

14. Canonical and normal form; collecting and sorting

14.1 Canonical and normal form

An important problem in symbolic computation is the test for equality:

```
[> e1 := x*(1+y); e2 := x + x*y;
```

To test for equality, we need to use the `evalb` (eval boolean) command:

```
[> e1 = e2; evalb(e1=e2);
```

While the expressions are mathematically equivalent, for Maple they are not the same.

The canonical form for an expression is *the unique* representation which reduces the test on equality to

1. bringing the expression to its canonical form;
2. testing whether canonical forms are the same.

For univariate polynomials, *the* canonical form is the expanded polynomial, after removing of superfluous terms (e.g.: $x - x = 0$), and reordering in descending order.

For multivariate polynomials, there are many possible ways to order the variables and monomials – so we no longer uniqueness. With the command **collect**, we view a polynomial in several variables as a polynomial in one (leading) variable:

```
[> e1c1 := collect(e1,x); e2c2 := collect(e2,y);
[> evalb(e1c1=e2c2);
```

The above instructions show that we have to settle for a convention. In this case we have to take the same leading variable:

```
[> e2c1 := collect(e2,x);
[> evalb(e1c1=e2c1);
```

A more involved normal form for multivariate polynomials is the Horner form.

Normalization is the name of the process of bringing a mathematical expression into a (not *the*) normal form. For rational expressions we have seen that `normal` removes all common factors and expands numerator and denominator. Depending on whether we expand or factor numerator and denominator, we have four possible choices of a normal form.

The expressions above, `e1` and `e2`, are toys to illustrate the problem. In reality, the expressions may be huge and require a lot of resources to manipulate. The remainder of this lecture treats commands to help to achieve a normal form.

14.2 Collection

The command `collect` allows us to view a multivariate polynomial as a polynomial in one variable.

```
[> p := sum(sum(sum(' (i+j+k)*x^i*y^j*z^k', 'i'=0..2), 'j'=0..2), 'k'=0..2);
```

We can view `p` as a polynomial in `x`:

```
[> collect(p,x);
```

We can view p as a polynomial in z with coefficients polynomials in y , whose coefficients are polynomials in x :

```
[> rp := collect(p, [z,y,x], 'recursive');
```

Observe however that the conversion to the Horner form takes the default order of variables:

```
[> convert(rp, 'horner');
[> convert(p, 'horner');
```

To change the order of variables :

```
[> convert(p, 'horner', [z,y,x]);
```

This illustrates that there is no such thing as “the” Horner form for multivariate polynomials.

14.3 Sorting

We have already seen the total degree and pure lexicographic order of multivariate polynomials. Note that here we first have to decide on the order of the variables, e.g. $x > y > z$, and then choose `tdeg` or `plex` to get total degree or pure lexicographical order respectively.

This extends to more general expressions :

```
[> gp := subs(x=sin(u), y=cos(v), z=tan(w), p);
[> collect(gp, sin(u));
[> sort(gp, [sin(u), cos(v), tan(w)], 'plex');
```

14.4 A Numerical Probability One Test on Equality

When even rewriting expressions into a normal form is too expensive or simply impossible, then we can resort to a numerical probability one test, provided we have a way to evaluate the expressions.

The test on equality between two expressions $e1$ and $e2$ is performed as follows:

1. pick a random number: r ;
2. evaluate $e1$ and $e2$ in r ;
3. compare $e1(r)$ with $e2(r)$.

Unless we are really unlucky in our choice of r , the test will be reliable – whence we call it a probability one test. To increase our confidence (we could have picked r from a common factor), we can repeat the test with other random numbers.

For the two expressions $e1$ and $e2$ defined above, we perform the random test as follows:

```
[> r := [rand(), rand()];           # we have two variables x and y
[> e1r := subs(x=r[1], y=r[2], e1); # evaluate e1 in r
[> e2r := subs(x=r[1], y=r[2], e2); # evaluate e2 in r
[> evalb(e1r=e2r);                 # check for equality
```

For truly numerical expressions, we may have to use `evalf` and work with a tolerance on the errors to decide whether a number is zero or not.

14.5 Assignments

1. Consider the polynomial $p = (x^2 + xy + x + y)(x + y)$.
 - (a) Write the polynomial as a polynomial in x with coefficients in y .
 - (b) Order the monomials in p in total degree using $x > y$.
 - (c) Order the monomials in p in pure lexicographic order, using $x > y$.
2. Verify (numerically) the following equality: $\ln(\tan(\frac{1}{2}x + \frac{1}{4}\pi)) = \operatorname{arcsinh}(\tan(x))$.