# Intro. to Polymake

- Written by Ewgenij Gawrilow, Michael Joswig, & various contributors.
- Website : www.polymake.org
- Can be run from the website as an application (limited capabilities), or installed and run in a Linux environment. (Instructions for installation on the website)

When the program starts, enter the following line to begin constructing a polytope:

**polytope> $p = new Polytope<Rational>();**

From this point, you can either enter the vertex description (by providing a list of points to take the convex hull of), or the halfspace description (by providing a list of linear inequalities). For this example, suppose we want to enter a 3-dimensional cube:

| **Vertex Description** | **Hyperplane Description** |
|---|---|
| Note: All points must be preceded by a 1 in the first coordinate (The leading 1 coordinate means homogenization) | Note: Every inequality must be entered in such a way that 0 is less than or equal to the dot product of the entered row with the row $(1, x_1, x_2, x_3, \ldots)$. |
| Command:<br>**polytope> $p->POINTS=<<".";** | Command:<br>**polytope> $p->INEQUALITIES=<<".";** |
| Sample Input:<br>polytope (2)> 1 0 0 0<br>polytope (3)> 1 1 0 0<br>polytope (4)> 1 0 1 0<br>polytope (5)> 1 1 1 0<br>polytope (6)> 1 0 0 1<br>polytope (7)> 1 1 0 1<br>polytope (8)> 1 0 1 1<br>polytope (9)> 1 1 1 1<br>polytope (10)> . | Sample Input:<br>polytope (2)> 1 -1 0 0<br>polytope (3)> 0 0 0 1<br>polytope (4)> 0 1 0 0<br>polytope (5)> 0 0 1 0<br>polytope (6)> 1 0 0 -1<br>polytope (7)> 1 0 -1 0<br>polytope (8)> . |

Note: You can enter in vertex (or hyperplane) information in one line. This method is more helpful when correcting errors. The following command shows how to enter the vertices of the cube in one-line notation (the method for hyperplanes is similar):

**polytope > $p->POINTS=[[1,0,0,0],[1,1,0,0],[1,0,1,0],[1,0,0,1], [1, 1, 1, 0], [1, 0, 1, 1], [1, 1, 0, 1], [1, 1, 1, 1]];**

If you are not running polymake online, you have the option of saving and loading your polytopes. The following commands illustrate how this is done:

**polytope> save($p, "filename");**
**polytope> $p = load("filename.poly");**

Once the polytope is constructed, you can have polymake return various pieces of information about the polytope. If the query has a yes/no answer, like "Is the polytope simplicial?", the program returns a "0" for no and a "1" for yes. The following table lists some useful commands as well as the information that they return. Polymake has far more commands than this, so check the website to see what's available.

| Command | Result |
|---|---|
| **polytope> print $p->VERTICES;** | prints the vertices of $p |
| **polytope> print $p->FACETS;** | prints the facet defining hyperplanes of $p |
| **polytope> print $p->N_VERTICES;** | prints the number of vertices of $p |
| **polytope> print $p->N_FACETS;** | prints the number of facets of $p |
| **polytope> print $p->FACET_SIZES;** | prints the number of vertices in each of the facets of $p |
| **polytope> print $p->F_VECTOR;** | prints the F-vector of $p |
| **polytope> print $p->DIM;** | prints the dimension of $p |
| **polytope> print $p->BOUNDED;** | tells you if $p is bounded |
| **polytope> print $p->SIMPLE;** | tells you if $p is simple |
| **polytope> print $p->SIMPLICIAL;** | tells you if $p is simplicial |
| **polytope> print $p->NEIGHBORLY;** | tells you if $p is neighborly |
| **polytope> print $p->CUBICAL;** | tells you if $p is cubical |
| **polytope> print $p->GRAPH->ADJACENCY;** | returns the vertex-edge adjacency graph of $p by listing the vertices adjacent to each vertex of $p |
| **polytope> print $p->GRAPH->EDGES;** | returns the edges of the vertex-edge adjacency graph of $p |

Polymake also offers several options to visualize various aspects of your polytope. These options are not available if you run Polymake from the website.

| Command | Result |
|---|---|
| **polytope> $p->VISUAL;** | renders a visualization of $p ($\leq 3$ dim) which can be exported as a postscript file |
| **polytope> $p->VISUAL(FacetStyle =>'hidden');** | renders a wireframe visualization of $p |
| **polytope> $p->VISUAL(FacetColor=> ['blue','yellow','blue','yellow','blue','yellow']);** | many characteristics of the visualization can be specified, like color, labels, vertex size, transparency, etc. |
| **polytope> $p->SCHLEGEL;** | renders the Schlegel diagram of $p (if dim $\leq 3$), if dim = 4, just use the VISUAL command above |
| **polytope> $p->SCHLEGEL->CONSTRUCTION;** | shows how the Schlegel diagram is constructed for 3-dim polytopes |
| **polytope> $p->GALE;** | produces a visual of the Gale diagram of $p |
| **polytope> $p->HASSE_DIAGRAM->VISUAL;** | produces a visual of the face lattice of $p |
| **polytope> $p->GRAPH->VISUAL;** | produces a visual of the vertex-edge adjacency graph of $p |

Polymake also offers many constructions which build new polytopes out of old polytopes. Here are a few options

| Command | Result |
|---|---|
| **polytope> $q=polarize($p);** | constructs the polar of $p and calls it $q |
| **polytope> prefer 'jreality'**<br>**polytope> $q = truncation($p,All);** | constructs a polytope by truncating all vertices of $p |
| **polytope> compose($p->VISUAL (FacetStyle=> "hidden",VertexStyle=> "hidden"),$q->VISUAL);** | shows visualizations of $p and $q from above, to see the effects of truncation |
| **polytope> $q=transform($p, matrix, store)** | applies a linear transformation to $p given by 'matrix', 'store' is a Boolean determining whether to store the reverse transformation as an attachment |

More common constructions:

**polytope>$r = simplex(n), cross(n), cube(n), conv($p, $q), pyramid($p), prism($p), intersection($p, $q)**

To exit polymake, enter the command **polytope> exit();**