# Parameterized Polyhedra and their Vertices

Vincent Loechner †, Doran K. Wilde ‡

†ICPS, Université Louis Pasteur de Strasbourg,

Pôle API, Boulevard Sébastien Brant, 67400 Illkirch, France

e-mail: `loechner@icps.u-strasbg.fr`

‡Brigham Young University

Department of Electrical and Computer Engineering

459 Clyde Building, Provo, Utah 84602

e-mail: `wilde@ee.byu.edu`

**Abstract**

Algorithms specified for parametrically sized problems are more general purpose and more reusable than algorithms for fixed sized problems. For this reason, there is a need for representing and symbolically analyzing linearly parameterized algorithms.

An important class of parallel algorithms can be described as systems of parameterized affine recurrence equations (PARE). In this representation, linearly parameterized polyhedra are used to describe the domains of variables. This paper describes an algorithm which computes the set of parameterized vertices of a polyhedron, given its representation as a system of parameterized inequalities. This provides an important tool for the symbolic analysis of the parameterized domains used to define variables and computation domains in PARE's.

A library of operations on parameterized polyhedra based on the Polyhedral Library has been written in C and is freely distributed.

## 1 Introduction

In order to improve the performance of scientific programs wherein most of the CPU time is spent executing DO-Loops, important analysis techniques for the transformation and parallelization of loop nest programs have been developed. In one such technique, the program is first transformed into a single assignment form called a *system of affine recurrence equations* (SARE). Many algorithms in the areas of linear algebra, graphics, string processing, and digital signal processing, as well as other fields, can be written as SARE's. In this form, a program can be abstracted, analyzed and transformed in order to exploit the parallelism in the program.

Most research in the analysis of SARE's has only considered unparameterized systems [11, 14]. However, in the description of an algorithm, the problem size is often specified in terms of parameters. For example, in specifying a matrix operation, you would most likely describe the algorithm in terms of an $N \times M$ matrix, where $N$ and $M$ are size parameters. Parameters need only to be instantiated when the algorithm is evaluated. Parameterizing an algorithm by the size of its inputs generalizes the algorithm and improves its re-usability. Quinton and Van Dongen [13] introduced *parameterized SARE's*, which they called PARE's, and briefly discussed how to derive schedules for them. However, they did not go into any detail about how to do symbolic analysis of parameterized problems.

The objective of this paper is to present a constructive method for finding the vertices of a parameterized polyhedron. This allows us to statically and symbolically analyze PARE's. We are particularly interested in symbolic dependence analysis, i.e., the computation of the dependence vectors, as function of the parameters. Many other important characteristics of a parallel program can also be determined symbolically as a function of the parameters, such as the latency, the processor count, the processor efficiency, and the number and nature of communications [2, 9].

The remainder of this paper is organized as follows: Section 2 presents some basic definitions and introduces our representation of polyhedra. Section 3 introduces the notion of parameterized polyhedron and shows how it can be represented by an extended polyhedron $\mathcal{D}'$ in the combined data and parameter space. This extended polyhedron is easily constructed from the original parameterized polyhedron. We prove in this section that the vertices of a polyhedron depending on $m$ independent parameters correspond to the $m$-faces (faces of dimension $m$) of its extended polyhedron $\mathcal{D}'$. In section 4, we describe the homogeneous representation of polyhedra and give an algorithm to find all of the $m$-faces of a polyhedron. Section 5 describes the algorithm that computes the vertices of parameterized polyhedra. An example and some applications to PARE parallelization are given in section 6. And finally, in section 7, we give our conclusions.

# 2 Representation of polyhedra

We briefly review some of the basic definitions used in this paper. For a complete treatment of polytope theory refer to [17], and to [15] for a more general theory of polyhedral convex sets.

## 2.1 Dual definitions of polyhedra

**Definition 1.** *The* implicit representation *of a polyhedron is defined as the intersection of a finite set of closed linear half-spaces. This representation is specified by a system of inequalities and equalities:*

$$\mathcal{D} : \{x \in \mathbb{Q}^n \mid Ax = b, \quad Cx \geq d\} \tag{1}$$

*where $A$ is a $j \times n$ matrix, $b$ is a $j$-vector, $C$ is a $k \times n$ matrix, and $d$ is a $k$-vector, and where $n$ is the dimension of the space containing the polyhedron, $k$ the number of inequalities, and $j$ the number of equalities.* The dimension of a polyhedron is defined to be the dimension of the smallest affine subspace which spans the polyhedron. A polyhedron of dimension $d$ is called a $d$-polyhedron.

**Definition 2.** *Any polyhedron given by Eq. (1) has an equivalent dual* Minkowski representation. *This representation is specified as a combination of lines, rays, and vertices:*

$$\mathcal{D} : \{x \in \mathbb{Q}^n \mid x = L\lambda + R\mu + V\nu, \quad \mu, \nu \geq 0, \quad \sum_i \nu_i = 1\} \tag{2}$$

*where $\lambda$, $\mu$ and $\nu$ are free-valued column vectors. The notation $\mu \geq 0$ means that all elements of vector $\mu$ are non-negative. The columns of $L$ are the vectors generating the lineality space $lin(\mathcal{D})$ (the* *largest linear subspace entirely contained in the polyhedron). The columns of $R$ are the extremal rays of $\mathcal{D}$ and the columns of $V$ are the vertices of $\mathcal{D}$. $\mathcal{D}$ is then defined as a linear combination of lines, a positive linear combination of unidirectional rays and a convex combination of vertices.*

The problem of computing the vertices and the rays used in Eq. (2) given the constraints from Eq. (1) has a solution of complexity $\mathcal{O}(k^{\lfloor n/2 \rfloor})$, where $k$ is the number of inequality constraints, $n$ the dimension of the space, and the input length is $\Omega(kn)$ [1]. This algorithm was implemented in the Polyhedral Library by Wilde [16]. It is based on the Motzkin double description method [12], and builds on the work done by Fernández and Quinton [5] and Le Verge [8].

## 2.2 Faces of a polyhedron

**Definition 3.** *A* supporting hyperplane *of an $n$-polyhedron $\mathcal{D}$ is a plane of dimension $n-1$ which intersects the hull of $\mathcal{D}$ but not its relative interior.*

**Definition 4.** *A* face *of a polyhedron $\mathcal{D}$ is the intersection of $\mathcal{D}$ and a supporting hyperplane of $\mathcal{D}$.*

Every face of $\mathcal{D}$ is also a polyhedron and is called a *$k$-face* if it is a $k$-polyhedron. We will represent the $i^{\text{th}}$ $k$-face of $\mathcal{D}$ as $F_i^k(\mathcal{D})$.

These definitions are generally given in terms of polytopes, but can be extended to work for polyhedra. The $(n-1)$-faces of an $n$-polyhedron are called the *facets.* The 0-faces of a polytope are its vertices and extremal rays. An extremal ray is a vertex at infinity in a certain direction.

Since the proper faces of a polyhedron are themselves polyhedra of smaller dimension, this representation is hierarchical. The relation "is a face of" is transitive and anti-symmetric and hence can be used to define a partial order among the faces of a polyhedron. This relation and the partially ordered set of all of the faces of a polyhedron along with the empty set form a lattice called the *face lattice* with the $n$-dimensional polyhedron at the top and the empty set (called the *(-1)-face*) at the bottom.

