The EUCLIDEAN ALGORITHM

Alain Lascoux
CNRS, Institut Gaspard Monge, Université Paris-Est, Marne-la-Vallée
Alain.Lascoux@univ-mlv.fr
`phalanstere.univ-mlv.fr/∼al`

Euler spent much time computing Taylor expansions

```
ACE> taylor(product(1-q^i,i=1..100), q,101);
         2    5    7    12    15    22    26    35    40
1 - q - q  + q  + q  - q   - q   + q   + q   - q   - q   +
     51    57    70    77    92    100        101
    q   + q   - q   - q   + q   + q    + O(q   )
```

Students are very happy to have only to push a button to do the same. A step further is to give the law of the exponents :

```
vv:=[0,1,2,5,7,12,15,22,26,35,40,51,57,70,77,92,100]:
ACE> interp([$0..16], %,x);
  6623974162912  4  22529983708  5   8069460064  6 948483778  7
- ------------- x  +----------- x  - ---------- x +--------- x
    212837625         1403325          1403325       637875
        +    ....     +
```

Does not work !

But taking only half of the exponents,

```
ACE> aa:=[seq(vv[2*i],i=1..nops(vv)/2)];
                 aa := [1, 5, 12, 22, 35, 51, 70, 92]


ACE> interp([$1..8],%, x);
                                              2
                                         3/2 x  - 1/2 x
```

One can begin to think....

Conclusion: Computers like functions of 1 variable.

Another try: the Euclidean division of two polynomials in x.

```
Reste:=proc(pol1,pol2,k) local i, f1, f2, f3;
 f1:=pol1;f2:=pol2;
  for i from 1 to k do
     f3:=rem(f1,f2,x); f3; f1:=f2; f2:=f3;
  od; f2
end:
ACE>Reste(x^4+a*x^3+b*x^2+c*x+d, x^3+e*x^2+f*x+g, 1);
   (-f+b-ea+e^2)x^2 + (-g-fa+fe+c)x-ga+d+ge
```

BEAUTIFUL

```
ACE> Reste(x^4+a*x^3+b*x^2+c*x+d, x^3+e*x^2+f*x+g, 2);
  3              2              2                  2          2
(f  - 2 g c - e a  g + b g a + b  f - f a b e + f e  b - 2 b f  - b
          3            2                                2
     - c e  + d f - d e  + e a d + 3 c f e - c b e + c e  a - 2 c f
                    2      2            2        2  2      /
     - 2 g f e + g e  a + c  + g f a - f  a e + f  a ) x  /
                    2 2               2                2        2
    (f - b + e a - e )  + (g e c + g e  b - 2 f b g - g  e + g a  f
                  2        2                      2
     + 2 d f e + b  g + d e  a - g a b e - d g + f  g - g a c + d c
          3                      /                 2 2
     - d e  - g a f e - d b e)  /  (f - b + e a - e )
                               /
```

NOT SO NICE

```
ACE> lprint(Reste(x^4+a*x^3+b*x^2+c*x+d, x^3+e*x^2+f*x+g, 3));
-(-g*a*b^3*f*d+g*a*b^2*f^2*d-2*g*a*b^2*f*c^2+4*g*a*b*f^2*c^2-6*g*a*b*d^2*f+
14*g*a*b*d^2*e^2-2*g^4*a*f*e-3*g*a^2*c^2*e^5-10*g*a^2*d^2*e^3-2*b*d^2*c*e^2*
a+8*b*d^2*g*f*e+2*d^2*f*c*e^2*a-d^3*f^2-d^3*e^4+4*d*f^5*b+14*f^2*a*b*e*d^2-\
22*f*a*b*e^3*d^2+8*f*a^2*b*e^2*d^2-4*d*b*f^3*c*a-12*d*b*f^3*g*e+2*d*c^2*e^3*
f*a+7*d*c*e^4*g*f+12*d*c*e^4*f^2*a-9*d*c*e^3*f^2*a^2+2*d*f^3*a^3*b*e-10*d*f*
e^2*b*g^2+3*d*f*e^5*b*c-10*d*f^2*e^3*b*c+18*d*f^2*e^3*b*g-2*d*f*e^2*b*c^2-6*
d*f^3*e^3*b*a+5*d*f^4*a*b*e+7*d*f^3*c*b*e-11*d*f^3*c*e^2*a+11*d*g*c*b^2*f-13
*d*g*c*b*f^2+6*d*b*g^2*a^2*f+3*d*b^3*f^2*a*e+3*d*b^2*f*c*e^3-5*d*b^2*f^2*c*e
+d*b^3*f*c*e+2*d*b^2*f^2*c*a+14*d*b^2*f^2*g*e-7*d*b^2*f^3*a*e-3*d*b^2*f*c*e^
2*a-3*d*f^2*a^2*b^2*e^2+6*d*f^2*a*b^2*e^3-6*d*f^4*b^2-6*d*e*a^3*g^2*f-6*d*f*
a*b*e^4*c+14*d*f^2*a*b*e^2*c+3*d*f*a^2*b*e^3*c-4*d*f^2*a^2*b*e*c+2*d*f*a*b*e
*c^2+d*f^3*e*a^2*g-11*d*c*f^2*e^2*g+4*d*c*f^3*e*a^2-d*c^2*e^2*a^2*f+2*d*c*e^
2*a^3*f^2-g*e^5*c*f^2*a-3*g*e^4*d*f^2*a+7*g*e^2*f^3*c*b-g^2*e*c*b^2*f+5*g^2*
e*c*b*f^2+2*g^2*e^2*b*f^3+6*g^3*e^2*c*b-5*g*e^4*b*d*c-4*g^2*e^2*c*f*a*b-7*g*
e^2*b^2*d*c-3*g^2*e^5*c*f+2*g^2*e^6*c*a+9*g^2*e^5*d*a+6*g^2*e^3*c*f^2+g^3*e^
4*f*a+18*g*e^2*b*d*c*f-8*g*e*b^3*f*d+5*g*e*b^2*f*c^2-7*g*e*b*f^2*c^2+3*g*e^2
*b^3*f*c+g*e^6*f*b*c-8*g*e^5*f*b*d-4*g*e^4*f^2*b*c-g^2*e^4*f^2*b-3*g^2*e*f^3
*c+g*e^4*f^3*c-8*g*e^3*f^3*d-2*g*e^2*f^4*c+5*g^3*e*b^2*a^2+2*g^4*e*b*a-4*g^3
*e*b^2*f+2*g^3*e*b*f^2-8*g*e^3*b*d^2+10*g*e^3*f*b*c^2-8*g*e^2*b^2*f^2*c-g*e^
6*c*d+5*g*e^5*c^2*f-3*g*e^5*c^2*b-7*g*e^3*c^2*f^2-2*g*e^2*c^3*f-3*g*e^3*c^2*
b^2+2*g*e^2*c^3*b+3*g*e*f^3*c^2-4*g*e*b^2*d^2+6*g^2*e^2*c^2*f-6*g^2*e^2*c^2*
b-6*g^3*e^2*c*f-4*g^3*e^3*f*b+6*g^3*e^3*b*a^2-7*g^2*e^2*b^2*f^2+3*g*a*c^2*b^
2*e^2-2*g*a*c^3*b*e-5*g*a*f^3*c*b*e-3*g*a^2*f^3*c*e^2-5*g^2*a*c*b^2*f+g^2*a*
c*b*f^2-6*g^2*a*c^2*f*e+6*g^2*a*c^2*b*e+6*g^3*a*c*f*e-2*g^3*a^4*e*b+6*g^2*a^
3*e^4*c+11*g^2*a^3*e^3*d-3*g^2*a^4*e^2*d-2*g^2*a^4*e^3*c+g^3*a^2*f^2*e-6*g^2
*a^2*c*e^5-3*g^2*a^2*d*f^2-15*g^2*a^2*d*e^4+e^3*a^3*g^2*b^2+4*e^5*d*f^2*g-3*
f*e^6*c*d*a-5*f^2*e^5*c*d-7*f^3*e*d^2*a-2*c^3*f*b*g-3*e^3*a^3*d^2*f+6*g*a^2*
b^2*e^3*d+6*e^5*a^2*g*b*d+6*g^2*a*b^3*e^3+3*g^2*a*b^4*e+3*e^5*a*g^2*b^2-c^2*
b^3*e*g-4*c*b^3*f^2*g+c*b^4*f*g+6*c*b^2*f^3*g-3*c*b^3*d*g-2*g^2*e^3*b^2*c+8*
g^2*e^2*b^3*f+4*d*b^2*e^2*g^2+2*d*b^4*e*g+3*d^2*b^2*e^3*a+3*e^3*a^4*d*g*f-3*
```

