Contents

1 Introduction **9**

1.1 History and Systems 10
1.1.1 The 'calculus' side 10
1.1.2 The 'group theory' side 11
1.1.3 A synthesis? 12
1.2 Expansion and Simplification 12
1.2.1 Expansion 13
1.2.2 Simplification 13
1.3 Algebraic Definitions 15
1.3.1 Algebraic Closures 18
1.4 Some Complexity Theory 18
1.5 Some Maple 19
1.5.1 The RootOf construct 19
1.5.2 The simplify command 19

2 Polynomials **21**

2.1 What are polynomials? 21
2.1.1 How do we manipulate polynomials? 23
2.1.2 Polynomials in one variable 23
2.1.3 A factored representation 25
2.1.4 Polynomials in several variables 26
2.1.5 Other representations 28
2.2 Rational Functions 31
2.3 Greatest Common Divisors 32
2.3.1 Polynomials in one variable 33
2.3.2 Subresultant sequences 37
2.3.3 Polynomials in several variables 38
2.3.4 Square-free decomposition 40
2.4 Non-commutative polynomials 41

3 Polynomial Equations **43**

3.1 Equations in One Variable 43
3.1.1 Quadratic Equations 43
3.1.2 Cubic Equations 44

3.1.3 Quartic Equations . . . 46
3.1.4 Higher Degree Equations . . . 46
3.1.5 Solutions in Real Radicals . . . 47
3.1.6 Equations of curves . . . 48
3.1.7 How many real roots? . . . 49
3.2 Linear Equations in Several Variables . . . 51
3.2.1 Linear Equations and Matrices . . . 51
3.2.2 Representations of Matrices . . . 51
3.2.3 Matrix Inverses: not a good idea! . . . 52
3.2.4 Over/under-determined Systems . . . 55
3.3 Nonlinear Equations in Several Variables . . . 56
3.3.1 Gröbner Bases . . . 59
3.3.2 How many Solutions? . . . 62
3.3.3 A Matrix Formulation . . . 64
3.3.4 Orderings . . . 65
3.3.5 Example . . . 67
3.3.6 The Gianni–Kalkbrener Theorem . . . 69
3.3.7 The Faugère–Gianni–Lazard–Mora Algorithm . . . 71
3.3.8 The Shape Lemma . . . 73
3.3.9 Triangular Sets . . . 74
3.3.10 Positive Dimension . . . 75
3.3.11 Conclusion . . . 78
3.3.12 Coefficients other than fields . . . 79
3.3.13 Non-commutative Ideals . . . 80
3.4 Equations and Inequalities . . . 80
3.4.1 Applications . . . 81
3.4.2 Quantifier Elimination . . . 82
3.4.3 Algebraic Decomposition . . . 84
3.4.4 Cylindrical Algebraic Decomposition . . . 86
3.4.5 Computing Algebraic Decompositions . . . 89
3.4.6 Complexity . . . 89
3.5 Conclusions . . . 90
3.5.1 Open Problems . . . 90

4 Advanced Algorithms 91
4.1 Modular Methods . . . 91
4.1.1 Gcd in one variable . . . 91
4.1.2 Polynomials in several variables . . . 98
4.1.3 Conclusions . . . 98
4.2 p-adic Methods . . . 98
4.2.1 Introduction to the factorization problem . . . 99
4.2.2 Modular methods . . . 100
4.2.3 Factoring modulo a prime . . . 100
4.2.4 The recombination problem . . . 102
4.2.5 Conclusions . . . 103

5 Calculus **105**
5.1 Introduction . . . 105
5.2 Integration of Rational Functions . . . 106
5.2.1 Integration of Proper Rational Functions . . . 107
5.2.2 Hermite's Algorithm . . . 108
5.2.3 The Horowitz–Ostrogradski Algorithm . . . 109
5.2.4 The Trager–Rothstein Algorithm . . . 110
5.3 Liouville's Theorem . . . 111
5.4 Integration of Logarithmic Functions . . . 112
5.5 Integration of Exponential Functions . . . 112

6 Algebra versus Analysis **113**
6.1 Fundamental Theorem of Calculus Revisited . . . 113
6.2 Constants Revisited . . . 113

A Algebraic Background **115**
A.1 The resultant . . . 115
A.2 Useful Estimates . . . 118
A.2.1 Matrices . . . 118
A.2.2 Coefficients of a polynomial . . . 119
A.2.3 Roots of a polynomial . . . 120
A.2.4 Root separation . . . 122
A.3 Chinese Remainder Theorem . . . 124

B Excursus **127**
B.1 The Budan–Fourier Theorem . . . 127
B.2 Equality of factored polynomials . . . 128
B.3 Karatsuba's method . . . 130
B.3.1 Karatsuba's method and sparse polynomials . . . 131
B.3.2 Karatsuba's method and multivariate polynomials . . . 132

C Systems **133**
C.1 Maple . . . 133
C.1.1 History . . . 133
C.1.2 Data structures . . . 134
C.1.3 Heuristic GCD . . . 136
C.1.4 Conclusion . . . 136

List of Figures

2.1 A polynomial SLP 30
2.2 Code fragment A — a graph 30
2.3 Code fragment B — a tree 30
2.4 DAG representation 31
2.5 Tree representation 31

3.1 Program for computing solutions to a cubic 45
3.2 Program for computing solutions to a quartic 46

4.1 Algorithm 11 97

C.1 Tree for `A`, `B` corresponding to table C.1 134
C.2 Tree for `A`, `B` corresponding to table C.2 134

Preface

This text is under active development, especially due to comments from colleagues, and students on the course CM30070 — Computer Algebra at the University of Bath. I am grateful to John Abbott (Genoa) for the polynomial f on page 99, and for the encouragement of people at CICM 2010 to write Chapter 6.

It is probably best cited as "J.H. Davenport, *Computer Algebra*. `http://staff.bath.ac.uk/masjhd/JHD-CA.pdf`" with a date.

Chapter 1

Introduction

Computer algebra, the use of computers to do algebra rather than simply arithmetic, is almost as old as computing itself, with the first theses [Kah53, Nol53] dating back to 1953. Indeed it was anticipated from the time of Babbage, when Ada Augusta, Countess of Lovelace, wrote

> We may say most aptly that the Analytical Engine weaves algebraical patterns just as the Jacquard loom weaves flowers and leaves. [oL43]

In fact, it is not even algebra for which we need software packages, computers by themselves can't actually do arithmetic: only a limited subset of it. If we ask Excel[1] to compute $e^{\pi\sqrt{163}} - 262537412640768744$, we will be told that the answer is 256. More mysteriously, if we go back and look at the formula in the cell, we see that it is now $e^{\pi\sqrt{163}} - 262537412640768800$. In fact, 262537412640768744 is too large a whole number (or integer, as mathematicians say) for Excel to handle, and it has converted it into floating-point (what Excel terms "scientific") notation. Excel, or any other software using the IEEE standard [IEE85] representation for floating-point numbers, can only store them to a given accuracy, about[2] 16 decimal places.[3] In fact, it requires twice this precision to show that $e^{\pi\sqrt{163}} \neq 262537412640768744$. Since $e^{\pi\sqrt{163}} = (-1)^{\sqrt{-163}}$, it follows from deep results of transcendental number theory [Bak75], that not only is $e^{\pi\sqrt{163}}$ not an integer, it is not a fraction (or rational number), nor even the solution of a polynomial equation with integer coefficients: essentially it is a 'new' number.

[1] Or any similar software package.

[2] We say 'about' since the internal representation is binary, rather than decimal.

[3] In fact, Excel is more complicated even than this, as the calculations in this table show.

i		1	2	3	4	...	10	11	12	...	15	16
a	10^i	10	100	1000	$1\ldots0$	...	$1\ldots0$	10^{11}	10^{12}	...	10^{15}	10^{16}
b	`a-1`	9	99	999	9999	$9\ldots9$	...	$9\ldots9$	10^{12}	...	10^{15}	10^{16}
c	`a-b`	1	1	1	1	1	...	1	1	...	1	0

We can see that the printing changes at 12 decimal digits, but that actual accuracy is not lost until we subtract 1 from 10^{16}.