# 3 Parameterized polyhedra

We are interested in describing a family of polyhedra $\mathcal{D}(p)$ as a linear function of $p$, an $m$-vector of parameters, as follows:

$$\mathcal{D}(p) = \{x \in \mathbb{Q}^n \mid Ax = Bp + b \ , \ Cx \geq Dp + d\} \, , \quad p \in \mathbb{Q}^m \tag{3}$$

where $A$, $B$, $C$ and $D$ are constant matrices and $b$ and $d$ are constant vectors.

The parameterized polyhedron $\mathcal{D}(p)$ given in Eq. (3) can be rewritten as a non-parameterized polyhedron in the combined data/parameter space as follows:

$$\mathcal{D}' = \left\{ \begin{pmatrix} x \\ p \end{pmatrix} \in \mathbb{Q}^{n+m} \mid A' \begin{pmatrix} x \\ p \end{pmatrix} = b \ , \ C' \begin{pmatrix} x \\ p \end{pmatrix} \geq d \right\} \tag{4}$$

where $A' = [A \mid -B]$ and $C' = [C \mid -D]$. Using Eq. 4, $\mathcal{D}'$ can always be constructed from $\mathcal{D}(p)$.

We define two projection functions, $\text{Proj}_d$ and $\text{Proj}_p$, which are used in the algorithm to construct parameterized vertices. The function $\text{Proj}_d$ projects the combined space $\mathbb{Q}^{n+m}$ onto the *data* space $\mathbb{Q}^n$. The function $\text{Proj}_p$ projects the combined space $\mathbb{Q}^{n+m}$ onto the *parameter* space $\mathbb{Q}^m$.

$$
\begin{tikzcd}
\mathcal{H}' \arrow[r, "\mathcal{H}(p) = Proj_d(\mathcal{H}' \cap \mathcal{S}(p))"] \arrow[d, "\text{Supports}"] & \mathcal{H}(p) \arrow[d, "\text{Supports}"] \\
\mathcal{D}' \arrow[r, "\substack{\mathcal{D}(p) = Proj_d(\mathcal{D}' \cap \mathcal{S}(p)) \\ \text{(Property 1)}}"'] \arrow[d, "\text{At}"] & \mathcal{D}(p) \arrow[d, "\text{At}"] \\
F_i^m(\mathcal{D}') \arrow[r, "\substack{v_i(p) = Proj_d(F_i^m \cap \mathcal{S}(p)) \\ \text{(Theorem 1)}}"'] & v_i(p)
\end{tikzcd}
$$

Figure 1: Relationship Diagram for Theorem 1 Proof

The following property relates a polyhedron $\mathcal{D}'$ in the combined data/parameter space to its equivalent parameterized polyhedron $\mathcal{D}(p)$. We call $\mathcal{S}(\hat{p})$ the affine subspace of $\mathbb{Q}^{n+m}$ that verifies:

$$\mathcal{S}(\hat{p}) : \left\{ \begin{pmatrix} x \\ p \end{pmatrix} \in \mathbb{Q}^{n+m} \mid p = \hat{p} \right\}$$

The hyperplane $\mathcal{S}(\hat{p})$ acts like a cutting plane that slices $\mathcal{D}'$, producing the cross sectional polyhedron $\mathcal{D}(\hat{p})$. The intersection of $\mathcal{D}'$ with $\mathcal{S}(\hat{p})$ fixes the parameter space of $\mathcal{D}'$ to a constant vector $\hat{p}$ and the result is projected onto the data space to obtain $\mathcal{D}(\hat{p})$ as given in the following property.

**Property 1.** $\mathcal{D}(p) = \mathrm{Proj}_d(\mathcal{D}' \cap \mathcal{S}(p))$

The following theorem is fundamental to this paper. It states that the vertices of $\mathcal{D}(p)$ correspond to projections of the $m$-faces of $\mathcal{D}'$.

**Theorem 1.**
*Let $v_i(p)$ be a parameterized vertex of the polyhedron $\mathcal{D}(p) \in \mathbb{Q}^n$ defined over $m$ linearly independent parameters $\in \mathbb{Q}^m$. The polyhedron $\mathcal{D}' \in \mathbb{Q}^{n+m}$ can be easily constructed from $\mathcal{D}(p)$ (Eq. 4). For each vertex $v_i(p)$ of $\mathcal{D}(p)$, there exists an $m$-face $F_i^m(\mathcal{D}')$ such that $v_i(p) = \mathrm{Proj}_d(F_i^m(\mathcal{D}') \cap \mathcal{S}(p))$.*

**Proof.**
Refer to figure 1 to visualize the relationships employed in this proof. If $v_i(p)$ is a vertex of $\mathcal{D}(p)$, then by the definition of a 0-face, there must exist a supporting hyperplane $\mathcal{H}(p)$ such that $\mathcal{H}(p) \cap \mathcal{D}(p) = v_i(p)$ and $\mathcal{H}(p) \cap \mathcal{D}(p)$ is dimension 0. If hyperplane $\mathcal{H}(p) = \{x \in \mathbb{Q}^n \mid ax + bp + c = 0\}$ with constant vectors $a$, $b$, and constant scalar $c$, then a supporting hyperplane of $\mathcal{D}'$, namely $\mathcal{H}' = \{\begin{pmatrix} x \\ p \end{pmatrix} \in \mathbb{Q}^{n+m} \mid ax + bp + c = 0\}$ can be constructed. It is easily verified that that $\mathcal{H}(p) = \mathrm{Proj}_d(\mathcal{H}' \cap \mathcal{S}(p))$. Using

this fact, along with property 1, we derive the following:

$$\begin{aligned} v_i(p) &= \mathcal{H}(p) \cap \mathcal{D}(p) \\ &= \mathrm{Proj}_d(\mathcal{H}' \cap \mathcal{S}(p)) \cap \mathrm{Proj}_d(\mathcal{D}' \cap \mathcal{S}(p)) \\ &= \mathrm{Proj}_d(\mathcal{H}' \cap \mathcal{D}' \cap \mathcal{S}(p)) \end{aligned} \tag{5}$$

$\mathcal{H}' \cap \mathcal{D}' \cap \mathcal{S}(p)$ gives the single point (0-polyhedron) $\begin{pmatrix} v_i(p) \\ p \end{pmatrix}$ for a fixed parameter vector $p$. In $\mathcal{H}' \cap \mathcal{D}'$, the vector $p$ is allowed to vary in $m$-dimensional space, which it can do since the $m$ parameters are independent. The other $n$ data dimensions are affine combinations of $p$, thus $\mathcal{H}' \cap \mathcal{D}'$ is an $m$-polyhedron. Since $\mathcal{H}'$ is a supporting hyperplane of $\mathcal{D}'$, $\mathcal{H}' \cap \mathcal{D}'$ is an $m$-face $F_i^m$ of $\mathcal{D}'$, such that the intersection of that $m$-face with $\mathcal{S}(p)$ is the single point referred to above.

Since $\mathcal{H}' \cap \mathcal{D}' = F_i^m(\mathcal{D}')$, Eq. (5) can be rewritten:

$$v_i(p) = F_i^0(\mathcal{D}(p)) = \mathrm{Proj}_d(F_i^m(\mathcal{D}') \cap \mathcal{S}(p)) \tag{6}$$

□

This theorem can be extended for the case that the $m$ parameters are not linearly independent. If there are $k$ independent equalities constraining the parameter space, then there are only $m-k$ degrees of freedom in the parameter space. In this case, the parameterized vertices are obtained from the $(m-k)$-faces of $\mathcal{D}'$.