f^3*e^4*a^2*d-2*e^4*a^3*g*b*d+2*e^7*d*g*b-b^3*g^2*e*c+3*c*b^3*g^2*a-3*e^4*g^
2*b^3-2*e^6*d*g^2+6*e^6*d^2*f+3*e^7*d^2*a-3*b^4*g^2*e^2+4*b^4*g^2*f+e^5*a^3*
d^2+c^3*b^2*g-11*f^2*e^4*d^2-6*e^6*a*d*g*b-9*e^4*a^3*d*g*f+9*e^5*d*g*a^2*f-7
*g^3*e^4*a*b-3*b*d^2*e^4*a^2+2*e^5*g^3*b-12*b^2*d*e^4*a*g-6*b^3*d*e^2*a*g+6*
b^2*d*e^5*g-b*d*e^4*g^2+6*b^3*d*e^3*g+6*b*d^2*e^5*a+12*e^4*a^2*d^2*f-15*e^5*
a*d^2*f-e^5*g^2*b*c+4*e^4*g^2*b^2*f-3*e^4*a^2*g^2*b^2-3*e^2*a^2*g^2*b^3-f^3*
a^4*d*e^2+6*g^2*c^2*f*b-4*g^2*f*e^3*b*c+f^2*a^3*b*e^3*d-3*f^2*a^2*e^4*b*d+3*
f*a^2*c*e^5*d-6*g^3*c*f*b-f^2*e^6*b*d-f*a^3*c*e^4*d+4*f^3*e^4*b*d+f*e^7*c*d+
f^3*e^5*a*d+3*f^3*a^3*e^3*d+4*g^2*b^2*f^3-3*g^2*c^2*b^2+2*f^5*e^2*d+3*g^3*c*
b^2+2*g^4*f*b-g^2*b*f^4-3*e^6*d*g*a*f-3*b*d^2*e^6-3*b^2*d^2*e^4+3*b^3*g^2*d-\
6*b^3*g^2*f^2-d^2*b^3*e^2+c^3*f^2*g-4*c*b*f^4*g-f^4*e^4*d+c*f^5*g+3*g^3*c*f^
2-3*g^2*c^2*f^2+3*g^3*a^3*e^4+g^3*a^5*e^2-12*g*a*c^2*e^4*f+6*g*a*c^2*e^4*b-\
14*g*a*d^2*f*e^2+3*g*a^3*e^2*d^2-g*a*f^4*d-2*g*a*f^3*c^2+3*g*a*b^2*d^2+3*g*a
*c^2*e^6-2*g*a*c^3*e^3+3*g*a*d^2*f^2+11*g*a*d^2*e^4-g^2*a^2*f^3*b-3*g^2*a^2*
c^2*e^2+3*g^3*a^2*c*e^2+11*g*a*c^2*f^2*e^2+3*f^2*e^5*b*d*a-10*b^2*g^3*a*e^2-\
2*g*a^2*b*f^3*c-g^2*a*b*f^3*e+10*g*a*b*d*c*e^3-6*g*a^2*b*d^2*e+7*g*a*b^2*d*c
*e-6*g^3*a*c*b*e-14*g*a*c*e^3*d*f-3*g*a^2*c*e^4*d+9*g*a^2*c^2*e^3*f+3*g^2*a*
c*e^4*f+3*g*a^2*c*e^4*f^2-3*g*a^3*c*e^3*f^2+6*g*a^2*d^2*f*e+11*g*a*d*f^2*c*e
-3*g^2*a*d*f^2*e-6*g^2*a*d*e^3*f-6*g*a^2*d*e^3*f^2+5*g*a^3*d*e^2*f^2+g*a^3*e
^3*d*c+g*a^3*c^2*e^4+g*a^2*c^3*e^2-6*g*a^2*f^2*b*e*d+6*g*a^2*f*b*e^3*d-7*g*a
^3*f*b*e^2*d-g*a^3*f*b*e^3*c-2*g*a^3*f^2*b*e*c+3*g^2*a^2*f^2*b*e^2+4*g*a^2*f
*b*e*c^2-3*g^2*a^2*f*e^4*b+2*g^2*a^3*e*b^2*f-g^2*a^4*e^2*f*b+3*g^2*a^3*e^3*f
*b+6*g^2*a^3*e*b*d-3*g^2*a^3*e^2*c*f+7*g^2*a^3*e^2*c*b-15*g^2*a^2*b*c*e^3-13
*g^2*a^2*b*d*e^2-8*g^2*a^2*b^2*c*e-6*g^3*a^2*b*f*e-6*g*a*b^2*f*c*e^3+11*g*a*
b^2*f*d*e^2+7*g*a*b^2*f^2*c*e-3*g*a*b^3*f*c*e+g*a^2*b^2*f^2*c+5*g^2*a*b^2*f^
2*e+5*g*a^2*b^2*f*e*d+3*g*a^2*b^2*f*c*e^2+12*g^3*a*f*e^2*b-3*g*a*f*e^5*b*c-\
12*g*a*f^2*e^2*b*d+9*g*a*f*e^4*b*d+6*g*a*f^2*e^3*b*c-14*g*a*f*e^2*b*c^2-18*g
*a*b*d*c*f*e-5*g*a^2*b*d*c*e^2+4*g^2*a*b*d*f*e+7*g*a^2*d*f*c*e^2+12*g^2*a^2*
d*f*e^2+g*a*f^3*b*d+7*g*a*f^3*d*e^2+g*a*f^4*c*e+g^3*a^3*b^2-g^4*a^2*e^2+g^3*
a^3*f^2+g^3*a*f^3-4*g*a^2*c^2*f^2*e-6*g^2*a*c*f^2*e^2+2*g*a^3*c*f^3*e-3*g*a^
2*c^2*b*e^3-2*g*a^3*c^2*e^2*f+g*a^4*c*e^2*f^2-2*g^3*a*f^2*e^2-3*g^3*a^3*e^2*

```
f+2*g^3*a^4*e*f-g^2*a^2*b^3*f+2*g^2*a^2*b^2*f^2-3*g^2*a^2*b^2*d+g^3*a^3*b*e^
2-2*g^3*a^3*b*f+g^2*a*f^3*c+g*a^2*f^4*c+6*g^2*a*c^2*e^3+7*g^3*a*b^2*f-5*g^3*
a*b*f^2-6*g^3*a*c*e^3-3*g^3*a^4*e^3+3*g*a^2*f*b*e^4*c+3*g*a*c*e^5*d+2*g*a*c^
3*f*e+8*g^2*a^2*c*f*b*e-7*b^2*f*e*a*d^2-d*f^6-2*f^4*d^2-b^2*d^3+6*f^3*b*d^2+
8*f^3*d^2*e^2+2*b^3*f*d^2-6*b^2*f^2*d^2+2*b*d^3*f-2*b*d^3*e^2+c*e^5*d^2+2*d^
3*f*e^2+2*d^3*e^3*a-e^2*a^2*d^3-4*d*f^3*g^2-d*f^3*c^2-d*f^5*a^2-d*b^4*f^2+4*
d*b^3*f^3+10*b^2*f*d^2*e^2-17*f^2*e^2*b*d^2+14*f*e^4*b*d^2+2*b*d^2*c*e^3+2*b
*d^3*e*a+b^2*d^2*c*e-2*c*e^3*d^2*f-2*c*e^4*a*d^2-2*d^3*f*e*a+d^2*f^2*c*e-4*d
^2*f^2*g*e+8*d^2*e^3*g*f+19*d^2*e^3*f^2*a-8*d^2*e^2*f^2*a^2+e^3*a^2*d^2*c+6*
d*g^2*f^2*e^2+3*d*f^4*a^2*e^2-2*d*f^4*a^3*e+5*d*f^3*g*c-7*d*f^4*e^2*b+7*d*f^
3*c*e^3-3*d*f^4*c*e+2*d*f^4*c*a+4*d*f^4*g*e-d*f^5*a*e-3*d*b^3*f^2*e^2-10*d*b
^2*f*g^2-d*b^2*f*c^2-d*b^2*f^3*a^2-3*d*f^2*e^4*b^2+8*d*f^3*e^2*b^2+11*d*b*f^
2*g^2+2*d*b*f^2*c^2+2*d*b*f^4*a^2-d*c^2*e^4*f+2*d*c^2*f^2*e^2-2*b*d^2*c*f*e-
e^4*g^4-b^5*g^2-e^8*d^2-g^4*b^2-g^4*f^2-2*d*c^2*f^2*e*a+g^2*e^5*f*b*a+3*g^2*
e^3*a^2*c*f-7*g^2*e*b^3*a*f-g^2*e*b^2*a*d-6*g^2*e^3*b^2*a*f+9*g^2*e^4*b*a*c+
8*g^2*e^3*b*a*d+10*g^2*e^2*b^2*a*c-2*g^2*e*b*a^3*f^2+3*g*e^4*b^2*f*c-16*g*e^
3*b^2*f*d-g^3*e^5*a^2-3*g^2*e^4*c^2+3*g^3*e^4*c+2*g^4*e^2*f+2*g^4*e^3*a-g*e^
7*c^2+g*e^4*c^3-4*g*e^5*d^2-3*b^3*g^3*a+4*b^2*g^3*e^3-2*b*g^4*e^2+2*b^3*g^3*
e-3*e^6*a^2*d^2-e^6*g^2*b^2)/(f^3-2*g*c-e*a^2*g+b*g*a+b^2*f-f*a*b*e+f*e^2*b-\
2*b*f^2-b*d+g^2-c*e^3+d*f-d*e^2+e*a*d+3*c*f*e-c*b*e+c*e^2*a-2*c*f*a-2*g*f*e+
g*e^2*a+c^2+g*f*a-f^2*a*e+f^2*a^2)^2
```

