Polynomials

Application of discrete models

1 Basics

Definition 1.1. A mathematical expression is called a *polynomial* if it contains only the following:

- constants (e.g. numbers),
- a variable (e.g. x),
- addition, subtraction, multiplication,
- exponentiation to an exponent in $\mathbb{N}$,
- parentheses.

For example, these are polynomials: $3x^5 - 5x^3 + x + 1/2$, $x^2(x+3)^5$ and 15, but these are not: $1/x$ (it contains division), x^{-5} (the exponent is not in $\mathbb{N}$) and $\sqrt{x}$ (square root is not allowed).

(Note: this definition can be generalized to allow more than one variables, e.g. then $x^3 + xy^2 + y$ is a bivariate polynomial, i.e. a polynomial with two variables. Then, the above definition defines *univariate polynomials*, i.e. polynomials with one variable. But in this subject, we only deal with the latters, so we omit the qualifier "univariate".)

Definition 1.2. Let $K[x]$ be the set of polynomials with constants from the set K and with the variable x.

For example, $\mathbb{R}[x]$ is the set of polynomials with real numbers, and $\mathbb{Z}[x]$ is the set of polynomials with integers (and of course $\mathbb{Z}[x] \subseteq \mathbb{R}[x]$). (Later we will define what K can be exactly, but until that, let K be a number set, e.g. $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ etc.)

A polynomial can be written in multiple forms, e.g. $(2x+1)^2 = 4x^2+4x+1 = x^5+1+x-x^5+4x^2+3x$. If we need a unique representation, then we can use a special form:

Definition 1.3. The *canonical form* of a polynomial $p \in K[x]$ is $p = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$, where all $a_k \in K$ and $a_n \neq 0$.

For example, the canonical form of $p = (2x+1)^2$ is $p = 4x^2+4x+1$. A polynomial can be transformed into canonical form by expanding the brackets, collecting the x's in each term into a single power (e.g. $2x \cdot 2x = 4x^2$), combining terms with equal exponents, and writing the terms in descending powers of x.

Definition 1.4. In the above canonical form,

- a_k's are the *coefficients* of the polynomial (more specifically, a_k is the *coefficient of degree* k),
- n is the *degree* of the polynomial – notation: $\deg p$ –,
- a_n is the *leading coefficient* of the polynomial and
- a_0 is the *constant term* of the polynomial.

For example, the coefficients of $p = 2x^3 - 5x + 4$ are 2, 0, -5 and 4 (note that $0x^2$ was omitted from the canonical form), its degree is 3, its leading coefficient is 2 and its constant term is 4.

But there is a polynomial which cannot be written in canonical form yet (by the above definition): the 0 (because we required that $a_n \neq 0$). So we need to handle it separately:

Definition 1.5. $0 \in K[x]$ is called the *zero polynomial*, its canonical form is 0, and its degree is $-\infty$. (Note: it is $-\infty$ to make it smaller than for any other polynomial. It cannot be defined 0, because that would break many statements about the degree of polynomials, such as Theorem 1.7 below. Some sources leave the degree of the zero polynomial undefined, and e.g. Sage defines it to be -1.)

Other special cases:

Definition 1.6.

- $p \in K[x]$ is a *constant polynomial* if $\deg p \leq 0$.
- $p \in K[x]$ is a *linear polynomial* if $\deg p \leq 1$.

For example, 2 and 0 are constant polynomials (they cannot have x in them), and $2x - 3$, x, 2 and 0 are all linear polynomials.

Arithmetic operations on polynomials affect their degree as follows:

Theorem 1.7. *If $p, q \in \mathbb{R}[x]$, then:*

1. $\deg(p + q) \leq \max(\deg p,\ \deg q)$ *(with equality if $\deg p \neq \deg q$);*
2. $\deg(p - q) \leq \max(\deg p,\ \deg q)$ *(with equality if $\deg p \neq \deg q$);*
3. $\deg(p \cdot q) = \deg p + \deg q$.

For example, if $p = 2x - 1$ and $q = x^2 + 2$, then $\deg p = 1$ and $\deg q = 2$, and indeed: $\deg(p + q) = \deg(x^2 + 2x + 1) = 2 = \max(1, 2)$ and $\deg(p \cdot q) = \deg(2x^3 - x^2 + 4x - 2) = 3 = 1 + 2$.

