following form.

$$\sum_{x\in\mathfrak{p}\cap\Lambda} h(x) = \sum_{\mathfrak{f}\in\mathcal{F}(\mathfrak{p})} \int_{\mathfrak{f}} D(\mathfrak{p}, \mathfrak{f}) \cdot h \tag{6}$$

where $\mathcal{F}(\mathfrak{p})$ is the set of all faces of $\mathfrak{p}$. For each face $\mathfrak{f}$, $D(\mathfrak{p}, \mathfrak{f})$ is a differential operator of infinite degree with constant coefficients associated to $\mathfrak{f}$. The operator $D(\mathfrak{p}, \mathfrak{f})$ is *local*, in the sense that it depends only on the intersection of $\mathfrak{p}$ with a neighborhood of any generic point of $\mathfrak{f}$. The integral on the face $\mathfrak{f}$ is taken with respect to the Lebesgue measure on $< \mathfrak{f} >$ defined by the lattice $\mathbb{Z}^d \cap \mathrm{lin}(\mathfrak{f})$. Here $< \mathfrak{f} >$ is the affine span of the face $\mathfrak{f}$ and $\mathrm{lin}(\mathfrak{f})$ is the linear subspace parallel to $< \mathfrak{f} >$.

Let us recall the construction of the operators $D(\mathfrak{p}, \mathfrak{f})$. We denote by $\mathfrak{t}(\mathfrak{p}, \mathfrak{f})$ the transverse cone to $\mathfrak{p}$ along $\mathfrak{f}$. Using the standard scalar product, $\mathfrak{t}(\mathfrak{p},\mathfrak{f})$ is described as the following affine cone in $\mathbb{R}^d$. Let $\text{lin}(\mathfrak{f})^\perp$ be the vector subspace orthogonal to $\text{lin}(\mathfrak{f})$. Then $\mathfrak{t}(\mathfrak{p},\mathfrak{f})$ is the orthogonal projection on $\text{lin}(\mathfrak{f})^\perp$ of the supporting cone of $\mathfrak{p}$ along $\mathfrak{f}$. The operator $D(\mathfrak{p},\mathfrak{f})$ is defined in terms of the transverse cone $\mathfrak{t}(\mathfrak{p},\mathfrak{f})$, as follows.

For every rational affine cone $\mathfrak{a} \subset V$, we construct in [2] an analytic function $\xi \mapsto \mu(\mathfrak{a})(\xi)$ on $\mathbb{R}^d$. This construction depends on the choice of a scalar product. Here we use the standard scalar product. These functions $\mu(\mathfrak{a})$ have nice properties which play a crucial role in our method. First, the assigment $\mathfrak{a} \mapsto \mu(\mathfrak{a})$ is a *valuation* on the set of affine cones with a given vertex. Second, it is *invariant under lattice translations.* Furthermore, $\mu(\mathfrak{a}) = 0$ *if* $\mathfrak{a}$ *contains a line.*

We define

$$D(\mathfrak{p},\mathfrak{f}) = D(\mu(\mathfrak{t}(\mathfrak{p},\mathfrak{f})))$$

as the differential operator of infinite degree with constant coefficients, with symbol $\mu(\mathfrak{t}(\mathfrak{p},\mathfrak{f}))(\xi)$. In other words, if $\xi = (\xi_1, \ldots, \xi_d)$, we obtain $D(\mathfrak{p},\mathfrak{f})$ by replacing ξ_i by $\frac{\partial}{\partial x_i}$ in the Taylor series of $\mu(\mathfrak{t}(\mathfrak{p},\mathfrak{f}))(\xi)$.

For any positive integer t, we consider the dilated polytope $t\mathfrak{p}$ and the corresponding sum

$$S(t\mathfrak{p},h) = \sum_{x \in t\mathfrak{p}\cap\Lambda} h(x).$$

From (6), it follows easily that the function $t \mapsto S(t\mathfrak{p},h)$ is given by a quasi-polynomial: there exist periodic functions $t \mapsto E_i(\mathfrak{p},h,t)$ on $\mathbb{N}$ such that

$$S(t\mathfrak{p},h) = \sum_{i=0}^{d+r} E_i(\mathfrak{p},h,t)t^i \tag{7}$$

whenever t is a positive integer. Moreover the coefficients $E_i(\mathfrak{p},h,t)$ are computed using the functions $\mu(\mathfrak{t}(t\mathfrak{p},t\mathfrak{f}))$. Indeed, let s be the vertex of $\mathfrak{t}(\mathfrak{p},\mathfrak{f})$ so that $\mathfrak{t}(\mathfrak{p},\mathfrak{f}) = s + \mathfrak{t}_0$. Then the dilated transverse cone is $\mathfrak{t}(t\mathfrak{p},t\mathfrak{f}) = ts + \mathfrak{t}_0$. As $\mathfrak{a} \mapsto \mu(\mathfrak{a})$ is invariant under lattice translations, we have

$$\mu(\mathfrak{t}((t+q)\mathfrak{p},t\mathfrak{f})) = \mu(\mathfrak{t}(t\mathfrak{p},t\mathfrak{f})),$$

if q is an integer such that qs is a lattice point for the projected lattice, or equivalently, such that $q < \mathfrak{f} >$ contains a lattice point. Thus, the coefficients $E_i(\mathfrak{p},h,t)$ depend only on $t \mod q$, where q is the smallest integer such that $q\mathfrak{p}$ has integral vertices.

