Cohomology of Line Bundles: A Computational Algorithm

— Script Manual —

March 29, 2010

1 Installation Guidelines

The *Mathematica 7* script found in the package is completely self-contained. Just copy the notebook file

`cohom-script.nb`

to some local directory and open it with *Mathematica 7*.[1]

2 Usage

2.1 Input format and structure

The script in its current form consists of three parts:

1. The main portion of the file contains the actual routines used for the program. Each time you open the script file, you briefly have to re-run this part, such that the *Mathematica* kernel working in the background has all the necessary definitions etc. available. This is done, for example, by pressing `Shift+Enter` while the cursor is still in the upper part of the script.

2. The second portion is clearly marked by the words "`User input`". Here you have to specify the relevant toric data, i.e. specifying the toric variety X for which the cohomology of the line bundle $\mathcal{O}_X(D)$ is to be computed. The following data is required in list form:

[1] The script does not use any commands special to Version 7 of *Mathematica*, i.e. there are no obstacles to using an older version. However, lacking such older software, we were unable to check if the notebook files can be properly opened in *Mathematica* versions prior to 7. In case you encounter any problems, just copy the contents of the plain text script `cohom-script.txt` found in this package into an empty project of your *Mathematica*. Please understand that we cannot offer any help if this still does not produce any useful data.

- *Coordinates:* A list of the coordinate names like {`u1,u2,u3,u4`}.
- *Stanley-Reisner ideal:* A list of lists corresponding to the generators of the Stanley-Reisner ideal, e.g. {{`u1,u2`},{`u3,u4`}}.
- *GLSM relations:* A list of lists corresponding the the GLSM charges or projective relations associated to each coordinate, for example {{`1,0`},{`1,0`},{`0,1`},{`0,1`}}. The number of those lists obviously has to be equal to the number of coordinates.

This data is assigned to a list variable, which for book-keeping purposes should carry a name suggesting the described toric variety, e.g. `P1xP1`.

3. As the next step the data referred to as "secondary" or "remnant" cohomology in the paper has to be computed, which is done by calling the procedure

```
GenerateSecondarySequences[P1xP1];
```

where `P1xP1` of course has to be replaced by the variable name you defined earlier.

4. Finally, the actual cohomology group dimensions can be computed. With respect to the choice of Picard generators implicated through the GLSM charges / projective relations, the weights can be specified directly as a list when calling the procedure

```
LinebundleCohomologyOf[{-9, 3}];
```

Note that the number of the provided numbers has to be equal to the number of the Picard generators.

2.2 Output format

After running the script, the output is basically self-explanatory. It is a list containing the dimension of the zeroth cohomology group in its first component, the dimension of the first cohomology group in its second component and so on. For instance it looks as

$$\mathrm{h}^*(\mathrm{O}(-9,3)) \;=\; \{0,\ 32,\ 0\}\,,$$

for the case considered above. Those of you, not using the graphical interface of *Mathematica* will probably see the same thing as

```
Superscript[h,*](O(-9,3))  =  {0, 32, 0}.
```

Therefore in this case we find that the zeroth as well as the second cohomology group of this line bundle do have vanishing dimension while the first cohomology

group is of dimension 32. So in general for a variety X of dimension n and $n+r$ vertices the output for a line bundle $\mathcal{O}_X(Q_1,...,Q_r)$ is of the form

$$\mathrm{h}^*(\mathrm{O}(Q_1,...,Q_r)) = \{\dim\left[H^0\left(X,\mathcal{O}_X\left(Q_1,...,Q_r\right)\right)\right],\ldots,\\ \ldots,\dim\left[H^n\left(X,\mathcal{O}_X\left(Q_1,...,Q_r\right)\right)\right]\}\,.$$

Since out algorithm does not know about the maps connecting the different spaces in the remnant sequences, it has to work only with the dimensions to determine the cohomologies. Therefore it may happen that cases appear, where no unique cohomology group can be determined but rather two or three different choices of such are possible. In such a case the program will automatically take the Serre duality into account in order to determine the only correct solution. We have tested the program with fairly complicated examples and could not find any case, where this procedure did not give a unique solution. Nevertheless, it may be that this non-uniqueness happens at some point. In this case the program will inform the user via the output

```
The algorithm does not return a unique solution.
Possible solutions are:
```

followed by a list of lists containing all possible solutions.

3 Example: Cohomology of dP^3

We work through a step by step example of using the script, which should make the steps outlined earlier clear. We consider the geometry of the del Pezzo-3 surface, whose toric data is summarized in Table 1.

vertices of the polyhedron / fan	coords	GLSM charges Q^1	Q^2	Q^3	Q^4	divisor class
$\nu_1 = (-1, -1)$	x_1	1	0	0	1	$H+Z$
$\nu_2 = (1, 0)$	x_2	1	0	1	0	$H+Y$
$\nu_3 = (0, 1)$	x_3	1	1	0	0	$H+X$
$\nu_4 = (0, -1)$	x_4	0	1	0	0	X
$\nu_5 = (-1, 0)$	x_5	0	0	1	0	Y
$\nu_6 = (1, 1)$	x_6	0	0	0	1	Z

$$\mathrm{SR}(dP_3) = \langle x_1x_2,\ x_1x_3,\ x_1x_6,\ x_2x_3,\ x_2x_5,\ x_3x_4,\ x_4x_5,\ x_4x_6,\ x_5x_6\rangle$$

Table 1: Toric data for the del Pezzo-3 surface.

After opening the script file in *Mathematica*, first execute the upper part of the script to make all the definitions available to the program. Then scroll down

to the "User Input" section and type in

```
dP3 = {
(*Coordinates*){u1, u2, u3, u4, u5, u6},
(*Stanley Reisner*){{u1, u2}, {u1, u3}, {u1, u6},
{u2, u3}, {u2, u5}, {u3, u4}, {u4, u5}, {u4, u6}, {u5, u6}},
(*Equivalence Relations*){{1, 0, 0, 1}, {1, 0, 1, 0},
{1, 1, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
};

GenerateSecondarySequences[dP3];
```

which should already be done in the script file. Then execute this second part. In the third block of the program you specify the line bundle that is to be computed for this example, e.g.

```
LinebundleCohomologyOf[{1, 120, 3, -33}];
```

and by executing this block, the dimensions of the corresponding cohomology groups $H^i(dP_3, \mathcal{O}_{dP_3}(1, 120, 3, -33))$ are computed. You will get the output

$$h^*(\text{O}(1,\ 120,\ 3,\ \text{-33}))\ =\ \{0,\ 7614,\ 0\}\,.$$