We will see throughout this book (for an early example, see page 36) that innocent-seeming problems can give rise to numbers far greater than one would expect. Hence there is a requirement to deal with numbers larger, or to greater precision, than is provided by the hardware manufacturers whose chips underlie our computers. Practically every computer algebra system, therefore, has a package for manipulating arbitrary-sized integers (so-called *bignums*) or real numbers (*bigfloats*).

But the fact that the systems *can* deal with large numbers does not mean that we *should* let numbers increase without doing anything. If we have two numbers with n digits, adding them requires a time proportional to n, or in more formal language (see section 1.4) a time $O(n)$. Multiplying them[4] requires a time $O(n^2)$. Calculating a g.c.d., which is fundamental in the calculation of rational numbers, requires $O(n^3)$, or $O(n^2)$ with a bit of care[5]. This implies that if the numbers become 10 times longer, the time is multiplied by 10, or by 100, or by 1000. So it is always worth reducing the size of these integers. We will see later (an early example is on page 91) that much ingenuity has been well-spent in devising algorithms to compute "obvious" quantities by "non-obvious" ways which avoid, or reduce, the use of large numbers. The phrase *intermediate expression swell* is used to denote the phenomenon where intermediate quantites are much larger than the input to, or outputs from, a calculation.

Notation 1 *We write algorithms in an Algol-like notation, with* `:=` *to indicate assignment, and* `=` *(as opposed to C's* `==`*) to indicate the equality predicate. We use indentation to indicate grouping*[6]*, rather than clutter the text with* **begin** ... **end**. *Comments are introduced by the # character, running to the end of the line.*

1.1 History and Systems

The first recorded use of computers to do computations of the sort we envisage was in 1951 [MW51], where $180\left(2^{127}-1\right)^2-1$, a 79-digit number, was prime. In 1953, two theses [Kah53, Nol53] kicked off the 'calculus' side of computer algebra with programs to differentiate expressions. In the same year, [Has53] showed that algorithms in group theory could be implemented on computers.

1.1.1 The 'calculus' side

The initial work [Kah53, Nol53] consisted of programs to do one thing, but the focus soon moved on to 'systems', capable of doing a variety of tasks. One

[4] In principle, $O(n \log n \log\log n)$ is enough [AHU74, Chapter 8], but no computer algebra system routinely uses this, for it is more like $20n \log n \log\log n$. However, most systems will use 'Karatsuba arithmetic' [KO63, and section B.3], which takes $O(n^{\log_2 3} \approx n^{1.585})$, once the numbers reach an appropriate length, often 16 words [SGV94].

[5] In principle, $O(n \log^2 n \log\log n)$ [AHU74, Chapter 8], but again no system uses it routinely.

[6] An idea which was tried in Axiom [JS92], but which turns out to be better in books than in real life.

early one was Collins' system SAC [Col71], written in Fortran. Its descendants continue in SAC-2 [Col85] and QEPCAD [Bro03]. Of course, computer algebra systems *can* be written in any computer language, even Cobol [fHN76].

However, many of the early systems were written in LISP, largely because of its support for garbage collection and large integers. The group at M.I.T., very influential in the 1960s and 70s, developed MACSYMA [MF71, PW85]. This system now exists in several versions [Ano07]. At about the same time, Hearn developed Reduce [Hea05], and shortly after a group at IBM Yorktown Heights produced SCRATCHPAD [GJY75]. This developed into AXIOM [JS92], a system that attempted to match the generality of mathematics with 'generic programming' to allow algorithms to be programmed in the generality in which they are conventionally (as in this book) stated.

These were, on the whole, very large software systems, and attempts were made to produce smaller ones. muMATH [RS79] and its successor DERIVE [RS92] were extremely successful systems on the early PC, and paved the way for the computer algebra facilities of many high-end calculators. Maple [CGGG83] pioneered a 'kernel+library' design.

> The basic Maple system, the *kernel*, is a relatively small collection of compiled C code. When a Maple session is started, the entire kernel is loaded. It contains the essential facilities required to run Maple and perform basic mathematical operations. These components include the Maple programming language interpreter, arithmetic and simplification routines, print routines, memory management facilities, and a collection of fundamental functions. [MGH$^+$03, p. 6]

1.1.2 The 'group theory' side

Meanwhile, those interested in computation group theory, and related topics, had not been idle. One major system developed during the 1970s/80s was CAYLEY [BC90]. This team later looked at Axiom, and built the system MAGMA [BCM94], again with a strong emphasis on genericity. Another popular system is GAP [BL98], whose 'kernel+library' design was consciously [Neu95] inspired by Maple.

The reader may well ask 'why two different schools of thought?' The author has often asked himself the same question. The difference seems one of mathematical attitude, if one can call it that. The designer of a calculus system envisages it being used to compute an integral, factor a polynomial, multiply two matrices, or otherwise operate on a mathematical *datum*. The designer of a group theory system, while he will permit the user to multiply, say, permutations or matrices, does not regard this as the object of the system: rather the object is to manipulate whole groups (etc.) of permutations (or matrices, or ...), i.e. a mathematical *structure*.

1.1.3 A synthesis?

While it is too early to say that the division has been erased, it can be seen that MAGMA, for example, while firmly rooted in the group-theory tradition, has many more 'calculus like' features. Conversely, the interest in polynomial ideals, as in Proposition 26, means that systems specialising in this direction, such as SINGULAR [Sch03] or COCOA [GN90], use polynomial algorithms, but the items of interest are the structures rather than the individual polynomials.

1.2 Expansion and Simplification

These two words, or the corresponding verbs 'expand' and 'simplify' are much used (abused?) in computer algebra, and a preliminary explanation may be in order. Computers of course, do not deal in mathematical objects, but, ultimately, in certain bit patterns which are *representations* of mathematical objects.

Definition 1 *A correspondence f between a class O of objects and a class R of representations is* a representation *of O by R if each element of O corresponds to one or more elements of R (otherwise it is not represented) and each element of R corresponds to one and only one element of O (otherwise we do not know which element of O is represented).* In other words "is represented by", is the inverse of a surjective function "represents" from a subset of R (the "legal representations") to O.

Hence we could represent any mathematical objects, such as polynomials, by well-formed strings of indeterminates, numbers and the symbols `+,-,*,(,)`. The condition "well-formed" is necessary to prevent nonsense such as `)x+++1y(`, and would typically be defined by some finite grammer [ALSU07, Section 2.2]. With no simplification rules, such a representation would regard `x-x` as just that, rather than as zero.

Definition 2 *A representation of a monoid (i.e. a set with a 0, and an addition operation) is said to be* normal *if the only representation of the object 0 is* `0`.

If we have a normal representation f, then we can tell if two objects a and b are equal by computing $f(a) - f(b)$: if this is zero, then a and b are equal, while is this is not zero, they must be unequal. However, this is an inefficient method, to put it mildly.

Definition 3 *A representation is said to be* canonical *if every object has only one representation, i.e. f is a bijection, or 1–1 mapping.*

With a canonical representation, we can say that two objects "*are* the same if, and only if, they *look* the same". **some comments on hashing**

1.2.1 Expansion

This word is relatively easy to define, as application of the distributive law, as seen in the first bullet point of Maple's description.

> - The expand command distributes products over sums. This is done for all polynomials. For quotients of polynomials, only sums in the numerator are expanded; products and powers are left alone.
> - The expand command also expands most mathematical functions, including ....

The precise meaning depends on the underlying polynomial representation used (recursive/distributed — see page 26), so Maple, which is essentially distributed, would expand $x(y+1)$ into $xy+x$, while Reduce, which is recursive, would not, but would expand $y(x+1)$ into $xy+x$, since its default ordering is 'x before y'.

