Multivariate Polynomial Division

BY

Alkis Akritas
ECE/UTh/Volos

Most symbolic computation involves mathematical expressions that contain more than one symbol or generalized variable. In order to manipulate these expressions, we must extend the concepts and algorithms described so far to polynomials with several variables. This generalization is the subject of these notes. Included is a description of coefficient domains and a recursive representation of multivariate polynomials (Section 1), recursive polynomial division (Section 2) and monomial-based polynomial division (Section 3).

1 Recursive Representation of Multivariate Polynomials

A multivariate polynomial u in the set of distinct symbols $\{x_1, x_2, \ldots, x_k\}$ is a finite sum with (one or more) monomial terms of the form

$$c \cdot x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k},$$

where the coefficient c is in a coefficient domain $\mathbb{K}$ and the exponents n_i are non-negative integers. The notation $\mathbb{K}[x_1,\ x_2,\ \ldots,\ x_k]$ represents the set of polynomials in the symbols $x_1, x_2, \ldots, x_k$ with coefficients in $\mathbb{K}$. For example, $\mathbb{Z}[x, y]$ represents all polynomials in x and y with coefficients that are integers.

A particularly important instance when $\mathbb{K}$ is not a field has to do with the *recursive representation* of multivariate polynomials. For example, let $\mathbb{Q}[x, y]$ be the polynomials in x and y with rational number coefficients. By collecting coefficients of powers of x, a polynomial $f(x, y)$ is represented as

$$f(x, y) = f_0(y) x^n + f_1(y) x^{n-1} + \cdots + f_n(y),$$

where the coefficients $f_i(y)$ are in $\mathbb{Q}[y]$. In this sense, $\mathbb{Q}[x, y]$ is equivalent to $\mathbb{K}[x]$, where $\mathbb{K} = \mathbb{Q}[y]$.

The above discussion suggests that a fruitful way to approach the study of polynomials in several variables is to consider polynomials in one variable $\mathbb{K}[x]$ where $\mathbb{K}$ may not satisfy all of the properties of a field. But what axioms should define $\mathbb{K}$? Let us take as a prototype the recursive view of polynomials in two variables with rational number coefficients $(\mathbb{Q}[y])[x] = \mathbb{K}[x]$. The coefficient domain is not a field since polynomials in $\mathbb{K} = \mathbb{Q}[y]$ with positive degree do not have inverses that are polynomials. $\mathbb{K}$ does, however, have all the properties of another algebraic system known as an *integral domain*.

Definition 1. *Let $\mathbb{K} = \{a, b, c, \ldots\}$ be a set of expressions and let $a+b$ and $a \cdot b$ be two operations defined for expressions a and b in $\mathbb{K}$. The set $\mathbb{K}$ is an* ***integral domain*** *if, it satisfies all the field axioms except that of the multiplicative inverse. The latter is replaced by the following axiom:*

IntDom: *If $a \cdot b = 0$, then either $a = 0$ or $b = 0$.*

A field is an integral domain, but the converse is not true. Both $\mathbb{Z}$ and $\mathbb{Q}[x]$ are integral domains although neither one is a field.

Although an integral domain may not contain inverses, the divisibility concept can still be defined.

Definition 2. *Let b and c be expressions in an integral domain $\mathbb{K}$.*

i. An expression $b \neq 0$ is a ***divisor*** *of (or* ***divides****) c if there is an expression d in $\mathbb{K}$ such that $b \cdot d = c$. We use the notation $b|c$ to indicate that b is a divisor of c and $b \nmid c$ if it does not. The expression d is called the* ***cofactor*** *of b in c and is represented by* ***cof***(b, c).

ii. A ***common divisor*** *of b and c is an expression $d \neq 0$ in $\mathbb{K}$ such that $d|b$ and $d|c$.*

Definition 3. *Let b, c, and d be expressions in an integral domain $\mathbb{K}$.*

i. An expression b is called a ***unit*** *if it has a multiplicative inverse.*

ii. Two expressions $c \neq 0$ and $d \neq 0$ are called ***associates*** *if $c = b \cdot d$, where b is a unit.*

In $\mathbb{Q}[x]$, the unit expressions are the non-zero polynomials of degree zero (i.e. the rational numbers), and whenever one polynomial f is a (non- zero) rational multiple of another polynomial g, the two are associates. In $\mathbb{Z}$, the only units are 1 and -1. At the other extreme, in a field all non-zero expressions are units, and any two non-zero expressions are associates.

