qskeleton

User's Guide

Version 1.00

Sergey Bastrakov

bastrakov@vmk.unn.ru

Lobachevsky State University of Nizhni Novgorod, Russia

April 8, 2015

Contents

1 Introduction

1.1 Theoretical preliminaries

Each convex polyhedron in $\mathbf{R}^d$ can be represented in the following two ways:

1. *Facet representation* as a set of solutions to a system of linear inequalities:

$$P = \left\{ x \in \mathbf{R}^d : Ax \geq b \right\} . \tag{1}$$

2. *Vertex representation* as the Minkowski's sum of the convex hull of some points and conical hull of some vectors in $\mathbf{R}^d$:

$$P = \operatorname{conv}(v_1, v_2, \ldots, v_n) + \operatorname{cone}(u_1, u_2, \ldots, u_m) . \tag{2}$$

The problem to compute representation (2) for given representation (1) is called the *vertex enumeration problem*. The converse one is called the *facet enumeration problem*, or in case of polytopes (bounded polyhedra) — *convex hull problem*.

Analogously, each polyhedral cone in $\mathbf{R}^d$ can be represented in the following two ways:

1. *Facet representation* as a set of solutions to a homogeneous system of linear inequalities:

$$C = \left\{ x \in \mathbf{R}^d : Ax \geq 0 \right\} .$$

2. *Vertex representation* as a set of all non-negative linear combinations of some vectors in $\mathbf{R}^d$:

$$C = \operatorname{cone}(u_1, u_2, \ldots, u_m) .$$

There is a standard way to reduce vertex/facet enumeration problem for polyhedra to the corresponding problem for polyhedral cones. The vertex and facet enumeration problems are dual to one another, thus the same algorithm can be used to solve both problems.

A closely related operation is *eliminating variables* from systems of linear inequalities. The result of eliminating variables $\{x_1, x_2, \ldots x_k\}$ from a system of linear inequalities is another system of linear inequalities without the variables from X, such that the solution sets of both systems over the remaining variables are the same.

1.2 What is qskeleton?

qskeleton is polyhedral computation software for solving the vertex and facet enumeration problems for convex polyhedra and polyhedral cones, and eliminating variables from systems of linear inequalities. It implements the double description method (DDM) and Fourier-Motzkin elimination (FME) with Chernikov rules. The input data can be integer or real numbers.

qskeleton is capable of solving the following problems:

- computing vertex representation of a polyhedron (or polyhedral cone), given the facet representation;
- computing facet representation of a polyhedron (or polyhedral cone), given the vertex representation; in particular, if a polyhedron is bounded, this is convex hull problem;
- eliminating variables from a system of linear inequalities, geometrically, projecting a polyhedron onto a linear subspace.

The following are also polyhedral computing problems, but *qskeleton* is not capable of solving those:

- computing a face lattice of a polyhedron;
- computing a triangulation of a polyhedron;
- finding integer points of a polyhedron.

qskeleton is publicly available sofrware distributed under the GNU GPLv2.

2 Building qskeleton

Download the source code of *qskeleton* from GitHub `https://github.com/sbastrakov/qskeleton` as an archive or clone the repository using git.

The build system is based on CMake that can be obtained from `http://www.cmake.org/download/`. To build *qskeleton* go to `build` directory and run `build_linux.sh` or `build_windows.bat` depending on your operating system. Running `build_linux.sh` generates a Makefile in a newly created subdirectory `build/unix_makefiles`. Running `build_windows.bat` generates a Visual Studio solution in a newly created subdirectory `build/visual_studio`.

The default compilers are g++ on Linux and Visual Studio 2010 on Windows. In case you use another compiler, modify the `-G and/or -D` parameters in the corresponding script, refer to CMake documentation for generator names. Note that is case Visual Studio is used the default configuration of the generated solution is Debug, while it is strongly recommended to build in Release configuration for performance considerations.

In case for some reason CMake can not be used, it should be rather easy to build *qskeleton* manually: the code does not use any non-standard language features and the only external dependence is tclap header-only template library located in `deps` directory.

