Lesson 11: Systems and resultants

```
> restart;
```

Real solutions of a system

We were looking at the system of equations $p1 = 0$, $p2 = 0$, where

```
> p1 := 2*x^4  - 2*y^3 + y ;
  p2 :=  2 * x^2 * y + 3*y^4 -  2*x ;
```

$$p1 := 2x^4 - 2y^3 + y$$

$$p2 := 2x^2 y + 3y^4 - 2x \tag{1.1}$$

We had found all the solutions using **solve**.

```
> S:= solve([p1 = 0, p2 = 0], [x,y]);
```

$$S := \left[[x=0, y=0], \left[x = \left(RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^4 \left(-4\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^2 - 4 + 9\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^5\right)\right) \Big/ \left(-4 + 12\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^5\right), y = RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)\right]\right] \tag{1.2}$$

We used **allvalues** and then **evalf** to get all the numerical values for the solution that involved **RootOf**.

```
> S2:= evalf([allvalues(S[2])]);
```

$$S2 := [[x = 0.4088835987 + 0.2213222308\,I, y = 0.6921985873 + 0.04574550742\,I], [x = -0.3867033783 + 0.8848037969\,I, y = 0.9644222739 + 0.4081484176\,I], [x = 0.7966066723 - 0.5214024214\,I, y = 0.2723934785 + 0.7358796527\,I], [x = -0.5214433312 - 0.8482128195\,I, y = 0.2420756341 + 0.8090287926\,I], [x = 0.1792182542 + 1.189433167\,I, y = -0.4485184522 + 1.089052952\,I], [x = -0.8166124849 - 0.08313922143\,I, y = -0.6059866650 + 0.4303097517\,I], [x = 0.1916065196 - 0.8707008718\,I, y = -0.7669819973 + 0.4552090965\,I], [x = 0.2968882966, y = -0.6992057186], [x = 0.1916065196 + 0.8707008718\,I, y = -0.7669819973 - 0.4552090965\,I], [x = -0.8166124849 + 0.08313922143\,I, y = -0.6059866650 - 0.4303097517\,I], [x = 0.1792182542 - 1.189433167\,I, y = -0.4485184522 - 1.089052952\,I], [x = -0.5214433312 + 0.8482128195\,I, y = 0.2420756341 - 0.8090287926\,I], [x = 0.7966066723 + 0.5214024214\,I, y = 0.2723934785 - 0.7358796527\,I], [x = -0.3867033783 - 0.8848037969\,I, y = 0.9644222739 - 0.4081484176\,I], [x = 0.4088835987 - 0.2213222308\,I, y = 0.6921985873 - 0.04574550742\,I]] \tag{1.3}$$

Most of the solutions are complex. We just want the real ones (those with no I).

```
> remove(has, S2, I);
```

$$[[x=0.2968882966, y=-0.6992057186]] \quad \textbf{(1.4)}$$

Here's what's going on in this bit of Maple magic.
has(A, B) checks whether a Maple expression **A** contains a certain subexpression **B**, returning either *true* or *false*. For example:

```
> has(x^2 + y, x);
```

$$\mathit{true} \quad \textbf{(1.5)}$$

```
> has(x^2 + y^3, y^2);
```

$$\mathit{false} \quad \textbf{(1.6)}$$

remove(f, A) takes a function **f** (which should return either *true* or *false*), applies it to each of the operands of **A** (e.g. the members of a list or set) and removes those where the functions return *true,* keeping those where it returns *false.*
select(f, A) does the opposite, keeping those where the function returns *true.*
Additional inputs to **f** can be included in the call to **remove** or **select** after the **A**.
Thus in our case, **remove(has, S2, I)** will evaluate **has(t, I)** for each of the members of the list S2, and returns the list consisting of those members **t** for which the result was *false,* i.e. those that contain only real numbers.

Resultants

What is that polynomial that y is supposed to be a root of?

```
> res := resultant(p1,p2,x); factor(res);
```

$$res := 324\,y^{16} - 288\,y^{13} + 144\,y^{11} + 64\,y^{10} + 320\,y^{8} - 176\,y^{6} - 64\,y^{3} + 32\,y$$
$$4\,y\left(81\,y^{15} - 72\,y^{12} + 36\,y^{10} + 16\,y^{9} + 80\,y^{7} - 44\,y^{5} - 16\,y^{2} + 8\right) \quad \textbf{(2.1)}$$

This is the **resultant** of $p1$ and $p2$, considered as polynomials in x (with coefficients that are polynomials in y). The most important property is that the resultant of two polynomials is 0 if and only if they have a common root.