HORRIBLE

In that case, a solution is brought by the theory of symmetric functions. The successive remainders are symmetric functions in two alphabets $\mathbb{A}, \mathbb{B}$, which are the roots of the two polynomials.

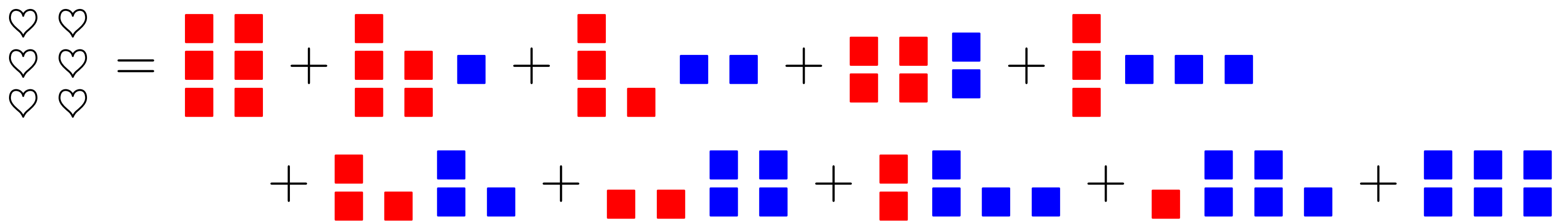
For example, the last remainder, which is the *resultant*, of $\prod_{a\in\mathbb{A}}(x-a)$ and $\prod_{b\in\mathbb{B}}(x-b)$, is just

$$R(\mathbb{A},\mathbb{B}) := \prod (a-b)$$

It will be formally represented by a rectangle (with dimensions the two degrees) :

$$\begin{array}{ccccc} \heartsuit & \heartsuit & \heartsuit & \heartsuit & \heartsuit \\ \heartsuit & \heartsuit & \heartsuit & \heartsuit & \heartsuit \\ \heartsuit & \heartsuit & \heartsuit & \heartsuit & \heartsuit \end{array}$$

and the theory of symmetric functions allows to decompose it into pieces corresponding to $\mathbb{A}$ and $\mathbb{B}$:



(experts will have recognized *Cauchy formula*; each diagram codes an explicit object: a *Schur function* in $\mathbb{A}$ or $\mathbb{B}$, which is, thanks to Jacobi, a polynomial in the coefficients of the original polynomials).

But what was the problem ?. We were given two polynomials and we have performed algebraic operations on them. These operations have preserved the symmetry between the roots of polynomials, so that in fact we have generated certain symmetric functions of the two sets of roots $\mathbb{A}$, $\mathbb{B}$.

But symmetric functions in two alphabets are more complicated than symmetric functions in only one alphabet, let us see how to reduce the difficulty.

The equations for the iterated division were

$$\begin{aligned} f &= q_1 g - \mathcal{R}_1, \\ g &= q_2 \mathcal{R}_1 - \mathcal{R}_2, \\ \mathcal{R}_1 &= q_3 \mathcal{R}_2 - \mathcal{R}_3, \\ &\dots \\ \mathcal{R}_{n-3} &= q_{n-1} \mathcal{R}_{n-2} - \mathcal{R}_{n-1}, \\ \mathcal{R}_{n-2} &= q_n \mathcal{R}_{n-1} \end{aligned}$$

One may rewrite the first equation as follows:

$$\frac{g}{f} = \frac{1}{q_1 - \frac{\mathcal{R}_1}{g}} \; .$$

Iterating, one gets a *continued fraction*:

$$\frac{g}{f} = \cfrac{1}{q_1 - \cfrac{1}{q_2 - \cfrac{1}{\ddots - \cfrac{1}{q_n}}}}$$

which codes in a compact manner the Euclidean algorithm. The same problem can lead to many different pictures !

