Macaulay2 Summer School 2014
Tutorial on Numerical Algebraic Geometry
Homework #1

1. **Getting acquainted with Bertini**
   Try to solve some simple problems with Bertini or Bertini.m2.
   (a) Solve $x^2 - 1 = y^2 + 1 = 0$ on your own. Here's the input file:

   ```
   variable_group x,y;
   function f,g;

   f=x^2-1;
   g=y^2+1;

   END;
   ```

   (b) Go through the screen output and the various output files suggested on the screen.
   (c) Now try solving $(x-1)^2 = y^2 + 1 = 0$. What has changed on the screen and in `main_data`?
   (d) Try cooking up some of your own polynomial system(s) to see what happens. Maybe see what Bertini does to one with positive-dimensional solution sets.

2. **Failure**
   The point of this problem is to see what Bertini does when paths fail.
   (a) First, solve $x^3 - 1 = y^2 - 1 = 0$, as in the first problem.
   (b) There should be six solutions. Rename the file `finite_solutions` as `start`.
   (c) Now try running the following input file:

   ```
   CONFIG

   UserHomotopy: 1;

   END;

   INPUT

   variable x,y;
   function f,g;
   pathvariable t;
   parameter s;

   s=t;

   f = (x^3-1)*s + (x^3+2)*(1-s);
   g = (y^2-1)*s + (y^2+0.5)*(1-s);
   ```

```
END;
```

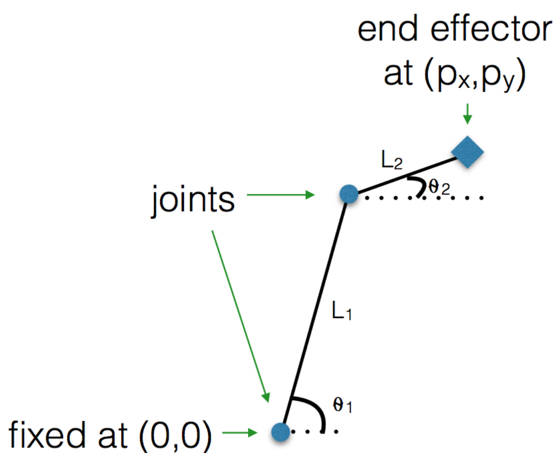
This, by the way, is how you set up your own homotopy in Bertini. Notice (a) the configuration indicating that this is a user-defined homotopy, (b) the replacement of `variable_group` with `variable`, (c) the inclusion of a pathvariable (which always runs from 1 to 0), (d) the inclusion of a parameter which depends only on the pathvariable, (e) and the appearance of only the parameter in the polynomials, never the pathvariable. Also notice that you have provided the startpoints to Bertini in the file `start`.

(d) Can you see what happened with the run? Look at the end of the screen output. Also, check out the endpoints of the paths in the file `output`. The file `failed_paths` also collects all path failures.

(e) Can you tweak the example to turn this into a near miss? What happened to the precision in your example? Spoiler alert: The answer is in this footnote.[1]

3. **Kinematics**

A 2R planar robot is a planar robotic arm with two links and two rotational joints:



The joints can spin freely (let's ignore collisions), the links have fixed lengths (L_1 and L_2), and the bottom joint is fixed at $(0, 0)$. The *forward kinematics problem* is to determine the location of the end effector, given the joint angles (θ_1 and θ_2). The *inverse kinematics problem* is to determine joint angles that will put the end effector at a particular point in the plane.

(a) Use the Pythagorean theorem to write down two (non-polynomial) equations that determine p_x and p_y (one equation each) from L_1, L_2, θ_1, and θ_2. Spoiler alert: The answer is in this footnote.[2]

(b) Replace $\cos(\theta_1)$ with new variable c_1 (and similarly for c_2, s_1, and s_2) and toss in the two equations $c_1^2 + s_1^2 - 1 = c_2^2 + s_2^2 - 1 = 0$ to get a 4-equation, 4-variable polynomial system.

[1] Try adding 1e-11*I to each of the target system constant terms.

[2] $L_1 \cos(\theta_1) + L_2 \cos(\theta_2) - p_x = L_1 \sin(\theta_1) + L_2 \sin(\theta_2) - p_y = 0.$

(c) Let $L_1 = L_2 = 1$ and set $(p_x, p_y) = (1, 1)$. What solution(s) do you expect? What do you get?

(d) Can you find a point (p_x, p_y) at which there is a single multiplicity 2 solution?

(e) What happens if (p_x, p_y) falls outside the *workspace* (the region of the plane that can be reached by the end effector)?

(f) Can you find a point (p_x, p_y) at which there are infinitely solutions? (Maybe try this one on Tuesday, though you could think about it on Monday.)

(g) Write the system for the 3R planar or 2R spatial and ask yourself similar questions.

4. **Regeneration and root counts**
Go to `https://bertini.nd.edu/BertiniExamples/` and grab the file `TwoSpheres_OneCone.input` (in Chapter 5).

   (a) Run it as is (with regeneration). How many paths were tracked?

   (b) Try commenting out "UseRegeneration: 1" and see how many paths a total degree run would take. (You could also spot this from the degrees!)

   (c) Try different choices of variable groups (without regeneration). How many paths were tracked? Can you make sense of this, i.e., come up with these root counts by hand?

5. **Parameter homotopy**
This is a special sort of homotopy that I will discuss Wednesday at 9. The idea is to solve an instance of a parameterized polynomial system with random parameter values, then move from the solutions at that random point to points of interest in parameter space. The procedure for doing this is as in #2: A first run, copy `finite_solutions` over to `start`, then do a user-defined homotopy (only in the coefficients).

   (a) Consider the intersection of the unit circle with other circles in the plane. Write a Bertini input file with the unit circle intersected with some general complex circle (perhaps using the Bertini data type *random*, which will select a random complex number for you). Solve that system. (If you use *random*, you can find Bertini's random numbers in the copy of the `input` file at the end of `main_data`, after the run.)

   (b) Try running a user-defined homotopy into various circles, e.g., one that intersects nicely, one for which the intersection is a unique singular point, and perhaps others.

   (c) If you can get to Bertini.m2, try automating this process, say on some mesh of choices in the parameter space.

6. **Open m-hom problem** Recall that there are many possible variable groupings, even for a relatively small number of variables. It is cost-prohibitive to cycle through all possible groupings, as described in the lecture. Create an algorithm to find a good (the optimal?) variable grouping without scanning through all such groupings. This problem has been eluding my group for a few years!

7. **Other examples** Check out the following websites to see some more examples:

- `http://homepages.math.uic.edu/~jan/` (click "Demos (tree)" to see them)
- `https://bertini.nd.edu/BertiniExamples/`

Pick some out and try running them yourself.

8. **Other software packages**
Install and try out Bertini.m2, HOM4PS-2.0, and/or PHCpack. Please report any issues with Bertini.m2.