To understand resultants, it's simplest to eliminate the distraction of the second variable, and start with polynomials with constant coefficients. I'll take these same polynomials, but fix $y = 3$.

```
> f1 := eval(p1, y=3);
  f2 := eval(p2, y=3);
```

$$f1 := 2\,x^4 - 51$$
$$f2 := 6\,x^2 + 243 - 2\,x \quad \textbf{(2.2)}$$

```
> resultant(f1,f2,x);
```

$$13519230468 \quad \textbf{(2.3)}$$

Suppose $f_1(x) = a\,x^n + \ldots$ and $f_2(x) = b\,x^m + \ldots$ are polynomials of degrees n and m respectively. In our example, $a = 2$, $n = 4$, $b = 6$, $m = 2$.
Let the n roots of $f_1(x)$ be $r_1, \ldots, r_n$ and the m roots of $f_2(x)$ be $s_1, \ldots, s_m$ (including repeated roots, if any). Then the **resultant** of f_1 and f_2 is $a^m\,b^n\left(\prod_{j=1}^{m}\left(\prod_{i=1}^{n}(r_i - s_j)\right)\right)$

The symbol

$\prod$ stands for product. We need to multiply all the differences $r_i - s_j$ where r_i is a root of $f_1(x)$ and s_j is a root of $f_2(x)$, and finally multiply the result by $a^m b^n$.

Note that this will be 0 if f_1 and f_2 have a root in common.

Let's calculate this for our example, using this definition. First we need the roots.

```
> Digits:= 16:
  r := fsolve(f1, x, complex);
  s := fsolve(f2, x, complex);
```

$r := -2.247165429865153, -2.247165429865153\, I, 2.247165429865153\, I, 2.247165429865153$

$s := 0.1666666666666667 - 6.361778227997438\, I, 0.1666666666666667 + 6.361778227997438\, I$ **(2.4)**

OK, four roots for f1 and two for f2. Now to plug in to the formula for the resultant.

The **mul** command will make a sequence of expressions, depending on an index variable, and multiply them together. For example, for $\prod_{i=1}^{3} c_i$:

```
> mul(c[i],i=1..3);
```

$$c_1\, c_2\, c_3 \qquad \textbf{(2.5)}$$

By the way, for addition we would use **add**.

```
> add(c[i],i=1..3);
```

$$c_1 + c_2 + c_3 \qquad \textbf{(2.6)}$$

In this case I need one **mul** inside another.

```
> 2^2 * 6^4 * mul(mul(r[i]-s[j], j=1..2), i=1..4);
```

$$1.351923046800000\; 10^{10} + 3.438906887494981\; 10^{-8}\, I \qquad \textbf{(2.7)}$$

```
> round(%);
```

$$13519230468 \qquad \textbf{(2.8)}$$

Up to roundoff error, this is the same as what Maple got for the resultant.

Resultant using division

If that was all there was to computing the resultant, it would be rather useless. Fortunately, there are algebraic ways to calculate it, which don't depend on actually getting the roots, just using the coefficients. This is why it is useful, especially for polynomials in several variables.

Note that $f_2(x) = b \left(\prod_{j=1}^{m} (x - s_j) \right)$

so the resultant of f_1 and f_2 must be $a^m \left(\prod_{i=1}^{n} f_2(r_i) \right)$

Similarly, the resultant is also

$$(-1)^{mn}\, b^n \left(\prod_{j=1}^{m} f_1(s_j)\right)$$

where the $(-1)^{mn}$ comes from the fact that each $s_j - r_i$ must be changed to $r_i - s_j$. Let's try these in our example.

```
> 2^2 * mul(eval(f2,x=r[i]), i=1..4);
```

$$1.351923046800000\ 10^{10} - 7.155308202911300\ 10^{-8}\ I \tag{3.1}$$

```
> (-1)^8 * 6^4 * mul(eval(f1,x=s[j]), j=1..2);
```

$$1.351923046800000\ 10^{10} + 0.\ I \tag{3.2}$$

This would still not be useful if we needed to calculate the roots of one of our polynomials.
But now we use division with remainder:

Given any two polynomials $f_1(x)$ and $f_2(x)$, $f_2(x) \neq 0$, we can write $\boldsymbol{f_1(x) = Q(x)\, f_2(x) + f_3(x)}$ **where $Q(x)$ and $f_3(x)$ are polynomials, and $f_3(x)$ has lower degree than $f_2(x)$.** We can get $Q(x)$ and $f_3(x)$ using **quo** and **rem**.

```
> Q:= quo(f1,f2,x);
  f3 := rem(f1,f2,x);
```

$$Q := \frac{1}{3}x^2 + \frac{1}{9}x - \frac{727}{54}$$

$$f3 := \frac{6441}{2} - \frac{1456}{27}x \tag{3.3}$$

```
> f1 = expand(Q * f2 + f3);
```