But one can forget that f and g are polynomials, one can as well divide formal series in z, this time according to increasing powers. Writing $f = f_{-1}$, $g = f_0$, one has now

$$
\begin{aligned}
f_{-1}(z) &= (1 - z\zeta_0)\, f_0(z) + \beta_1 z^2\, f_1(z)\ , \\
f_0(z) &= (1 - z\zeta_1)\, f_1(z) + \beta_2 z^2\, f_2(z)\ , \\
f_1(z) &= (1 - z\zeta_2)\, f_2(z) + \beta_3 z^2\, f_3(z)\, , \ldots, \\
f_{i-1}(z) &= (1 - z\zeta_i)\, f_i(z) + \beta_{i+1} z^2\, f_{i+1}(z)\, , \ldots
\end{aligned}
$$

Thus, the Euclidean algorithm may be interpreted as a process with input a pair of unitary series (f, g), and output two infinite sequences

$$
[\zeta_0,\, \zeta_1,\, \zeta_2,\, \ldots] \qquad \& \qquad [\beta_1,\, \beta_2,\, \ldots]\, .
$$

However, one does not change the output by taking the pair

$$(f/g, 1), \quad \text{or} \ (1, g/f) :$$

all the equations are divided by the same factor g or f.

We could not do this with polynomials: the quotient of two polynomials is not, in general, a polynomial, but

the quotient of two series is a series.

Now, we can take back our computer :

```
DivideSeries:=proc(f,N)  local i,out,f1,f2;
 f1:=1; f2:=f;
 for i from 1 to N do
  out:=OneStepDivideSeries(f1,f2);
  f1:=f2; f2:=op(2,out) # out=[ linear factor, new series]
 od;
 f2
end:
```

As an input, we take the generating function of complete functions h_i (sums of all monomials of a given degree), so that the output will be a symmetric function.

```
f:=1-z*h1+z^2*h2-z^3*h3+z^4*h4-z^5*h5+z^6*h6-z^7*h7+z^8*h8;
ACE> DivideSeries(f,2);

        2       6      2    2      2          2    4      2
- (-z h3  h1 - z  h8 h2  - z  h4 h2  + z h3 h2  - z  h6 h2

         2                2     3
     - h3  + h4 h2 - h4 h1  - h2  + 2 h3 h1 h2 + z h3 h4

                        2    2          2      2    2
     - z h5 h2 + z h5 h1  + z  h6 h2 - z  h6 h1  - z  h3 h5

        3          3      2    3      2    3          4
     - z  h7 h2 + z  h7 h1  + z  h5 h2  + z  h3 h6 + z  h8 h2

        4      2    4          5          5      2
     - z  h8 h1  - z  h3 h7 + z  h3 h8 + z  h7 h2  - z h1 h2 h4

        2             2             3             3
     + z  h3 h1 h4 + z  h1 h2 h5 - z  h1 h2 h6 - z  h3 h1 h5

        4             4             5             5
     + z  h1 h2 h7 + z  h3 h1 h6 - z  h1 h2 h8 - z  h3 h1 h7

        6             /
```

```
     + z  h8 h1 h3)  /  (
      3                 2     2
    h2  - h4 h2 + h4 h1  + h3  - 2 h3 h1 h2)

ACE> Tos(numer(%),collect);
                                            2                3
s[2, 2, 2] - s[3, 2, 2] z + s[4, 2, 2] z  - s[5, 2, 2] z  +
                 4
    s[6, 2, 2] z  + .....

ACE> Tos(numer(DivideSeries(f,3)),collect);
                                                     2
s[3, 3, 3, 3] - s[4, 3, 3, 3] z + s[5, 3, 3, 3] z  +   .....
```

But, if we prefer, we can compute with symmetric functions in explicit variables. This time we (formally) factorise the series :

