ISSN 1749-3889 (print), 1749-3897 (online)
International Journal of Nonlinear Science
Vol.31(2021) No.2/3,pp.128-133

Enumerating the Vertices of a Parametric Polyhedron in MAPLE

Yuegang Zhao, Wuqiao Liu*, Fa Yao
School of Mathematical Sciences, Jiangsu University, Zhenjiang, Jiangsu 212013, P.R. China
(Received 2 March 2021, accepted 5 May 2021)

Abstract: Parametric polyhedron plays an important role in parallel computing. Usually, they are represented by a system of parameterized linear inequalities, which is called the H-representation. A critical problem is to compute the V-representation, i.e. enumerating the vertices of a parametric polyhedron. Considering the absence of parametric polyhedral computation in MAPLE, we present a library named `ParametricVertices` to enumerate the vertices of a parametric polyhedron.

Keywords: Parametric polyhedron; H-representation; V-representation

1 Introduction

Let A be a matrix of size $m \times n$ over $\mathbb{Z}$ and $\mathbf{b}$ be a column vector of size m over $\mathbb{Z}$. The collection of solutions $A\mathbf{x} \leq \mathbf{b}$ encoding a polyhedron in $\mathbb{Q}^n$. Let $P = \{\mathbf{x} \,|A\mathbf{x} \leq \mathbf{b}\}$, then P is called a polyhedron in $\mathbb{Q}$ and the linear system $A\mathbf{x} \leq b$ is called the H-representation of P. Geometrically, the polyhedron P contains vertices $\mathbf{v}_1, \ldots, \mathbf{v}_t$, rays $\mathbf{r}_1, \ldots, \mathbf{r}_q$ and lines $\mathbf{l}_1, \ldots, \mathbf{l}_s$, then P can be represented by the set

$$\{\sum_{1\leq i\leq t} \alpha_i \mathbf{v}_i + \sum_{1\leq i\leq q} \beta_i \mathbf{r}_i + \sum_{1\leq i\leq s} \gamma_i \mathbf{l}_i \mid \alpha_1 + \cdots + \alpha_t = 1 \text{ and } \alpha_i, \beta_j \geq 0\}. \tag{1}$$

Equation (1) is called the V-representation of P. The conversion between H-representation and V-representation is a classical topic in computational geometry. Lots of algorithms have been proposed to solve this problem. The most famous one is the DDmethod (short for double description method) [1, 2]. A lot of research work claimed the DDmethod is equivalent to Fourier-Motzkin elimination [3], which is a famous variable elimination method for linear inequalities. It has been implemented in many libraries, like Polylib, PPL, CDD, as well as in the popular mathematical software MAPLE.

In the H-representation $\{A\mathbf{x} \leq \mathbf{b}\}$ of a polyhedron P, replacing the constant vector $\mathbf{b}$ with a linear system $B\boldsymbol{\theta} + \mathbf{c}$, where B, $\mathbf{c}$ and $\boldsymbol{\theta}$ are constant matrix, constant vector and parametric vector respectively, the new system $\{A\mathbf{x} \leq B\boldsymbol{\theta} + \mathbf{c}\}$ will represent a parametric polyhedron $P(\boldsymbol{\theta})$. In this case, enumerating the vertices is much more complicated than the problem for non-parametric polyhedron. For different parameters, a parametric polyhedron may have different vertices. To enumerate the vertices, we need to partition the parameter space first. In 2008, Leochner et al. proposed a method to enumerate the parametric vertices [4] . Since then, most libraries related to polyhedral computation have been using this method to compute the vertices of a parametric polyhedron. Moreover, their method have been used to count the integer points in a parametric polyhedron [5, 6]. Since parametric system with parametric polyhedron is an important tool in the field of parallel computing, program verification and so on, it is of great importance to study the efficient method of enumerating the vertices of a parametric polyhedron.

There are several choices to deal with the parametric systems in MAPLE, like the command `solve` which can solve the parametric systems, the sub-library `ParametricSystemTools` in `RegularChains` which can simplify the parametric polynomial systems. However, we did not find any implementations for enumerating the vertices of a parametric polyhedron. This motivates us to fill this gap.

Our main contributions are listed in the following:

*Corresponding author. *1531230204@qq.com*

1. We construct an object named `ParametricPolyhedron`, whose attributes consist of variables, parameters, constraints for variables and constraints for parameters. This object contains all the information required by a parametric polyhedron. It is helpful for the further parametric polyhedral computation.