(Note: these rules also work for the zero polynomial, e.g. $0 \cdot x^2 = 0$, and indeed, $-\infty + 2 = -\infty$.)

2 Polynomials in Sage

Sage can calculate with polynomials like with any other symbolic expression:

```
sage: (2*x - 1)*(x^2 + 2)
(x^2 + 2)*(2*x - 1)
sage: expand(_)
2*x^3 - x^2 + 4*x - 2
sage: type(_)
<class 'sage.symbolic.expression.Expression'>
```

(Reminder: `x` is a built-in symbolic variable, as if created by `var("x")`, see the lecture notes: SageMath introduction.)

But this is slightly cumbersome (e.g. it requires `expand()`), and not very efficient for polynomials. Sage has structures optimized specifically for polynomials: `ZZ["x"]`, `QQ["x"]` etc., corresponding to the mathematical sets $\mathbb{Z}[x]$, $\mathbb{Q}[x]$ etc.:

```
sage: P = ZZ["x"]
sage: p = P(2*x - 1)    # convert a symbolic expression to a polynomial
sage: p
2*x - 1
sage: q = P([2, 0, 1])  # create from coefficients (starting from the constant)
sage: q
x^2 + 2
```

```
sage: p * q             # no need for expand() (and very fast)
2*x^3 - x^2 + 4*x - 2
sage: type(_)
<class 'sage.rings.polynomial.[...]'>
```

Beware that `x` has two different meanings here: in the entered commands, `x` is still a symbolic variable (type: `Expression`), but in the printed results, `x` is the variable of the polynomials (type: polynomial). Don't mix them:

```
sage: p*(x^2 + 2)     # DON'T do this! (polynomial * symbolic expression)
(x^2 + 2)*(2*x - 1)
sage: type(_)         # we have lost the advantages of polynomials:
<class 'sage.symbolic.expression.Expression'>
```

(Note: as a slightly simplified explanation, Sage represents a symbolic expression as a sequence of operations, e.g. $x^2 + 2$ is represented as "square x, then add 2". This is very general, but not very efficient to calculate with. Polynomials on the other hand are represented using a sequence of coefficients, e.g. x^2+2 is represented as $2, 0, 1$. This makes much more efficient calculations possible.)

To avoid confusing symbolic expressions with polynomials, we can define `x` to be a polynomial too:

```
sage: P = ZZ["x"]    # let P be the set of integer polynomials
sage: x = P.gen()    # let x be the variable of the polynomials in P
```

There is also a simplified syntax that defines both `P` and `x`:

```
sage: P.<x> = ZZ[]   # same as: P = ZZ["x"]; x = P.gen()
sage: P
Univariate Polynomial Ring in x over Integer Ring
sage: type(x)              # x is also redefined
<class 'sage.rings.polynomial.[...]'>
sage: (2*x - 1)*(x^2 + 2)  # now these are polynomials too
2*x^3 - x^2 + 4*x - 2
sage: var("x")             # reset x
sage: type(x)
<class 'sage.symbolic.expression.Expression'>
```

We can query the properties of polynomials:

```
sage: P.<x> = ZZ[]
sage: p = 2*x^3 - 5*x + 4
sage: p.degree()
3
sage: p.list()      # list of coefficients (starting from the constant term)
[4, -5, 0, 2]
sage: p[1]          # coefficient of degree 1
-5
sage: p.leading_coefficient()   # or simply: p.lc()
2
sage: p.constant_coefficient()  # or simply: p[0]
4
```