$$f = (1 - zx_1)(1 - zx_2)(1 - zx_3)\cdots$$

```
ACE> f:=convert([seq(1-z*cat(x,i),i=1..5)],`*`);
  f := (1 - z x1) (1 - z x2) (1 - z x3) (1 - z x4) (1 - z x5)


ACE> numer(DivideSeries(f,2));

     3   2   2       3   3        2     3   3   2
-z x4  x5  x2  - z x2  x1  x5 - x5  x4 z  x3  x1  x2
       2   3   2   2          2   2   2   2   2
    + z  x4  x3  x1  x5 + 2 z  x4  x1  x2  x5
         2   2   2   2   2      2   2   2   2   2
    + 2 z  x4  x1  x3  x5  + 2 z  x5  x2  x4  x3
       2   3   2   2        2     3   2      3
    + z  x4  x1  x2  x5 - x5  x4 z  x3  x1 x2
       2   3   3          2   3   3          2   3   3
    + z  x3  x4  x2 x5 + z  x3  x4  x1 x2 + z  x2  x4  x3 x5
       2   3   2   2       2   3   3          2   3   2   2
    + z  x4  x1  x2  x3 + z  x2  x4  x1 x3 + z  x4  x3  x2  x5
       2   3   2   2         2   2      2   2
    + z  x4  x1  x3  x2 + 5 z  x1  x2 x3  x4  x5
       2   3   3          2   3   3          2   3   2   2
```

```
+ z  x1  x4  x2 x5 + z  x1  x4  x2 x3 + z  x1  x5  x4  x3
   2   3   3          2   3   2   2       2   3   2   2
+ z  x1  x4  x3 x5 + z  x1  x5  x4  x2 + z  x4  x2  x3  x1
   2   3   3              3  3   2      2
+ z  x2  x4  x1 x5 - x5 x4  z  x3  x1 x2
    3   2  3   2           3     3   2   2
- x5  x4  z  x3  x1 x2 - x5  x4 z  x3  x1  x2
      2     3   2   2   2      2   2   3
- 2 x5  x4 z  x3  x1  x2  + 3 z  x4  x5  x1 x2 x3
     2   3      2          2   3   2   2
+ 3 z  x3  x1 x4  x2 x5 + z  x3  x1  x4  x2
   2   3   2   2       2   3   2   2
+ z  x3  x1  x4  x5 + z  x1  x2  x5  x3
     2   3      2          2   2   2   3
+ 3 z  x3  x1 x5  x2 x4 + z  x1  x2  x3  x5
     2   2   2      2         2   2      3
+ 5 z  x1  x2  x3 x4  x5 + 3 z  x1  x2 x3  x4 x5
     2   2      2   2         2      3   2
+ 5 z  x1  x2 x3  x5  x4 + 3 z  x1 x2  x3  x4 x5
     2   3      2          2   3   3
+ 3 z  x1  x2 x3  x4 x5 + z  x1  x3  x4 x5
   2   3   3            2      2   2   2
+ z  x5  x1  x2 x4 + 5 z  x1 x2  x3  x5  x4
   2   3   3            2   2   2   2   2
+ z  x1  x2  x4 x5 + 2 z  x1  x2  x3  x5
   2   3   2   2         2   3      2
+ z  x3  x1  x5  x4 + 3 z  x4  x5 x1  x2 x3
   2   3   2   2       2   3   3          2   3   3
+ z  x1  x2  x3  x5 + z  x2  x5  x1 x3 + z  x3  x4  x1 x5
   2   3   2   2       2   3   3          2   3   2   2
+ z  x5  x1  x4  x3 + z  x5  x1  x2 x3 + z  x5  x1  x4  x2
```

```
   2   3   2   2       2   3   3          2   3   3
+ z  x5  x1  x3  x2 + z  x1  x2  x3 x4 + z  x2  x3  x1 x4
   2   3   2   2       2   3   3          2   3   2   2
+ z  x2  x3  x4  x5 + z  x2  x3  x4 x5 + z  x2  x3  x4  x1
   2   3   3          2   3   2   2       2   3   2   2
+ z  x1  x3  x2 x4 + z  x1  x3  x4  x2 + z  x1  x3  x4  x5
   2   3   2   2         2   3   2
+ z  x4  x5  x1  x2 + 3 z  x4  x5  x1 x2 x3
   2   3   2   2       2   2   2   3
+ z  x2  x5  x4  x1 + z  x1  x2  x3  x4
   2   3   2   2       2   3   3
+ z  x1  x3  x5  x4 + z  x1  x3  x2 x5
     2   3      2          2   3   2   2
+ 3 z  x5  x1 x3  x2 x4 + z  x3  x2  x4  x1
   2   3   2   2       2   3   2   2
+ z  x3  x2  x5  x4 + z  x3  x2  x4  x5
   2   3   2   2       2   3   2   2       2   3   3
+ z  x5  x4  x2  x1 + z  x2  x3  x5  x4 + z  x2  x3  x1 x5
   2   3   2   2       2   3   2   2
+ z  x5  x4  x3  x1 + z  x5  x1  x2  x4
   2   3   2   2       2   3   2   2
+ z  x4  x5  x2  x1 + z  x5  x1  x3  x4
     2   3      2          2   3   2   2
+ 3 z  x4  x5 x2  x1 x3 + z  x5  x1  x2  x3
   2   3   3            2   3      2
+ z  x5  x1  x3 x4 + 3 z  x5  x1 x2  x3 x4
   2   3   3          2   3   3          2   3   2   2
+ z  x3  x5  x1 x4 + z  x3  x5  x2 x4 + z  x5  x4  x2  x3
   2   3   2   2       2   3   2   2
+ z  x3  x5  x4  x2 + z  x3  x5  x4  x1
   2   3   2   2       2   3   2   2
```

```
+ z  x3  x5  x1  x2 + z  x3  x5  x2  x1
   2   3   2   2       2   3   2   2
+ z  x2  x5  x4  x3 + z  x5  x4  x3  x2
     2   3      2          2   3   2   2
+ 3 z  x2  x1 x5  x3 x4 + z  x2  x1  x4  x3
   2   3   2   2         2   3      2
+ z  x2  x1  x5  x3 + 3 z  x5  x4 x1  x2 x3
   2   3   2   2       2   3   2   2
+ z  x2  x1  x3  x4 + z  x2  x1  x4  x5
     2   2   3               2   3      2
+ 3 z  x2  x1  x3 x4 x5 + 3 z  x2  x1 x4  x3 x5
   2   3   2   2       2   3   2   2       2   3   3
+ z  x4  x5  x3  x1 + z  x4  x5  x2  x3 + z  x2  x5  x3 x4
   2   3   2   2       2   3   2   2
+ z  x1  x3  x5  x2 + z  x5  x3  x2  x1
   2   3   2   2         2   3      2
+ z  x1  x2  x4  x5 + 3 z  x1  x2 x4  x3 x5
     2   2   2   2            2   2   2   2
+ 5 z  x4  x5  x1  x2 x3 + 5 z  x4  x5  x2  x1 x3
   2   3   2   2         2   2   2      2
+ z  x4  x5  x1  x3 + 5 z  x1  x2  x3 x5  x4
   2   3   3          2   3   2   2
+ z  x1  x2  x3 x5 + z  x1  x2  x3  x4
     2   2   2   2          2   3   2   2
+ 5 z  x1  x2  x3  x4 x5 + z  x1  x2  x5  x4
   2   3   2   2       2   3   2   2
+ z  x2  x1  x5  x4 + z  x4  x5  x3  x2
   2   3   2   2         2   2   2   2   2
+ z  x1  x2  x4  x3 + 2 z  x1  x2  x3  x4
   2   3   3          2   3   3
+ z  x4  x5  x1 x3 + z  x4  x5  x1 x2
```

$$+ 3 z^{2} x4^{3} x5 x3^{2} x1 x2 + 5 z^{2} x4^{2} x5^{2} x3^{2} x1 x2$$

$$+ z^{2} x4^{3} x5^{3} x2 x3 + 3 z^{2} x1 x2^{2} x3^{3} x4 x5$$

$$+ 3 z^{2} x1^{2} x2^{3} x3 x4 x5 + 3 z^{2} x1^{3} x2 x5^{2} x3 x4$$

$$+ z^{2} x5^{3} x3^{2} x2^{2} x4 + z^{2} x2^{3} x5^{3} x1 x4 + z^{2} x2^{3} x5^{2} x3^{2} x1$$