$$2x^4 - 51 = 2x^4 - 51 \tag{3.4}$$

Now the key observation is that $f_1(s_i) = f_3(s_i)$, and so (except for the factor $(-1)^{mn}\, b^m$, which will have to be adjusted) the resultant of f_1 and f_2 is the same as the resultant of f_3 and f_2.

```
> (-1)^8 * 6^4 * mul(eval(f3,x=s[j]), j=1..2);
```

$$1.351923046800000\ 10^{10} + 0.\ I \tag{3.5}$$

```
> (-1)^2 * 6^1 * mul(eval(f1,x=s[j]), j=1..2) = evalf(resultant
  (f3,f2,x));
```

$$6.258902994444444\ 10^{7} + 0.\ I = 6.258902994444444\ 10^{7} \tag{3.6}$$

```
> resultant(f1,f2,x) = (-1)^(8-2)*6^(4-1)*resultant(f3,f2,x);
```

$$13519230468 = 13519230468 \tag{3.7}$$

So instead of calculating the resultant of f_1 (of degree 4) and f_2 (degree 2), we only have f_3 (of degree 1) and f_2 to deal with. Next, take the remainder in dividing f_2 by f_3.

```
> f4 := rem(f2, f3, x);
```

$$f4 := \frac{91254805659}{4239872} \tag{3.8}$$

```
> resultant(f3,f2,x)=(1456/27)^2*resultant(f3,f4,x);
```

$$\frac{1126602539}{18} = \frac{1126602539}{18} \tag{3.9}$$

Now f_4 is just a number, so it has no roots at all. The products are just 1, so the resultant of f_3 and f_4 is $f_4^{degree(f_3)}$.

```
> resultant(f3,f4,x) = f4^1;
```

$$\frac{91254805659}{4239872} = \frac{91254805659}{4239872} \tag{3.10}$$

A procedure for the resultant

Here's how we could program **resultant** if we had to, using the ideas from the last section. I'll call it the procedure **MyResultant**.

```
> MyResultant:= proc(f1, f2, x)
    local a,b,m,n,f3;
    m:= degree(f1,x);
    a:= coeff(f1,x,m);
    n:= degree(f2,x);
    b:= coeff(f2,x,n);
    if m = -infinity or n = -infinity then  0
    elif n = 0 then  b^m
    elif m = 0 then  a^n
    elif m < n then
      f3:= rem(f2,f1,x);
      a^(n-degree(f3,x))*MyResultant(f1,f3,x)
    else
      f3:= rem(f1,f2,x);
      b^(m-degree(f3,x))*MyResultant(f3,f2,x)
    end if;
  end proc;
```

```
MyResultant := proc(f1, f2, x)                                        (4.1)
    local a, b, m, n, f3;
    m := degree(f1, x);
    a := coeff(f1, x, m);
    n := degree(f2, x);
    b := coeff(f2, x, n);
    if m = −infinity or n = −infinity then
        0
    elif n = 0 then
        b^m
    elif m = 0 then
        a^n
    elif m < n then
        f3 := rem(f2, f1, x); a^(n − degree(f3, x)) * MyResultant(f1, f3, x)
    else
        f3 := rem(f1, f2, x); b^(m − degree(f3, x)) * MyResultant(f3, f2, x)
    end if
end proc
```

Just like **resultant**, it takes three inputs: two polynomials **f1** and **f2**, and the variable name **x**. We'll use local variables **a**, **b**, **m**, **n** and **f3**. First it finds the degrees **m** and **n** of **f1** and **f2** and the leading coefficients **a** and **b**: the **coeff** command can be used to find those coefficients. Now there are a number of cases to consider, which is taken care of by an **if** statement.
The syntax of the **if** statement is

if *conditional expression 1* **then**
  statements 1
elif *conditional expression 2* **then**
  statements 2
else
  statements 3
end if

Here's how it works. First it checks *conditional expression 1* (which must be something that evaluates to either *true* or *false*). If it's *true*, then Maple executes *statements 1* (which can be any number of statements) and then skips to whatever is after the **end if**. If it's *false*, then Maple checks *conditional expression 2*. If that one is *true*, then Maple executes *statements 2* and skips to what's after the **end if**. Finally, if none of the conditional expressions is *true*, Maple executes *statements 3*. The **elif** clause is optional, and there can be any number of them. The **else** clause is also optional: if there's no **else** clause and the conditional expressions are all *false*, Maple just skips to what's after the **end if**.