3 Evaluation

Definition 3.1. The *value* of the polynomial $p \in K[x]$ at c is obtained by replacing all occurences of x in p by the value c. Notation: $p(c)$. This computation is called the *evaluation* of p at c.
For example, if $p = x^2 - 3x + 5$ and $c = 2$, then $p(2) = 2^2 - 3 \cdot 2 + 5 = 3$.
Note: it is not required that $c \in K$, e.g. we can evaluate the same $p = x^2 - 3x + 5 \in \mathbb{Z}[x]$ at $c = 1/2 \notin \mathbb{Z}$ too: $p(1/2) = (1/2)^2 - 3 \cdot (1/2) + 5 = 15/4$.
The most naive way of evaluating a polynomial is to use its canonical form directly, i.e. $p(c) = a_n c^n + \ldots + a_1 c + a_0$. But this is not efficient, as it needs n multiplications for $a_n c^n$, n–1 multiplications for $a_{n-1}c^{n-1}$ etc., which is $n+(n-1)+\ldots+2+1 = n(n+1)/2$ multiplications in total, plus n additions. (For example, if $\deg p = 10$, then it needs 55 multiplications and 10 additions.)
This can be improved by going from right to left, and using the previous value of c^k to calculate the next c^{k+1} with only one extra multiplication: $c^{k+1} = c^k \cdot c$. This methods needs only $2n-1$ multiplications and n additions (which for $\deg p = 10$ is 19 multiplications and 10 additions).
But an even more efficient way is *Horner's method* (or *Horner's scheme*), which turns the polynomial into a form that allows very efficient evaluation. For example:

$$\begin{aligned} p &= 2x^4 + 3x^3 - 2x^2 - 4x + 3 = \\ &= (2x^3 + 3x^2 - 2x - 4)x + 3 = \\ &= ((2x^2 + 3x - 2)x - 4)x + 3 = \\ &= (((2x + 3)x - 2)x - 4)x + 3. \end{aligned}$$

In each step, we extract x from all the terms we can, and continue this process until all the exponents disappear. In general, this means the following transformation:

$$\begin{aligned} p &= a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0 = \\ &= ((\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_2)x + a_1)x + a_0. \end{aligned}$$

This form makes evaluation very efficient: in each step, we multiply a parenthesized subexpression (first a_n) by x (or rather c), then add the next coefficient to get the next parenthesized subexpression, and finally the result. This process requires only n multiplications and n additions.
Horner's method can be written as an algorithm as follows:

1. Input: the coefficients $a_0, a_1, \ldots, a_n$ and the value c.
2. $b := a_n$
3. Loop from $k = n-1$ to 0:
   - $b := b \cdot c + a_k$
4. The result: b

In each step, the value of b equals to one of the parenthesized subexpressions (e.g. after the first step, $b = a_n c + a_{n-1}$, then $b = (a_n c + a_{n-1})c + a_{n-2}$ etc.).
(Note: this algorithm doesn't work directly for the zero polynomial, because it doesn't have any coefficients. But this can be fixed easily by giving a single zero coefficient: $n = 0$ and $a_0 = 0$. In general, the algorithm doesn't require that $a_n \neq 0$, so there might be redundant 0's at the end of the coefficient list.)
In Sage, polynomials can be evaluated using an expression like `p(2)`, where `p` is a Sage polynomial. (Note: Sage also uses Horner's method for this.)