$$+ z^{2} x5^{3} x3^{3} x1 x2 + 5 z^{2} x1 x2^{2} x3^{2} x4^{2} x5$$

$$- 3 z x1 x2^{3} x3 x4 x5 - 6 z x1 x2 x3^{2} x4^{2} x5$$

$$- 3 z x2^{2} x1^{2} x4^{2} x5 - 6 z x2^{2} x1 x4^{2} x3 x5$$

$$+ z^{2} x1^{2} x2^{3} x3^{2} x5 - 3 z x1^{2} x2^{2} x5^{2} x4$$

$$- 2 z x1^{2} x2^{3} x4 x5 - 6 z x1^{2} x2 x3 x4^{2} x5$$

$$- 2 z x1^{3} x3^{2} x5 x4 - z x4^{3} x5^{3} x2 - z x4^{3} x5^{3} x1$$

$$- z x5^{3} x1^{2} x4^{2} - 2 z x2^{2} x3^{3} x4 x5 - z x4^{3} x5^{3} x3$$

$$- z x1^{3} x3^{2} x5^{2} - z x1^{3} x3^{3} x5 - z x5^{3} x1^{2} x3^{2}$$

$$- z x2^{3} x3^{2} x4^{2} - z x2^{3} x3^{2} x5^{2} - z x2^{3} x5^{3} x3$$

$$- z x2^{3} x3^{3} x4 - z x2^{3} x5^{3} x1 - z x2^{3} x1^{2} x4^{2}$$

$$- z x1^{3} x2^{3} x3 - 2 z x4^{3} x3^{2} x1 x2 - 2 z x4^{3} x3 x2^{2} x1$$

3 3 3 3 3 3

```
- z x2  x4  x3 - z x3  x4  x2 - z x4  x2  x5
      3   2   2       3   3          3   3
- z x1  x3  x4  - z x1  x3  x2 - z x3  x4  x5
      3   2   2         2   3               2   3
- z x1  x5  x4  - 2 z x1  x3  x4 x5 - 2 z x1  x4  x2 x3
      3   3          3   2   2       3   2   2
- z x1  x4  x5 - z x4  x1  x2  - z x4  x1  x3
      3   2   2       3   3          3   2   2
- z x3  x1  x4  - z x2  x3  x1 - z x5  x4  x2
      3   2   2       3   2   2       3   3
- z x5  x3  x4  - z x3  x5  x4  - z x1  x2  x4
      3   2   2       3   2   2       3   3
- z x2  x1  x5  - z x1  x2  x5  - z x4  x1  x2
        3      2            3   2
- 2 z x1  x2 x3  x5 - 2 z x1  x2  x3 x5
        2         2          3   2   2       3   3
- 6 z x1  x2 x3 x5  x4 - z x3  x2  x4  - z x3  x2  x5
        2   2   2            2   2   2
- 3 z x5  x1  x3  x4 - 3 z x1  x2  x3  x4
        3   2               2   3
- 2 z x1  x2  x3 x4 - 2 z x1  x2  x3 x4
           2   3          3   2   2         3   2
- 2 z x1 x2  x3  x4 - z x3  x2  x5  - 2 z x4  x5  x1 x3
        2      3               3   2
- 2 z x1  x2 x3  x4 - 2 z x1 x2  x3  x5
        2      2   2         3      2
- 3 z x1  x2 x3  x5  - 2 z x1  x2 x5  x4
        3      2            2   3
- 2 z x5  x1 x2  x4 - 2 z x3  x5  x1 x2
        2      2             3   2   2         3      2
- 6 z x3  x1 x2  x4 x5 - z x4  x5  x1  - 2 z x4  x5 x2  x3
```

```
        3      2            3      2
- 2 z x3  x1 x4  x5 - 2 z x1  x2 x3  x4
           2      2          3   2   2       3   3
- 6 z x1 x2  x3 x5  x4 - z x3  x1  x5  - z x1  x4  x3
      3   3          3   2   2       3   3
- z x4  x2  x1 - z x4  x2  x3  - z x1  x3  x4
      3   3            2   3               3      2
- z x4  x3  x1 - 2 z x2  x3  x1 x5 - 2 z x1  x2 x4  x3
        3                     3   2             2   3   2
- 3 z x1  x2 x3 x4 x5 - 2 z x1  x2  x5 x4 - z x1  x2  x3
      3   2   2         2   2   2            3      2
- z x1  x2  x3  - 3 z x5  x4  x3  x2 - 2 z x5  x4 x1  x3
        3      2            3      2
- 2 z x5  x4 x1  x2 - 2 z x3  x2 x5  x4
        3      2            3      2
- 2 z x3  x1 x5  x2 - 2 z x2  x3 x4  x5
        2   2   2            2   3
- 3 z x1  x3  x4  x2 - 2 z x3  x5  x2 x4
        2   2   2          2   2   3       3   3
- 3 z x2  x5  x4  x1 - z x1  x2  x3  - z x5  x1  x4
      3   2   2       3   2   2       3   3
- z x5  x1  x2  - z x4  x5  x3  - z x2  x5  x4
      3   2   2       3   2   2       3   3
- z x2  x5  x4  - z x5  x3  x2  - z x3  x5  x2
      3   3          3   3            3      2
- z x3  x5  x1 - z x3  x5  x4 - 2 z x4  x5 x1  x3
        2   2   2            3      2
- 3 z x2  x3  x4  x5 - 2 z x5  x1 x2  x3
        3      2            3      2
- 2 z x3  x2 x4  x5 - 2 z x1  x3 x5  x4
        2   3               2   2   2
```

```
- 2 z x4  x5  x1 x2 - 3 z x4  x5  x3  x1
        2   2                  2   3
- 6 z x4  x5  x1 x2 x3 - 2 z x4  x5  x2 x3
        3      2            2   2   2
- 2 z x2  x1 x4  x5 - 3 z x4  x5  x2  x3
        2   3               3      2
- 2 z x3  x2  x4 x5 - 2 z x4  x5 x1  x2
        2   2   2            2   3
- 3 z x3  x5  x2  x4 - 2 z x4  x5  x1 x3
        3   2               3      2
- 2 z x5  x1  x2 x3 - 2 z x5  x4 x3  x1
        3                     2   3
- 3 z x5  x4 x1 x2 x3 - 2 z x5  x4  x2 x3
        3      2            3
- 2 z x3  x1 x5  x4 - 3 z x3  x5 x1 x2 x4
        3      2            2      2
- 2 z x3  x1 x4  x2 - 6 z x3  x5 x1  x2 x4
           2   2   2         2   2   2
- 3 z x1 x2  x3  x5  - 3 z x1  x2  x3  x5
              2   2               2   2   2
- 6 z x1 x2 x3  x5  x4 - 3 z x1 x2  x3  x4
        2      3            3      2
- 2 z x1  x2 x3  x5 - 2 z x1  x3 x4  x5
        3      2            3      2
- 2 z x2  x1 x3  x4 - 2 z x4  x5 x3  x2
        3      2            3      2
- 2 z x4  x5 x3  x1 - 2 z x2  x1 x5  x4
        2      2               2   2   2
- 6 z x1  x3 x2  x4 x5 - 3 z x1  x3  x4  x5
        2   2      2         2   2   2
- 3 z x1  x2  x3 x5  - 3 z x4  x5  x1  x2
```

```
        2   2   2            3      2
- 3 z x4  x5  x1  x3 - 2 z x2  x1 x5  x3
        3      2            2   2   2
- 2 z x2  x1 x4  x3 - 3 z x1  x2  x4  x3
        2   3               3      2
- 2 z x1  x2  x3 x5 - 2 z x1  x2 x5  x3
        3                     3      2
- 3 z x4  x5 x1 x2 x3 - 2 z x4  x5 x2  x1
        3      2            3      2
- 2 z x5  x3 x2  x4 - 2 z x2  x3 x5  x4
        3      2          3   3          3   3
- 2 z x1  x2 x4  x5 - z x5  x1  x3 - z x5  x1  x2
        3   2             3   2   2          3   3      3
- 2 z x4  x5  x1 x2 - z x1  x2  x4  - x5 x4 z  x3  x1 x2
    3     3         3     3     3      3
- x5  x4 z  x3 x1 x2  - x5  x4 z  x3 x1  x2
         3      3   3     3   3  3
- x5 x4 z  x3 x1  x2  - x5  x4  z  x3 x1 x2
    3     3   3              3  3         3
- x5  x4 z  x3  x1 x2 - x5 x4  z  x3 x1 x2
       3  3   3              2  3      3   2
- x5 x4  z  x3  x1 x2 - x5 x4  z  x3 x1  x2
       3  3      3             3   3   3
- x5 x4  z  x3 x1  x2 - x5 x4 z  x3  x1  x2
         2  3   2   2   2     2   3  3         2
- 2 x5 x4  z  x3  x1  x2  - x5  x4  z  x3 x1 x2
    2     3   2   3          2   2  3   2   2
- x5  x4 z  x3  x1  x2 - 2 x5  x4  z  x3  x1  x2
    3   2  3      2        2   3  3      2
- x5  x4  z  x3 x1  x2 - x5  x4  z  x3 x1  x2
    2     3      3   2        2  3   3   2
```

```
- x5  x4 z  x3 x1  x2  - x5 x4  z  x3  x1  x2
    3   2  3         2        2  3   3      2
- x5  x4  z  x3 x1 x2  - x5 x4  z  x3  x1 x2
       2  3      2   3          3   3   2   2
- x5 x4  z  x3 x1  x2  - x5 x4 z  x3  x1  x2
       3  3      2   2        3  3   2   2
- x5 x4  z  x3 x1  x2  - x5 x4  z  x3  x1  x2
         3   2   3   2          3   2   2   3
- x5 x4 z  x3  x1  x2  - x5 x4 z  x3  x1  x2
    2     3   3      2     2   2  3      3
- x5  x4 z  x3  x1 x2  - x5  x4  z  x3 x1  x2
       2  3   2      3        2  3   2   3
- x5 x4  z  x3  x1 x2  - x5 x4  z  x3  x1  x2
      2   2  3   2      2     2   2  3   3
- 2 x5  x4  z  x3  x1 x2  - x5  x4  z  x3  x1 x2
    2     3      2   3     2   3  3   2
- x5  x4 z  x3 x1  x2  - x5  x4  z  x3  x1 x2
    2   2  3         3     3     3   2      2
- x5  x4  z  x3 x1 x2  - x5  x4 z  x3  x1 x2
    3     3      2   2       2   2  3      2   2
- x5  x4 z  x3 x1  x2  - 2 x5  x4  z  x3 x1  x2
    3              3              3
+ x5  x1 x3 x4 + x1  x3 x5 x4 + x5  x1 x2 x3
      2   2           3              3
+ 2 x1  x3  x5 x4 + x4  x5 x1 x2 + x4  x5 x1 x3
      2     2           2      2          2      2
+ 2 x5  x4 x3  x1 + 2 x5  x4 x2  x3 + 2 x5  x4 x1  x3
    3                   2   2             2   2
+ x4  x5 x2 x3 + 2 x1 x2  x3  x4 + 2 x1 x2  x3  x5
      2   2             2   2             2      2
+ 2 x1  x2  x3 x5 + 2 x1  x2  x3 x4 + 2 x1  x2 x5  x4
```

