MACAULAY2 - RTG SEMINAR

AUGUST 28TH, 2023

Step 1 (find a partner): First, go find someone you don't know well, and who optimally has a different level of computer-background than you, and sit next to them. Try to work on todays exercises as a team!

Our goal for today is to learn to load Macaulay2 files, and to learn about loops in Macaulay2.

Step 2 (create a file): You can edit Macaulay2 files in any editor, and then load them in Macaulay2 (in emacs, or in a terminal).

If you want to learn to edit in emacs, there are some advantages (vscode and atom have extensions that let you do similar things). You got an email about this last week hopefully.

Create a new file in your editor of choice. Save it, and call it `test.m2`. Add the following line to the file.

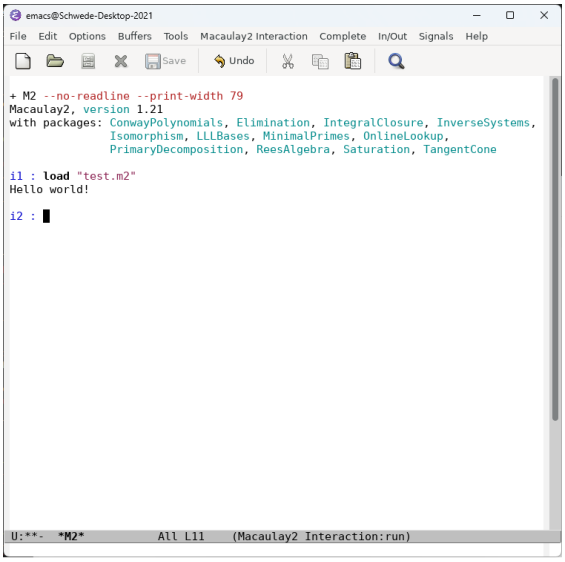
```
print "Hello world!";
```

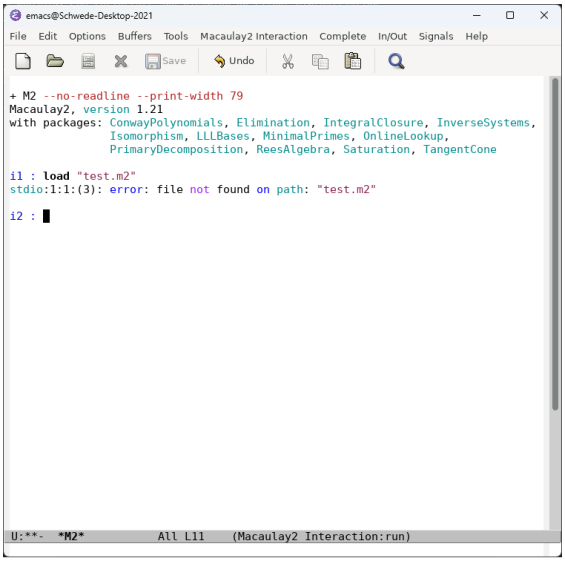
Note the semicolon ";".

Now, start up Macaulay2 and run the command:

```
load "test.m2"
```

There are at least two things that can happen (pictured below).





If you got the second one (the error), that means that *Macaulay2 cannot find your file.* See the next page for how to try to fix it.

Fixing that error depends on your system a bit.

(1) You can give a complete path to the file, instead of just the file name, might be able to run `load "/home/schwede/test.m2"` on some machines.
(2) If using the web based M2, you will need to click the button "UPLOAD FILE" and then chose "LOCAL FILE UPLOAD".
(3) If you are running Macaulay2 locally on linux (for instance lab computers), or Mac, you will need to either:
   (a) Add the location of your file to your Macaulay2 path, ie from Macaulay2 run
   ```
   path = path | {"/my/path/to/folder"}
   ```
   (although this is a temporary solution though, you need to modify your `.Macaulay2/init.m2` file to make it remembered after restarting Macaulay2), or
   (b) Save that file to a location that is already in your Macaulay2 path (for instance your home directory), or
   (c) Run emacs/Macaulay2 from the location where the file is.

   I'm happy to help with this process. Raise your hand or see if someone nearby can help.
(4) If you are on Windows, running Macaulay2 locally on WSL2, you will probably need to move your file / edit your file in the WSL2 filesystem (try `WindowsKey + R`, and then enter `\\wsl.localhost`). If you are stuck, ask for help.
(5) If you are ssh'ing into the computer system, you will need to either:
   (a) use the editor on departmental system (emacs, vscode, nano, vim), note you can open another terminal and ssh in again.
   (b) `sftp` (that is, upload) your file into the departmental system. There are programs that can help with this too. For instance Filezilla (Mac or windows or linux) or WinSCP (Windows) or Cyberduck (Mac).

Another error:

Sometimes you make a mistake in syntax, and will get an error like:

```
   test.m2:3:5:(3):[5]: error: no method for adjacent objects:
--            pirmaryDecomposition (of class Symbol)
--                   3    2
--    SPACE   ideal(y  + x ) (of class Ideal)
```

or

```
   test.m2:4:6:(3):[5]: error: division by zero
```

The a key thing to look at is the number right after `test.m2`.

The debugger:

In such a case, you will also enter the debugger, Macaulay2 will say `ii4 :` instead of `i4 :` . To exit the debugger simply run `break`

```
   i3 : load "test.m2"
test.m2:4:6:(3):[5]: error: division by zero
test.m2:4:6:(3):[5]: --entering debugger (type help to see debugger commands)
test.m2:4:5-4:7: --source code:
n = 1/0;

ii4 : break

i5 :
```

We'll talk about using the debugger later (or see the documentation).

