03_reticulas_polaridad

August 4, 2021

1 Uso de polymake: Retículas de caras y polaridad

En este cuaderno describiremos algunas de las funcionalidades que presenta polymake para estudiar las retículas de caras de polítopos, y para construir polítopos polares.

Definamos a `$p` como una pirámide de dimensión 3 sobre un poliedro:

[ ]:
```
$p = bipyramid(cube(3));
$p->SCHLEGEL;
```

Podemos consultar la dimensión de `$p`, listar sus vértices, sus facetas y la incidencia de vértices en facetas:

[ ]:
```
print "Dimensión: ";
print $p->DIM;
print "\n\nVértices:\n";
print $p->VERTICES;
print "\nFacetas:\n";
print_constraints($p->FACETS);
print "Incidencia vértices-facetas:\n";
print ($p->VERTICES_IN_FACETS);
```

La propiedad `HASSE_DIAGRAM` devuelve un objeto con información sobre la retícula del polítopo. El diagrama de Hasse de la retícula puede dibujarse accediendo al método `VISUAL` de este objeto:

[ ]:
```
$p->HASSE_DIAGRAM->VISUAL;
```

La propiedad `FACES` del objeto de la retícula nos devuelve una lista de todas las caras del polítopo, ordenadas por su dimensión y representadas por los conjuntos de sus vértices:

[ ]:
```
print $p->HASSE_DIAGRAM->FACES;
```

Utilizando subíndices podemos acceder a cada una de las caras de la lista. Por ejemplo, para acceder a la tercera cara empleamos la expresión:

[ ]:
```
print $p->HASSE_DIAGRAM->FACES->[2];
```

A menudo, queremos consultar únicamente las caras de una determinada dimensión. Para ello podemos usar el método `nodes_of_dim`, que nos devuelve los índices de las caras dentro del arreglo `FACES` que tienen una dimensión especificada:

[ ]:
```
print $p->HASSE_DIAGRAM->nodes_of_dim(2);
print "\n---\n";
print map { $p->HASSE_DIAGRAM->FACES->[$_] }␣
 ↪@{$p->HASSE_DIAGRAM->nodes_of_dim(2)};
```

Si solamente nos interesa la cantidad de nodos de una dimensión, podemos usar el método `size`:

[ ]:
```
print $p->HASSE_DIAGRAM->nodes_of_dim(1)->size;
```

El siguiente código muestra la cantidad de caras de dimensión k, para $k \in \{-1, \ldots, 4\}$:

[ ]:
```
print map {$_, ": ", $p->HASSE_DIAGRAM->nodes_of_dim($_)->size,"\n" } (-1..4);
```

El método `INVERSE_RANK_MAP` nos permite consultar los índices de cada cara en el arreglo de caras de la retícula, agrupados según su rango:

[ ]:
```
print $p->HASSE_DIAGRAM->INVERSE_RANK_MAP;
```

A veces, nos interesa extraer solamente una parte del diagrama de Hasse. Con la función `lower_hasse_diagram` extraemos los niveles inferiores, hasta un rango determinado (recordar que la dimensión de cada cara es igual a su rango menos 1).

[ ]:
```
$HD_partial = lower_hasse_diagram($p->VERTICES_IN_FACETS,2);
$HD_partial->VISUAL;
```

[ ]:
```
print $HD_partial->FACES;
```

Observar que es posible almacenar el objeto retícula asociado a la retícula de caras de un polítopo en una variable independiente:

[ ]:
```
$reticula = $p->HASSE_DIAGRAM;
print $reticula->FACES;
```

1.1 Construcción de polítopos polares

La función `polarize` construye el polítopo polar de un polítopo dado.

[ ]:
```
$q = polarize($p);
$q->HASSE_DIAGRAM->VISUAL;
```

Para visualizar el polítopo `$q`, es conveniente re-crearlo en una nueva variable a partir de sus vértices. Recordar que el polar $P^{\triangle}$ de un polítopo P es un polítopo, si P contiene al cero como un punto interior. En nuestro caso, esta condición se cumple.

[ ]:
```
$q2 = new Polytope(POINTS=>$q->VERTICES);
$q2->VISUAL;
```

En caso de ser necesario, podemos hacer uso de la función `translate` antes de llamar a `polarize`. Esta función traslada un polítopo de acuerdo a un vector de desplazamiento.

[ ]:
```
$p = pyramid(cube(3));
print $p->VERTICES;
print "\n---\n";
### la función polarize no retorna un polítopo, porque el cero no es punto␣
 ↪interior de $p
$q = polarize($p);
$q2 = new Polytope(POINTS=>$q->VERTICES);
print($q2->VERTICES);
### la función VISUAL genera un error
$q2->VISUAL;
```

[ ]:
```
$p = pyramid(cube(3));
print $p->VERTICES;
print "\n---\n";
### empleando translate, movemos a $p para que cero sea punto interior
$t = new Vector(0,0,0,-1/2);
$p2 = translate($p,$t);
print $p2->VERTICES;
print "\n---\n";
$q = polarize($p2);
$q2 = new Polytope(POINTS=>$q->VERTICES);
### ahora $q2 es un polítopo...
print($q2->VERTICES);
### ... y la función VISUAL se puede ejecutar
$q2->VISUAL;
```

1.2 Ejercicio

1. Definamos un polítopo `$h` como un prisma sobre un hexágono

[ ]:
```
$h = prism( n_gon( 6 ) );
$h->VISUAL;
```

2. Examinemos el número de caras (no triviales) de cada dimensión que tiene `$h`:

[ ]:
```
print map{ $_, ": ", $h->HASSE_DIAGRAM->nodes_of_dim( $_ )->size, "\n" } (0..2);
```

3. Revisemos la forma que tienen las caras de dimensión 2:

[ ]:
```
print map{$h->HASSE_DIAGRAM->FACES->[$_],␣
 ↪"\n"}@{$h->HASSE_DIAGRAM->nodes_of_dim(2)};
```

4. Dibujemos el diagrama de Hasse de la retícula de caras de `$h`:

[ ]:
```
$h->HASSE_DIAGRAM->VISUAL;
```

5. Definamos a `$m` como el polar de `$h`:

[ ]:
```
### verifiquemos primero que el cero es punto interior de $h
print $h->VERTICES;
```

[ ]:
```
$m = polarize($h);
```

6. Examinemos el número de caras (no tirviales) de cada dimensión que tiene `$m`:

[ ]:
```
print map{ $_, ": ", $m->HASSE_DIAGRAM->nodes_of_dim( $_ )->size, "\n" } (0..2);
```

7. Revisemos la forma que tienen las caras de dimensión 2 de `$m`:

[ ]:
```
print map{$m->HASSE_DIAGRAM->FACES->[$_],␣
↪"\n"}@{$m->HASSE_DIAGRAM->nodes_of_dim(2)};
```

8. Dibujemos el diagrama de Hasse de la retícula de caras de `$m`:

[ ]:
```
$m->HASSE_DIAGRAM->VISUAL;
```

9. Visualicemos el polítopo polar, re-creando una copia del mismo a partir de sus vértices. Observar que `$h` contiene al origen como punto interior, por lo que no es necesario trasladar el polítopo antes de llamar a `polarize`:

[ ]:
```
$m2=new Polytope(POINTS=>$m->VERTICES);
$m2->VISUAL;
```

[ ]:
```

```