4 Roots

Definition 4.1. The value c is a *root* of the polynomial $p \in K[x]$ if $p(c) = 0$.
For example, $c = 1$ is a root of $p = 2x^2 - 3x + 1$, because $p(1) = 2 \cdot 1^2 - 3 \cdot 1 + 1 = 0$.
Just like for evaluation in general, it is not required that $c \in K$, e.g. the above p (which is $p \in \mathbb{Z}[x]$) also has the root $c = 1/2 \notin \mathbb{Z}$: $p(1/2) = 2 \cdot (1/2)^2 - 3 \cdot (1/2) + 1 = 0$. For another example: $q = x^2 + 1 \in \mathbb{R}[x]$, but it has no real roots, only complex: $x_1 = i$ and $x_2 = -i$.
Theorem 4.2. *$c \in K$ is a root of $p \in K[x]$ if and only if $\exists q \in K[x] : p = (x - c)\, q$.*
For example, $c = 1$ is a root of the above $p = 2x^2 - 3x + 1$, and indeed: $p = (x - 1)(2x - 1)$.
Definition 4.3. The polynomial $x - c \in K[x]$ is called the *root factor* corresponding to c.
So the above theorem says that c is a root of p if and only if the root factor corresponding to c can be extracted from p.
It can happen that from $p = (x - c)\, q$, the polynomial q also has the root c. Then, the same root factor can be extracted from q too: $q = (x - c)\, r$, i.e. $p = (x - c)^2\, r$, which means that $x - c$ can be extracted from p more than once. Then we say that c is a *multiple root* of p, more precisely:
Definition 4.4. $c \in K$ is a *root of multiplicity k* of the polynomial $p \in K[x]$ if $\exists q \in K[x]$ such that $p = (x - c)^k\, q$, but c is not a root of q. Terminology: if $k = 1$, then c is a *simple root*, if $k = 2$, it is a *double root* etc., and if $k \geq 2$, then c is a *multiple root* of p.
So c is a root of multiplicity k if $x - c$ can be extracted exactly k times. For example, $c = 1$ is a double root of $p = 2x^3 - 3x^2 + 1$, because $p = (x - 1)^2\, (2x + 1)$, but 1 is not a root of $2x + 1$.
The following theorem follows from the root factor extraction:
Theorem 4.5. *If K is a number set, then a polynomial $p \in K[x] \setminus \{0\}$ of degree n can have at most n roots in K. This is true even if the roots are counted with multiplicities, i.e. the sum of the multiplicities of the roots is also at most n.*
For example, a cubic polynomial (a polynomial of degree 3) can have at most 3 roots. But if it has a double root (e.g. $c = 1$ for the above p), then it can only have one more root, because the double root "counts twice".
(Proof: extract as many root factors from p as possible: $p = (x - c_1) \cdots (x - c_m)\, q$, where q has no more roots. Then p can have no roots other than $c_1, \ldots, c_m$, because if $p(c) = 0$, then one of the factors must be 0, i.e. either some $c_i = c$, or q would not be rootless. But every extraction decreases the degree by one, so there can be at most $n = \deg p$ root factors.)
(Note: the condition "if K is a number set" will be explained and generalized in a later part.)
Some special cases:

- A polynomial of degree one, i.e. $p = ax + b$ can have only one root (a simple root): $x = -b/a$ (but e.g. in $\mathbb{Z}$, this division may not be possible).
- A polynomial of degree zero (i.e. nonzero constant polynomial) has no roots.
- But for the zero polynomial, all $c \in K$ are roots (this is why it had to be excluded from the above theorem).

Earlier, we defined the canonical form of polynomials. Using root factors, we can define another important form:
Definition 4.6. The *root factor form* of the polynomial $p \in K[x]$ is $p = a(x - c_1)(x - c_2) \cdots (x - c_m)$, where $a, c_1, \ldots, c_m \in K$ and $a \neq 0$.

For example, the polynomial $p = 2x^2 - 8x + 6$ (which is in canonical form) can be written in root factor form as $p = 2(x-1)(x-3)$. This form shows all the roots of the polynomial, along with their multiplicities. (And note that the constant factor (a, in this case 2) is the leading coefficient of the polynomial.)

But the root factor form doesn't always exist, because it requires the polynomial to have exactly n roots in the base set K (counted with multiplicities), where n is its degree, and the above theorem only guarantees that it has *at most* n roots. Furthermore, the number of roots depends on the base set K. For example, the polynomial $p = 16x^4 - 1$ has no roots in $\mathbb{Z}$, two simple roots in $\mathbb{R}$: $x_1 = 1/2$ and $x_2 = -1/2$, and four roots in $\mathbb{C}$: $x_1 = 1/2$, $x_2 = -1/2$, $x_3 = i/2$, $x_4 = -i/2$; therefore, it has a root factor form only in $\mathbb{C}$: $p = 16(x - 1/2)(x + 1/2)(x - i/2)(x + i/2)$.

The following theorem says that the set of complex numbers ($\mathbb{C}$) is very special in this regard:

Theorem 4.7 (Fundamental theorem of algebra)**.** *Every polynomial $p \in \mathbb{C}[x] \setminus \{0\}$ of degree n has exactly n roots in $\mathbb{C}$, counted with multiplicities.*

This guarantees that *any* (nonzero) polynomial over $\mathbb{C}$ has a root factor form.