In our case, the first conditional expression is **m = -infinity or n = -infinity**, which would mean **f1** or **f2** is 0. In that case, the resultant should be 0. The *statements 1* just says 0.
Since what comes after the **end if** is **end proc**, i.e. the end of the procedure, 0 would be the last thing evaluated, so the procedure would return the value 0.
If that first conditional expression is not true, Maple tries the second conditional expression **n = 0**. If that's true, **f2** is a nonzero constant, and the result should be **b^m.**
If neither of the first two is true, Maple tries **m = 0**, and if that's true (i.e. **f1** is a nonzero constant) it returns the result **a^n**.
The next thing it tries (if nothing was true yet) is **m < n**. If that's the case, Maple uses **rem** to find the remainder in dividing **f2** by **f1**, assigns that to **f3**, and then returns the appropriate power of **a** times the resultant of **f1** and **f3**. That is done using this same procedure **MyResultant**. This means that **MyResultant** is a **recursive** procedure: a procedure that calls itself.
Finally (if none of the conditional expressions was true), Maple finds the remainder in dividing **f1** by **f2**, assigns that to **f3**, and returns the appropriate power of b times the resultant of **f3** and **f2**, again done using a recursive call to MyResultant.

Recursive procedures are used very often in Maple. One thing you have to be careful of is that the chain of recursive calls must eventually come to an end. For example, this would be bad:

```
> badproc:= proc(x)
     x * badproc(x-1)
  end proc;
                    badproc := proc(x)  x * badproc(x − 1)  end proc                    (4.2)
```

```
> badproc(10);
Error, (in badproc) too many levels of recursion
```

In this case the chain didn't come to an end: **badproc(10)** called **badproc(9)**, which called **badproc**

(8), and so on... If you stopped at 1, you'd be computing 10!, but there's nothing to tell Maple to stop there, so it continues with 0, -1, -2, ..., until eventually there are so many recursive calls that Maple stops and gives you an error message. If it didn't do that, it would use up all available memory and crash the computer.

Fortunately, this shouldn't be a problem in **MyResultant**: when the procedure calls itself recursively, it does so with a new polynomial **f3** of lower degree than either of **f1** and **f2**. Eventually we must get a polynomial of degree 0 or $-\infty$, at which point the chain stops.

Now let's test out **MyResultant**, comparing its result to Maple's **resultant**.

```
> MyResultant(f1,f2,x) = resultant(f1,f2, x) ;
```

$$13519230468 = 13519230468 \tag{4.3}$$

```
> MyResultant(p1,p2, x)=
  resultant(p1,p2, x);
```

$$4\,y\,(81\,y^{15} - 72\,y^{12} + 36\,y^{10} + 16\,y^9 + 80\,y^7 - 44\,y^5 - 16\,y^2 + 8) = 324\,y^{16} - 288\,y^{13} + 144\,y^{11} + 64\,y^{10} + 320\,y^8 - 176\,y^6 - 64\,y^3 + 32\,y \tag{4.4}$$

```
> expand(%);
```

$$324\,y^{16} - 288\,y^{13} + 144\,y^{11} + 64\,y^{10} + 320\,y^8 - 176\,y^6 - 64\,y^3 + 32\,y = 324\,y^{16} - 288\,y^{13} + 144\,y^{11} + 64\,y^{10} + 320\,y^8 - 176\,y^6 - 64\,y^3 + 32\,y \tag{4.5}$$

It seems to work.
I should emphasize that this was just an exercise, not really meant to be a serious replacement for Maple's **resultant**. If you want the resultant, there's no reason not to use **resultant**.

The other variable

Now back to two equations in two variables. We wanted to solve the equations $p_1 = 0$ and $p_2 = 0$.

```
> p1,p2;
```

$$2\,x^4 - 2\,y^3 + y,\ 2\,x^2\,y + 3\,y^4 - 2\,x \tag{5.1}$$

The resultant of these (considered as polynomials in x) should be a polynomial in y which is 0 exactly when p_1 and p_2 have a common root.

```
> factor(resultant(p1,p2,x));
```

$$4\,y\,(81\,y^{15} - 72\,y^{12} + 36\,y^{10} + 16\,y^9 + 80\,y^7 - 44\,y^5 - 16\,y^2 + 8) \tag{5.2}$$

There are some technicalities here: what's actually true is that if y is a root of the resultant and the coefficients of the highest powers of x in p_1 and p_2 are nonzero for that value of y, then there is some x that makes $p_1(x, y) = p_2(x, y) = 0$. That's not a problem here, because the highest powers of x don't contain y.

Now, what is that x ? In the case $y = 0$, it's easy: you just plug in $y = 0$ to the equations and immediately see that $x = 0$ is the solution for both. But in general, especially if y can only be represented as a complicated RootOf, you really don't want to have to solve some complicated equation with coefficients involving that RootOf. Fortunately, you don't have to.