As $p$ varies over the parameter space, vertices may split, shift, and merge. The corollary below gives the range of parameter space for which a particular vertex exists. From theorem 1 above, we can conclude that vertex $v_i(p)$ exists whenever $F_i^m(\mathcal{D}') \cap \mathcal{S}(p) \neq \emptyset$.

**Corollary 1.**
*The range of $p$ over which $v_i(p)$ exists (i.e., where $F_i^m(\mathcal{D}') \cap \mathcal{S}(p) \neq \emptyset$) is $\mathrm{Proj}_p(F_i^m(\mathcal{D}'))$.*

Theorem 2 shows how the vertices of $\mathcal{D}(p)$ can be represented as affine functions of the parameter $p$.

**Theorem 2.**
*For each $m$-face $F_i^m$ of $\mathcal{D}'$, and for the set of all points $\begin{pmatrix} x \\ p \end{pmatrix} \in F_i^m$, one of the following is true:*

*i.) $x$ is an affine function of $p$, i.e., $x = v_i(p)$ where $v_i(p)$ is a parameterized vertex of $\mathcal{D}(p)$.*

*ii.) $x$ is not constrained, i.e., for a given value of $p$, more than one value of $x$ is feasible.*

**Proof.**
From the definition of a face, we know that all $\begin{pmatrix} x \\ p \end{pmatrix} \in F_i^m$ lie in an affine subspace of dimension $m$, and thus $x$ and $p$ are related according to the following set of $n$ equations:

$$\begin{bmatrix} A \\ \bar{C}_i \end{bmatrix} x = \begin{bmatrix} B \\ \bar{D}_i \end{bmatrix} p + \begin{bmatrix} b \\ \bar{d}_i \end{bmatrix}$$

where $\bar{C}_i$, $\bar{D}_i$, and $\bar{d}_i$ are a subset of the inequalities in Eq. (3) changed to equalities.

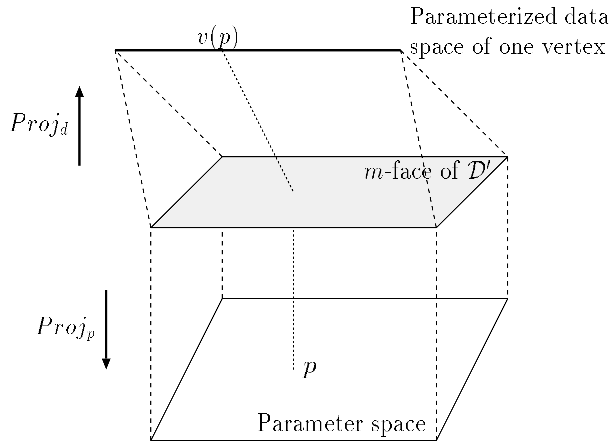


Figure 2: Computation of $v(p)$

The matrix $\begin{bmatrix} A \\ \bar{C}_i \end{bmatrix}$ is an $n \times n$ matrix, $\begin{bmatrix} B \\ \bar{D}_i \end{bmatrix}$ is an $n \times m$-matrix, and $\begin{bmatrix} b \\ \bar{d}_i \end{bmatrix}$ is an $n$-vector.

If the matrix $\begin{bmatrix} A \\ \bar{C}_i \end{bmatrix}$ admits an inverse $\begin{bmatrix} A \\ \bar{C}_i \end{bmatrix}^{-1}$, then this system of equations can be rewritten and solved as:

$$x = \begin{bmatrix} A \\ \bar{C}_i \end{bmatrix}^{-1} \begin{bmatrix} B \\ \bar{D}_i \end{bmatrix} p + \begin{bmatrix} A \\ \bar{C}_i \end{bmatrix}^{-1} \begin{bmatrix} b \\ \bar{d}_i \end{bmatrix} = v_i(p) \tag{7}$$

and $x$ is an affine function of $p$ (case i. of this theorem).

If no inverse exists, then certain elements of $x$ will not be constrained by $p$ at all on that $m$-face, and there exists more than one point in the $m$-face $F_i^m$ corresponding to each parameter value $p$, and this face does not correspond to a vertex of $\mathcal{D}(p)$ (case ii. of this theorem).

□

# 4 Computing the faces of a polyhedron

In this section, we introduce a slightly different way to represent polyhedra. In the equations and discussions so far, polyhedra have been given in their *inhomogeneous* form. Every inhomogeneous system of equations also has an equivalent *homogeneous* (or conic) form, which is more convenient for doing geometric computations [7, 16]. It is obtained by applying the transformation $x \to \begin{pmatrix} \xi x \\ \xi \end{pmatrix}$, $\xi \geq 0$, to the inhomogeneous system. The following is the equivalent homogeneous form of Eq. (1):

$$\mathcal{C} \;=\; \left\{ \begin{pmatrix} \xi x \\ \xi \end{pmatrix} \in \mathbb{Q}^{n+1} \mid \hat{A} \begin{pmatrix} \xi x \\ \xi \end{pmatrix} = 0 \;,\; \hat{C} \begin{pmatrix} \xi x \\ \xi \end{pmatrix} \geq 0 \right\} \tag{8}$$

where $\hat{A} = \left[\; A \mid -b \;\right]$ and $\hat{C} = \left[ \begin{array}{c|c} C & -d \\ \hline 0 \cdots 0 & 1 \end{array} \right]$.

The original polyhedron $\mathcal{D}$ is the intersection between $\mathcal{C}$ and the hyperplane of equation $\xi = 1$.

The vertices and rays of an inhomogeneous polyhedron have a unified and homogeneous representation as rays in the homogeneous form. The dual of Eq. (8) and homogeneous form of Eq. (2) is the following Minkowski representation:

$$\mathcal{C} \;=\; \left\{ \begin{pmatrix} \xi x \\ \xi \end{pmatrix} \in \mathbb{Q}^{n+1} \mid \begin{pmatrix} \xi x \\ \xi \end{pmatrix} = \hat{L}\lambda' + \hat{R}\mu',\ \mu' \geq 0 \right\} \tag{9}$$

where $\hat{L} = \left[ \begin{array}{c} L \\ \hline 0 \cdots 0 \end{array} \right]$ and $\hat{R} = \left[ \begin{array}{c|c} R & V \\ \hline 0 \cdots 0 & 1 \cdots 1 \end{array} \right]$, $\lambda'$ and $\mu'$ are free-valued column vectors.

The unification of vertices and rays simplifies the representation and computation of polyhedra in the homogeneous form. The cost of this new representation is the one additional dimension in the homogeneous representation.

## 4.1 Saturation and the incidence matrix

The algorithm to find the set of $m$-faces of a polyhedron relies heavily on the notion of saturation and on the properties of the incidence matrix [16]. In this section, these concepts will be introduced.

