Computing Homology and Cohomology Using Macaulay2

Macaulay2 is a free software system for computations in commutative algebra. To use it, do one of the following:

(1) Download and install it from the website: `http://www2.macaulay2.com/Macaulay2/`
There is documentation available there as well.
(2) Use the online interface: `http://habanero.math.cornell.edu:3690`
(3) Log into your math account, open a terminal window, and type "M2". (I hope this works.)

When you start Macaulay2, you'll see something like this:

```
Macaulay2, version 1.6
with packages: ConwayPolynomials, Elimination, IntegralClosure, LLLBases,
               PrimaryDecomposition, ReesAlgebra, TangentCone
i1 :
```

You can compute the kernel of a matrix directly:

```
i1 : A=matrix{{1,2,3},{4,5,6}}

o1 = | 1 2 3 |
     | 4 5 6 |

              2       3
o1 : Matrix ZZ  <--- ZZ

i2 : ker A

o2 = image | -1 |
           | 2  |
           | -1 |

                               3
o2 : ZZ-module, submodule of ZZ
```

Here `i1` and `i2` are input that I have typed in, and `o1` and `o2` are Macaulay's output. (You can refer to `o1`, etc., as variables later in the session.)

By default, M2 works over $\mathbb{Z}$ (that's what `ZZ` means). Here I have defined a 2×3 matrix A corresponding to a linear transformation $\mathbb{Z}^3 \to \mathbb{Z}^2$ (Macaulay writes the arrows right to left) and computed its kernel, which Macaulay has described as the image (i.e., column space) of the matrix $B = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$.

If you have a $\mathbb{Z}$-module and you just want to know what it is up to isomorphism, you can use the prune command:

```
i3 : prune ker A

       1
o3 = ZZ

o3 : ZZ-module, free

i4 : M = matrix{{4,2,6},{2,6,4},{6,2,4}}; prune coker M

              3        3
o4 : Matrix ZZ  <--- ZZ

o5 = cokernel | 24 0 0 |
              | 0  2 0 |
              | 0  0 2 |

                                     3
o5 : ZZ-module, quotient of ZZ

i6 : N = matrix{{4,-4},{-4,4}}; prune coker N

              2        2
o6 : Matrix ZZ  <--- ZZ
```

In these last two computations, Macaulay is actually producing the Smith normal form. So coker $M \cong \mathbb{Z}_{24} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$ and coker $N \cong \mathbb{Z} \oplus \mathbb{Z}_4$. (The zero row corresponds to the $\mathbb{Z}$ summand.)

Macaulay also has a package to work with simplicial complexes (although not Δ-complexes). First you have to define a polynomial ring in variables corresponding to vertices; then you can specify the complex by its facets. Macaulay has built-in commands to compute the simplicial chain complex and homology groups.

By the way, if you want to suppress the output from a Macaulay command (e.g., if you are defining a variable or loading a package), end your input with a semicolon. I'll suppress the `i7:`'s, etc., from the following input to make it easier to read (and to cut and paste into your own M2 session).

```
load "SimplicialComplexes.m2";
R = ZZ[a..d]; -- make a polynomial ring with variables a,b,c,d
X1 = simplicialComplex{a*b, a*c, b*c};  -- the 1-skeleton of the 2-simplex
X2 = simplicialComplex{a*b*c}; -- the full 2-simplex
X3 = simplicialComplex{a*b*c, a*b*d};  -- two 2-simplices joined at an edge
X4 = simplicialComplex{a*b*c, a*b*d, a*c*d, b*c*d}; -- a hollow tetrahedron
```

Now we construct the chain complex of one of these simplicial complexes, say `X3`. This can be done with a dedicated command:

```
i13 : C = chainComplex X3 -- compute the simplicial chain complex

        1       4       5       2
o13 = ZZ  <-- ZZ  <-- ZZ  <-- ZZ

      -1      0       1       2

o13 : ChainComplex
```

The numbers on the bottom are dimensions. So X_3 has one (-1)-simplex (of course), four 0-simplices, five 1-simplices and two 2-simplices. You can extract a single boundary map as a matrix:

```
i14 : C.dd_2 -- extract the boundary map d2 (from 2-chains to 1-chains)

o14 = | -1 -1 |
      | 1  0  |
      | 0  1  |
      | -1 0  |
      | 0  -1 |
```