Consider the process that we used to get the resultant, using division with remainder.

```
> p3:= rem(p1,p2,x);
```

$$p3 := -\frac{2\,(-1 + 3\,y^5)\,x}{y^3} + \frac{1}{2}\,y\,(-4\,y^2 - 4 + 9\,y^5) \tag{5.3}$$

Since $p_1 = q\,p_2 + p_3$ for some polynomial q, the solutions of the system $\{p_1, p_2\}$ will be exactly the same as the solutions of $\{p_3, p_2\}$. Similarly with $\{p_3, p_4\}$ at the next step.

```
> p4:= rem(p2,p3,x);
```

$$p4 := \frac{1}{8} \frac{y^4 (81 y^{15} - 72 y^{12} + 36 y^{10} + 16 y^9 + 80 y^7 - 44 y^5 - 16 y^2 + 8)}{(-1 + 3 y^5)^2} \tag{5.4}$$

Since p_4 does not depend on x, the process stops here. But look at p_3. It has degree 1 in x, so you can solve for x in terms of y very easily.

```
> xp3:= solve(p3=0,x);
```

$$xp3 := \frac{1}{4} \frac{y^4 (-4 y^2 - 4 + 9 y^5)}{-1 + 3 y^5} \tag{5.5}$$

In general, when you come to the last step of our algorithm for resultant, when one of the polynomials has no x, the solution for x can be found from the other polynomial. In many cases (including this one) that polynomial will be of degree 1 in x, so the solution will make x a rational function of the y value.

I want to compare this result with what we got from solve for the system.

```
> S:=solve([p1=0,p2=0],[x,y]);
```

$$S := \Big[[x = 0, y = 0], \Big[x = \big(RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^4 \big(-4\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^2 - 4 + 9\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^5\big)\big) \Big/ \big(-4 + 12\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^5\big), y = RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)\Big]\Big] \tag{5.6}$$

The second member of the list, **S[2]**, is the interesting one. But I want y instead of that RootOf in the equation for x. Let's first get that RootOf.

```
> eval(y, S[2]);
```

$$RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8) \tag{5.7}$$

Now to substitute y for that in the equation for x.

```
> Substitution :=  % = y;
```

$$Substitution := RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8) = y \tag{5.8}$$

```
> eval(x, S[2]);
```

$$\big(RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^4 \big(-4\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^2 - 4 + 9\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^5\big)\big) \Big/ \big(-4 + 12\, RootOf(81\_Z^{15} - 72\_Z^{12} + 36\_Z^{10} + 16\_Z^9 + 80\_Z^7 - 44\_Z^5 - 16\_Z^2 + 8)^5\big) \tag{5.9}$$

```
> eval(%, Substitution);
```

$$\frac{y^4\left(-4y^2-4+9y^5\right)}{-4+12y^5} \qquad (5.10)$$

This should be the same as **xp3**.

```
> % - xp3;
```

$$\frac{y^4\left(-4y^2-4+9y^5\right)}{-4+12y^5}-\frac{1}{4}\,\frac{y^4\left(-4y^2-4+9y^5\right)}{-1+3y^5} \qquad (5.11)$$

```
> normal(%);
```

$$0 \qquad (5.12)$$

But sometimes our second-last polynomial may be of higher degree. It might factor, so you'll get several solutions with different formulas for x, or you may get another RootOf containing the RootOf for y. For example:

```
> p4:= y^5 + y - 3;
  p3:= x^3 * y + x + 1;
```

$$p4 := y^5+y-3$$
$$p3 := x^3\,y+x+1 \qquad (5.13)$$

```
> S:= solve({p4,p3},{x,y});
```

$$S := \left\{x = RootOf\left(3\,\_Z^3+\left(1+RootOf\left(\_Z^5+\_Z-3\right)^4\right)\,\_Z+1+RootOf\left(\_Z^5+\_Z-3\right)^4\right), y = RootOf\left(\_Z^5+\_Z-3\right)\right\} \qquad (5.14)$$

```
> Substitution:= (eval(y,S) = y);
```

$$Substitution := RootOf\left(\_Z^5+\_Z-3\right)=y \qquad (5.15)$$

```
> eval(eval(x, S),Substitution);
```