Definition 4. *Two expressions b and c in an integral domain $\mathbb{K}$ are **relatively prime** if any common divisors of b and c is a unit.*

For example, in $\mathbb{Z}$ the integers 3 and 5 are relatively prime because the only common divisors are the units 1 and -1.

In the field $\mathbb{Q}$, *any two* rational numbers are relatively prime because all common divisors are units.

On the other hand, in the integral domain $\mathbb{Z}[x]$, the polynomials $2x+2$ and $2x-2$ are not relatively prime because 2, which is not a unit, divides both polynomials.

Definition 5. *An expression $b\neq 0$ in an integral domain $\mathbb{K}$ is **reducible** if there are non-unit expressions c and d such that $b=c\cdot d$. The expression b is **irreducible** if it is not reducible.*

Example 6. Consider the expression $f=2x+2$ as a member of the integral domain $\mathbb{Q}[x]$. In this context, f is irrreducible. However, when f is viewed as a member of the integral domain $\mathbb{Z}[x]$, f is reducible with the factorization $f=2(x+1)$.

Definition 7. *A **unique factorization domain** $\mathbb{K}$ is an integral domain that satisfies the following axiom:*

UFD: *Each $a\neq 0$ in $\mathbb{K}$ that is not a unit has a factorization in terms of non-unit, irrreducible expressions in $\mathbb{K}$*

$$a=a_1\cdot a_2\cdots a_k.$$

The factorization is unique up to the order of the factors and associates of the factors.

The set of integers $\mathbb{Z}$ is a unique factorization domain and so is the polynomial domain $\mathbb{F}[x]$, where $\mathbb{F}$ is a field.

The next theorem states that the integral domain property of the coefficient domain is inherited by the polynomial domain $\mathbb{K}[x]$.

Theorem 8. *If $\mathbb{K}$ is an integral domain, then $\mathbb{K}[x]$ is also an integral domain.*

By utilizing the recursive nature of multivariate polynomials, we extend the integral domain property to multivariate polynomials.

Theorem 9. *If* $\mathbb{K}$ *is an integral domain, then* $\mathbb{K}[x_1,x_2,\ldots,x_k]$ *is also an integral domain.*

2 Recursive Polynomial Division in $\mathbb{K}[x]$

Suppose f, g are polynomials with coefficients in a field. The division operation of f by g is defined using recurrence relations, which include the field computation

$$\frac{\mathrm{lc}(r_i, x)}{\mathrm{lc}(g, x)},$$

where lc(poly, x) is the leading coefficient of poly with respect to x. This operation causes a problem when the coefficient domain $\mathbb{K}$ is not a field because $\mathrm{lc}(g, x)^{-1}$ may not exist in $\mathbb{K}$.

For f, g in $\mathbb{K}[x]$, polynomial division is often used to determine if $g|f$. If this is the goal, one way to define the division process is to continue the iteration as long as $\mathrm{lc}(v, x)|\mathrm{lc}(r_i, x)$, that is, as long as $\mathrm{cof}(\mathrm{lc}(v, x), \mathrm{lc}(r_i, x))$ exists in $\mathbb{K}$.

The general division process is called ***recursive polynomial division*** because it depends on a division process in the coefficient domain that determines if $\mathrm{lc}(g, x)$ divides $\mathrm{lc}(r_i, x)$. For multivariate polynomials in $\mathbb{K}[x_1,x_2,\ldots,x_k]$, this means that division in terms of the main variable x_1 depends recursively on division of polynomials in $\mathbb{K}[x_2,\ldots,x_k]$.