**Definition 5.** *A ray $r$ is said to* **saturate** *an inequality $a^T x \geq 0$ when $a^T r = 0$, it* **verifies** *the inequality when $a^T r > 0$, and it* **does not verify** *the inequality when $a^T r < 0$. Likewise, a ray $r$ is said to* **saturate** *an equality $a^T x = 0$ when $a^T r = 0$, and it* **does not verify** *the equality when $a^T r \neq 0$. Equalities and inequalities are collectively called* **constraints**. *A constraint is* **satisfied** *by a ray if the ray saturates or verifies the constraint.*

The incidence matrix $I$ is a boolean matrix which has a row for every constraint (rows of $\hat{A}$ and $\hat{C}$, Eq. 8) and a column for every line or ray (columns of $\hat{L}$ and $\hat{R}$, Eq. 9). Each element $I_{ij}$ in $I$ is defined as follows:

$$I_{ij} = \begin{cases} 1, & \text{if constraint } c_i \text{ is saturated by ray(line) } r_j, \text{ i.e., } c_i^T r_j = 0 \\ 0, & \text{otherwise, i.e., } c_i^T r_j > 0 \;\; (c_i \text{ is verified by } r_j, \text{ but not saturated}) \end{cases}$$

Substituting the equation for $\begin{pmatrix} \xi x \\ \xi \end{pmatrix}$ in Eq. 9 into the equations involving $\begin{pmatrix} \xi x \\ \xi \end{pmatrix}$ in Eq. 8, we obtain:

$$\forall (\mu \geq 0,\ \lambda)\ :\ \left\{ \begin{array}{l} \hat{A}\hat{L}\lambda + \hat{A}\hat{R}\mu = 0 \\ \hat{C}\hat{L}\lambda + \hat{C}\hat{R}\mu \geq 0 \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} \hat{A}\hat{L} = 0,\ \hat{A}\hat{R} = 0 \\ \hat{C}\hat{L} = 0,\ \hat{C}\hat{R} \geq 0 \end{array} \right.$$

where rows of $\hat{A}$ and $\hat{C}$ are equalities and inequalities, respectively, and where columns of $\hat{L}$ and $\hat{R}$ are lines and rays, respectively.

From the above demonstration, we know that all rows of the $I$ matrix associated with equations ($A$) are 1 (saturated), and all columns of the $I$ matrix associated with lines ($L$) are also 1 (saturated). Only entries associated with inequalities ($C$) and rays ($R$) can have 1's (saturations) as well as 0's (non-saturations). This is illustrated in the following diagram representing the incidence matrix $I$.

| I | $\hat{L}$ | $\hat{R}$ |
|---|---|---|
| $\hat{A}$ | (1) | (1) |
| $\hat{C}$ | (1) | (0 or 1) |

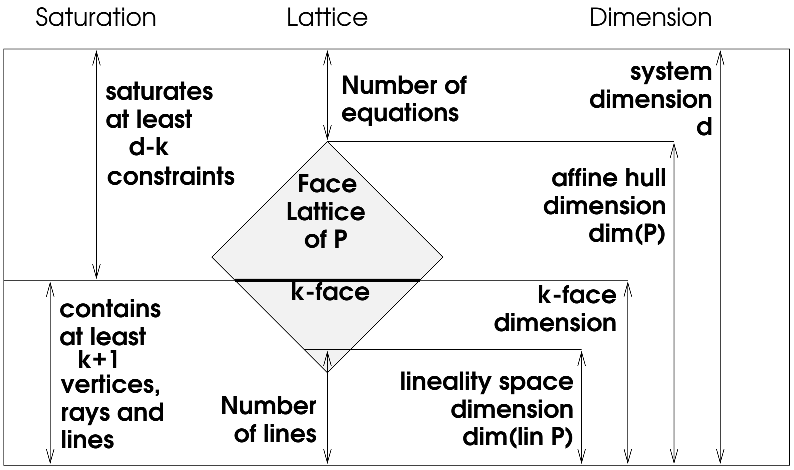


Figure 3: The $k$-face in context of the face lattice

The algorithm to construct all of the $k$-faces of a polyhedron is based on relationships given in Prop. 2. below. Figure 3 illustrates these relationships between the face lattice, the dimension of the polyhedron, and the $k$-face.

**Property 2.** *Given any polyhedron $\mathcal{P}$, and letting:*
*(number of lines) $\leq k \leq$ (d $-$ number of equalities), then the following are true:*

*a. dim($\mathcal{P}$) = d - (number of equations)*

*b. dim(lin $\mathcal{P}$) = (number of lines)*

*c. Every $k$-face contains at least $k+1$ vertices, rays, or lines, which must include all of the lines and at least one vertex.*

*d. Every vertex, ray, and line in a $k$-face saturates at least $d-k$ equalities and inequalities, which must include all of the equalities.*

## 4.2 Algorithm to find the set of $k$-faces

Given a polyhedron $\mathcal{P}$ of dimension $d$,

1. For all combinations of $(d - k)$ constraints (which always includes all of the equations in $\mathcal{P}$),

2. Intersect the selected rows of the incidence matrix,

3. Count the number of ones in the resulting saturation vector, which is equal to the number of vertices, rays, and lines saturated by *all* of the selected constraints. If the number of saturations is equal to or greater than $k+1$ and the set contains at least one vertex, then the set of saturated vertices, rays, and lines constitutes a viable $k$-face of the polyhedron.

4. *Elimination of redundant k-faces*
   If another constraint that has already been considered saturates the same set of vertices, rays, and lines, then this face has already been reported.
   Otherwise, this face is new and is reported.

5. Continue to the next combination of constraints (step 1.).

This algorithm works for any polyhedron, not just for polytopes. It assumes that the polyhedron $\mathcal{P}$ is represented using Motzkin's double description, as the set of the vertices of $\mathcal{P}$, the set of the facets of $\mathcal{P}$, and the incidence matrix relating the vertices and facets (see section 4.1). Given this representation, the algorithm to find all of the $k$ faces can be implemented recursively as follows:

**k-Face Generation Algorithm**

*Input:* rays/vertices of $\mathcal{P}$.
facets (inequalities) of $\mathcal{P}$ numbered 1 to $f$.
$I[i], 1 \le i \le f$, the set of rays/vertices saturated by each facet $i$.

*Output:* the set of $k$-faces of $\mathcal{P}$ represented by
$\{\mathcal{V}, \mathcal{F}\}$, subsets of rays/vertices and facets of $\mathcal{P}$

```
procedure ScanFaces ( k : dimension of faces
                      i : current facet (inequality) being considered;
                      needed : number of facets needed to complete a k-face;
                      F : a set of facets.
                      V : the set of all vertices/rays saturated by the facets in F
                    )

begin
  if needed= 0 then                          — {V, F} is a candidate k-face
    for j = 1 ··· i − 1 do                   — redundancy check
      if j not in F and I[j] ∩ V = V then    — is there a previous facet that saturates V ?
        return;                              — face is redundant if there is a another
    end                                      —    facet j numbered less than i that also
    output({V, F})                           —    saturates all of V but is not found in F
  end
  if i + needed > f + 1 then return          — not possible to complete a k-face
  V' = V ∩ I[i]                              — compute a new V with facet i included
  if |V'| ≥ k + needed then                  — viability: are there enough vertices/rays left?
    ScanFaces ( k, i + 1, needed − 1, F ∪ i, V' ); — keep facet i, and try adding i + 1
  end
  if i + needed ≥ f + 1 then return          — not possible to complete a k-face
  ScanFaces ( k, i + 1, needed, F, V );      — don't keep facet i, try adding i + 1
end
```

*To generate all k-faces, call:*
ScanFaces ( k, 1, (dim $\mathcal{P}$ - k), { all equalities }, { all lines, rays/vertices } );

### 4.3 Discussion and Related Work

The above combinatorial algorithm finds candidate $k$-faces by first generating all combinations of $d-k$ facets (inequalities) out of the total set of $n$ facets. Not all combinations are viable $k$-faces. A simple test for viability counts the number of rays/vertices in the set $\mathcal{V}$ in $\mathrm{O}(m)$ time. Candidate $k$-faces that have passed the viability test are then tested for redundancy in $\mathrm{O}(nm)$. Thus the total worst case time for this algorithm is $\mathrm{O}\left(\binom{n}{d-k}(nm+m)\right)$. By duality, this algorithm could also be written to run in $\mathrm{O}\left(\binom{m}{k+1}(nm+m)\right)$. Either way, execution time is bounded from below by $\Omega\left(f(nm+m)\right)$, where $f$ is the number of $k$-faces.