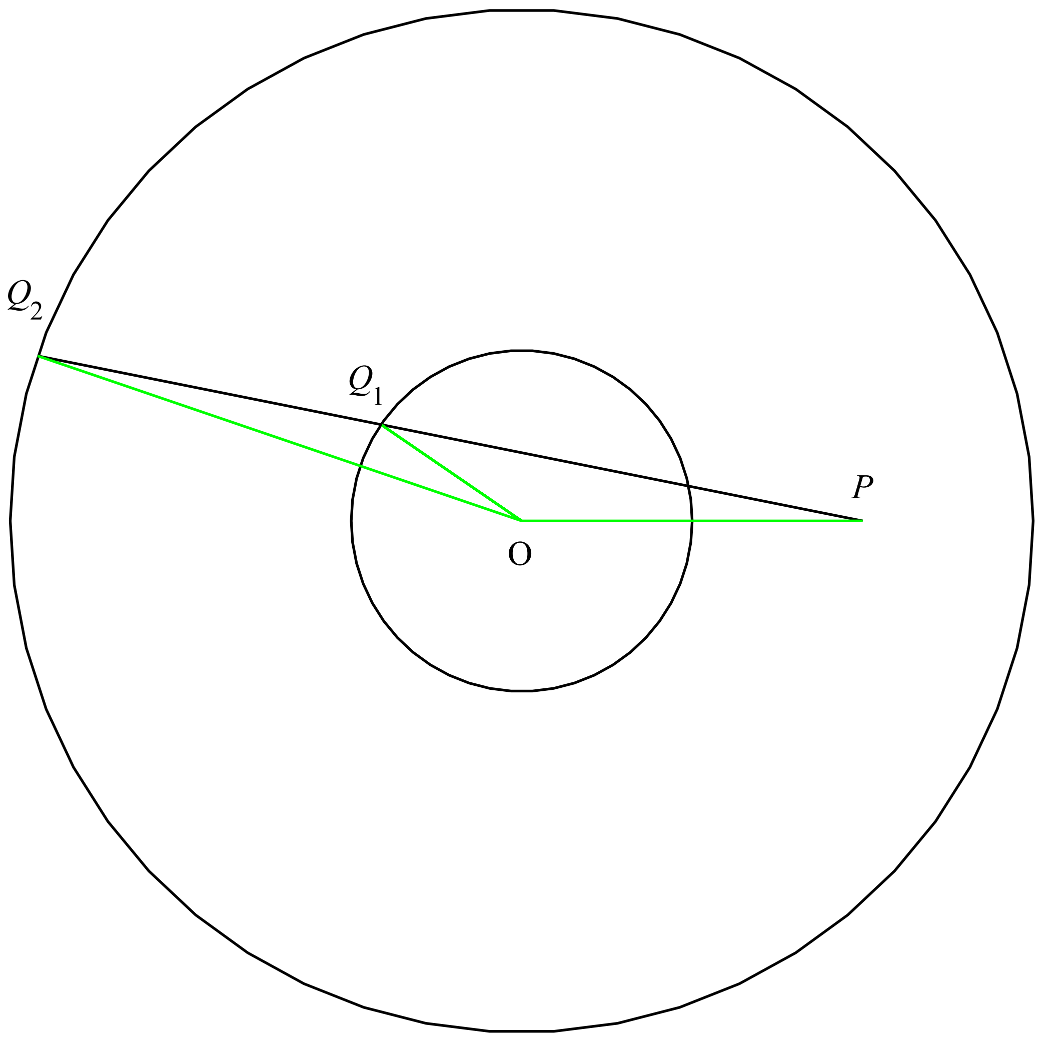
$$RootOf\left(3\,\_Z^3+\left(y^4+1\right)\,\_Z+1+y^4\right) \qquad (5.16)$$

(note that $1+y^4=\dfrac{3}{y}$ when $y^5+y-3=0$)

Some geometry

Here's a nice little application of resultants to a geometrical problem. We're given two concentric circles with radii r_1 and r_2. From a given point P at a distance d from the centre of the circles, we want to draw a line intersecting the circles at points Q_1 and Q_2 so that the distances PQ_2 and PQ_1 have a given ratio w. How should we do it?
Here's a picture in the case where $r_1=1$, $r_2=3$, $d=2$. I'll try for $w=2$ (but this line isn't the one that achieves that ratio).

```
> restart:
  with(plots): with(plottools):
  display([
      circle([0,0],1),
      circle([0,0],3),
    plot([[2,0],[-2.840,.968]],colour=black),
 plot([[2,0],[0,0],[-.825,.565],[0,0],[-2.840,.968]],
colour=green),
    textplot([[0,-.2,O],[2,0.2,P],[-0.9,0.8,Q[1]],
      [-2.9,1.3,Q[2]]])],
    axes=none, scaling=constrained);
```



A few things to notice about how I drew this picture:

- The circles were drawn with a command called **circle** in the **plottools** package. You tell it the coordinates of the centre of the circle (as a list [*x*,*y*]) and the radius. You could also give it a **colour** option. The default is black.
- The labels were printed with the **textplot** command, which is in the **plots** package. You give it a list consisting of *x* and *y* coordinates and the text you want it to print, or a list of such lists. I moved the coordinates slightly off the point I wanted to label, because I didn't want the label to be actually on the point. The text to print could be a string (enclosed in quotes), or a Maple expression.
- The **display** command is used to put all the pieces together in one plot.
- I gave the **display** command the option **axes = none**, because I didn't want axes interfering with the picture, and **scaling = constrained**, to make sure the circles look like circles.