2. We build a library named `ParametricVertices` in MAPLE. As far as we know, this is the first library enabling to compute the vertices of a parametric polyhedron. The vertices output by this library are objects, which we name as `ParametricVerticeswithDomain`. Each object `ParametricVerticeswithDomain` have two attributes: the vertices and their domain for parameters.

The basic knowledges are introduced in Section 2 and the core implementations are presented in Section 3. At the end, we provide the related work and our future work in Section 4.

2 Background

In this section, we review some basic concepts in polyhedral geometry related to enumerating the vertices of a parametric polyhedron. All through this paper, we use bold letters, e.g. $\mathbf{v}$ to denote column vectors and we use capital letters, e.g. A to denote matrices.

2.1 Polyhedron

A subset $P \subseteq \mathbb{Q}^n$ is called a *convex polyhedron* (or simply a *polyhedron*) if $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ holds, for a matrix $A \in \mathbb{Q}^{m\times n}$ and a vector $\mathbf{b} \in \mathbb{Q}^m$, where n, m are positive integers; we call the linear system $\{A\mathbf{x} \leq \mathbf{b}\}$ an *H-representation* of P. Hence, a polyhedron is the intersection of finitely many half-spaces. By Minkovski theorem, a polyhedron can be generated by the vertices, rays and lines contained in it with the form of Equation (1). We call the set in Equation (1) a *V-representation*.

If $\mathbf{c}$ is a non-zero vector, and $\delta = \max\{\mathbf{c}^t\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, the affine hyperplane $\{\mathbf{c}^t\mathbf{x} = \delta\}$ is called a *supporting hyperplane* of P. A subset F of P is called a *face* of P if $F = P$ or if F is the intersection of P with a set of supporting hyperplane of P.

Let P be a subset of $\mathbb{Q}^n$, the *dimension* $\dim(\mathrm{P})$ of P is $a-1$ where a is the maximum number of affinely independent points in P. We say P is *full-dimensional* if we have $\dim(\mathrm{P}) = \mathrm{n}$. Algebraically, P is full-dimensional if and only if there is no implicit equations in its H-representation. A face of dimension zero is called a vertex.

2.2 Parametric polyhedron and their vertices

A subset $P(\boldsymbol{\theta}) \subseteq \mathbb{Q}^n$ is called the *parametric polyhedron* if $P(\boldsymbol{\theta}) = \{\mathbf{x} \mid A\mathbf{x} \leq B\boldsymbol{\theta} + \mathbf{c}\}$, for matrices $A \in \mathbb{Q}^{m\times n}$, $B \in \mathbb{Q}^{m\times t}$ and a vector $\mathbf{c} \in \mathbb{Q}^m$. Here $\mathbf{x}$ and $\boldsymbol{\theta}$ are the column vectors of variables and parameters, respectively. For each specific value of the parameters $\boldsymbol{\theta}_0$, one can obtain a set of vertices of the non-parametric polyhedron $P(\boldsymbol{\theta}_0)$, say $\mathbf{v}_1(\boldsymbol{\theta}_0), \ldots, \mathbf{v}_r(\boldsymbol{\theta}_0)$. Each $\mathbf{v}_i$ is a linear combination of $\boldsymbol{\theta}_0$. When the parameters vary, the vertices may shift, split, merge et al. However, they still maintain to be the linear combinations of the parametric vector $\boldsymbol{\theta}$. For each vertex $\mathbf{v}(\boldsymbol{\theta})$ of the parametric polyhedron $P(\boldsymbol{\theta})$, it maintains to be a vertex when $\boldsymbol{\theta}$ varies in some specific domain $D(\boldsymbol{\theta})$, which is a linear constraints of the vector $\boldsymbol{\theta}$. In this sense, we can obtain a set of vertices of $P(\boldsymbol{\theta})$ in the following form:

$$\begin{cases}\mathbf{v}_1(\boldsymbol{\theta})\\ D_1(\boldsymbol{\theta})\end{cases} \quad \begin{cases}\mathbf{v}_2(\boldsymbol{\theta})\\ D_2(\boldsymbol{\theta})\end{cases} \quad \cdots \quad \begin{cases}\mathbf{v}_s(\boldsymbol{\theta})\\ D_s(\boldsymbol{\theta})\end{cases} \tag{2}$$

Note that the domains $D_1(\boldsymbol{\theta}), \ldots, D_s(\boldsymbol{\theta})$ in Equation (2) is a partition of the parameter space. However, they may have non-empty common spaces. For a specific parameter $\boldsymbol{\theta}_0$, the vertices of $P(\boldsymbol{\theta}_0)$ consist of all the vertices $\mathbf{v}_i(\boldsymbol{\theta}_0)$ which satisfy $\boldsymbol{\theta}_0 \in D(\boldsymbol{\theta})$.