Fukuda and Rosta [6] described another combinatorial algorithm that finds the *complete* face lattice (not just the $k$-faces). The complexity of their algorithm is $\mathrm{O}(f^2dn)$ where $n$ is the number of facets, $d$ is the dimension of the polyhedron, and $f$ is the total number of *all* faces in the lattice found by the algorithm. The algorithm in the present paper differs from the Fukuda-Rosta algorithm in that it only finds the $k$-faces, not the entire lattice, and does so without generating or storing other faces in the lattice. It also differs in the representation of the polyhedron and faces. The algorithm in this paper uses a *double description* representation, consisting of the set of vertices, the set of facets, and the incidence matrix relating the two sets.

# 5 Computing the vertices of a parameterized polyhedron

To facilitate this computation, we move to the homogeneous form that was discussed in section 4. The homogeneous representation of the polyhedron $\mathcal{D}'$ (Eq. 4) is:

$$\mathcal{C}' = \left\{ \begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix} \in \mathbb{Q}^{n+m+1} \mid \hat{A}\begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix} = 0,\ \hat{C}\begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix} \geq 0 \right\} \tag{10}$$

and its dual homogeneous Minkowski representation is:

$$\mathcal{C}' = \left\{ \begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix} \in \mathbb{Q}^{n+m+1} \mid \begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix} = \hat{L}\lambda + \hat{R}\mu\ ,\ \mu \geq 0 \right\} \tag{11}$$

where $x$ is an $n$-dimensional column vector, $p$ is an $m$-dimensional column vector, and where $\hat{A} = \left[\begin{array}{c|c|c} A & -B & -b \end{array}\right]$, $\hat{C} = \left[\begin{array}{c|c|c} C & -D & -d \\ \hline 0\cdots0 & 0\cdots0 & 1 \end{array}\right]$, $\hat{L} = \left[\begin{array}{c} L \\ \hline 0\cdots0 \end{array}\right]$ and $\hat{R} = \left[\begin{array}{c|c} R & V \\ \hline 0\cdots0 & 1\cdots1 \end{array}\right]$, and $\lambda$ and $\mu$ are free-valued column vectors.

Placing Eq. 7 in homogeneous form, we get

$$\begin{pmatrix}\xi x\\ \xi\end{pmatrix} = T_i \begin{pmatrix}\xi p\\ \xi\end{pmatrix} \tag{12}$$

where:

$$T_i = \left[\begin{array}{c|c} \begin{bmatrix}A\\ \bar{C}_i\end{bmatrix}^{-1}\begin{bmatrix}B\\ \bar{D}_i\end{bmatrix} & \begin{bmatrix}A\\ \bar{C}_i\end{bmatrix}^{-1}\begin{bmatrix}b\\ \bar{d}_i\end{bmatrix} \\ \hline 0\cdots0 & 1 \end{array}\right]$$

The matrix $T_i$ can be computed using the above equation, however a simpler dual formulation is also available. We first need to define new homogeneous data and parameter projection functions $\mathrm{Proj}_d$ and $\mathrm{Proj}_p$:

$$\mathrm{Proj}_d\left(\left[\begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix}\cdots\right]\right) = \left[\begin{pmatrix}\xi x\\ \xi\end{pmatrix}\cdots\right]$$
$$\mathrm{Proj}_p\left(\left[\begin{pmatrix}\xi x\\ \xi p\\ \xi\end{pmatrix}\cdots\right]\right) = \left[\begin{pmatrix}\xi p\\ \xi\end{pmatrix}\cdots\right]$$

For each $m$-face $F_i^m$, we create a matrix $M_i$ whose columns consist of the set of $k$ ($\geq m+1$) extremal rays (in homogeneous form) lying in $F_i^m$. Then, we can set up a system of equations similar to Eq. 12 as follows:

$$\mathrm{Proj}_d(M_i) = T_i\,\mathrm{Proj}_p(M_i)$$

If a linear relationship $T_i$ exists, it can be easily computed by:

$$T_i = \mathrm{Proj}_d(M_i)\,\mathrm{Proj}_p(M_i)^{-R}$$

where $\mathrm{Proj}_p(M_i)^{-R}$ is the right matrix inverse of $\mathrm{Proj}_p(M_i)$. However, if the right inverse of matrix $\mathrm{Proj}_p(M_i)$ does not exist, then this face does not correspond to a single vertex of $\mathcal{D}(p)$ (Theorem 2).

## 5.1 The parametric vertex enumeration problem

The result of this paper is to show how to compute the matrix $V(p)$ containing the vertices of the polyhedron, in the data space alone as piecewise affine function of the parameters as in the equation:

$$\mathcal{D}(p) = \{x \in \mathbb{Q}^n \mid x = L\lambda + R\mu + V(p)\nu\ ,\ \ \mu,\nu \geq 0\ ,\ \ \sum_i \nu_i = 1\}$$

Notice that the directions of rays and lines in a parameterized polyhedron never depend on the parameters. This is a direct consequence of the following theorem:

**Theorem 3.** *Let $\mathcal{D}(p)$ be an $n$-dimensional parameterized polyhedron. The linear space supporting any face $F_i^k(\mathcal{D}(p))$ does not depend on the parameter $p$.*

**Proof.**
Any facet $F_i^{n-1}(\mathcal{D}(p))$ of a parameterized polyhedron is supported by a hyperplane $H_i(p)$ defined by:

$$H_i(p) = \{x \in \mathbb{Q}^n \mid a_i x + b_i p + c_i = 0\}$$

where $a_i$ and $b_i$ are constant rational vectors and $c_i$ is a constant rational scalar. This hyperplane is orthogonal to constant vector $a_i$ in the space $\mathbb{Q}^n$ for any given value of $p$.

Since all faces of dimension $k$ are the intersection of $n-k$ facets, the affine space supporting them is orthogonal to the corresponding set of constant vectors $a_i$. As an immediate result, the linear space

supporting each face is constant.
□

The faces of a parameterized polyhedron may shift position depending on the parameters, but not change direction. Therefore, the lines and rays of a parameterized polyhedron do not depend on the parameters. Lines in matrix $L$ and rays in matrix $R$ may be easily determined as functions of the lines and rays in $\mathcal{D}'$.

$V(p)$ is defined as the set of parameter dependent vertices $v_i(p)$, each defined over a fixed domain in the parameter space. Each $v_i(p)$ is defined as an affine function of the parameters $p$ in the homogeneous data space as follows:

$$\begin{pmatrix} \xi v_i(p) \\ \xi \end{pmatrix} = \begin{cases} T_i \begin{pmatrix} \xi p \\ \xi \end{pmatrix} & \text{if } p \in \mathrm{Proj}_p(F_i^m) \\ 0 & \text{otherwise} \end{cases}$$

where $T_i$ is a constant $(n+1) \times (m+1)$ matrix and the validity domain $\mathrm{Proj}_p(F_i^m)$ is the $m$-dimensional subdomain of the parameter space $\mathbb{Q}^m$ on which this vertex is defined. Outside this parameter subdomain, the vertex is defined to be the vector 0 which is a null ray in the homogeneous space.

We represent $V(p)$ as a set of pairs $[T_i, \mathrm{Proj}_p(F_i^m)]$.