Actually, let's write a procedure that will produce the plot for any Q_1 on the inner circle. We'll give it as input the angle $\theta = POQ_1$.

```
> drawpicture:= proc(theta)
     uses plots, plottools;
     local x1,y1,yline,x2,y2,x,r;
     x1:= cos(theta);
     y1:= sin(theta);
     yline:= y1*(2-x)/(2-x1);
     x2:= fsolve(yline^2 + x^2 = 9, x = -3 .. 0);
     y2:= eval(yline, x=x2);
     r := evalf(sqrt(((x2-2)^2 + y2^2)/((x1-2)^2 + y1^2)));
     display([
      circle([0,0],1),
      circle([0,0],3),
      plot([[2,0],[x2,y2]],colour=black),
   plot([[2,0],[0,0],[x1,y1],[0,0],[x2,y2]],     colour=green),
      textplot([[0,-.2,O],[2,0.2,P],[x1,y1+0.2,Q[1]],
        [x2,y2+0.2,Q[2]]])],
      axes=none, scaling=constrained, title=(ratio=r));
  end proc;
```

drawpicture := **proc**(theta) **(6.1)**

local *x1*, *y1*, *yline*, *x2*, *y2*, *x*, *r*;

x1 := cos(theta);

y1 := sin(theta);

yline := *y1* * (2 − *x*) / (2 − *x1*);

x2 := *fsolve*(*yline*^2 + *x*^2 = 9, *x* = −3 ..0);

y2 := *eval*(*yline*, *x* = *x2*);

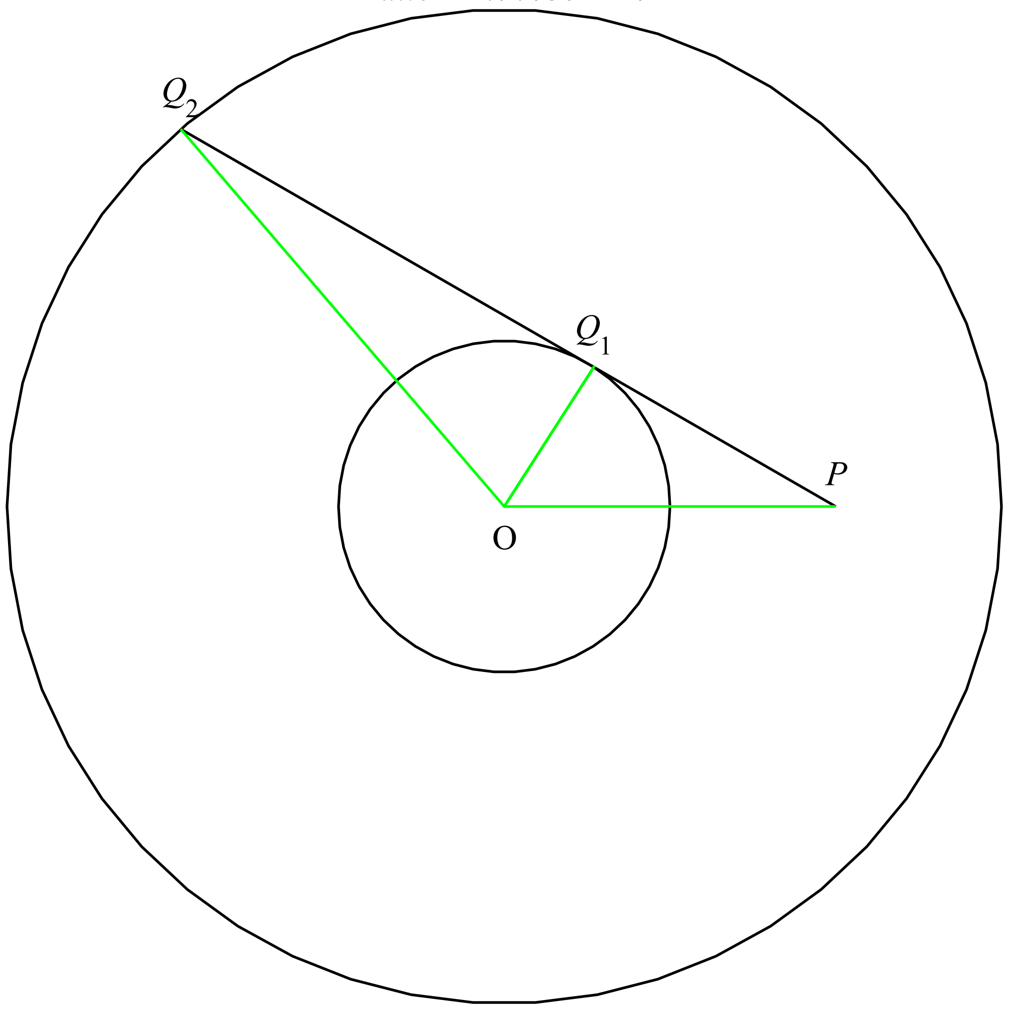
r := *evalf*(sqrt(((*x2* − 2)^2 + *y2*^2) / ((*x1* − 2)^2 + *y1*^2)));

