Veronika Steffanová

User documentation for MATLAB library for polyhedra

Prague 2012

1. Introduction

Thank you that you decided to try our MATLAB library for polyhedra. It contains several functions to operate with polyhedra, e.g. conversions between vertex and facet descriptions in both directions or computations of convex union and intersection of two polyhedra and removing redundant vertices or facets from vertex or facet description.

This library is fully compatible with freeware Octave too, but before usage change the installed packages according to the instructions from the last chapter.

There are used strict rules for the input parameters of the functions. If they are not filled in correctly, the results would be empty values and a message would be received by console output.

The library belongs to Charles University in Prague, so it cannot be used for commercial purposes.

2. Input and output data structures

There are three data structures used in the library which can be used as input or received as output.

1. **Facets:** are represents as a list of inequalities or equalities. The list is composed of two parts: A stands for a matrix of the left side, and b stands for the right side. One instance of the structure is used either for equalities or inequalities, never for both of them. Type is always implied for the context. For the creation of an instance you can use the constructor or type it manually.

   Example:

   $$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \leq \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

   ```
   Nerovnice.A = [1 2;3 4];
   Nerovnice.b = [5;6]
   %or
   Nerovnice = inequalities([1 2;3 4],[5;6])
   ```

2. **Vertices:** are simply stored in a list, one vertex on each rows.

   Example: $\triangle ABC, A[0;0], B[0;1], C[1;0]$

   ```
   ABC = [0 0; 0 1; 1 0];
   ```

3. **Double description:** has a little more complicated structure, but it allows you to find the relations between vertices and inequalities in constant time. Let have a polyhedron P. Then P.c returns a list of coordinates of all vertices, P(i).c returns coordinates of ith vertex. P(i).ineq stands for inequalities of all facets, where the ith vertex lies in. P(i).ineq(j) gives you information about jth facet. There are two fields: inequal, which contains a inequality of the facet $a_1 + a_2 + \ldots + a_d \leq b$ like an array $[a_1, a_2, \ldots, a_d, b]$, and vert, which contains a list of indices of all vertices which lies in jth facet.

   Example: $\triangle ABC, A[0;0], B[0;1], C[1;0], AB : -y \leq 0, AC : -x \leq 0, BC : x + y \leq 1$

   ```
   ABC.c
     ans = 0 0
     ans = 0 1
     ans = 1 0

   ABC(1).c
     ans = 0 0
   ```

```
ABC(1).ineq(1)
  inequal = 0 -1 0
  vert = 1 2

ABC(1).ineq(2)
  inequal = -1 0 0
  vert = 1 3

%return all inequalities
Nerovnice = extractineq(ABC)

%return all vertices in matrix
Vrcholy = extractvertices(ABC)
```

3. From facet to vertex description

The conversion from facets to vertices is implemented in the function

```
[Vertices, Edges] = ineqtovertices(Inequalities,Equalities,'S')
```

where the third parameter is optional.

Input: The first parameter Inequlities contains a list of all inequalities of facets in the format (1). The second parameter Equalities contains a list all equalities in the same format. In the area of linear programming there are often used both equalities and inequalities for specification of the polyhedron boundaries, therefore there is an optional usage of equalities as an input. If you do not want to use both inequalities and equalities, you can replace any of the parameters by []. The third parameter is optional and can have only the value 'S', which says you want as a result a polyhedron in double description.

Output: The first parameter Vertices contains list of all vertices of the input polyhedron according to (2) or the polyhedron in double description if you used the third input parameter. The second parameter Edges contains list of vectors which defines unbounded edges of the polyhedron. If there are no unbounded edges, it returns an empty array. If there are any unbounded edges, it means the polyhedron unbounded and this information is always displayed in console.

As an input you can use combination of equalities and inequalities of dimension one and higher. If the polyhedron has no vertex, the Vertices stays empty. It is possible that there are returned some unbounded edges, but without a vertex it is difficult to say if there are really edges and if the algorithm found all of them. But you can be sure that the rays specified by the vector lies in the polyhedron.

4. From vertex to facet description

The conversion from vertex to facet description is implemented by two functions

```
Polyhedron = verticestoineqcomb(Vertices, 'S')
Polyhedron = verticestoineqdual(Vertices, 'S')
```

where the first one uses incremental algorithm and the second one uses dual graph-traversal algorithm. But the input and output parameters are same. we recommend to use the first one, because usually it is faster. But if you have much vertices (more than 50 or 100), but low dimension (2 or 3), the second one can be faster.

Input: The first parameter Vertices contains list of all vertices in the format (2). The second parameter is optional and can have only the value 'S', which says you want as a result a polyhedron in double description.

Output: The first parameter Polyhedron contains a list of all facet in the form of inequalities according to (1) of the input polyhedron or the polyhedron in double description, if you used the second input parameter.

The input vertices can be of one or higher dimension and there has to exist their affine independent subset of $d + 1$ vertices, where d is the dimension.

5. Convex union and intersection

Functions for convex union and intersection are quite similar, so we describe them in one chapter.

```
[Polyhedron, Edges] = union(Polyedr1, Polyedr2, output)
[Polyhedron, Edges] = intersection(Polyedr1, Polyedr2, output)
```

Input: Both function have three input parameters. The first two parameters stands for the polyhedra which should be unite or intersect. They can be in any format what was specified in the second chapter. They must only have the same dimension. The third parameter specifies the format of output. It can has three values: 'V' – out will be in vertex description, 'F' – output will be in facet description, 'S' – output will be in double description.

Output: The first parameter stands for the polyhedron in the format according to the third input parameter. The second parameter is a list of all unbounded edges. Unbounded edges have a sense only for union if at least one input polyhedron is in the facet description, or for intersection if both polyhedra are in the facet description (in all other cases the output polyhedron is always bounded, so it cannot have unbounded edges).

6. Remove redundancies

There are two function for removing redundancies from the vertex or facet representation.

```
Inequalities = rmineq(Inequalities)
Vertices = rmivertices(Vertices)
```

Both take as an input the polyhedron in the chosen description, and return a polyhedron where all vertices are extremal (convex hull without any vertex is different from the convex hull of all vertices), or each inequality defines a facet.

7. Installing instructions for Octave

If you want to use our library for MATLAB in Octave, at first take the file linprog.m, what is attached to the library, and replace the original file with the same name in the installed package *optim*. The path to the file looks like that:

(Octave/gcc-*version*/share/octave/packeges/optim-*version*)

The new file can do everything what the old file, but there some repaired bugs and also add the possibility to study the third parameter of the original linprog from MATLAB, which says if the result or the linear program is defined (Octave code 180, MATLAB code 1) or if it is not (all other codes, we set it on 0).