Computations over $\mathbb{Q}$ vs. $\mathbb{C}$

A slightly subtle computational aspect that has been discussed in the exercise session a couple of times, but not in the lectures is the following: There is no way to truly work over $\mathbb{R}$ or $\mathbb{C}$ in a computer, since that would require encoding equivalence classes of Cauchy sequences or Dedekind cuts (or whatever your favorite way of defining these fields is).

Here, we will discuss a few different approaches for dealing with this difficulty.

0.1. **Floating point arithmetic.** One approach is to "model" $\mathbb{C}$ or $\mathbb{R}$ by working with floating point approximations. For applications, this is often good enough (and it often makes calculations very fast), but one has to be careful with **rounding errors**, and in many cases, it is useful to use **interval arithmetic** to be able to draw precise conclusions from floating point calculations.

In the course, we have used floating point and interval arithmetic every time we have used `HC.jl`. We have also used it a couple of times to solve univariate polynomial equations in `Oscar`. (Recall that the way we created floating point models with interval arithmetic of $\mathbb{C}$ and $\mathbb{R}$ in `Oscar` was by setting `CC=AcbField(64)` and `RR=ArbField(64)`, respectively. These models work great with for example the `roots` command.)

However, it is **not** a good idea to use these models for ideal-theoretic calculations such as Gröbner bases, since the rounding errors can give rise to completely incorrect results.

Example 0.1. If we approximate the ideal $\langle x^2 - 1, x + 1\rangle \subseteq \mathbb{C}[x]$ (which is equal to $\langle x + 1\rangle$), by $\langle x^2 - 1.00001, x + 1\rangle$ we obtain an ideal that is equal to $\langle 1\rangle$, since

$$(x^2 - 1.00001) - (x - 1)(x + 1) = 0.00001 \neq 0\,.$$

0.2. **Working over $\mathbb{Q}$.** Most systems we work with are defined in terms of generators with *rational coefficients*. This allows us to do the calculations over $\mathbb{Q}$, even if we want to work over $\mathbb{R}$ or $\mathbb{C}$, thanks to the following result:

Proposition 0.2. *Fix a monomial ordering $\prec$ on $\mathbb{Z}^n_{\geq 0}$. Let $f_1, \ldots, f_s \in \mathbb{Q}[x_1, \ldots, x_n]$, and suppose that $G \subseteq [x_1, \ldots, x_n]$ is a Gröbner basis with respect to $\prec$ for the ideal $I_{\mathbb{Q}} = \langle f_1, \ldots, f_s\rangle$ formed in $\mathbb{Q}[x_1, \ldots, x_n]$. Then G is also a basis with respect to $\prec$ for the ideal $I_{\mathbb{K}} = \langle f_1, \ldots, f_s\rangle$ formed in $\mathbb{K}[x_1, \ldots, x_n]$, where $\mathbb{K}$ equals $\mathbb{R}$ or $\mathbb{C}$ (or any other extension of $\mathbb{Q}$).*

Proof. Apply the Buchberger criterion. □

Example 0.3. The result of the following computation will be a Gröbner basis in the ring $\mathbb{Q}[x, y]$, but also $\mathbb{R}[x, y]$ and $\mathbb{C}[x, y]$:

```
R, (x,y) = polynomial_ring(QQ,["x","y"])
I = ideal([x^2+y^2+1//3, x^2+x*y+1//3*x])
G = groebner_basis(I)
```

On the other hand, the following computation incorrectly gives $\{1\}$ as a Gröbner basis:

```
CC = AcbField(64)
R, (x,y) = polynomial_ring(CC,["x","y"])
I = ideal([x^2+y^2+1//3, x^2+x*y+1//3*x])
G = groebner_basis(I)
```

Remark 0.4. We raised the issue that this can be confusing to the `Oscar` developers, and it seems like they agree and that in future versions, `groebner_basis` will throw an error if you try to use them for fields with floating point arithmetic. See Issue #3207 on Github.

0.3. **Finite field extension.** If you have a finite generating set for an ideal in $\mathbb{C}[x_1, \ldots, x_n]$, there will be a most finitely many non-rational coefficients appearing in it, say $\alpha_1, \ldots, \alpha_r \in \mathbb{C}$, and we can extend $\mathbb{Q}$ to $\mathbb{Q}(\alpha_1, \ldots, \alpha_r) \subset \mathbb{C}$ which can be encoded with exact arithmetic in `Oscar`. The practicalities of doing this is beyond the scope of the course, but you can read more in the number theory part of the Oscar documentation.