Expansion can, of course, cause exponential blow-up in the size of the expression: consider $(a+b)(c+d)(e+f)\ldots$, or $\sin(a+b+c+\ldots)$. The second bullet point of Maple's description can lead to even more impressive expansion, as in

```
expand(BesselJ(4,t)^3);
```

(just where did the number 165888 come from?) or

```
expand(WeierstrassP(x+y+z,2,3));
```

1.2.2 Simplification

This word is much used in algebra, particularly at the school level, and has been taken over by computer algebra, which has thereby committed the sin of importing into a precise subject a word without a precise meaning. Let us first consider a few examples.

1. Does $\frac{x^2-1}{x-1}$ simplify to $x+1$? For most people, the answer would be 'yes', but some would query "what happens when $x=1$", i.e. would ask whether we are dealing with abstract formulae, or representations of functions. This is discussed further for rational functions on page 32, and in item 5 below.

2. Does $\frac{x^{1000}-1}{x-1}$ simplify to $x^{999}+\cdots+1$ (assuming the answer to the previous question is 'yes')? Here the fraction is much shorter than the explicit polynomial, and we have misled the reader by writing $\cdots$ here[7].

[7] The construct $\cdots$ is very common in written mathematics, but has had almost (but see [SS06]) no analysis in the computer algebra literature.

3. Does $\sqrt{1-x}\sqrt{1+x}$ simplify to $\sqrt{1-x^2}$? Assuming the mathematical validity, which is not trivial [BD02], then the answer is almost certainly yes, since the second operations involves fewer square roots than the first.

4. Does $\sqrt{x-1}\sqrt{x+1}$ simplify to $\sqrt{x^2-1}$? This might be thought to be equivalent to the previous question, but consider what happens when $x = -2$. The first one simplifies to $\sqrt{-3}\sqrt{-1} = i\sqrt{3} \cdot i = -\sqrt{3}$, while the second simplifies to $\sqrt{3}$. Note that evaluations result in *real* numbers, though they proceed via complex ones. Distinguishing this case from the previous one is a subject of active research [CDJW00].

5. Assume we are working modulo a prime p, i.e. in the field $\mathbf{F}_p$. Does $x^p - x$ simplify to 0? As polynomials, the answer is no, but as functions $\mathbf{F}_p \to \mathbf{F}_p$, the answer is yes, by Fermat's Little Theorem. Note that, as functions $\mathbf{F}_{p^2} \to \mathbf{F}_{p^2}$, say, the answer is no.

Now we give a few illustrations of what simplification means to different audiences.

Teachers At a course on computer algebra systems for teachers of the French 'concours', among the most competitive mathematics examinations in the western world, there was a round table on the meaning of simplification. Everyone agreed that the result should be 'mathematically equivalent', though it was less clear, prompted by example 1, exactly what this meant. The response to example 2 was along the lines of "well, you wouldn't ask such a question". The author wishes he had had examples 3 and 4 to hand at the time!

The general consensus was that 'simplification' meant 'give me the answer I want'. This answer is not *effective*, in the sense that it cannot be converted into a set of rules.

Moses [Mos71] This is a seminal paper, but describes approaches to simplification rather than defining it. Inasmuch as it does define it, it talks about 'utility for further operations', which again is not effective. However, the principle is important, since a request to factor would find the expression $x^{999} + \cdots + 1$ appropriate[8], whereas $\frac{x^{1000}-1}{x-1}$ is in a field, and factorisation is not a relevant question.

Carette [Car04] He essentially defines simplification in terms of the length of the result, again insisting on mathematical equivalence. This would regard examples 1 (assuming they were deemed to be mathematical equivalent) and 3 as simplifications, but not 2, since the expression becomes longer.

Numerical Analysts A numerical analyst would be shocked at the idea that $x^2 - y^2$ was 'simpler' than $(x + y)(x - y)$. He would instantly quote an example such as the following [Ham07].

[8] In fact, knowing the expression *came from* that quotient would be relevant [BD89] to factorisation algorithms, but that's beside the point here.

> For simplicity, assume decimal arithmetic, with perfect rounding and four decimal places. Let $x = 543.2$ and $y = 543.1$. Then x^2 evaluates to 295100 and y^2 evaluates to 295000, so $x^2 - y^2$ becomes 100, while $(x + y)(x - y)$ evaluates to 108.6, a perfect rounding of the true answer 108.63.
>
> Furthermore, if we take $x = 913.2$ and $y = 913.1$, $x^2 - y^2$ is still 100, while the true result is 182.63.

This is part of the whole area of *numerical stability*: fascinating, but outside the scope of this text.

One principle that can be extracted from the above is "if the expression is zero, please tell me": this would certainly meet both the teachers' and Carette's views. This can be seen as a call for simplification to return a normal form *where possible* [Joh71, Ric97].

Maple's description of the 'simplify' command is as follows.

- The `simplify` command is used to apply simplification rules to an expression.
- The `simplify/expr` calling sequence searches the expression, `expr`, for function calls, square roots, radicals, and powers. It then invokes the appropriate simplification procedures.
- **symbolic** Specifies that formal symbolic manipulation of expressions is allowed without regard to the analytical issue of branches for multi-valued functions. For example, the expression `sqrt(x^2)` simplifies to `x` under the symbolic option. Without this option, the simplified result must take into account the different possible values of the (complex) sign of `x`.

Maple does its best to return a normal form, but can be fooled: for example

$$RootOf\left(\_Z^4 + b\_Z^2 + d\right) - 1/2\sqrt{-2\,b + 2\sqrt{b^2 - 4\,d}},$$

which is actually zero (applying figure 3.2), does not simplify to zero under Maple 11.

Because simplification may often require expansion, e.g. to take $(x-1)(x+1)$ to $x^2 - 1$, the two are often confused, and indeed both Macsyma and Reduce (internally) used `ratsimp` and `*simp` (respectively) to denote what we have called expansion.

1.3 Algebraic Definitions

In this section we give some classic definitions from algebra, which we will return to throughout this book. Other concepts are defined as they occur, but these ones are assumed.

Definition 4 *A set R is said to be a* ring *if it has two binary operations $+$ and $*$, a unary operation $-$ and a distinguished element 0, such that, for all a, b and c in R:*

1. $a + (b + c) = (a + b) + c$ *(associativity of $+$);*
2. $a * (b * c) = (a * b) * c$ *(associativity of $*$);*
3. $a + b = b + a$ *(commutativity of $+$);*
4. $a + (-a) = 0$*;*
5. $a + 0 = a$*;*
6. $a * (b + c) = (a * b) + (a * c)$ *(distributivity of $*$ over $+$);*
7. $a * b = b * a$ *(commutativity of $*$).*

Not every text includes the last clause, and they would call a 'commutative ring' what we have called simply a 'ring'. In the absence of the last clause, we will refer to a 'non-commutative ring'.

Definition 5 *If R is a (possibly non-commutative) ring and $\emptyset \neq I \subseteq R$, then we say that I is a (left-)ideal of R, written $I \triangleleft R$, if the following two conditions are satisfied*[9]*:*

(i) $\forall f, g \in I, f - g \in I$,

(i) $\forall f \in R, g \in I, fg \in I$.

There are also concepts of *right ideal* and *two-sided ideal*, but all concepts agree in the case of commutative rings. Non-trivial ideals (the trivial ideals are $\{0\}$ and R itself) exist in most rings: for example, the set of multiples of m is an ideal in $\mathbf{Z}$.

Proposition 1 *If I and J are ideals, then $I + J = \{f + g : f \in I, g \in J\}$ and $IJ = \{fg : f \in I, g \in J\}$ are themselves ideals.*

Definition 6 *A ring is said to be* noetherian*, or to satisfy the* ascending chain condition *if every ascending chain $I_1 \subset I_2 \cdots$ of ideals is finite.*

Theorem 1 (Noether) *If R is a commutative noetherian ring, then so is $R[x]$ (Notation 4, page 22).*

Corollary 1 *If R is a commutative noetherian ring, then so is $R[x_1, \ldots, x_n]$.*

Definition 7 *A ring R is said to be an* integral domain *if, in addition to the conditions above, there is a neutral element 1 such that $1 * a = a$ and, whenver $a * b = 0$, at least one of a and b is zero.*

[9] We write $f - g \in I$ since then $0 = f - f \in I$, and then $f + g = f - (0 - g) \in I$.