In Sage, we can find the roots of a polynomial using the member function `roots()`:

```
sage: P.<x> = ZZ[]
sage: p = 16*x^4 - 1
sage: p.roots()
[]
sage: p.roots(QQ)          # giving a larger base set
[(1/2, 1), (-1/2, 1)]      # the 1's are the multiplicities
sage: p.roots(QQ, False)   # but they can be turned off
[1/2, -1/2]
sage: p.roots(CC)          # RR and CC only allows numerical calculation
[(-0.5000000000, 1), (0.5000000000, 1), (-0.5000000000*I, 1), (0.5000000000*I, 1)]
```

5 Division with remainder

Just like for integers, we cannot generally divide polynomials and get another polynomial (e.g. $x^2/(x+1)$ is not a polynomial, in the same way as $2/3 \notin \mathbb{Z}$). But just like for integers, we can do *division with remainder* for polynomials. For integers, it is based on the following theorem:

Theorem 5.1 (Division theorem)**.** *If $a, b \in \mathbb{Z}$ and $b \neq 0$, then there are unique $q, r \in \mathbb{Z}$ for which $a = qb + r$ and $0 \leq r < |b|$.*

Here q is the quotient, and r is the remainder. For example, if $a = 36$ and $b = 10$, then $q = 3$ and $r = 6$, because $36 = 3 \cdot 10 + 6$ and $0 \leq 6 < 10$.

A very similar theorem exists for polynomials:

Theorem 5.2 (Division theorem for polynomials)**.** *If $f, g \in \mathbb{R}[x]$ and $g \neq 0$, then there are unique $q, r \in \mathbb{R}[x]$ for which $f = qg + r$ and $\deg r < \deg g$.*

For example, if $f = 2x^3 + x^2 - 2x - 1$ and $g = x^2 - 2x + 1$, then $q = 2x + 5$ and $r = 6x - 6$, because $f = 2x^3 + x^2 - 2x - 1 = (2x + 5)(x^2 - 2x + 1) + (6x - 6)$ and $\deg(6x - 6) < \deg(x^2 - 2x + 1)$.

Definition 5.3. In the above theorem, q is the *quotient polynomial* of f and g, and r is their *remainder polynomial*.

Division with remainder can be performed manually very similarly to integer long division, but instead of going by the positions of the digits, one goes by the degrees of the terms. Starting from f, multiples of g are subtracted from it to cancel its leading coefficient, thus decreasing the degree of the polynomial step by step. This is repeated until the degree goes below $\deg g$, in which case the result is the remainder. For the above example ($f = 2x^3 + x^2 - 2x - 1$ and $g = x^2 - 2x + 1$):

$$\begin{aligned} f_0 &:= f &&= 2x^3 + x^2 - 2x - 1, \\ f_1 &:= f_0 - 2x \cdot g &&= 5x^2 - 4x - 1, \\ f_2 &:= f_1 - 5 \cdot g &&= 6x - 6, \end{aligned}$$

where $\deg f_2 < \deg g$, so the remainder is $r := f_2 = 6x - 6$, and the quotient can be assembled from the multipliers of g: $q = 2x + 5$.

In Sage, polynomial division works with the same syntax as for integers: `f // g`, `f % g` etc. (But regular division, `f / g` should be avoided, because its result type is not a Sage polynomial, even if there is no remainder.)

An interesting special case is when the divisor is of the form $g = x - c$, i.e. it is a root factor. Then, since it is a degree one polynomial, the remainder is a constant polynomial ($\deg r < \deg g = 1$). If c is a root of f, then $f = (x - c)\, q$, i.e. the remainder is 0. But if it is not, then the remainder is $f(c)$, i.e. the value of f at c. (This can be proven easily by substituting c into $f = (x - c)\, q + r$.) Furthermore, if $f(c)$ is calculated using Horner's method (see earlier above), then we automatically get the quotient polynomial q too:

Theorem 5.4. *If $f \in \mathbb{R}[x]$ and $c \in \mathbb{R}$, then the remainder polynomial of $f = a_n x^n + \ldots + a_1 x + a_0$ and $g := x - c$ is the constant polynomial $f(c)$, and their quotient polynomial is $q = b_{n-1} x^{n-1} + \ldots + b_1 x + b_0$, where the coefficients b_k are defined by the following recursion: $b_{n-1} := a_n$ and $b_k := b_{k+1} c + a_{k+1}$.*

Note that these b_k's are the intermediate values of b in Horner's method, as described in its algorithm above. For example, if $f = 2x^3 - 3x^2 - 4x + 7$ and $c = 2$, then, according to Horner's scheme, $f(c) = ((2c - 3)c - 4)c + 7$, i.e. $b_2 = 2$ (the leading coefficient), $b_1 = b_2 c - 3 = 1$, $b_0 = b_1 c - 4 = -2$ and $b_{-1} = b_0 c + 7 = 3$, so $q = b_2 x^2 + b_1 x + b_0 = 2x^2 + x - 2$ and $r = f(c) = b_{-1} = 3$.