In paper [4], the authors provide a way to partition the parameter space and give the vertices along with them. We introduce the brief idea here. For more details, please refer to paper [4]. Rewriting the parametric polyhedron $P(\boldsymbol{\theta})$ in a non-parametric form as $P' = \{(\mathbf{x}, \boldsymbol{\theta}) \mid A\mathbf{x} \leq B\boldsymbol{\theta} + \mathbf{c}\}$. We define two projection functions, Proj_t and Proj_n, as following:

- Proj_n projects the space $\mathbb{Q}^{n+t}$ onto the data space $\mathbb{Q}^n$;
- Proj_t projects the space $\mathbb{Q}^{n+t}$ onto the parameter space $\mathbb{Q}^t$.

Let $\hat{\boldsymbol{\theta}}$ be a vector in $\mathbb{Q}^t$, denote $S(\hat{\boldsymbol{\theta}}) = \left\{ \begin{pmatrix} \mathbf{x} \\ \hat{\boldsymbol{\theta}} \end{pmatrix} \mid \hat{\boldsymbol{\theta}} = \boldsymbol{\theta} \right\}$. Then $S(\hat{\boldsymbol{\theta}})$ is an affine subspace of $\mathbb{Q}^{n+t}$. By Theorem 1 of [4], the vertices of $P(\boldsymbol{\theta})$ correspond to projections of the t-faces of P'. This provides the main idea of the algorithm proposed in paper [4].

3 Main algorithms and their implementations

In this section, firstly we discuss the implementation of the parametric polyhedron. Then, we show how to enumerate the vertices of a parametric polyhedron. The algorithms we rely on are provided in paper [4].

3.1 Data-Types

The data-type `ParametricPolyhedron` is implemented, encoding a parametric polyhedron. It is implemented in an "object-oriented" fashion using the MAPLE language construct of a module, which offers "get" methods to access the different attributes of a parametric polyhedron, see Figure 1.

```
> vars := [i, j];
  paras := [p1, p2];
  consts := [j >= 0, j <= p1, j <=i, i<=p2];
  pp := ParametricPolyhedron(consts, vars, paras, []);
                        vars := [i, j]
                       paras := [p1, p2]
              consts := [0 ≤ j, j ≤ p1, j ≤ i, i ≤ p2]
```

$$pp := \begin{cases} \text{Relations} : \begin{cases} 0 \le j \\ j \le p1 \\ j \le i \\ i \le p2 \end{cases} \\ \text{Variables} : [i, j] \\ \text{Parameters} : [p1, p2] \\ \text{ParameterConstraints} : \{ \end{cases}$$

```
> GetConstraints(pp);
                  [0 ≤ j, j ≤ p1, j ≤ i, i ≤ p2]
> GetVariables(pp);
                        [i, j]
> GetParameters(pp);
                       [p1, p2]
> GetParametersConstraints(pp);
                         [ ]
```

Figure 1: Parametric Polyhedron

3.2 Solvers

We illustrate our implementation by one example from paper [4]. One can refer to the paper [4] for the detailed algorithm. Here we only show the implementation in MAPLE. The command `ParametricVertices` takes as input a type of parametric polyhedron and outputs the parametric vertices along with their parameter spaces. The parametric vertices and their domain are implemented in an object `ParametricVerticeswithDomain`. We can use "get" methods to access the vertices and their parametric domains in a `ParametricVerticesandDomain`object. The output with our implementation is shown in Figure 2 while the result from Paper [4] can be found in Table 1.