3 Facet and vertex enumeration using qskeleton

qskeleton directly solves ray enumeration problem which is a vertex enumeration problem for polyhedral cones as described at subsection 3.1. Facet enumeration

for polyhedral cones as well as facet and vertex enumeration problems for geenral polyhedra can be easily reduced to ray enumeration, as described at the following subsections.

3.1 Ray enumeration for polyhedral cones

The ray enumeration problem for polyhedral cones is given a matrix $A \in \mathbf{R}^{n\times d}$ find a set of extreme rays $\{u_1, u_2, \dots u_m\}$ such that $\{x : Ax \geq 0\} = \text{cone}\,(u_1, u_2, \dots, u_n)$.

To use *qskeleton* one should create an input file that contains the matrix A. Input file is a text file with the first line containing two integer numbers — the number of rows and columns of the matrix, and the rest of the file containing the elements of the matrix in row-major order. Numbers are separated by spaces and blank lines.

For example, if you want to enumerate the extreme rays of the cone C defined as a set of solution to the system

$$\begin{cases} x_1 + x_3 \geq 0, \\ -x_1 + x_3 \geq 0, \\ x_2 + x_3 \geq 0, \\ -x_2 + x_3 \geq 0. \end{cases}$$

The corresponding input file is

```
4 3
1 0 1
-1 0 1
0 1 1
0 -1 1
```

To run *qskeleton* just type in the command prompt:

```
ddm filename
```

where `ddm` is a path to the ddm executable (by default `bin/ddm`) and `filename` is a path to the input file. The input file for the above-mentioned example is in `examples/example.ine` subdirectory, so ddm can be invoked from the *qskeleton* root directory with the input file `examples/example.ine` and output file `examples/example.ext` as

```
bin/ddm examples/example.ine -o examples/example.ext
```

(replace / with \ on Windows).

In case the output file name is not provided, the output will be done to stdout. The output format is the same as input: the first line contains number of extreme rays and dimensionality and the rest of the output is ray matrix in the row-major order, each row corresponds to an extreme ray.

In our example the output (`examples/example.ext`) is 4 extreme rays:

```
4 3
1 -1 1
-1 -1 1
1 1 1
-1 1 1
```

The set of extreme rays of the original cone is

$$\operatorname{cone}\left((1,-1,1)^\top,(-1,-1,1)^\top,(1,1,1)^\top,(-1,1,1)^\top\right).$$

3.2 Facet enumeration for polyhedral cones

Facet enumeration problem for polyhedral cones is the inverse to ray enumeration. These two problems are mutually dual, thus for facet enumeration one just needs to write rays as rows of a matrix, perform ray enumeration for this matrix and interpret rows of the resulting matrix as coefficients of inequalities.

To find the facet representation of the example cone from the previous subsection run

```
bin/ddm examples/example.ext
```

The result is

```
4 3
1 0 1
-1 0 1
0 1 1
0 -1 1
```

which is exactly (up to reordering) the inequalities of the original cone in `examples/example.ine`.

3.3 Vertex enumeration for convex polyhedra

The vertex enumeration problem for convex polyhedra can be reduced to ray enumeration of polyhedral code using the standard homogenization procedure.

Consider a polyhedron $P = \left\{x \in \mathbf{R}^d : Ax \geq b\right\}$. Introduce a new variable x_{d+1} and consider the cone $C = \left\{x \in \mathbf{R}^{d+1} : Ax \geq bx_{d+1}, x_{d+1} \geq 0\right\}$. Obviously, $P = C \cap \left\{x \in \mathbf{R}^{d+1} : x_{d+1} = 1\right\}$. Note that in case interior of P contains 0 the inequality $x_{d+1} \geq 0$ is redundant in C and can be omitted. Now construct extreme rays $U = \left\{u^{(1)}, u^{(2)}, \ldots, u^{(m)}\right\} \subset \mathbf{R}^{d+1}$ of C. Each extreme ray of C corresponds to either a vertice or extreme ray of P. If the $(d+1)$-st component of extreme ray $u \in U$ is not zero ($u_{d+1} \neq 0$), it corresponds to a vertice $v = (u_1/u_{d+1}, u_2/u_{d+1}, \ldots, u_d/u_{d+1})$ of the polyhedron P. If the $(d+1)$-st component of extreme ray u is zero, it corresponds to an extreme ray $v = (u_1, u_2, \ldots, u_d)$ of the polyhedron.