Step 3 (loops): Once you have files loading, let's learn about loops. There are two basic kinds of loops in Macaulay2. `while` loops and `for` loops. I tend to use `while` loops as I always forget the syntax for M2's `for` loops... Here's the documentation for both which you can either google, view by googling for it, running `viewHelp "for"` or `viewHelp "while"` (or replace `viewHelp` with `help`), or open this file in pdf viewer, and click the links below:

`https://macaulay2.com/doc/Macaulay2-1.21/share/doc/Macaulay2/Macaulay2Doc/html/_for.html`

`https://macaulay2.com/doc/Macaulay2-1.21/share/doc/Macaulay2/Macaulay2Doc/html/_while.html`

If you add the following code to your file to load `test.m2` and then `load` it.

```
i = 1;
print "some primes";
while (i < 100) do (
    if (isPrime i) then (print i);
    i = i+1;
);
print "some squares";
for i from 1 to 10 do (
    print i^2;
);
```

We still had lots of semicolons. Notice we had some *conditional statements* there too. Notice `if` and the `then`.

You can *nest* loops. That is, you can also run something like this:

```
i = 0;
while (i < 100) do (
    j = 0;
    while (j < 100) do (
        if (i^2 == j^3) then print (i,j);
    );
    i = i+1;
)
```

What does the above code do?

Warning: If you ever start an infinite loop in Macaulay2, simply hit `ctrl-c` twice. If you then see something like `ii44` with a doubled `i`, you are in the debugger. Execute `break` to leave it.

Step 4 (rational points): Let's figure out how to make a random degree 3 curve. Try running the following code.

```
p = 7
R = ZZ/p[x,y,z]
f = random(3, R)
```

That will make a random degree 3 equation. You can also do things like this:

```
f = y^2*z + x*(x-z)*(x-random(0, R)*z)
```

How do we determine if a particular point (in projective space) is on the curve? Try the following code, and perhaps look up the help on `sub`.

```
sub(f, {x=>1, y=>2, z=>3})
```

Most likely, you didn't find a point.

Using `sub` you can also control how your "random" equations are created. What does the following do?

```
f = y^2*z + sub(random(3, R), {y=>0});
```

Another way to check if something evaluates to a point is to use a function, like we did previously. Frequently, this is faster than `sub` for Macaulay2 to execute.

```
phi = map(ZZ/7, R, {x=>1, y=>2, z=>3})
phi(f)
```

Step 5 (nested loops):

In a separate file, make a nested loop (ie, insert one loop inside another) which will run through all possible rational points on a plane curve (over a prime-order-field). Combine what we did above to print out all the rational points you find, ie `print (i,j,k)`.

Try it out, is there any way to make it so it only prints out one point per projective-space-equivalence-class? How can you make it not display the point `(0,0,0)`?

Step 6 (bigger fields): Macaulay2 also has support for bigger fields. Look up the help on the command **GF**. For example, try the following code.

```
K = GF(81, Variable=>g)
L = GF(2,4, Variable=>h)
```

The Variable is super important, it's a letter for a cyclic generator of the multiplication group of the finite field (in this case, $\mathbb{F}_{81}$ or $\mathbb{F}_{16}$).

Write some code to find the rational points (in some finite field) on a curve.

Step 7: Use your skills to try to make intelligent guesses as to what percentage of the points in $\mathbb{A}^3_K$, are on a fixed elliptic curve. This time make it in two variables, ie work in the ring `R = K[x,y]`, in other words set $z = 1$.

- As `K = ZZ/p` and `p` goes to infinity. (see `nextPrime`)
- As `K = GF(p,e, Variable=>g)` and `e` goes to infinity.

Think about what you expect to get before you try them. If you know a bit about cubics, what do you expect will happen if you tried a cuspidal or nodal cubic curve?

Some tips.

(1) To move one element `f` from one ring `R` to another `S` you can always do `sub(f, S)`.
(2) To convert a number to a decimal, simply multiply by `1.0`.

Compiling Macaulay2 from source.

You might want to compile Macaulay2 from source, for example to modify its source code to add new features to the C++, or to improve something that already exists.

First go to:

`https://github.com/Macaulay2/M2`

Click the green button "¡¿ Code" and copy the url to the clipboard.

Now, from a terminal, you need to run `git clone` on that text, that is, run the following command

```
git clone https://github.com/Macaulay2/M2.git
```

(or you can possibly do this with a git repository gui as well).

Warning: Note, before compiling you may want to switch to a different branch (google how to switch branches in git). The default sometimes will have errors when compiling packages.

This will create a folder M2. Go to `M2/M2/` and look at the file `INSTALL` or `INSTALL.mac`. You can also look at `INSTALL-CMake.md` for a faster compilation experience (although you might have more issues too).

Basically, you will need a ton of libraries to compile Macaulay2. Follow the instructions there for how to get them (and getting them depends on your system).

After you have them, on linux (including WSL2 in Windows) you execute the following commands (assuming they don't have errors).

```
make
./configure --enable-download
make
```

You can then run the version you just compiled from the `M2/M2` folder by executing `./M2`. You can install the version you just installed by running `sudo make install`.

I'm less familiar with installing them on a Mac, but the instructions look pretty straightforward.

Regardless, compiling Macaulay2 can take hours, especially because all 100+ packages are loaded and their help is compiled (this latter part takes a long time).