Horner's method is often written in table form, where the first row contains the coefficients of f, and the b_k's are calculated in the second row:

	2	-3	-4	7	$\longleftarrow$ $f = 2x^3 - 3x^2 - 4x + 7$
$c = 2$	2	1	-2	3	$\longleftarrow$ $q = 2x^2 + x - 2$, $f(c) = 3$

6 Greatest common divisor

Besides division with remainder, many other integer operations can also be applied to polynomials. For example:

Definition 6.1. $p \in K[x]$ is a *divisor* of $q \in K[x]$ if $\exists r \in K[x] : q = pr$. Notation: $p \mid q$.

For example, $p = x - 1$ is a divisor of $q = 2x^2 - x - 1$, because $q = (x - 1)(2x + 1) = p\,(2x + 1)$.

Why is divisibility interesting for polynomials? Because it has a strong connection with the roots:

Theorem 6.2. *If $p \mid q$ and c is a root of p, then c is also a root of q, and its multiplicity in q is at least as big as its multiplicity in p.*

For example, 1 is a root of the above $p = x - 1$, and indeed, it is also a root of q: $q(1) = 2 - 1 - 1 = 0$.

(Proof: if $x-c$ can be extracted from p, then also from q, because $q = pr$, and at least as many times. Note: "$x-c$ can be extracted from p" can now be written as $x-c \mid p$.)
Divisibility can be checked using division with remainder (just like for integers):
Theorem 6.3. *$p \in \mathbb{R}[x] \setminus \{0\}$ is a divisor of $q \in \mathbb{R}[x]$ if and only if the remainder of q by p is 0.*
(Note: the definition allows the zero polynomial to be a divisor of itself, even though we can't *divide* by it – just like the 0 for integers. This is why we had to exclude it from the theorem: $\mathbb{R}[x] \setminus \{0\}$.)
After divisors, we can also define the following, in exactly the same way as for integers:
Definition 6.4. A *greatest common divisor* of $p \in K[x]$ and $q \in K[x]$ is a polynomial $r \in K[x]$ for which $r \mid p \ \wedge \ r \mid q \ \wedge \ \forall s \in K[x] : s \mid p \ \wedge \ s \mid q \implies s \mid r$. Notation: $\gcd(p, q)$. Furthermore, p and q are *coprime* if their greatest common divisor is 1.
For example, a greatest common divisor of $p = x^2 - 2x + 1$ and $q = 2x^2 - x - 1$ is $r = x - 1$, which can also be seen from the forms $p = (x-1)^2$ and $q = (x-1)(2x+1)$.
But there is a problem: for polynomials, the greatest common divisor is not unique! For example, for the above p and q in $\mathbb{R}[x]$, the following polynomials all qualify as their greatest common divisors: $r_1 = x - 1$, $r_2 = -x + 1$, $r_3 = 2x - 2$, $r_4 = -x/2 + 1/2$ etc. (But in $\mathbb{Z}[x]$, only r_1 and r_2 do, because there $r_3 \nmid p$.) But these polynomials are related:
Theorem 6.5. *If $r_1 \in K[x]$ and $r_2 \in K[x]$ are both greatest common divisors of $p \in K[x]$ and $q \in K[x]$, then $\exists c \in K \setminus \{0\} : r_1 = c \cdot r_2$.*
So the greatest common divisors are just constant multiples of each other. (In the above example, $r_1 = -r_2 = 1/2 \cdot r_3 = -2r_4$.) Therefore, having multiple gcd's are not really that big of a problem, because they are "essentially" the same (e.g. their roots are the same). If we need uniqueness, we can e.g. require that the leading coefficient of the gcd must be 1. (If it's possible, see e.g. $\mathbb{Z}[x]$.)
Why is polynomial gcd interesting? Because it tells something important about the roots:
Theorem 6.6. *If $r \in K[x]$ is a greatest common divisor of $p \in K[x]$ and $q \in K[x]$, then c is a root of r if and only if it is a root of both p and q, and its multiplicity in r is the smaller of the multiplicities in p and q.*
For example, the above p has only one root, 1 (but it is a double root); q has two roots, 1 and $-1/2$ (simple roots); therefore $r = \gcd(p, q)$ (or any other r_i) has the only root 1 (as a simple root).
So if we need to find the common roots of two polynomials, we only need to calculate the roots of their gcd. This is easier, because the latter has usually much smaller degree; and if it's 1, i.e. the two polynomials are coprime, then we know immediately that they cannot have common roots.
But how do we calculate the gcd of two polynomials? In the same way as for integers: using the Euclidean algorithm (see the lecture notes: Greatest common divisor), which can also be applied for polynomials without any modification, because they also have a remainder operation. For example, the gcd of $p = x^4 - 4x^3 + 1$ and $q = x^3 - 3x^2 + 1$ can be calculated as follows:

$$\begin{aligned}
r_0 &:= p = x^4 - 4x^3 + 1, \\
r_1 &:= q = x^3 - 3x^2 + 1, \\
r_2 &:= r_0 \bmod r_1 = -3x^2 - x + 2, \\
r_3 &:= r_1 \bmod r_2 = 16/9 \cdot x - 11/9, \\
r_4 &:= r_2 \bmod r_3 = -27/256, \\
r_5 &:= r_3 \bmod r_4 = 0,
\end{aligned}$$

so $\gcd(p,q) = -27/256$. But as we have seen, the gcd is not unique, and we can multiply it by any nonzero constant, e.g. $\gcd(p,q) = 1$ is also true. (In general, if the result is constant, then any other nonzero constant is also correct.) This means that p and q are coprime, i.e. they cannot have common roots. Note: since constant multipliers don't change the correctness of the result, we can also multiply the earlier polynomials to simplify the calculations, e.g. we can use $r_3' := 9r_3 = 16x - 11$ instead of r_3.

Not only the regular Euclidean algorithm, but also its extended version can be applied to polynomials – in exactly the same way as for integers (again, see: Greatest common divisor). The extended Euclidean algorithm calculates the following polynomials:

Theorem 6.7 (Bézout's identity for polynomials)**.** *If $r \in \mathbb{R}[x]$ is a greatest common divisor of $p \in \mathbb{R}[x]$ and $q \in \mathbb{R}[x]$, then there exist $u, v \in \mathbb{R}[x]$ for which $pu + qv = r$.*

In Sage, `gcd(p, q)` and `xgcd(p, q)` can be used for polynomials too.

7 Formal derivative

From calculus, we know the *derivative*, and it has useful properties for polynomials too. But here we don't want to use all the machinery of calculus (e.g. limits, continuity), and in some cases, we can't even do that (e.g. there is no continuity in $\mathbb{Z}$). Therefore, instead of using calculus, we simply define the derivative for polynomials:

Definition 7.1. For a polynomial $p = a_n x^n + \ldots + a_k x^k + \ldots + a_2 x^2 + a_1 x + a_0 \in K[x]$, its *formal derivative* is the polynomial $p' = n a_n x^{n-1} + \ldots + k a_k x^{k-1} + \ldots + 2a_2 x + a_1$.

(The word "formal" means that this definition is not based on calculus.)

For example, the derivative of $p = x^3 + 2x^2 - 3x + 5$ is $p' = 3x^2 + 4x - 3$.

In Sage, the derivative of a polynomial can be calculated using the member function `p.diff()`.

The formal derivative (like the derivative in calculus) has the following properties ($p, q \in K[x]$):

1. if p is constant ($p \in K$), then $p' = 0$;
2. $x' = 1$;
3. $(p+q)' = p' + q'$;
4. $(p-q)' = p' - q'$;
5. $(c \cdot p)' = c \cdot p'$ if c is constant ($c \in K$);
6. $(p \cdot q)' = p' \cdot q + p \cdot q'$;
7. $(p^n)' = n \cdot p' \cdot p^{n-1}$.