Another way of stating the last is to say that R *has no zero-divisors* (meaning none other than zero itself).

Definition 8 *An element u of a ring R is said to be a* unit *if there is an element u^{-1} such that $u * u^{-1} = 1$. u^{-1} is called the* inverse *of u.*

Definition 9 *If $a = u * b$ where u is a unit, we say that a and b are* associates.

Proposition 2 *If a and b are associates, and b and c are associates, then a and c are associates. In other words, being associates is an equivalence relation.*

For the integers, n and $-n$ are associates, whereas for the rational numbers, any two non-zero numbers are associates.

Definition 10 *If R is an integral domain such that every ideal I is* principal*, i.e. $I = (f)$ for some $f \in R$, then R is called a* principal ideal domain*, or P.I.D.*

Classic P.I.D.s are the integers $\mathbf{Z}$, and polynomials in one variable. Inside a principal ideal domain, we have the classic concept of a greatest common divisor.

Proposition 3 *Let R be a P.I.D., $a, b \in R$ and $(a, b) = (g)$. Then g is a* greatest common divisor *of a and b, in the sense that any other common divisor divides g, and $g = ca + db$ for $c, d \in R$.*

Definition 11 *If F is a ring in which every non-zero element is a unit, F is said to be a* field.

The "language of fields" therefore consists of two constants (0 and 1), four binary operations ($+$, $-$, $*$ and $/$) and two unary operations ($-$ and $^{-1}$, which can be replaced by the binary operations combined with the constants). The rational numbers and real numbers are fields, but the integers are not. For any m, the integers modulo m are a ring, but only if m is prime do they form a field. The only ideals in a field are the trivial ones.

Definition 12 *If R is an integral domain, we can always form a field from it, the so called* field of fractions*, consisting of all formal fractions*[10] $\frac{a}{b} : a, b \in R, b \neq 0$*, where a/b is zero if and only if a is zero. Addition is defined by $\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd}$, and multiplication by $\frac{a}{b} * \frac{c}{d} = \frac{ac}{bd}$. So $\frac{a}{b} = \frac{c}{d}$ if, and only if, $ad - bc = 0$.*

In particular, the rational numbers are the field of fractions of the integers.

Definition 13 *If F is a field, the* characteristic *of F, written* $\mathrm{char}(F)$*, is the least positive number n such that $\underbrace{1 + \cdots + 1}_{n \text{ times}} = 0$. If there is no such n, we say that the characteristic is zero.*

[10] Strictly speaking, equivalence classes of formal fractions, under the equality we are about to define.

So the rational numbers have characteristic 0, while the integers modulo p have characteristic p, as do, for example, the rational functions whose coefficients are integers modulo p.

Proposition 4 *The characteristic of a field, if non-zero, is always a prime.*

1.3.1 Algebraic Closures

Some polynomial equations have solutions in a given ring/field, and some do not. For example, $x^2 - 1 = 0$ always has two solutions: $x = -1$ and $x = 1$. The reader may protest that, over a field of characteristic two, there is only one root, since $1 = -1$. However, over a field of characteristic two, $x^2 - 1 = (x - 1)^2$, so $x = 1$ is a root with multiplicity two.

Definition 14 *A field F is said to be* algebraically closed *if every polynomial in one variable over F has a root in F.*

Proposition 5 *If F is algebraically closed, then every polynomial of degree n with coefficients in F has, with multplicity, n roots in F.*

Theorem 2 ("Fundamental Theorem of Algebra") [11] **C**, *the set of complex numbers, is algebraically closed.*

Definition 15 *If F is a field, the* algebraically closure *of F, denoted $\overline{F}$ is the field generated by adjoining to F the roots of all polynomials over F.*

It follows from proposition 53 that the algebraic closure is in fact algebraically closed.

1.4 Some Complexity Theory

As is usual in computing we will use the so-called "Landau notation"[12] to describe the computing time (or space) of operations.

Notation 2 ("Landau") *Let N be some measure of the size of a problem, and f some function. If $t(N)$ is the time taken, on a given hardware/software configuration, to solve the hardest problem of size N, we say that*

$$t(N) = O(f(N)) \tag{1.1}$$

if $\exists C \in \mathbf{R}, M : \forall N > Mt(N) < Cf(N)$.

We will also use "soft O" notation.

[11] The title is in inverted commas, since **R** and **C** are constructs of *analysis*, rather than algebra.

[12] Though apparently first introduced by[Bac94, p. 401]. See [Knu74].

Notation 3 ("soft O") *Let N be some measure of the size of a problem, and f some function. If $t(N)$ is the time taken, on a given hardware/software configuration, to solve the hardest problem of size N, we say that*

$$t(N) = \tilde{O}(f(N)) \tag{1.2}$$

if $\forall \epsilon > 0 \exists C \in \mathbf{R}, M : \forall N > M \quad t(N) < Cf(N)^{1+\epsilon}$.

We should note that "$= O$" and "$= \tilde{O}$" should really be written with "$\in$" rather than "=", and this use of "=" is not reflexive, symmetric or transitive. Also, many authors use $\tilde{O}$ to mean "up to logarithmic factors", which is included in our, more general, definition. [DL08].

The key results of complexity theory for elementary algorithms are that it is possible to multiply two N-bit integers in time $\tilde{O}(N)$, and two degree-N polynomials with coefficients of bit-length at most τ in time $\tilde{O}(N\tau)$. [AHU83]

However, the reader should be warned that $\tilde{O}$ expressions are often far from the reality experienced in computer algebra, where data are small enough that the limiting processes in equations (1.1) and (1.2) have not really taken hold (see note [4], or are quantised (in practice integer lengths are measured in words, not bits, for example).

1.5 Some Maple

1.5.1 The RootOf construct

1.5.2 The simplify command

This Maple command has been discussed before. It is worth noting that simplification in this sense does not commute with substitution, and it can be argued [Zim07] that it is a 'user-oriented' command that should not be used inside Maple programs.

Chapter 2

Polynomials

Polynomials are fundamental to much of mathematics, even when the objects under discussion are apparently not polynomials, such as differential equations. Equally, polynomials underpin much of computer algebra. But what, precisely, are they?

2.1 What are polynomials?

There are numerous definitions. From our point of view, computer algebra, we will adopt the following definition for *commutative*[1] polynomials, leaving non-commutative polynomials to be discussed in section 2.4.

Definition 16 (Polynomials) *A (commutative)* polynomial *is built up from* coefficients*, which are assumed to form a ring (definition 4), and certain* indeterminates *(often called* variables*), by the* algebraic operations *of addition, subtraction and multiplication. These are subject to the following laws, where* a, b, c *are polynomials,* m, n *coefficients, and 0 and 1 certain distinguished coefficients.*

1. $a + b = b + a$;
2. $(a + b) + c = a + (b + c)$;
3. $a + 0 = a$
4. $a + (-a) = 0$;
5. $a * b = b * a$;
6. $a * (b * c) = (a * b) * c$;

[1] "Commutative" meaning $a * b = b * a$. Strictly speaking, we should also worry whether addition is commutative, i.e. whether $a + b = b + a$, but we will always assume that addition is commutative.

7. $a * 1 = a$;

8. $a * (b + c) = (a * b) + (a * c)$;

9. $m + n = m \oplus n$;

10. $m * n = m \otimes n$;

where we have used $\oplus$ and $\otimes$ to denote the operations of addition and multiplication on coefficients, which are assumed to be given to us.