For example consider finding the vertex representation of a 3-dimensional cube $-1 \leq x_i \leq 1.i = 1, 2, 3$. After homogenization the system in form $Ax \geq 0$

is

$$\begin{cases} x_1 + x_4 \geq 0, \\ -x_1 + x_4 \geq 0, \\ x_2 + x_4 \geq 0, \\ -x_2 + x_4 \geq 0, \\ x_3 + x_4 \geq 0, \\ -x_3 + x_4 \geq 0, \end{cases}$$

($x_4 \geq 0$ is redundant and thus ommited).

The corresponding input file (`examples/cube3.ine`) is

```
6 4
1 0 0 1
-1 0 0 1
0 1 0 1
0 -1 0 1
0 0 1 1
0 0 -1 1
```

and the output (`examples/cube3.ext`) is

```
8 4
-1 -1 -1 1
-1 1 -1 1
1 -1 -1 1
1 1 -1 1
-1 -1 1 1
-1 1 1 1
1 1 1 1
1 -1 1 1
```

There are 8 extreme rays of the corresponding code, the last component of each ray is 1, thus there are 8 vertices of the cube:

$$v_1 = (-1,-1,-1)^\top, v_2 = (-1,-1,1)^\top,$$
$$v_3 = (-1,1,-1)^\top, v_4 = (-1,1,1)^\top,$$
$$v_5 = (1,-1,-1)^\top, v_6 = (1,-1,1)^\top,$$
$$v_7 = (1,1,-1)^\top, v_8 = (1,1,1)^\top.$$

3.4 Facet enumeration for convex polyhedra

Facet enumeration problem for convex polyhedra is dual to vertex enumeration. However, unlike the cones, one should first make sure 0 is a point of a polyhedron which can be obtained by coordinate shift.

Increase dimension by 1 augmenting each vertex with 1 and extreme ray with 0 and submit those as input matrix. The resulting matrix should be interpreted as $(A| - b)$ and the system in inequalities is $Ax \geq b$.

For example, to find the inequalities of the cube from the previous subsection, the input file is

```
8 4
-1 -1 -1 1
-1 1 -1 1
1 -1 -1 1
1 1 -1 1
-1 -1 1 1
-1 1 1 1
1 1 1 1
1 -1 1 1
```

and the output file is

```
6 4
0 0 1 1
0 1 0 1
1 0 0 1
0 0 -1 1
0 -1 0 1
-1 0 0 1
```

3.5 Options

For the list of options run `ddm` with `-h` option.

4 Fourier-Motzkin elimination using qskeleton

4.1 Fourier-Motzkin elimination for homogeneous systems

To perform Fourier-Motzkin elimination for a system of linear inequalities use binary `bin/fme`.

The input is a matrix of a system $Ax \geq 0$. The input file format is similar to the input for `ddm`: the first line contains the number of rows and columns of the matrix, followed by elements of the matrix in row-major order.

A set of eliminated variables are defined by a file provided by `-e` argument; by defailt all variables are eliminated. Elimination file consists of the number of eliminated varialbes followed by indexes of variables, indexes are 0-based (i.e. the index of the first variable is 0).

For example consider elimination of variables x_0 and x_1 from the system

$$\begin{cases} x_0 + x_2 \geq 0, \\ -x_0 + x_2 \geq 0, \\ x_1 + x_2 \geq 0, \\ -x_1 + x_2 \geq 0. \end{cases}$$

The input file is `examples/example.ine`:

```
4 3
1 0 1
-1 0 1
```

0 1 1
0 -1 1

Elimination file is `examples/example.elim`

2
0 1

The result is

2 3
0 0 1
0 0 1

Note that unlike facet and vertex enumeration, the result of elimination can contain redundant inequalities.

4.2 Fourier-Motzkin elimination for non-homogeneous systems

In case elimination is applied to a system $Ax \geq b$, reduce it the the homogenious case by using matrix $(A| - b)$.

4.3 Options

For the list of options run `fme` with `-h` option.