## 5.2 Procedure to construct $V(p)$

The preliminary step of this algorithm consists of removing all dependencies between the parameters in order to have $m$ independent parameters. Then the following procedure will compute the pairs $[T_i, \mathrm{Proj}_p(F_i^m)]$ which define $V(p)$:

1. Convert the homogeneous representation of the parameterized domain from the implicit form:

$$A \begin{pmatrix} \xi x \\ \xi p \\ \xi \end{pmatrix} = 0 \ , \ C \begin{pmatrix} \xi x \\ \xi p \\ \xi \end{pmatrix} \geq 0$$

   to the Minkowski form (Eq. 11) using the Polyhedral Library [16] for example:

$$\begin{pmatrix} \xi x \\ \xi p \\ \xi \end{pmatrix} = \hat{L}\lambda + \hat{R}\mu, \quad \mu \geq 0$$

2. Compute the $m$-faces using the algorithm given in section 4.2

   Each $m$-face $F_i^m$ of $\mathcal{D}'$ is represented by a set of lines, rays and vertices, or in the homogeneous space by a set of lines and rays:

$$\left\{ \begin{pmatrix} \xi x \\ \xi p \\ \xi \end{pmatrix} \in \mathbb{Q}^{n+m+1} \mid \begin{pmatrix} \xi x \\ \xi p \\ \xi \end{pmatrix} = \hat{L}\lambda + \hat{R}_i\mu \ , \quad \mu \geq 0 \right\}$$

where each column of $\hat{L}$ is a bidirectional ray and each column of $\hat{R}_i$ is a unidirectional ray. The columns of $\hat{R}_i$ are a subset of the columns of $\hat{R}$.

Let $M_i = [\hat{L} \mid \hat{R}_i]$.

The matrix $M_i$ has $(d+1)$ rows and at least $(m+1)$ columns.

The last two steps are done for each $m$-face $F_i^m$:

3. If the matrix $\mathrm{Proj}_p(M_i)$ has no right inverse, this $m$-face does not correspond to a single vertex of $\mathcal{D}(p)$. Proceed to the next $m$-face and go back to step 3.

   Compute $T_i = \mathrm{Proj}_d(M_i)\,\mathrm{Proj}_p(M_i)^{-R}$.

4. Compute the validity domain $\mathrm{Proj}_p(F_i^m)$ by projecting the $m$-face defined by $\hat{L}$ and $\hat{R}_i$ on the parameter space $\mathbb{Q}^m$. Report $[T_i, \mathrm{Proj}_p(F_i^m)]$ as a vertex of $\mathcal{D}(p)$.

   Proceed to the next $m$-face and go back to step 3.

### 5.3 Complexity analysis

The complexity of the procedure is:

$$\mathrm{O}(f(m^3 + m^2 n))$$

where f is the number of $m$-faces of the polyhedron $\mathcal{D}'$, $n$ is the dimension of the data space, and $m$ is the dimension of the parameter space. This is derived from the fact that for each $m$-face, we do a matrix inversion ($\Theta(m^3)$) and a matrix multiplication ($\Theta(m^2 n)$). We assume here again that the polyhedron is stored in its double description form, consisting of the set of vertices, the set of facets, and the incidence matrix (step 1 is then trivial).

In practical examples $m$ and $n$ are generally small.

On a Pentium 133MHz, it took less than 0.03 seconds to compute the 34 parameterized vertices of a 4-polyhedron with 4 parameters, containing 126 faces, and 0.22 seconds to compute the 89 parameterized vertices of a 5-polyhedron with 5 parameters, containing 1170 faces.

# 6 Applications

## 6.1 An example

Let us consider the following parameterized polyhedron depending on two parameters $p_1$ and $p_2$ :

$$\mathcal{D}(p_1, p_2) = \{0 \le j \le p_1, j \le i \le p_2\}$$

This polyhedron is represented in figure 4 if condition $p_1 \le p_2$ is verified, and in figure 5 if $p_1 \ge p_2$ Using the algorithm described above, we will find its vertices as function of the parameters.

Polyhedron $\mathcal{D}'$ associated with $\mathcal{D}(p_1, p_2)$ is given by :

$$\mathcal{D}' = \left\{ \begin{pmatrix} i \\ j \\ p_1 \\ p_2 \end{pmatrix} \in \mathbb{Q}^4 \mid 0 \le j \le p_1, j \le i \le p_2 \right\}$$

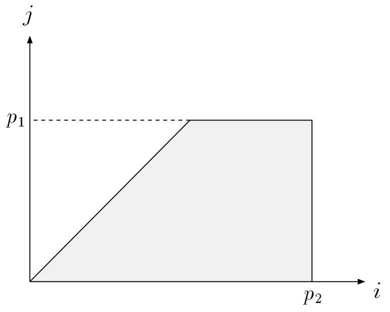


Figure 4: Domain $\mathcal{D}(p_1,p_2), p_1 \leq p_2$

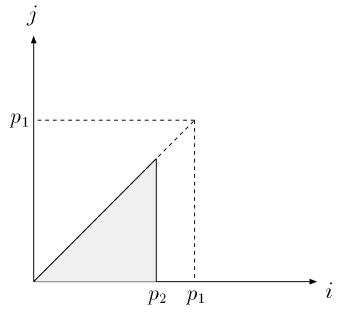


Figure 5: Domain $\mathcal{D}(p_1,p_2), p_1 \geq p_2$

The implicit homogeneous form of $\mathcal{D}(p_1,p_2)$ is:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \xi i \\ \xi j \\ \xi p_1 \\ \xi p_2 \\ \xi \end{pmatrix} \geq 0$$

and its Minkowski representation (given by the Polyhedral Library):

$$\begin{pmatrix} \xi i \\ \xi j \\ \xi p_1 \\ \xi p_2 \\ \xi \end{pmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{pmatrix}, \quad \mu_i \geq 0$$

The polyhedron $\mathcal{D}'$ has one vertex $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ and 4 rays $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.

Since the number of parameters $m$ is 2, the parameterized vertices correspond to the 2-faces of $\mathcal{D}'$. There are six such 2-faces. The calculation of $T_i$ $(1 \leq i \leq 6)$ and the polyhedra $\mathrm{Proj}_p(F_i^m)$ are presented on table I.

The algorithm finds five vertices, depending on the values of the parameter.

- If the condition $p_2 \geq p_1 \geq 0$ holds, the vertices of $\mathcal{D}(p_1,p_2)$ are: $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} p_1 \\ p_1 \end{pmatrix}$, $\begin{pmatrix} p_2 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} p_2 \\ p_1 \end{pmatrix}$.
- If $p_1 \geq p_2 \geq 0$, the vertices of $\mathcal{D}(p_1,p_2)$ are: $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} p_2 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} p_2 \\ p_2 \end{pmatrix}$.