$$+ 3 x_1 x_2 x_3^2 x_5 x_4 + x_1^2 x_2^2 x_4^2 + x_1^3 x_2 x_4^2$$
$$+ x_1^2 x_2^3 x_4 + x_1^2 x_2^3 x_5 + x_4^3 x_5^2 x_1 + x_4^3 x_5^2 x_2$$
$$+ x_5^2 x_1^3 x_3 + x_4^3 x_5^2 x_3 + x_2^3 x_1 x_5 x_4 + x_1 x_2 x_3^3 x_4$$
$$+ 2 x_1 x_2^2 x_3 x_5^2 + 3 x_1 x_2^2 x_3 x_5 x_4 + x_1^2 x_2^2 x_3^2$$
$$+ 2 x_1^2 x_2 x_3^2 x_4 + 2 x_1 x_2^2 x_3 x_4^2 + 2 x_1 x_2 x_3^2 x_4^2$$
$$+ x_1 x_2^3 x_3 x_4 + x_1^3 x_3^3 + 2 x_4^2 x_5 x_3^2 x_1$$
$$+ 2 x_4^2 x_5 x_2^2 x_3 + 2 x_4^2 x_5 x_1^2 x_3 + x_3^3 x_4^3$$
$$+ 3 x_5^2 x_4 x_1 x_2 x_3 + x_2^3 x_1^2 x_3 + x_3^3 x_5 x_1 x_2$$
$$+ 2 x_5^2 x_3^2 x_1 x_2 + x_2^3 x_5 x_1 x_3 + x_1^3 x_2^3 + x_1^3 x_4^3$$
$$+ x_2^3 x_4^3 + x_4^3 x_5^3 + x_3^3 x_2^3 + x_1^3 x_2^2 x_3 + x_3^3 x_2^2 x_1$$
$$+ x_4^2 x_5^3 x_1 + x_2^2 x_1^2 x_5^2 + x_1^3 x_2 x_5^2 + x_4^3 x_5 x_2^2$$
$$+ x_4^3 x_5 x_1^2 + x_4^3 x_5 x_3^2 + x_4^2 x_5^3 x_2 + x_4^2 x_5^2 x_1^2$$
$$+ x_4^2 x_5^2 x_2^2 + x_4^2 x_5^3 x_3 + x_4^2 x_5^2 x_3^2 + x_2^3 x_1 x_4^2$$
$$+ x_2^3 x_1 x_5^2 + x_1^3 x_2^2 x_5 + x_1^3 x_2^2 x_4 + x_1^3 x_5 x_4^2$$
$$+ x_1^3 x_3^2 x_2 + x_2^3 x_3^2 x_1 + x_3^3 x_1^2 x_2 + x_4^3 x_3 x_1 x_2$$

3 2 3 2 2 3 3 2

```
+ x3  x5  x4 + x3  x5 x4  + x3  x5  x2 + x5  x4 x2
    2   3        2   3        2   3        2   3
+ x2  x4  x1 + x2  x4  x3 + x1  x4  x2 + x1  x4  x3
    2   3        2   3        3      2     3
+ x3  x4  x2 + x3  x4  x1 + x5  x3 x2  + x2  x3 x5 x4
    3                2   2             2      2
+ x3  x1 x5 x4 + 2 x2  x3  x5 x4 + 2 x3  x2 x5  x4
    3              3              3
+ x5  x4 x2 x3 + x3  x2 x5 x4 + x5  x1 x2 x4
      2                   2      2          2      2
+ 3 x4  x5 x1 x2 x3 + 2 x2  x1 x4  x5 + 2 x1  x2 x4  x5
    3   3     3   3     3   3     3
+ x5  x1  + x3  x5  + x2  x5  + x1  x2 x3 x4
    3                2      2          2      2
+ x1  x2 x3 x5 + 2 x4  x5 x3  x2 + 2 x2  x1 x5  x4
    3                2   2             2         2
+ x1  x2 x5 x4 + 2 x1  x2  x5 x4 + 2 x1  x2 x3 x4
      2      2          2                   2   2
+ 2 x1  x2 x3  x5 + 3 x1  x2 x3 x5 x4 + 2 x4  x5  x1 x3
      2   2             2   2           2   2   2
+ 2 x4  x5  x2 x3 + 2 x4  x5  x1 x2 + x5  x1  x3
    3   2        3   2        3      2     2   2   2
+ x1  x3  x5 + x1  x3  x4 + x1  x3 x4  + x1  x3  x4
    2   3        3   2        3   2        3   2
+ x1  x3  x4 + x5  x1  x3 + x5  x1  x2 + x5  x1  x4
    2   3        2   3        2   3        2   2   2
+ x5  x1  x4 + x2  x5  x1 + x5  x3  x1 + x5  x3  x2
    3      2     3      2     2   3        3      2
+ x3  x5 x2  + x3  x5 x1  + x3  x5  x1 + x5  x4 x3
    3      2     3      2     3      2     3   2
+ x3  x2 x4  + x3  x2 x5  + x3  x1 x4  + x2  x3  x4
```

$$+ x_2^3 x_3^2 x_5 + x_2^3 x_3 x_4^2 + x_2^2 x_3^2 x_4^2 + x_2^2 x_3^3 x_4$$

$$+ x_2^3 x_3 x_5^2 + x_2^3 x_5^2 x_4 + x_2^3 x_5 x_4^2 + 2 x_5^2 x_1^2 x_2 x_3$$

Same difficulty as in the first computation ?