Definition 10. *Let* f *and* $g \neq 0$ *be polynomials in* $\mathbb{K}[x]$. *The* ***recursive polynomial division*** *of* f *by* g *is defined by the following sequence of quotients and remainders:*

$$\begin{aligned}
q_0 &= 0\\
r_0 &= f\\
q_i &= q_{i-1} + \mathrm{cof}(\mathrm{lc}(g, x), \mathrm{lc}(r_{i-1}, x))\, x^{\deg(r_{i-1},x)-\deg(g,x)},\\
r_i &= r_{i-1} + \mathrm{cof}(\mathrm{lc}(g, x), \mathrm{lc}(r_{i-1}, x))\, x^{\deg(r_{i-1},x)-\deg(g,x)} g.
\end{aligned}$$

The iteration terminates when either

$$\deg(r_i, x) < \deg(g, x)$$

or

$$\mathrm{lc}(g, x) \nmid \mathrm{lc}(r_i, x).$$

If the process stops after $i = \sigma$ iterations, then q_σ and r_σ are, respectively, the ***recursive quotient*** *and* ***recursive remainder*** *on dividing f by g.*

Example 11. Consider the polynomials $f = x^2y^2 + x$ and $g = xy + 1$ in $\mathbb{Q}[x, y]$, with x the main variable. On dividing f by g we obtain

$$f = q_1\, g + r_1 = (x y)(x y + 1) + x(1 - y)$$

and the division process stops after one iteration because

$$\mathrm{lc}(g, x) = y \nmid (1 - y) = \mathrm{lc}(r_1, x).$$

Notice that $\deg(g, x) = \deg(r_1, x) = 1$, which means that the remainder does *not* satisfy the Euclidean property — which requires the degree of the remainder to be less than the degree of the divisor.

In `Sympy`, recursive division is performed by specifying the coefficient domain. So we have:

```
Python] from sympy import *
Python] x, y = var('x y')
Python] f = x**2 * y**2 + x
Python] g = x * y + 1
Python] quo( f, g, domain = QQ[y] )

    x*y

Python] rem( f, g, domain = QQ[y] )

    x*(-y + 1)
```

On the other hand, when y is the main variable, the process terminates after two iterations with

$$f = q_2\, g + r_2 = (x y - 1)(x y + 1) + x + 1.$$

Now we have $0 = \deg(r_2, y) < \deg(g, y) = 1$, and so in this case the remainder does satisfy the Euclidean property.

```
Python] quo( f, g, domain = QQ[x] )

    x*y - 1

Python] rem( f, g, domain = QQ[x] )
```

```
x + 1
```

If the division process stops because $\mathrm{lc}(g, x) \nmid \mathrm{lc}(r_i, x)$, then the representation $f = q_j\, g + r_j$ may not be unique. This point is illustrated in the next example.

Example 12. Consider $f = xy + x + y$ and $g = xy$ as polynomials in $\mathbb{Q}[x, y]$ with main variable x. Since

$$\mathrm{lc}(g, x) = y \nmid (y + 1) = \mathrm{lc}(f, x)$$

the process terminates with $j = 0$, $q_0 = 0$, and $r_0 = xy + x + y$. In this case, f has the following representations

$$\begin{aligned} f = q_0\, g + r_0 &= 0 \cdot g + (xy + x + y) \\ f = q_0\, g + r_0 &= 1 \cdot g + (x + y). \end{aligned}$$

3 Monomial-Based Division in $\mathbb{Q}[x_1, \ldots, x_k]$

Another approach to polynomial division is called monomial-based division. Like recursive division, monomial-based division can determine if $g \mid f$. However, because the process is based on the monomial structure of polynomials rather than the recursive structure, it may produce a different quotient and remainder when $g \nmid f$.

To simplify the presentation, we describe the process for the polynomial domain $\mathbb{Q}[x_1, \ldots, x_k]$. For the remainder of this section, we view polynomials in the form

$$f = f_1 + f_2 + \cdots + f_l,$$

where each f_i is a monomial of the form

$$f_i = c \cdot x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k}.$$

Let us begin with a simple example in which the divisor g is a monomial.

Example 13. Consider the polynomials f, g with x as the main variable:

$$f = 2x^2 y + 3x^2 + 4xy + 5x + 6y + 7, \; g = xy.$$

First, doing recursive polynomial division of f by g, we notice that

$$\mathrm{lc}(g, x) = y \nmid 2y + 3 = \mathrm{lc}(f, x),$$

and, hence, the process terminates immediately with $q_0 = 0, r_0 = f$. Indeed,

```
Python] quo(2*x**2 * y + 3*x**2 + 4*x*y + 5*x + 6*y + 7, x*y,
        domain = QQ[y])

    0

Python] rem(2*x**2 * y + 3*x**2 + 4*x*y + 5*x + 6*y + 7, x*y,
        domain = QQ[y])

    x**2*(2*y + 3) + x*(4*y + 5) + 6*y + 7
```

On the other hand, monomial-based division recognizes that the monomial $g = xy$ divides the first and third monomials in f and we obtain

$$f = qg + r = (2x + 4)(xy) + 3x^2 + 5x + 6y + 7.$$

In `Sympy` to perform monomial polynomial division use the function `reduced`, where the second argument can be a list of polynomials.

```
Python] reduced(2*x**2 * y + 3*x**2 + 4*x*y + 5*x + 6*y + 7,
        [x*y])
```

Notice that g does not divide any of the monomials in the remainder r.