Notice that if $p_1 = p_2$ or $p_1 = 0$ or $p_2 = 0$, some vertices are redundant. Over the rest of the parameter space $\mathbb{Q}^m$, i.e., $p_1 < 0$ or $p_2 < 0$, the polyhedron is empty.

| $\hat{M}_i$ | $\mathrm{Proj}_d(M_i)$ | $\mathrm{Proj}_p(M_i)^{-R}$ | $T_i \quad \rightarrow \quad v_i(p)$ | $\mathrm{Proj}_p(F_i^m)$ |
|---|---|---|---|---|
| $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\perp$ | – | – |
| $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ | $p_1 \geq 0$<br>$p_2 \geq 0$ |
| $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} -1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} p_1 \\ p_1 \end{pmatrix}$ | $p_2 \geq p_1 \geq 0$ |
| $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} p_2 \\ 0 \end{pmatrix}$ | $p_1 \geq 0$<br>$p_2 \geq 0$ |
| $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} -1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} p_2 \\ p_1 \end{pmatrix}$ | $p_2 \geq p_1 \geq 0$ |
| $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} p_2 \\ p_2 \end{pmatrix}$ | $p_1 \geq p_2 \geq 0$ |

Table I: Calculation of $T_i$ and $\mathrm{Proj}_p(F_i^m)$

## 6.2 Applications to automatic parallelization

A parameterized system of affine recurrence equations (PARE) is a set of equations of the form :

$$z \in \mathcal{D}(p) \quad \rightarrow \quad X[z] = f(..., Y[g(z,p)], ...) \tag{13}$$

where $X$ and $Y$ are variable names, $\mathcal{D}(p) \subset \mathbb{Z}^n$ is a parameterized $n$-polyhedron called the iteration space, and $g(z,p)$ is an affine function called the index function, $g(z,p) = Rz + Qp + r$.

A parallel implementation of a system of parameterized affine recurrence equations can be found by choosing an affine space-time mapping. The affine per variable allocation function is expressed by: $Alloc_X(z) = \sigma_X z + \gamma_X$. The row vectors of $\sigma_X$ are the basis vectors of the virtual processor space. The affine schedule function is chosen as:

$$t_X(z) = \lambda z + \alpha_X \tag{14}$$

Notice that $t_X(z)$ can be a vector in the case of a multidimensional schedule [4]. The row vectors of $\lambda$ are orthogonal to the affine spaces of simultaneous computations called *fronts*. The space-time mapping is expressed by a *full rank* $n \times n$ transformation matrix and an $n$-vector for each variable:

$$T_X = \begin{pmatrix} \lambda \\ \sigma_X \end{pmatrix} \quad , \quad c_X = \begin{pmatrix} \alpha_X \\ \gamma_X \end{pmatrix}$$

The dependence information related to a PARE such as Eqn. 13 is given by its *utilization* and *emission* sets. A utilization set related to variable $Y$ in Eqn. 13 corresponds to the set of points using the result computed at a given point $z_0$: $Util_Y(z_0, p)$ is a polyhedron parameterized by $z_0 \in \mathcal{D}(p)$ and $p$:

$$Util_Y(z_0, p) = \{z \in \mathcal{D}(p) \mid g(z, p) = z_0\}$$

The emission set related to variable $Y$ in Eqn. 13 is the set of points where a result is computed that needs to be transmitted to a utilization set:

$$\begin{aligned} Emit_Y(p) &= \{z_0 \in \mathbb{Z}^n \mid \exists z \in \mathcal{D}(p) \text{ such that } z_0 = g(z, p)\} \\ &= \{Rz + Qp + r \mid z \in \mathcal{D}(p)\} \end{aligned}$$

There are many applications of the algorithm given in this paper in the context of PARE's. Some of them are described below.

### i) Finding a schedule: the vertex method

In affine systems, the dependence between variables $Y$ and $X$ is expressed by a set of affine *dependence vectors* of the form $d = z - z_0$, where $z \in Util_Y(z_0, p)$ and $z_0 \in Emit_Y(p)$. Quinton and Van Dongen [13] proved that this set of dependence vectors defines a convex polyhedron of $\mathbb{Z}^n$ parameterized by $p$:

$$Range_Y(p) = \{(Id - R)z - Qp - r \mid z \in \mathcal{D}(p)\}.$$

The validity condition on the schedule is expressed by:

$$t_Y(z_0) \prec t_X(z) \tag{15}$$

where $\prec$ is the lexicographical *lower than* operator. This condition ensures that a data is computed before being used by another computation. Assuming an affine timing of the form in Eqn. 14 and assuming that the polyhedron $Range_Y(p)$ has a Minkowski representation consisting of a set of vertices $v_i(p)$, rays $r_j$, and lines $l_k$, then Quinton and Van Dongen showed that the constraints on the timing function in Eqn. 15 can be given as:

$$\begin{array}{rll} \text{i.} & \forall v_i(p) \in Range_Y(p): & \lambda v_i(p) + \alpha_X - \alpha_Y \succ 0 \\ \text{ii.} & \forall r_j \in Range_Y(p): & \lambda r_j \succ 0 \\ \text{iii.} & \forall l_k \in Range_Y(p): & \lambda l_k = 0 \end{array}$$

This of course has to be done for each parameter validity domain given by the algorithm. The above set of constraints on the schedule matrix $\lambda$ can be solved in order to derive a valid schedule.

Notice that condition *i.* is non-linear. Fortunately, it can be broken down into several linear conditions, one for each extremal ray or vertex of the domain of $p$.

Another way of finding the constraints on the schedule is to consider the whole set of dependence vectors and to derive a set of linear constraints from them [3]. This set of constraints is easier to generate, but the complexity of the resulting system is greater in terms of number of variables and constraints.

### ii) Communications optimization

The communications on a virtual processor array result from the dependences between the variables. If the data computed on an emission point can not be broadcast onto its whole utilization set due to architectural constraints, the pipeline method has to be applied in order to reduce the number of distant communications. This pipeline consists of sending the data from the emission point $z_0$ to the first activated utilization point $v_{min}$, and then transmitting it from neighbor to neighbor within the utilization set. This later transmission can be done using local communications on the processor array since the allocation function is affine and the utilization set is a convex polyhedron. But the first communication vector is the projection of one of the dependence vectors, called $d_{min} = v_{min} - z_0$. It is a parameterized vector depending both on $p$ and $z_0$ and therefore the resulting communication might be distant (non-local).

In order to build an efficient circuit or parallel implementation of the program, these distant communications have to be avoided. By choosing one of the vertices of the utilization set as $v_{min}$, we can find the set of linear constraints on the allocation matrices $\sigma_X$ and $\sigma_Y$ in order to project this parameter-dependent vector onto a constant communication vector in the processor array. The set of candidates $v_i(z_0, p)$ is found by the algorithm given in this paper. The choice of one of them as $v_{min}$ also implies a set of constraints on the schedule $\lambda$. However, this technique can only be applied if the extremal dependence vector $v_i(z_0, p)$ is integer. Since the algorithm given in this paper deals with rational vertices of rational parameterized polyhedra, $v_i(z_0, p)$ may be rational. Further work may lead to the computation of the first *integer* dependence vector as function of the parameters.

This procedure is fully described and a heuristic for minimizing the number of distant communications is given in [10].

### iii) Counting integer points in a parameterized polyhedron

This subsection describes how to derive the parametric expression for the number of integer points contained in a parameterized polytope $\mathcal{P}(p)$. This expression is in a form called an *Ehrhart polynomial.* The results given here are fully explained in [2].

A *q-periodic number* $u(p)$ is defined by a $q$-dimensional array $u_I$, indexed by vector $I = (i_1, ..., i_q)$, and of maximum index size $s_1 \times ... \times s_q$ such that: $u(p) = u_I$ with $I = (p_1 \bmod s_1, \ldots, i_q = p_q \bmod s_q)$.

Example: $(-1)^{p_1}$ is equal to the 1-periodic number $[1, -1]_{p_1}$.

An Ehrhart polynomial $E(p)$ is a polynomial in which the coefficients are periodic numbers. Its *pseudo-period* is the least common multiple of the periods ($s$'s) of its coefficients. The *denominator of a rational polytope* $\mathcal{P}(p)$ is the least common multiple of the denominators of its rational vertices. A *homothetic-bordered polytope* is a polytope whose vertices are fixed affine functions of the parameters over a domain $\mathcal{D}_p$.