Example 1 *Given a parametric polyhedron* $pp := \{(i,j) \mid j \ge 0, j \le p1, j \le i, i \le p2\}$*, we have the following parametric vertices:*

Vertices	$(p2, p2)$	$(p2, p1)$	$(p1, p1)$	$(p2, 0)$	$(0, 0)$
Domain	$\begin{cases} -p2 \le 0, \\ -p1 + p2 \le 0 \end{cases}$	$\begin{cases} -p1 \le 0, \\ p1 - p2 \le 0 \end{cases}$	$\begin{cases} -p1 \le 0, \\ p1 - p2 \le 0 \end{cases}$	$\begin{cases} -p2 \le 0, \\ -p1 \le 0 \end{cases}$	$\begin{cases} -p2 \le 0, \\ -p1 \le 0 \end{cases}$

Table 1: Parametric Vertices

```
> with(ParametricVertices);
        [ParameterizeVertices, ParametricPolyhedron, ParametricVerticeswithDomain]
> pvd := ParameterizeVertices(pp);
pvd := [{Vertices : [p2, p2]                  , Domain : [-p2 ≤ 0, -p1 + p2 ≤ 0]},
        {Vertices : [p2, p1], [p1, p1]        , Domain : [-p1 ≤ 0, p1 - p2 ≤ 0]},
        {Vertices : [p2, 0], [0, 0]           , Domain : [-p2 ≤ 0, -p1 ≤ 0]}]
> map(GetVertices, pvd);map(GetDomain, pvd);
        [[[p2, p2]], [[p2, p1], [p1, p1]], [[p2, 0], [0, 0]]]
        [[-p2 ≤ 0, -p1 + p2 ≤ 0], [-p1 ≤ 0, p1 - p2 ≤ 0], [-p2 ≤ 0, -p1 ≤ 0]]
```

Figure 2: Parametric Vertices

```
> consts := GetConstraints(pp);
                        consts := [0 ≤ j, j ≤ p1, j ≤ i, i ≤ p2]
> P := PolyhedralSets:-PolyhedralSet(eval(consts, [p1 = 2, p2 = 1]));
                        P := {Coordinates : [i, j]
                              Relations   : [-j ≤ 0, -i + j ≤ 0, i ≤ 1]
> PolyhedralSets:-VerticesAndRays(P);
                        [[1, 0], [1, 1], [0, 0]], [ ]
```

Figure 3: Vertices and Rays

We collect the vertices with the same domain to one object of `ParametricVerticeswithDomain`*, as is shown in Figure 2.*

Note that the domains for different branches in the output of `ParametricVertices` cover the whole parameter space. But they may have intersections, as is shown in Figure 4. Once the parameters are specified, we collect the all the vertices whose domains contain the specific parameters. The vertices when we specify the parameters to be $p1 = 2, p2 = 1$ is shown in Figure 3. They are exactly the collections of the first and the third branch in Figure 2.

4 Related work and future work

In this paper, we provide a library `ParametricVertices` to enumerate the vertices of a parametric polyhedron. It is closely related to the research work on polyhedral model. We list some of them in the following:

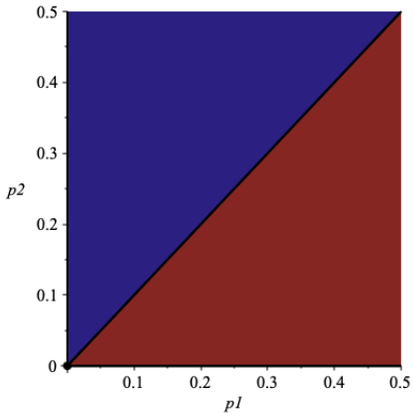

Figure 4: Doamins

1. *Counting and enumerating integer points in a parametric polyhedron* [6–8].

   In program analyses and optimizations, counting the integer points of a parametric polyhedron has important applications. It can decide how many memory locations are touched, how many cache misses do a loop generate, how many cache lines are touched by a loop et al. Barvinok proposed a method for counting the integer points of a non-parametric polyhedron in [9]. His method relies on enumerating the vertices of a polyhedron. Due to the complicate situtions for parameters, this method cannot be applied to a parametric polyhedron easily. Based on the method for enumerating the vertices in paper [4], Clauss et al. proposed a method for counting the integer points in a parametric polyhedron [6]. This method has been widely used in libraries related to polyhedral computation, like Polylib, Integer Set Library.

2. *Parametric linear programming and parametric integer programming* [10–12].

   In a standard linear programming, the optimal solution is taken on some vertices of the polyhedron represented by the constraints. However, it is not a wise idea to enumerate all the vertices to solve a linear programming. In paper [11], the author mainly relied on the projection of a polyhedron to solve the parametric linear programming problem. Similar to the idea in [4], they gave a partition of the parameter space and obtained an optimal value with a uniform expression in each parametric subspace.

3. *Fourier-Motzkin elimination* [2, 13–16].