The reader can think of the coefficients as being numbers, though they need not be, and may include other indeterminates that are not the "certain indeterminates" of the definition. However, we will use the usual 'shorthand' notation of 2 for $1 \oplus 1$ etc. The associative laws (2 and 6 above) mean that addition and multiplication can be regarded as n-ary operations. A particular consequence of these rules is

8' $m * a + n * a = (m \oplus n) * a$

which we can think of as 'collection of like terms'.

Proposition 6 *Polynomials over a ring form a ring themselves.*

If it is the case that a polynomial is only zero if it can be deduced to be zero by rules 1–10 above, and the properties of $\oplus$ and $\otimes$, then we say that we have a *free* polynomial algebra. Free algebras are common, but by no means the only one encountered in computer algebra. For examples, trigonometry is often encoded by regarding $\sin\theta$ and $\cos\theta$ as indeterminates, but subject to $\sin^2\theta + \cos^2\theta = 1$ [Sto77].

Notice what we have *not* mentioned: division and exponentiation.

Definition 17 ((Exact) Division) *If $a = b * c$, then we say that b* divides *a, and we write $b = \frac{a}{c}$.*

Note that, for the moment, division is only defined in this context. We note that, if c is not a zero-divisor, b is unique.

Definition 18 (Exponentiation) *If n is a natural number and a is a polynomial, then we define a^n inductively by:*

- $a^0 = 1$;
- $a^{n+1} = a * a^n$.

Notation 4 *If K is a set of coefficients, and V a set of variables, we write $K[V]$ for the set of polynomials with coefficients in K and variables in V. We write $K[x]$ instead of $K[\{x\}]$ etc.*

2.1.1 How do we manipulate polynomials?

We have defined the abstract, almost Platonic, concept of polynomials as mathematical objects, and polynomial algebra as rules for these objects. What do we mean by the *representation* of these objects in a computer system?

One option would be for a computer algebra system essentially to do nothing, simply recording the computations requested by the user, so that `a+b` would become simply $a + b$. However, we would not be very happy with a calculator which computed `1+1` as "1+1", as we would rather see "2". In particular, if the answer is 0, we would like to see that shown, i.e. we would like the representation to be normal (definition 2).

2.1.2 Polynomials in one variable

We will first consider polynomials in one variable, say x. If the coefficients come from a domain K, the polynomials in x over K are denoted by $K[x]$ (Notation 4). One obvious route to a canonical representation (definition 3) is to insist that polynomials be expanded, i.e. that multiplication is only applied to coefficients and variables, not to general polynomials. This is achieved by applying distributivity, rule 8 from definition 16, where necessary, ensuring that multiplication is not applied to additions. Once this is done, the polynomial is of the form $\sum_{i=0}^{n} a_i x^i$, where the a_i are coefficients.

Notation 5 *We assume that $a_n \neq 0$. In this case, n is called the* degree *of the polynomial, denoted* $\deg(f)$*, or* $\deg_x(f)$ *if we wish to make it clear which variable is being considered. a_n is called the* leading coefficient *of f, and denoted* $\mathrm{lc}(f)$ *or* $\mathrm{lc}_x(f)$*. If $\mathrm{lc}(f) = 1$, we say that f is* monic*. $f - \mathrm{lc}(f)x^{\deg(f)}$, i.e. f minus its leading term, is known as the* reductum *of f.*

There is then an important distinction, which does not really occur when doing algebra by hand: does one represent the zero coefficients, or not?

Definition 19 *A representation*[2] *is said to be* dense *if every coefficient, zero or not, is represented, while it is* sparse *if zero coefficients are not stored.*

Hence the polynomial x^2+0x-1, normally written as x^2-1, would be stored as `<1,0,-1>` in a dense representation, but `<<2,1>,<0,-1>>` in a sparse representation. As is implicit in the previous sentence, the normal "human" representation is sparse. Those systems that automatically expand, e.g. Reduce [Hea05], use a sparse representation, since a system would look fairly silly if it was unable to represent $x^{1000000000}+1$ since it could not store the 999,999,999 zeros. However, dense representations are often used internally in some algorithms.

Proposition 7 *Both the dense and the sparse expanded representation are canonical (definition 3), provided that:*

- *leading (in the dense case) or all (in the sparse case) zeros are suppressed;*

[2] In the current case, we are dealing with polynomials. But the concept is more general — see section 3.2.2 for sparse matrices, for example.

- *(in the sparse case) the individual terms are stored in a specified order, generally*[3] *sorted.*

Addition is fairly straight-forward in either representation. In Lisp, we can do addition in a sparse representation as follows. We use a representation in which the `CAR` of a polynomial is a term, whilst the `CDR` is another polynomial: the initial polynomial minus the term defined by the `CAR`, i.e. the reductum (Notation 5). A term is a `CONS`, where the `CAR` is the exponent and the `CDR` is the coefficient. Thus the LISP structure of the polynomial $3x^2+1$ is `((2 . 3) (0 . 1))`, and the list `NIL` represents the polynomial 0, which has no non-zero coefficients, and thus nothing to store. In this representation, we must note that the number 1 does not have the same representation as the polynomial 1 (which is `((0 . 1))`), and that the polynomial 0 is represented differently from the other numerical polynomials.

```
(DE ADD-POLY (A B)
  (COND ((NULL A) B)
        ((NULL B) A)
        ((GREATERP (CAAR A) (CAAR B))
            (CONS (CAR A) (ADD-POLY (CDR A) B)))
        ((GREATERP (CAAR B) (CAAR A))
            (CONS (CAR B) (ADD-POLY A (CDR B))))
        ((ZEROP (PLUS (CDAR A) (CDAR B)))
          ; We must not construct a zero term
            (ADD-POLY (CDR A) (CDR B)))
        (T (CONS (CONS (CAAR A) (PLUS (CDAR A) (CDAR B)))
                 (ADD-POLY (CDR A) (CDR B))))))

(DE MULTIPLY-POLY (A B)
  (COND ((OR (NULL A) (NULL B)) NIL)
;  If a = a0+a1 and b = b0+b1, then ab = a0b0 + a0b1 + a1b
        (T (CONS (CONS (PLUS (CAAR A) (CAAR B))
                       (TIMES (CDAR A) (CDAR B)))
                 (ADD-POLY (MULTIPLY-POLY (LIST (CAR A))
                                          (CDR B))
                           (MULTIPLY-POLY (CDR A) B))))))
```

If A has m terms and B has n terms, the calculating time (i.e. the number of LISP operations) for `ADD-POLY` is bounded by $O(m+n)$, and that for `MULTIPLY-POLY` by $O(m^2n)$ ($(m(m+3)/2-1)n$ to be exact).[4] There are multiplication algorithms which are more efficient than this one: roughly speaking, we ought to sort the terms of the product so that they appear in decreasing

[3] But we should observe that Maple, for example, which uses a hash-based order, is still canonical, even though it may not seem so to the human eye.

[4] It is worth noting the asymmetry in the computing time of what is fundamentally a symmetric operation. Hence we ought to choose A as the one with the fewer terms. If this involves counting the number of terms, many implementors 'cheat' and take A as the one of lower degree, hoping for a similar effect.

order, and the use of `ADD-POLY` corresponds to an insertion sort. We know that the number of coefficient multiplications in such a 'classical' method is mn, so this emphasises that the extra cost is a function of the exponent operations (essentially, comparison) and list manipulation. Of course, the use of a better sorting method (such as "quicksort") offers a more efficient multiplication algorithm, say $O(mn \log m)$ [Joh74]. But most systems use an algorithm similar to the procedure given above.

In a dense representation, we can use radically different, and more efficient, methods based on Karatsuba's method [KO63, and section B.3], which takes time $O(\max(m,n)\min(m,n)^{0.57\cdots}$, or the Fast Fourier Transform [AHU74, chapter 8], where the running time is $O(n \log n)$.