You can compute homology with the command `HH`, either one group at a time or all at once:

```
                5        2
o14 : Matrix ZZ  <--- ZZ

i15 : prune HH_2 X4 -- compute just one homology group

        1
o15 = ZZ

o15 : ZZ-module, free

i16 : prune HH X4 -- compute all the homology groups

o16 = -1 : 0

       0 : 0

       1 : 0

             1
       2 : ZZ

o16 : GradedModule
```

By the way, let's verify that the chain complex is a chain complex:

```
i17 : C.dd_1 * C.dd_2 -- verify that boundary-squared = 0

o17 = 0

                4        2
o17 : Matrix ZZ  <--- ZZ
```

What if you want to compute cellular homology or cohomology? The construction of the cellular chain complex of a cell complex cannot be automated as it can in the simplicial setting (which is more or less the same statement that a general cell complex cannot be specified purely combinatorially as a simplicial complex can). However, you can define matrices by hand and make them into the boundary maps of a chain complex.

For example, consider the lens space of problem #5 on HW #4 (Hatcher, §2.1, #8). We'll set $n = 3$ to keep the matrices small. The following sequence of Macaulay commands constructs its cellular chain complex. First, define the boundary maps:

```
d1 = matrix{{1,1,1,0,0},{-1,-1,-1,0,0}};
d2 = map(source d1, , matrix{{1,0,-1,-1,0,1},{-1,1,0,1,-1,0},{0,-1,1,0,1,-1},
      {1,1,1,0,0,0},{0,0,0,1,1,1}});
d3 = map(source d2, , matrix{{-1,0,1},{1,-1,0},{0,1,-1},{-1,0,1},{1,-1,0},{0,1,-1}});
```

Note that we have to make sure Macaulay knows that the image of `d2` is the domain ("source") of `d1`, and similarly for `d3` and `d2`. (For reasons of its own, the syntax for the `map` command is `map(target,source,matrix)`; here we have left the sources of `d2` and `d3` blank.) Now we build the chain complex by specifying its groups and maps explicitly:

```
C = new ChainComplex;
C.ring = ZZ;
C#0 = target d1; C#1 = target d2; C#2 = target d3; C#3 = source d3;
C.dd#1 = d1; C.dd#2 = d2; C.dd#3 = d3;
```

Now we compute homology. Fortunately, the answer comes out the way Hatcher said it would.

```
i17 : prune HH C

            1
o17 = 0 : ZZ

      1 : cokernel | 3 |

      2 : 0

            1
      3 : ZZ

o17 : GradedModule
```

In other words,

$$H_3(X) = \mathbb{Z}, \qquad H_2(X) = 0, \qquad H_1(X) = \mathbb{Z}_3, \qquad H_0(X) = \mathbb{Z}. \tag{1}$$

How about cohomology? First, we pass to the dual chain complex. However, Macaulay changes the grading so that the dual of the j^{th} piece of C is now the $j-1$st piece of C^* (which makes sense algebraically, but less so topologically):

```
i20 : D = dual(C)

        3       6       5       2
o20 = ZZ  <-- ZZ  <-- ZZ  <-- ZZ

      -3      -2      -1      0
```

The upshot is that $H^k(X)$ is actually the $(-k)^{th}$ cohomology group of this complex `D` as far as Macaulay is concerned. So the computation

```
i23 : prune HH D

             1
o23 = -3 : ZZ

      -2 : cokernel | 3 |

      -1 : 0

             1
       0 : ZZ
```

should be interpreted as saying that

$$H^3(X) = \mathbb{Z}, \qquad H^2(X) = \mathbb{Z}_3, \qquad H^1(X) = 0, \qquad H^0(X) = \mathbb{Z} \tag{2}$$

which is consistent with (1), via the Universal Coefficient Theorem for Cohomology (UCTH).

(Exercise: Get Macaulay to compute the homology and cohomology groups of the Klein bottle with integer coefficients, using your favorite cell or simplicial structure, and verify that the results are consistent with UCTH.)

By the way, all this just scratches the surface of what Macaulay can do — it can also calculate homological-algebra functors such as $\otimes$, Hom, Tor, and Ext, and it can perform calculations over polynomial rings (ideal membership, ideal intersection, Hilbert series, etc.) using Gröbner bases.