When $\mathfrak{a}$ is a *unimodular* affine cone of dimension 1 or 2, the functions $\mu(\mathfrak{a})$ have an explicit form, in terms of the functions $B(X,u)$ introduced in (4).

Let $\mathfrak{d}$ be a one dimensional affine cone of the form $(s+\mathbb{R}_+)V$ where V is a primitive vector and $s \in \mathbb{Q}$. We have

$$\mu(\mathfrak{d})(\xi) = B(\langle \xi, V\rangle, ceil(s) - s). \tag{8}$$

Let $\mathfrak{a}$ be a two dimensional *unimodular* affine cone. Let V_1, V_2 be primitive generators of its edges, such that $\det(V_1, V_2) = 1$. For $\xi = (\xi_1, \xi_2) \in \mathbb{R}^2$, let $y_i = \langle \xi, V_i \rangle$, for $i = 1, 2$, be the coordinates of ξ relative to the dual basis (V_1^*, V_2^*). We write the vertex of $\mathfrak{a}$ as $s_1V_1 + s_2V_2$ with $s_i \in \mathbb{Q}$. Let $\epsilon_i = ceil(s_i) - s_i$, and let $C_i = \frac{\langle V_1, V_2\rangle}{\langle V_i, V_i\rangle}$, for $i = 1, 2$. With these notations, we have

$$\mu(\mathfrak{a})(\xi) = \frac{e^{\epsilon_1 y_1 + \epsilon_2 y_2}}{(1-e^{y_1})(1-e^{y_2})} + \frac{1}{y_1}B(y_2 - C_1 y_1, \epsilon_2) + \frac{1}{y_2}B(y_1 - C_2 y_2, \epsilon_1) - \frac{1}{y_1 y_2}. \tag{9}$$

The function $\mu(\mathfrak{a})(\xi)$ is actually analytic, although this is not obvious on (9). In order to compute the contribution of a vertex s of $\mathfrak{p}$ to the sum (6), we need to compute $\mu(\mathfrak{c})(\xi)$ when $\mathfrak{c}$ is the two-dimensional supporting cone at s. The crucial tool here is Barvinok's decomposition into unimodular cones. The valuation property of $\mathfrak{a} \mapsto \mu(\mathfrak{a})$ makes it possible to reduce the computation to the unimodular case, and use (9). Notice that (9) returns a function of the relative coordinates (y_1, y_2), which we must convert back to a function of the standard coordinates (ξ_1, ξ_2), in order to add the contributions of the various unimodular cones in Barvinok's decomposition. Actually, since $\mu(\mathfrak{a}) = 0$ if the cone $\mathfrak{a}$ contains a line, we use the variant of Barvinok's decomposition described in the first method.

References

[1] **Barvinok A. I.**, *A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Math. Oper. Res. **19** (1994), 769-779.

[2] **Berline N. and Vergne M.**. *Local Euler-Maclaurin formula for polytopes*, Moscow Math. Journal, **7** (2007), 355–386. arXiv:math.CO/0507256.

[3] **Brion M.**, *Points entiers dans les polyèdres convexes*, Ann. Sci. Ecole Norm. Sup. **21** (1988), 653-663.

[4] **De Loera J.A., Haws D., Hemmecke R., Huggins H., Tauzer J. and Yoshida R.**, *A Users Guide for LattE v1.1, 2003*, software package LattE, available at http://www.math.ucdavis.edu∼latte.

[5] **Koeppe M.**, *A primal Barvinok algorithm based on irrational decompositions*, SIAM Journal on Discrete Mathematics, **21** (2007), pp. 220-236. Software *LattE macchiato* available at http://www.math.ucdavis.edu∼mkoeppe/latte/.

[6] **Verdoolaege S. and Bruynooghe M.**, *Algorithms for weighted counting over parametric polytopes: A survey and a practical comparison*, Eighth ACES Symposium, Edegem, Belgium, September 2008. https://lirias.kuleuven.be/handle/123456789/197757. Software available at http://freshmeat.net/projects/barvinok/.

Universita di Roma Tor Vergata, Dipartimento di Matematica, via della Ricerca Scientifica, 00133 Roma, Italy
E-mail address: `baldoni@mat.uniroma2.it`

Ecole Polytechnique, Centre de mathématiques Laurent Schwartz, 91128, Palaiseau, France
E-mail address: `berline@math.polytechnique.fr`

Institut de Mathématiques de Jussieu, Théorie des Groupes, Case 7012, 2 Place Jussieu, 75251 Paris Cedex 05, France

Ecole Polytechnique, Centre de mathématiques Laurent Schwartz, 91128, Palaiseau, France
E-mail address: `vergne@math.polytechnique.fr`