Ehrhart's main result is that the pseudo-period of any Ehrhart polynomial $E(p)$ associated with a homothetic-bordered polytope $\mathcal{P}(p)$ is at most equal to the denominator of $\mathcal{P}(p)$. The algorithm given in this paper can be used to compute the denominator of $\mathcal{P}(p)$.

Thus, the general form of the Ehrhart polynomial can be found: its period is equal to the denominator of $\mathcal{P}(p)$ and its maximum degree is of course the dimension of $\mathcal{P}(p)$. Numeric evaluations of the Ehrhart polynomial for a few small fixed values of the parameters can be easily found. From these initial values, we can compute the Ehrhart polynomial coefficients associated with $\mathcal{P}(p)$ by solving a system of linear equalities.

The result is a symbolic expression for the number of integer points contained in a parameterized polytope that is given as a set of pairs consisting of a validity domain and an Ehrhart polynomial: $(\mathcal{D}_p, E(p))$.
Example (from 6.1):

$$\begin{array}{rcl} \#\,\mathcal{D}(p_1,p_2) & = & \#\{(i,j)\in \mathbb{Z}^2 \mid 0\le j\le p_1, j\le i\le p_2\} \\ & = & \left\{ \begin{array}{llll} \text{if} & p_1\le p_2 & \text{then} & -\frac{1}{2}{p_1}^2+p_1p_2+\frac{1}{2}p_1+p_2+1 \\ \text{if} & p_2\le p_1 & \text{then} & \frac{1}{2}{p_2}^2+\frac{3}{2}p_2+1 \end{array} \right. \end{array}$$

This result is very useful in the analysis of a parallel implementation of a system of PAREs. The set of fronts associated with a schedule function can be determined as a function of the parameters $p$ and the time step $t$ (which is a vector in the case of multi-dimensional schedule). The Ehrhart polynomial $E(p,t)$ associated to these fronts is the number of processors active at each time step. The maximum of this function over $t$ is the peak parallelism of the implementation, i.e. the maximum number of active processors at any given time step. A complete example can be found in [2].

Another use of the Ehrhart polynomial is the computation of the number of communications related to a dependence. When the pipeline method is being used, the number of distant communications (when they have not been eliminated by an efficient allocation) is equal to the number of points in the emission set. This number is also the number of broadcast communications when using the broadcast method.

Since $Emit_Y(p)$ is the image of the integer parameterized polyhedron $\mathcal{D}(p)$ by the affine function $g(z,p)$, it is a sparse polyhedral lattice. An *integer* parameterized polyhedron, homothetic to that lattice can be easily found by projecting $\mathcal{D}(p)$ along the vectors of $Ker(R)$, where $g(z,p)=Rz+Qp+r$. Finally, the Ehrhart polynomial associated with this polyhedron is the number of points of $Emit_Y(p)$.

# 7 Conclusion

We have shown how to represent parameterized polyhedra and we have presented an algorithm to find the vertices of such polyhedra as functions of the parameters.

This algorithm is used to analyze parameterized systems of affine recurrence equations, parametrically specified in terms of the size of the inputs. It is also used as part of another important analysis procedure to find the enumeration of integer points in a polyhedron as a function of the parameters. It can be used to do symbolic dependence analysis for computing a schedule or for analyzing communication costs. The results of these analyses guide the transformation of systems of recurrence equations in order to synthesize regular array circuits or generate parallel code.

Another important tool that is useful in analyzing parameterized systems is PIP (Parametric Integer Programming [3]) which finds the lexicographic minimum (or maximum) of a parameterized polyhedron using a modified simplex method. PIP gives the set of parameterized vertices of the polyhedron, qualified by subdomains of the parameter space, that are the lexicographic minimum (maximum) point in the polyhedron as a function of the parameters. The tool described in this paper differs from PIP in that it finds the complete set of parameterized vertices for a given polyhedron.

The procedure was originally written to be used by the toolbox OPERA [9] in development at the ICPS laboratory. It is written in C and has been placed in the public domain. Subsequently, it has been integrated into the Polyhedral Library [16], jointly developed by IRISA and BYU, and is freely distributed with the library (`http://www.ee.byu.edu/~wilde/polyhedra.html`).


## Acknowledgments

We would like to acknowledge Sanjay Rajopadhye for insights he provided on early versions of this paper, Catherine Mongenet for a careful reading and commentary of the paper, and the referees for their thoughtful and constructive critique.


# References


[1] Avis, D., Bremner, D. How Good are Convex Hull Algorithms? In *11th Annual ACM Symposium on Computational Geometry*, Vancouver, B.C., June 1995, pp 20–28.

[2] Clauss, Ph. and Loechner V. Parametric Analysis of Polyhedral Iteration Spaces. In *International Conference ASAP'96*, Chicago, Illinois, August 1996, pp 415–424.

[3] Feautrier, P. Parametric Integer Programming. *RAIRO Recherche Opérationelle*, **22**(3), pp. 243–268, 1988.

[4] Paul Feautrier. Some efficient solutions to the affine scheduling problem, Part I, One dimensional time, and Part II, Multidimensional time. *International Journal of Parallel Programming*, 21(5 and 6), 1992.

[5] Fernandez, F., and Quinton, P. Extension of Chernikova's Algorithm for Solving General Mixed Linear Programming Problems. Technical Report 437, IRISA, Rennes, 1988.

[6] Fukuda, K., and Rosta, V. Combinatorial face enumeration in Convex Polytopes. In *Computational Geometry*, no. 4, pp. 191–198, 1994.

[7] Goldman, A. J. Resolution and separation theorems for polyhedral convex sets. *Linear inequalities and related systems*, Kuhn and Tucker Ed., Princeton U., NJ, 1956.

[8] Le Verge, H. A note on Chernikova's Algorithm. Technical Report 635, IRISA, Rennes, 1992.

[9] Loechner, V., and Mongenet, C. OPERA : A Toolbox for Loop Parallelization. *First International Workshop on Software Engineering for Parallel and Distributed Systems, PDSE'96*, Berlin, March 1996.

[10] Loechner, V., and Mongenet, C. Communication minimization in the context of affine problems. ICPS Technical Report 97-01, 1997. `http://icps.u-strasbg.fr/pub-97/`

[11] Mongenet, C., Clauss, Ph., and Perrin, G. R. A geometrical coding to compile affine recurrence equations on regular arrays. *5th International Parallel Processing Symposium, IPPS'91*, Anaheim, California, 1991.

[12] Motzkin, T. S., Raiffa, H., L. Thompson, G. and Thrall, R. M. The double description method., 1953, republished in *Theodore S. Motzkin: Selected Papers*, Birkhauser, Boston, 1983.

[13] Quinton, P., and Van Dongen, V. The Mapping of Linear Recurrence Equations on Regular Arrays. *Journal of VLSI Signal Processing,* **1**, pp. 95–113, 1989.

[14] Rajopadhye, S. Synthesizing systolic arrays with control signals from recurrence equations. *Distributed Computing*, **3**, pp. 88–105, 1989.

[15] Schrijver, A. *Theory of linear and integer programming.* John Wiley and Sons, NY, 1986.

[16] Wilde, D. K. A library for doing polyhedral operations. Master's thesis, Oregon State University, Corvallis, Oregon, Dec 1993. Also published in IRISA technical report PI 785, Rennes, France, 1993.

[17] Ziegler, Günter. *Lectures on Polytopes.* Graduate Texts in Mathematics, no. 152, Springer-Verlag, NY, 1995.