NOT SO, we have functions of only one set of variables, we can try to express them in different bases, for example Schubert polynomials (which contain as a sub-family Schur functions).

```
ToY0(%);

                                             3
Y[0, 0, 0, 3, 3] - z Y[0, 0, 1, 3, 3] - z  Y[1, 1, 1, 3, 3]
        2
     + z  Y[0, 1, 1, 3, 3]
```

RULE of indices CLEAR, even if we do not know what the symbols mean.

We could experiment this time
– in the case of two sets of variables, we could not !

Let us see now how to interpret the Euclidean algorithm in terms of 2×2 matrices.

The intermediate fractions (stopping at level k, instead of n) are rational functions N_k/D_k, which are called *convergents* of the continued fraction.

For example,

$$\frac{N_1}{D_1} = \frac{1}{q_1}, \quad \frac{N_2}{D_2} = \frac{q_2}{q_1 q_2 - 1}, \ \frac{N_3}{D_3} = \frac{q_2 q_3 - 1}{q_1 q_2 q_3 - q_1 - q_3} \ .$$

With these functions, the recursions are now

$$\begin{aligned} D_k &= q_k D_{k-1} - D_{k-2} \ , \\ N_k &= q_k N_{k-1} - N_{k-2} \ , \end{aligned}$$

with initial conditions $D_k = 0 = N_k$ if $k < 0$, $D_0 = 1$, $N_0 = 0$, $N_1 = 1$.

Instead of a system of equations, one can use 2×2 matrices:

$$\begin{bmatrix} N_k & N_{k-1} \\ D_k & D_{k-1} \end{bmatrix} = \begin{bmatrix} N_{k-1} & N_{k-2} \\ D_{k-1} & D_{k-2} \end{bmatrix} \begin{bmatrix} q_k & 1 \\ -1 & 0 \end{bmatrix},$$

and the preceding system is replaced by

$$\begin{bmatrix} N_k & N_{k-1} \\ D_k & D_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ q_1 & 1 \end{bmatrix} \begin{bmatrix} q_2 & 1 \\ -1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_k & 1 \\ -1 & 0 \end{bmatrix}.$$

with our linear factor q_i as entries.

But there is another matrix which also contains the information. It is a tridiagonal matrix, due to Jacobi.

We rewrite the equations between the denominators of the successive convergents :

$$x\, D_n = -\beta_n D_{n-1} + \zeta_n D_n + D_{n+1} \ .$$

On the left, we have "multiplication by x", the interpretation has changed ! One has the infinite space of polynomials in one variable x, with basis

$$\{D_0(x),\, D_1(x),\, D_2(x),\, \ldots\}$$

and the equations describe the images of the elements of this basis under multiplication by x.

The matrix describing the multiplication is

$$\mathfrak{Jac} := \begin{bmatrix} \zeta_0 & 1 & 0 & \cdots & 0 & \cdots \\ -\beta_1 & \zeta_1 & 1 & \ddots & 0 & \cdots \\ & \ddots & \ddots & \ddots & \vdots & \\ & & \ddots & \zeta_{n-2} & 1 & \ddots \\ & & & -\beta_{n-1} & \zeta_{n-1} & \cdots \\ & & & & \ddots & \ddots \end{bmatrix} .$$

The i-th row of this matrix, $i = 0, 1, 2, \ldots$ describes the decomposition of $xD_i(x)$ in the basis $\{D_0(x),\, D_1(x), \ldots\}$.

There are many things that one can do with a matrix, compared to a system of equations, for example, take powers. $\mathfrak{Jac}^n$ represents multiplication by x^n.

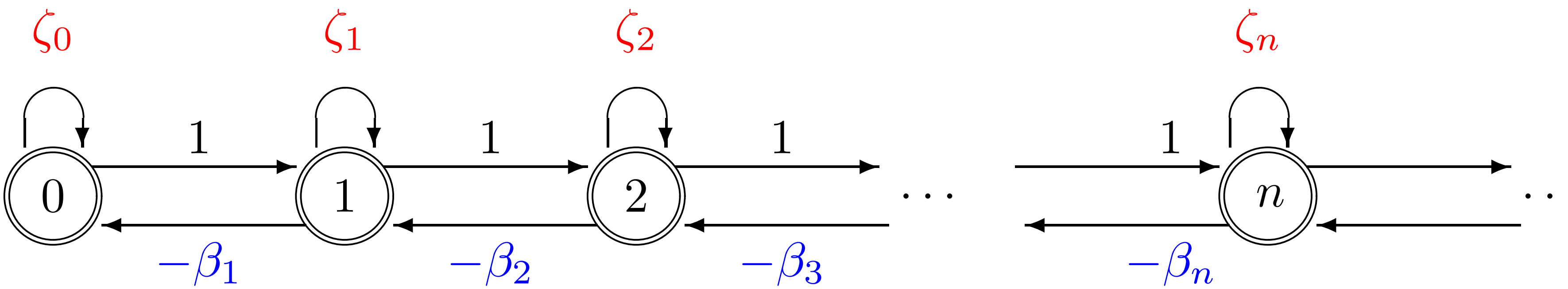
The formula for the elements of a product of two matrices M, N is

$$(MN)_{i,j} = \sum_k M_{ik} N_{kj}$$

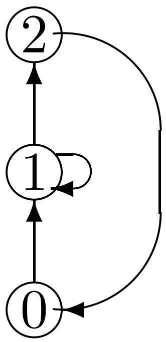
To go from i to j, one goes from i to k using M, then from k to j using N.

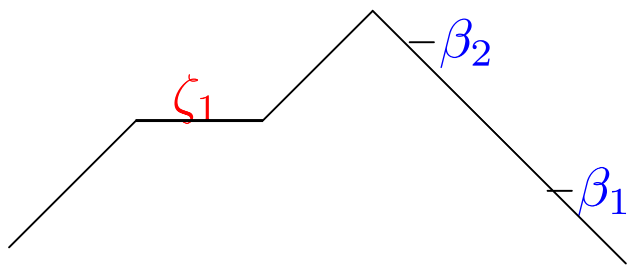
Thus, one can interpret powers of a matrix in terms of paths.

For Jacobi matrix, the paths are clear: loops corresponding to the diagonal, or going left or right by one step, with weight 1 or β_i :



But the path in the Jacobi automaton displayed on the left can be transformed to the picture on the right by adding an horizontal time coordinate :





Now, we have another combinatorial object, the Motzkin paths, with weights on the steps.

In fact, this construction allows to recover the original input (the coefficients of the series that we were dividing by 1), from the parameters ζ_i and β_j occuring in the Euclidean division.

In summary, the Euclidean algorithm is a process which, starting with two polynomials, or two series, produces a double sequence ζ_i, β_j. By introducing different combinatorial objects, and varying our point of view, we have almost avoided any computation, and related the algorithm to different classical topics of mathematics.