plots:-*display*([*plottools*:-*circle*([0, 0], 1), *plottools*:-*circle*([0, 0], 3), *plot*([[2, 0], [*x2*, *y2*]], *colour* = *black*), *plot*([[2, 0], [0, 0], [*x1*, *y1*], [0, 0], [*x2*, *y2*]], *colour* = *green*), *plots*:-*textplot*([[0, −0.2, O], [2, 0.2, *P*], [*x1*, *y1* + 0.2, *Q*[1]], [*x2*, *y2* + 0.2, *Q*[2]]])], *axes* = *none*, *scaling* = *constrained*, *title* = (*ratio* = *r*))

end proc

```
> drawpicture(1);
```

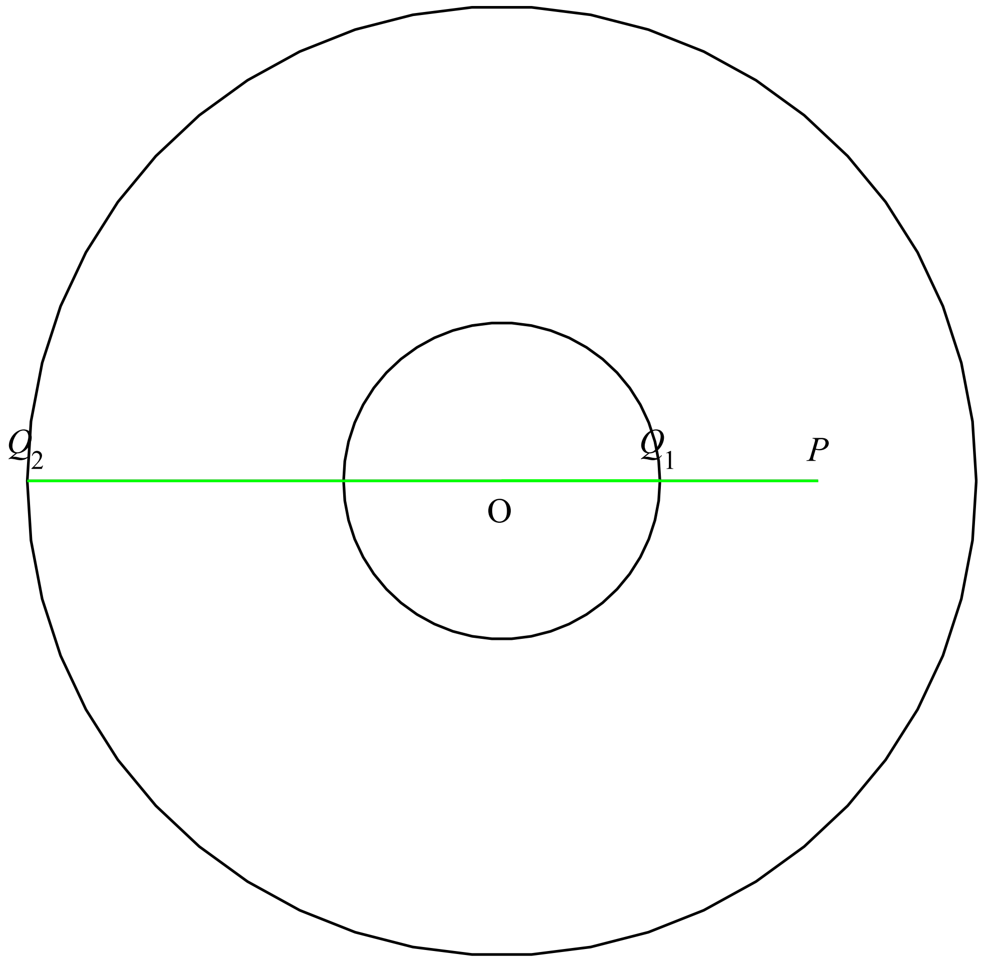
$ratio = 2.707354247$



And an animation.

```
> animate(drawpicture,[theta], theta=0..2*Pi);
```

$ratio = 5.000000000$
$\theta = 0.$



Maple objects introduced in this lesson

if ... then ... elif ... else ... end if
plottools package
circle
textplot