Why is the derivative useful for polynomials? Because of the following property about the roots:

Theorem 7.2. *If $c \in \mathbb{R}$ is a root of multiplicity $k \geq 1$ of $p \in \mathbb{R}[x]$, then c is a root of multiplicity $k-1$ of p'.*

For example, $c = 3$ is a double root of $p = x^3 - 4x^2 - 3x + 18 = (x+2)(x-3)^2$, and the same c is a simple root of $p' = 3x^2 - 8x - 3 = (x-3)(3x+1)$. (Also, p has a simple root $c = -2$, which is not a root of p', i.e. it has $c = -2$ as a "root of multiplicity zero".)

(Proof: by the definition of roots of multiplicity k, $p = (x-c)^k\, q$, where c is not a root of q. Using the above properties of the derivative, we can derive that $p' = (x-c)^{k-1}\,(kq + (x-c)q')$, i.e. c is *at least* of multiplicity $k-1$ of p'. To prove that it is *exactly* of multiplicity $k-1$, we need to check that c is not a root of $kq + (x-c)q'$, and indeed, it cannot be, because it is not a root of q.)

Of course, p' can have additional roots to those of p, e.g. $c = -1/3$ for the above p'. If we want to keep only those roots that are also roots of p, we can take their greatest common divisor:

Theorem 7.3. *$c \in \mathbb{R}$ is a root of multiplicity $k \geq 1$ of $p \in \mathbb{R}[x]$ if and only if it is a root of multiplicity $k-1$ of $\gcd(p, p')$.*

(Note the difference from the previous theorem: "if and only if", i.e. $\gcd(p, p')$ has no other roots.)

Continuing the above example, $\gcd(p, p') = x - 3$, whose only root is the simple root $c = 3$.

This property of $\gcd(p, p')$ is useful for multiple reasons:

1. If we need the multiple roots of a polynomial p, then we only need to find the roots of $\gcd(p, p')$, which is easier, because it has lower degree than p, and the gcd can be calculated efficiently using the Euclidean algorithm.
2. As a special case, if $\gcd(p, p')$ is a constant polynomial, then we know immediately that p cannot have multiple roots, without calculating any roots.
3. The polynomial $p \,/\, \gcd(p, p')$ (it is a polynomial because $\gcd(p, p') \mid p$) has exactly the same roots as p, but only as simple roots (since each $(x-c)^k$ was divided by $(x-c)^{k-1}$). So if we need to find the roots of a polynomial p, we can first divide it by $\gcd(p, p')$, because doing so will not change the roots (only their multiplicities), but it may reduce the degree.

We have seen Horner's method, which calculated the value $p(c)$ of a polynomial p efficiently. If we also need the value of the derivative, $p'(c)$, we can of course calculate it in the same way from p'. For this, we need the polynomial p', whose calculation requires further multiplications (and more memory). But we can avoid that: Horner's method can be extended to calculate $p'(c)$ directly, along with $p(c)$, without calculating p' itself. This is due to the following theorem:

Theorem 7.4. *If $p \in K[x]$, $c \in K$ and the quotient of p by $x-c$ is $q \in K[x]$, i.e. $p = (x-c)\,q + p(c)$, then $p'(c) = q(c)$.*

(Proof: $p' = ((x-c)\,q + p(c))' = q + (x-c)\,q'$, and substituting c into this gives $p'(c) = q(c)$.)

This is useful because Horner's method calculates q automatically (see the section above: Division with remainder), so we can just do another iteration of Horner's method to evaluate $q(c)$, which is the same as $p'(c)$. Also, these two iterations can be combined, so we don't need to store the whole polynomial q. (This combined iteration requires approx. $2n$ multiplications and $2n$ additions, whereas the calculation of p' would require further n multiplications.)

For example, for $p = x^4 - 3x^3 + 6x + 3$ and $c = 2$, the extended Horner's method can be written as:

	1	-3	0	6	3	$\longleftarrow\ p = x^4 - 3x^3 + 6x + 3$
$c = 2$	1	-1	-2	2	7	$\longleftarrow\ q = x^3 - x^2 - 2x + 2,\ \ p(c) = 7$
$c = 2$	1	1	0	2		$\longleftarrow\ q(c) = p'(c) = 2$

8 Author

These lecture notes were written by Uray M. János, partially using Nagy Ádám's notes in Hungarian: https://compalg.elte.gitlab-pages.hu/dimoa-web/tematika/dimoa

Please report errors and suggestions here: uray.janos@inf.elte.hu.