There is a technical, but occasionally important, difficulty with this procedure, reported in [ABD88]. We explain this difficulty in order to illustrate the problems which can arise in the translation of mathematical formulae into computer algebra systems. In `MULTIPLY-POLY`, we add $a_0 b_1$ to $a_1 b$. The order in which these two objects are calculated is actually important. Why, since this can affect neither the results nor the time taken? The order *can* dramatically affect the maximum memory space used during the calculations. If a and b are dense of degree n, the order which first calculates $a_0 b_1$ should store all these intermediate results before the recursion finishes. Therefore the memory space needed is $O(n^2)$ words, for there are n results of length between 1 and n. The other order, $a_1 b$ calculated before $a_0 b_1$, is clearly more efficient, for the space used at any moment does not exceed $O(n)$. This is not a purely theoretical remark: [ABD88] were able to factorise $x^{1155} - 1$ with REDUCE in 2 megabytes of memory, but they could not remultiply the factors without running out of memory, which appears absurd.

Division is fairly straight-forward: to divide f by g, we keep subtracting appropriate (meaning $c_i = (\mathrm{lc}(f)/\mathrm{lc}(g))x^{\deg f - \deg g}$) multiples of g from f until the degree of f is less than the degree of g. If the remaining term (the remainder) is zero, then g divides f, and the quotient can be computed by summing the $c_i x^i$. This is essentially the process known to schoolchildren as "long division".

2.1.3 A factored representation

Instead of insisting that multiplication not be applied to addition, we could insist that addition not be applied to multiplication. This would mean that a polynomial was represented as a product of polynomials, each the sum of simple terms:

$$f = \prod_i f_i = \prod_i \left(\sum_{j=0}^{n_i} a_{i,j} x^j \right). \tag{2.1}$$

In practice, repeated factors are stored explicitly, as in the following format:

$$f = \prod_i f_i^{d_i} = \prod_i \left(\sum_{j=0}^{n_i} a_{i,j} x^j \right)^{d_i}. \tag{2.2}$$

We have a choice of using sparse or dense representations for the f_i, but usually sparse is chosen. It is common to insist that the f_i are square-free[5] and relatively prime[6] (both of which necessitate only g.c.d. computations[7] — lemma 3), but not necessarily[8] irreducible. Hence this representation is generally known as *partially factored*. In this format, the representation is not canonical, since the polynomial x^2-1 could be stored either as that (with 2 terms), or as $(x-1)(x+1)$ (with 4 terms): however, it is normal in the sense of definition 2. For equality testing, see excursus B.2

Multiplication is relatively straight-forward, we check (via g.c.d. computations[9]) for duplicated factors between the two multiplicands, and then combine the multiplicands. Addition can be extremely expensive, and the result of an addition can be exponentially larger than the inputs: consider

$$(x+1)(x^2+1)\cdots(x^{2^k}+1)+(x+2)(x^2+2)\cdots(x^{2^k}+2),$$

where the input has $4(k+1)$ non-zero coefficients, and the output has 2^{k+1} (somewhat larger) ones.

This representation is not much discussed in the general literature, but is used in Redlog [DS97] and Qepcad [CH91], both of which implement cylindrical algebraic decomposition (see section 3.4), where a great deal of use can be made of corollaries 13 and 14.

2.1.4 Polynomials in several variables

Here the first choice is between factored and expanded. The arguments for, and algorithms handling, factored polynomials are the same as in the case of one variable. The individual factors of a factored form can be stored in any of the ways described below for expanded polynomials, but recursive is more common since it is more suited to g.c.d. computations (sections 4.1 and 4.2), which as we saw above are crucial to manipulating factored representations.

If we choose an expanded form, we have one further choice to make, which we explain in the case of two variables, x and y, and illustrate the choices with $x^2y+x^2+xy^2-xy+x-y^2$.

recursive — $C[x][y]$. We regard the polynomials as polynomials in y, whose coefficients are polynomials in x. Then the sample polynomial would be

[5] Which almost certainly improves compactness, but see [CD91], where a dense polynomial of degree 12 was produced (13 terms), whose square had only 12 nonzero terms, and the process can be generalised. Twelve is minimal [Abb02].

[6] Which often improves compactness, but consider $(x^p-1)(x^q-1)$ where p and q are distinct primes, which would have to be represented as $(x-1)^2(x^{p-1}+\cdots+1)(x^{q-1}+\cdots+1)$.

[7] These g.c.d. computations, if carried out by modular or p-adic methods (pages 91 and 98), *should* be cheap if the answer is "no simplification", and otherwise should lead to greater efficiency later.

[8] If we were to insist on irreducibility, we would need to store x^p-1 as $(x-1)(x^{p-1}+\cdots+1)$, with $p+2$ terms rather than with 2. Furthermore, irreducibility can be expensive to prove [ASZ00].

[9] Again, these should be cheap if there is no factor to detect, and otherwise lead to reductions in size.

$(x-1)y^2 + (x^2-x)y + (x^2+x)y^0$. We have made the y^0 term explicit here: in practice detailed representations in different systems differ on this point.

recursive — $C[y][x]$. We regard the polynomials as polynomials in x, whose coefficients are polynomials in y. Then the sample polynomial would be $(y+1)x^2 + (y^2-y+1)x + (-y^2)x^0$.

distributed — $C[x,y]$. We regard the polynomials as polynomials in x and y, whose coefficients are numbers. With 6 terms (as in this example), there are $6! = 720$ possible orders. It is usual to impose two additional constraints on the order on terms, or more accurately on the *monomials*[10], i.e. ignoring the coefficients, which we will denote[11] as $>$.

Definition 20 *An ordering is said to be an* admissible *ordering if it satisfies the following conditions.*

- *Compatibility with multiplication: if $a > b$ then, for all monomials c, $ac > bc$.*
- *Well-foundedness: for all non-trivial monomials a, $a > 1$.*

These requirements greatly reduce the available orders for our sample polynomial. One possibility would be to sort by total degree (i.e. the sum of the degrees in each variable), using degree in x as a tie-breaker. This would give us $x^2y + xy^2 + x^2 - xy - y^2 + x$. There is a fuller discussion of such orderings in section 3.3.4. However, we should note one important property of admissible orders here.

Theorem 3 (Descending Chain Condition; Dickson's Lemma) *Any decreasing sequence (with respect to an admissible ordering) of monomials is finite. [Dic13]*

In general, if there are n variables, there are $n!$ possible recursive representations, but an infinite number of possible distributed representations, though clearly only finitely many different ones for any one given polynomial or finite set of polynomials.

In both cases, we use sparse, rather than dense, representations, since any reasonable multivariate polynomial had better be sparse: degree 6 in each of 6 variables means $7^6 = 117649$ terms. The same results as for univariate polynomials apply.

Proposition 8 *For a fixed ordering, both recursive and distributed representations are canonical (definition 2). Partially factored representations are normal, but not canonical.*

[10] We use the term *monomial* to mean a product of (possibly repeated) variables, as in xyz or x^2y, without any coefficient. Usage on this point differs.

[11] We are not making any *numerical* evaluation of the monomials, merely saying which order we put the monomials in.

It makes no sense to compare polynomials in different representations, or the same representation but different orderings. We have spoken about 'representations', but in fact the division between recursive and distributed goes deeper. While characterisations 3 and 2 of a Gröbner base (theorem 13) can make sense in either view, characterisations 4 and 1 (the only effective one) only make sense in a distributed view. Conversely, while the abstract definitions of factorisation and greatest common divisors (definition 22) make sense whatever the view, the only known algorithms for computing them (algorithm 1 or the advanced ones in chapter 4) are inherently recursive[12].