The above example leads us to the following definition — at least when the divisor is a monomial:

Definition 14. *Let $f = f_1 + f_2 + \cdots + f_l$ be a polynomial and let $g \neq 0$ be a* ***monomial*** *in $\mathbb{K}[x]$. Dividing f by g gives*

$$f = q(f, g) \cdot g + r(f, g)$$

where

$$Q(f, g) = \sum_{1 \leqslant i \leqslant l, g \mid f_i} \frac{f_i}{g} \tag{1}$$

and

$$r(f, g) = \sum_{1 \leqslant i \leqslant l, g \nmid f_i} f_i. \tag{2}$$

The formula of the sum for the quotient, above, includes all monomials f_i from f such that $g|f_i$, while the formula of the sum for the remainder includes all monomials f_i from f such that $g\nmid f_i$.

3.1 Lexicographical Ordering of Monomials

To complete the definition of the new division process, we must show how it is defined when g is a sum of monomials. The approach we use is to repeatedly apply a process similar to that in Equations (1) and (2), but with g replaced by the leading monomial of g which is defined by the order relation in the next definition.

Definition 15. *Let f and g be monomials in $\mathbb{Q}[x_1, x_2, \ldots, x_k]$.*

- *i. f is less than g in lexicographical order with respect to the list $L = [x_1, x_2, \ldots, x_k]$ if one of the following conditions is true:*
  - *a) $\deg(f, x_1) < \deg(g, x_1)$, or*
  - *b) for some j, $1 < j \leqslant k$, $\deg(f, x_i) = \deg(g, x_i)$ for $i = 1, 2, \ldots, j-1$ and $\deg(f, x_j) < \deg(g, x_j)$.*

  The condition that f is less than g in lexicographical order is denoted by $f \prec g$.
- *ii. The monomials f and g are called **equivalent** monomials if they have the same term (or variable part).[1] The condition that f is equivalent to g is represented by $f \equiv g$.*
- *iii. The condition that $f \prec g$ or $f \equiv g$ is denoted by $f \preceq g$.*

Example 16. For $L = [x, y, z]$,

$$\begin{aligned} x^2y^3z^4 &\prec x^2y^4z^3 \\ xy^3z &\prec x^2y \\ yz^5 &\prec x \\ 2xy &\equiv 3xy \end{aligned}$$

---

[1] **Term** or **variable part** is the monomial without its numerical coefficient. The terminology has not been standardized yet.

Lexicographical order depends on the order of the symbols in the list L. In the previous example, when the order of the symbols is changed to $L=[z, y, x]$, the lexicographical order of the first three examples above is reversed.

The designation of this order relation as "lexicographical" suggests that it is similar to an "alphabetical" order. Indeed, the most significant factor in the order relation is the main variable x_1 . It is only when f and g have the same degree in this variable that the next variable x_2 is significant. If the degree of f and g with respect to x_2 are also the same, then x_3 is significant and so forth.

The leading monomial of a polynomial is defined in terms of the lexicographical order of its monomials.

Definition 17. *Let f be in $\mathbb{Q}[x_1, x_2, ..., x_k]$, and let $L=[x_1, x_2, ..., x_k]$. If f is a sum of monomials, then the* ***leading monomial*** *is the monomial of f that is greatest in the lexicographical order. If f is a monomial, then the leading monomial is just f itself. The operator notation*

$$\operatorname{lm}(f, L)$$

denotes the leading monomial of f. The inclusion of the list L in this notation emphasizes that the leading monomial is defined with respect to the order of the symbols in L.