   As is shown in [2], enumerating the vertices of a full-dimensional polyhedron is equivalent to computing the projections of a polyhedron. Fourier-Motzkin elimination is a well-known method for computing the projections. During the process of Fourier-Motzkin elimination, large amount of redundant inequalities can be generated, which lead to the low-efficiency of the implementation. Lots of work has been working on removing the redundant inequalities [16–20]. Enumerating the vertices of a polyhedron, as well as a parametric polyhedron, faces the same problem. In our implementation, we may need to compute lots of linear equation systems in order to find a true vertex. Acctually, most of the linear systems will not return to a vertex. How to detect those non-necessary systems with a low cost is our next aim. This is hopeful to improve the efficiency of our present implementation.

Our next work is to improve the efficiency of our library. Moreover, we will study furthermore how to obtain the independent parametric domains for different set of vertices. Following this, we can study the parametric linear and integer programming later on.

Acknowledgments

This research was funded by the Jiangsu University Student Innovation and Entrepreneurship Training Program(NO.202010299515X).

References

[1] T. S. Motzkin. The double description method, in contributions to the theory of games ii. *Annals of Mathematics Study*, 28(1953).

[2] K. Fukuda and A. Prodon. Double description method revisited. In Combinatorics and computer science, 91–111. Springer. 1996.

[3] T. S. Motzkin. Beiträge zur Theorie der linearen Ungleichungen. Azriel Press. 1936.

[4] V. Loechner and D. K. Wilde. Parameterized polyhedra and their vertices. *International Journal of Parallel Programming*, 25(1997)(6):525–549.

[5] P. Clauss. Counting solutions to linear and nonlinear constraints through ehrhart polynomials: applications to analyze and transform scientific programs. In Proceedings of the 10th international conference on Supercomputing, ICS '96. 1996.

[6] S. Verdoolaege et al. Counting integer points in parametric polytopes using barvinok's rational functions. *Algorithmica*, 48(2007)(1):37–66.

[7] B. Meister and P. Clauss. Uniform random sampling in polyhedra. In IMPACT 2020, 10th International Workshop on Polyhedral Compilation Techniques. 2020.

[8] M. Köppe and S. Verdoolaege. Computing parametric rational generating functions with a primal barvinok algorithm. *Electronic Journal of Combinatorics*, 15(2007)(1):229–247.

[9] A. I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Math. Oper. Res.*, 19(1994)(4):769–779.

[10] P. Feautrier. Parametric integer programming. *RAIRO Recherche Opérationnelle*, 22(1988).

[11] C. N. Jones, E. C. Kerrigan and J. M. Maciejowski. On polyhedral projection and parametric programming. *Journal of Optimization Theory and Applications*, 138(2008)(2):207–220.

[12] F. Borrelli, A. Bemporad and M. Morari. Geometric algorithm for multiparametric linear programming. *Journal of optimization theory and applications*, 118(2003)(3):515–540.

[13] G. B. Dantzig and B. C. Eaves. Fourier-motzkin elimination and its dual. *J. Comb. Theory, Ser. A*, 14(1973)(3):288–297.

[14] J.-L. Imbert. Fourier's elimination: Which to choose? In PPCP, 117–129. 1993.

[15] M. Le Fur. Scanning parameterized polyhedron using fourier-motzkin elimination. *Concurrency - Practice and Experience*, 8(1996)(6):445–460.

[16] R. Jing, M. M. Maza and D. Talaashrafi. Complexity estimates for fourier-motzkin elimination. In Computer Algebra in Scientific Computing - 22nd International Workshop, CASC 2020, Linz, Austria, September 14-18, 2020, Proceedings, vol. 12291 of *Lecture Notes in Computer Science*, 282–306. Springer. 2020.

[17] C. Chen and M. M. Maza. Solving parametric polynomial systems by realcomprehensivetriangularize. In International Congress on Mathematical Software. 2014.

[18] S. I. Bastrakov, A. V. Churkin and N. Y. Zolotykh. Accelerating fourier–motzkin elimination using bit pattern trees. *Optimization Methods and Software*, (2020):1–14.

[19] T. Huynh, C. Lassez and J. L. Lassez. Practical issues on the projection of polyhedral sets. *Annals of Mathematics & Artificial Intelligence*, 6(1992)(4):295–315.

[20] A. Maréchal and M. Périn. Efficient elimination of redundancies in polyhedra using raytracing. *Springer International Publishing*, (2017).