2.1.5 Other representations

Sparse representations take up little space *if* the polynomial is sparse. But shifting the origin from $x = 0$ to $x = 1$, say, will destroy this sparsity, as might many other operations. The following example, adapted from [CGH+03], illustrates this. Let $\Phi(Y, T)$ be

$$\exists X_1 \ldots \exists X_n (X_1 = T + 1) \wedge (X_2 = X_1^2) \wedge \cdots \wedge (X_n = X_{n-1}^2) \wedge (Y = X_n^2). \quad (2.3)$$

The technology described in section 3.4.2 will convert this to a polynomial equation

$$\Psi(Y, T) : Y = (1 + T)^{2^n}. \quad (2.4)$$

Dense or sparse representations have problems with this, in the sense that expression (2.3) has length $O(n)$, but expression (2.4) has length $O(2^n)$ or more. A factored representation could handle the right-hand side, *assuming* that we are not representing the equations as polynomial = 0. But changing the last conjunct of Φ to $(Y = (X_n + 1)^2)$ changes Ψ to

$$Y = \left(1 + (1 + T)^{2^{n-1}}\right)^2, \quad (2.5)$$

whose factored representation now has length $O(2^n)$.

Factored representations display a certain amount of internal structure, but at the cost of an expensive, and possibly data-expanding, process of addition. Are there representations which do not have these 'defects'? Yes, though they may have other 'defects'.

Expression tree This representation "solves" the cost of addition in the factored representation, by storing addition as such, just as the factored representation stored multiplication as such. Hence $\left((x + 1)^3 - 1\right)^2$ would be legal, and represented as such. Equation (2.5) would also be stored compactly *provided* exponentiation is stored as such, e.g. Z^2 requiring one copy of Z, rather than two as in $Z \cdot Z$. This system is not canonical, or even normal: consider $(x + 1)(x - 1) - (x^2 - 1)$. This would be described by Moses [Mos71] as a "liberal" system, and generally comes

[12] At least for commutative polynomials. Factorisation of non-commutative polynomials is best done in a distributed form.

with some kind of `expand` command to convert to a canonical representation. Assuming now that the leaf nodes are constants and variables, and the tree's internal nodes are (binary, i.e. with two arguments) addition, subtraction and multiplication, then a tree with maximal depth p can represent a polynomial with maximum total degree 2^p. It would need to have $2^p - 1$ internal nodes (all multiplication), and 2^p leaf nodes. The degree is easy to bound, by means of a tree-walk, but harder to compute, especially if cancellation actually occurs. Similarly, the leading coefficient can be computed via a simple tree-walk *if* no cancellation occurs.

Expression DAG also known as **Straight-Line Program (SLP)** [IL80]. This is essentially[13] the representation used by Maple — it looks like the previous representation, but the use of hashing in fact makes it a directed acyclic graph (DAG). Again, a straight-line program of length l (i.e. a DAG of depth $l-1$) can store a polynomial of degree 2^{l-1}. The difference with the expression tree representation above is that we only need l nodes, since the nodes can be reused.

This format is essentially immune to the "change of origin" problem mentioned above, since we need merely replace the x node by a tree to compute $x + 1$, thus adding two nodes, and possibly increasing the depth by one, irrespective of the size of the polynomial. The general 'straight-line' formalism has advantages where multi-valued functions such as square root are concerned: see the discussions around figures 3.1 and 3.2.

However, there is one important caveat about straight-line programs: we must be clear what operations are allowed. If the only operations are $+$, $-$ and $\times$, then evidently a straight-line program computes a polynomial. Equally, if division is allowed, the program *might* not compute a polynomial. But might it? If we look at figure 2.1, we see that $p = x^2 - 1$ and $q = x - 1$, so the result is $\frac{p}{q} = \frac{x^2-1}{x-1} = x + 1$. Or is it? If we feed in $x = 1$, we in fact get $\frac{0}{0}$, rather than 2. This is a singularity of the kind known as a removable singularity, because $\lim_{x\to 1} \frac{p(x)}{q(x)} = 2$. In fact [IL80, Theorem 3], deciding if two straight-line programs are equivalent is undecidable if division is allowed.

We said earlier that the only known algorithms for computing greatest common divisors were recursive. This is essentialy true, and means that the computation of greatest common disivors of straight-line programs is not a straight-forward process [Kal88].

Additive Complexity This [Ris85, Ris88] is similar to a straight-line program, except that we only count the number of (binary) addition/subtraction nodes, i.e. multiplication and exponentiation are 'free'. Hence the degree is unbounded in terms of the additive complexity, but for a given

[13] Maple uses n-ary addition and multiplication, rather than binary, as described in section C.1.

Figure 2.1: A polynomial SLP

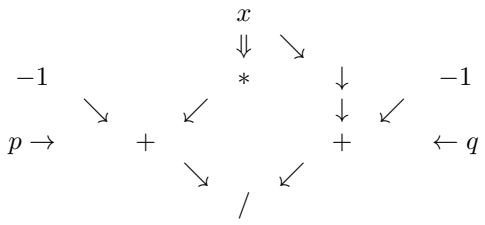


Figure 2.2: Code fragment A — a graph

```
p:=x+1;
q:=p;
r:=p*q;
```

> expression (tree/DAG) can be bounded by a tree-walk. A univariate polynomial of additive complexity a has at most C^{a^2} real roots for some absolute constant C: conjecturally this can be reduced to 3^a. These bounds trivially translate to the straight-line program and expression tree cases.

"Additive complexity" is more of a measure of the 'difficulty' of a polynomial than an actual representation. Of the others, the first was used in Macsyma for its "general expression", and the second is used in Maple[14]. In fact, Macsyma would[15] *allow* general DAGs, but would not force them. Consider the two code fragments in figures 2.2 and 2.3. In the case of figure 2.2, both systems would produce the structure in figure 2.4. For figure 2.3, Macsyma would produce the structure[16] in figure 2.5. Maple would still produce the structure of figure 2.4, since the hashing mechanism would recognise that the two $x+1$ were identical.

[14] Until such time as operations such as `expand` are used!

[15] Not explicitly, but rather as a side-effect of the fact that Macsyma is implemented in Lisp, which cares little for the difference. The basic function `EQUAL` does not distinguish between acyclic and cyclic structures

[16] The x is shown as shared since the Lisp implementation will store symbols unqiuely.

Figure 2.3: Code fragment B — a tree

```
p:=x+1;
q:=x+1;
r:=p*q;
```

Figure 2.4: DAG representation

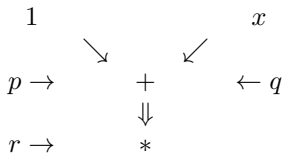
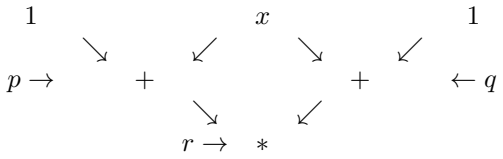


Figure 2.5: Tree representation



2.2 Rational Functions

Of course, we want to manipulate a wider class of expressoins, and even $\frac{1}{x}$ is not a polynomial.

Definition 21 *A* rational function *is built up from* coefficients*, which are assumed to form an integral domain (definition 7), and certain* indeterminates*, by the* algebraic operations *of addition, subtraction, multiplication and division (except that division by zero is not permitted). In addition to the laws in definition 16 (but with a, b and c interpreted as rational functions), the following law is obeyed.*

10+1 $a * (1/a) = 1$.

Proposition 9 *Any rational function can be put over a common denominator, i.e. written as* n/d *where* n *and* d *are polynomials, known as the* numerator *and* denominator *respectively.*

Proposition 10 *In common denominator format,* $\frac{a}{b} = \frac{c}{d}$ *if, and only if,* $ad - bc = 0$.

We can in fact characterise three simple forms of equality.

common coefficients An example of this would be.

$$\frac{x^2 - 2x + 1}{x^2 - 1} \quad \text{versus} \quad \frac{2x^2 - 4x + 2}{2x^2 - 2}.$$

Here we need to remove the g.c.d. of the contents (definition 26) of the two polynomials.