Example 18. If $f=3x^2y+4xy^2+y^3+x+1$, then

$$\begin{aligned} \operatorname{lm}(f, [x, y]) &= 3x^2y \\ \operatorname{lm}(f, [y, x]) &= y^3 \end{aligned}$$

In `Sympy` we have the function `LM` which returns the variable part of the leading monomial and the function `LT` which returns the leading monomial.

```
Python] LM( 3*x**2 * y + 4*x*y**2 + y**3  + x + 1, [x, y] )

    x**2*y

Python] LM( 3*x**2 * y + 4*x*y**2 + y**3  + x + 1, [y, x] )

    y**3

Python] LT( 3*x**2 * y + 4*x*y**2 + y**3  + x + 1, [x, y] )

    3*x**2*y

Python] LT( 3*x**2 * y + 4*x*y**2 + y**3  + x + 1, [y, x] )

    y**3
```

By contrast, in `Xcas/Giac` the function `lcoeff` does the same job as `LT`.

```
> lcoeff( 3*x**2 * y + 4*x*y**2 + y**3  + x + 1, [x, y] )
```

$$3\cdot x^2\cdot y$$

```
> lcoeff( 3*x**2 * y + 4*x*y**2 + y**3  + x + 1, [y, x] )
```

$$y^3$$

3.2 Monomial-Based Division Algorithm

Definition 19. *Let f and $g\neq 0$ be polynomials in $\mathbb{Q}[x_1,x_2,\ldots,x_k]$, and let $L=[x_1, x_2,\ldots,x_k]$. Suppose that $g_l=\operatorname{lm}(g,L)$, and define the iteration scheme*

$$\begin{aligned} q_0 &= 0\\ r_0 &= f\\ \text{and for } i\geqslant 1 &\\ f_i &= Q(r_{i-1},g_l)\\ q_i &= q_{i-1}+f_i\\ r_i &= r_{i-1}-f_i\,g, \end{aligned}$$

where the function Q is defined in Equation (1)*. The iteration terminates when $Q(r_i\,,g_l\,)=0$. If $i=\tau$ is the first such index, then q_τ is the **monomial-based quotient** of f divided by g, and r_τ is the **monomial-based remainder**.*

Example 20. Let $f=x^3+3x^2y+4xy^2$ and $g=xy+2y+3y^2$, and let $L=[x,y]$ define the variable order. Then, we start with

$$q_0=0, r_0=f, g_l=xy, g_r=3y^2+2y,$$

and after the first iteration we have

$$f_1=Q(r_0,g_l)=3x+4y, q_1=3x+4y,$$

and

$$\begin{aligned} r_1 &= (f-f_1\,g_l)+(-f_1\,g_r)\\ &= (f-(3x^2y+4xy^2))+(-9xy^2-6xy-8y^2-12y^3)\\ &= x^3-9xy^2-6xy-8y^2-12y^3. \end{aligned}$$

The next iteration gives

$$f_2 = Q(r_1, g_l) = -9y - 6$$

and

$$q_2 = 3x - 5y - 6, r_2 = x^3 + 28y^2 + 15y^3 + 12y.$$

Since g_l does not divide any monomial of r_2, $Q(r_2, g_l) = 0$ and the iteration terminates with $\tau = 0$; the monomial-based quotient and remainder are q_2, r_2 above.

Indeed, performing monomial-based division with `Sympy` we verify that our computations are correct:

```
Python] f = x**3 + 3*x**2*y + 4*x*y**2
Python] g = x*y + 2*y + 3*y**2
Python] reduced( f, [g], [x, y] )
    ([3*x - 5*y - 6], x**3 + 15*y**3 + 28*y**2 + 12*y)
```

On the other hand, recursive polynomial division terminates after $\sigma = 0$ iterations with $q_\sigma = 0$ and $r_\sigma = f$.

```
Python] quo( f, g, domain = QQ[y] )
    0
Python] rem( f, g, domain = QQ[y] )
    x**3 + 3*x**2*y + 4*x*y**2
```

However, if the variable order is changed to $[y, x]$, then both monomial-based division and recursive division give the same result:

```
Python] reduced( f, [g], domain = QQ[x] )
    ([4*x/3], x**3 + y*(5*x**2/3 - 8*x/3))
Python] quo( f, g, domain = QQ[x] )
    4*x/3
Python] rem( f, g, domain = QQ[x] )
    x**3 + y*(5*x**2/3 - 8*x/3)
```