# COMPUTATIONS IN MACAULAY2

OSKAR HENRIKSSON

While not a part of the official course content (and therefore not something that will be tested on the exam), we highly encourage you to play with the concepts of the course computationally, as part of your self-studies! This is a great way to build intuition, quickly check calculations you have done by hand, and test conjectures on open-ended problems.

One of the most popular tools for commutative algebra calculations is `Macaulay2`, which you can run online, directly in your web browser at

https://www.unimelb-macaulay2.cloud.edu.au/.

In this document, we will collect useful `Macaulay2` commands that are related to the course content thus far. (The document will be updated continuously throughout the course.)

If you have questions about the code, or want to learn more about computational algebraic geometry, just talk to any of us on the teaching team! If you think this is interesting and want to learn more about how the underlying algorithms work, we recommend taking our course *Applied Algebra and Geometry* in Block 2!

**Radicals and Minimal Primes.** It is easy to define rings and ideals, determine primary decompositions, and find the minimal associated primes of an ideal in `Macaulay2`.

```
R = QQ[x,y,z]
J = ideal(x^3*z^2-2*x^3*z+x^3-z^3+2*z^2-z, y*z-y)
I = radical(J)
primaryDecomposition(J)
minimalPrimes(J)
```

**Algebra Homomorphisms.** It is easy to form algebra homomorphisms by telling the program where to send each generator. *Note:* The codomain is written **before** the domain in `Macaulay2`.

```
R = QQ[x,y,z]
S = QQ[t]
phi = map(S, R, {t,t^2,t^3})
ker(phi)
```

**Finiteness of Algebra Homomorphisms.** You can check if a homomorphism is finite via the command `isModuleFinite` from the `PushForward` package.

```
needsPackage("PushForward")
phi = map(QQ[x1,x2]/(x1*x2), QQ[y], {x1+x2})
isModuleFinite(phi)
```

**Noether Normalizations.** You can find a (random) Noether normalization of an algebra via the `noetherNormalization` command in the `NoetherNormalization` package.

```
needsPackage("NoetherNormalization")
R = QQ[x1,x2]/(x1*x2)
noetherNormalization(R)
```

**Homogenization and Saturation.** You can also homogenize ideals and find saturations with the following commands:

```
R = QQ[x,y,z,t]
I = ideal(x^3-z, z^4-r)
Ih = homogenize(I,t)
Ih = saturate(Ih,t)
```

A couple of things are important to note:

- The homogenization variable (in the above case `t`) needs to be included in the ring where the ideal is defined (in the above case `R`).
- The command `homogenize(I,t)` only homogenizes the generators that define the ideal `I`. To get the correct homogenization, we therefore need to saturate with respect to the homogenization variable `t`.

By default, a polynomial ring in `Macaulay2` is graded with all variables of degree 1. You can obtain a customized grading in the following way:

```
R = QQ[x,y,z,t, Degrees => {{1},{2},{3},{1}}]
f = x+y+z
fh = homogenize(f,t)
```

**Grassmannians.** To get generators of the homogeneous ideal of the Plücker embedding of a Grassmannian, we use the `Grassmannian` command. Note that `Macaulay2` uses the projective convention, where the Grassmannian $G(k,n)$ corresponds to `Grassmannian(k-1, n-1)`. For example, to get the ideal of $G(3,6)$ and verify that `Macaulay2` only needs 35 Plücker relations to generate the ideal (as opposed to the 225 used in the general construction given in the lecture), we write the following:

```
I = Grassmannian(2,5)
numgens(I)
```

**Hilbert Functions and Polynomials.** One can compute the Hilbert function and polynomial of a homogeneous ideal $I$ in a graded ring $R$ as follows:

```
R = QQ[x,y,z,t]
I = ideal(x^3-z^3, w^4-r^4)
hilbertFunction({3},I)
hilbertPolynomial(I, Projective => false)
```

Note that if the keyword `Projective` is set to `true`, then `Macaulay2` will express the Hilbert polynomial in terms of Hilbert polynomials of projective spaces.