"up to sign" An example of this would be.

$$\frac{-x^2+2x-1}{x^2-1} \quad \text{versus} \quad \frac{x^2-2x+1}{-x^2+1}.$$

These are 'clearly equal' but "computers don't do clearly". We need a convention, and the common one[17] is 'leading coefficient positive' in the denominator. However, this does not generalise so easily to other domains of coefficients [DT90].

common factors An example of this would be

$$\frac{x^2-2x+1}{x^2-1} \quad \text{versus} \quad \frac{x-1}{x+1}.$$

If we put the difference between the two over a common denominator, we get $\frac{0}{x^2-1} = 0$. The reader may complain that $\frac{x^2-2x+1}{x^2-1}$ "is undefined when $x = 1$", whereas $\frac{x-1}{x+1}$ "has the value 0". However, we have not defined what we mean by such substitutions, and for the purposes of this chapter, we are concerned with *algebraic equality* in the sense of proposition 10.

2.3 Greatest Common Divisors

The following definition is valid whenever we have a concept of division.

Definition 22 *h is said to be a* greatest common divisor*, or* g.c.d.*, of f and g if, and only if:*

1. *h divides both f and g;*
2. *if h' divides both f and g, then h' divides h.*

This definition clearly extends to any number of arguments. The g.c.d. is normally written $\gcd(f, g)$.

Note that we have defined *a* g.c.d, whereas it is more common to talk of *the* g.c.d. However, 'a' is correct. We normally say that 2 is *the* g.c.d. of 4 and 6, but in fact -2 is equally a g.c.d. of 4 and 6.

Proposition 11 *If h and h' are greatest common divisors of a and b, they are associates (definition 9).*

Greatest common divisors need not exist. For example, let us consider the set of all integers with $\sqrt{-5}$. 2 clearly divides both 6 and and $2+2\sqrt{-5}$. However, so does $1+\sqrt{-5}$ ($6 = (1+\sqrt{-5})(1-\sqrt{-5})$), yet there is no multiple of both 2 and $1+\sqrt{-5}$ which divides both.

[17] By no means the only possible one: 'leading coefficient negative' would be equally valid, as would 'trailing coefficient positive'.

Definition 23 *An integral domain (definition 7) in which any two elements have a greatest common divisor is known*[18] *as a* g.c.d. domain.

If R is a g.c.d. domain, then the elements of the field of fractions (definition 12) can be simplified by cancelling a g.c.d. between numerator and denominator, often called "reducing to lowest terms". While this simplifies fractions, it does not guarantee that they are normal or canonical. One might think that $\frac{0}{1}$ was the unique representation of zero required for normality, but what of $\frac{0}{-1}$? Equally $\frac{-1}{2} = \frac{1}{-2}$, and in general we have to remove the ambiguity caused by units. In the case of rational numbers, we do this automatically by making the denominator positive, but the general case is more difficult [DT90].

Definition 24 *h is said to be a* least common multiple*, or* l.c.m.*, of f and g if, and only if:*

1. *both f and g divide h ;*
2. *if both f and g divide h', then h divides h'.*

This definition clearly extends to any number of arguments. The l.c.m. is normally written $\mathrm{lcm}(f, g)$.

Proposition 12 *If* $\gcd(f, g)$ *exists, then* $fg/\gcd(f, g)$ *is a least common multiple of f and g.*

This result is normally written as $fg = \gcd(f, g)\mathrm{lcm}(f, g)$, but this is only true up to associates. We should also note that this result does *not* extend to any number of arguments.

2.3.1 Polynomials in one variable

For univariate polynomials over a field, we can define a more extended version of division.

Definition 25 *If a and $b \neq 0$ are polynomials in $K[x]$, K a field, and $a = qb+r$ with* $\deg(r) < \deg(b)$*, then we say that b* divides *a* with quotient q and remainder r*, and q and r are denoted* $\mathrm{quo}(a, b)$ *and* $\mathrm{rem}(a, b)$.

It is clear that q and r exist, and are unique. Division in the previous sense then corresponds to the case $r = 0$.

Theorem 4 (Euclid) *If K is a field, the univariate polynomials $K[x]$ form a g.c.d. domain.*

Algorithm 1 (Euclid)
Input: $f, g \in K[x]$.
Output: $h \in K[x]$ *a greatest common divisor of f and g*

[18] Normally known as a *unique factorisation domain*, but, while the existence of greatest common divisors is equivalent to the existence of unique factorisation, the ability to *compute* greatest common divisors is not equivalent to the ability to *compute* unique factorisations [FS56, DGT91], and hence we wish to distinguish the two.

$i := 1$;
if $\deg(f) < \deg(g)$
  then $a_0 := g$; $a_1 := f$;
  else $a_0 := f$; $a_1 := g$;
while $a_i \neq 0$ **do**
    $a_{i+1} = \mathrm{rem}(a_{i-1}, a_i)$;
    $\#q_i$:=the corresponding quotient: $a_{i+1} = a_{i-1} - q_i a_i$
    $i := i + 1$;
return a_{i-1};

Proof. We must first show that this is an algorithm, i.e. that the potentially infinite loop actually terminates. But $\deg(a_i)$ is a non-negative integer, strictly decreasing each time round the loop, and therefore the loop must terminate. So $a_i = 0$, but $a_i = \mathrm{rem}(a_{i-2}, a_{i-1})$, so a_{i-1} divides a_{i-2}. In fact, $a_{i-2} = q_{i-1}a_{i-1}$. Now $a_{i-1} = a_{i-3} - q_{i-2}a_{i-2}$, so $a_{i-3} = a_{i-1}(1 + q_{i-2}q_{i-1})$, and so on, until we deduce that a_{i-1} divides a_0 and a_1, i.e. f and g in some order. Hence the result of this algorithm is a common divisor. To prove that it is a greatest common divisor, we must prove that any other common divisor, say d, of f and g divides a_{i-1}. d divides a_0 and a_1. Hence it divides $a_2 = a_0 - q_1 a_1$. Hence it divides $a_3 = a_1 - q_2 a_2$, and so on until it divides a_{i-1}.

We should note that our algorithm is asymmetric in f and g: if they have the same degree, it is not generally the case that $\gcd(f, g) = \gcd(g, f)$, merely that they are associates.

Lemma 1 *In these circumstances, the result of Euclid's algorithm is a linear combination of f and g, i.e. $a_{i-1} = \lambda f + \mu g$: $\lambda, \mu \in K[x]$.*

Proof. a_0 and a_1 are certainly such combinations: $a_0 = 1 \cdot f + 0 \cdot g$ or $1 \cdot g + 0 \cdot f$ and similarly for g. Then $a_2 = a_0 - b_1 a_1$ is also such a combination, and so on until a_{i-1}, which is the result.

The above theory, and algorithm, are all very well, but we would like to compute (assuming they exist!) greatest common divisors of polynomials with integer coefficients, polynomials in several variables, etc. So now let R be any g.c.d. domain.

Definition 26 *If $f = \sum_{i=0}^{n} a_i x^i \in R[x]$, define the* content *of f, written* $\mathrm{cont}(f)$*, or* $\mathrm{cont}_x(f)$ *if we wish to make it clear that x is the variable, as* $\gcd(a_0, \ldots, a_n)$*. Technically speaking, we should talk of* a *content, but in the theory we tend to abuse language. and talk of* the *content. Similarly, the* primitive part*, written* $\mathrm{pp}(f)$ *or* $\mathrm{pp}_x(f)$*, is $f/\mathrm{cont}(f)$. f is said to be* primitive *if* $\mathrm{cont}(f)$ *is a unit.*

Proposition 13 *If f divides g, then* $\mathrm{cont}(f)$ *divides* $\mathrm{cont}(g)$ *and* $\mathrm{pp}(f)$ *divides* $\mathrm{pp}(g)$*. In particular, any divisor of a primitive polynomial is primitive.*

The following result is in some sense a converse of the previous sentence.

Lemma 2 (Gauss) *The product of two primitive polynomials is primitive.*

Proof. Let $f = \sum_{i=0}^{n} a_i x^i$ and $g = \sum_{j=0}^{m} b_j x^j$ be two primitive polynomials, and $h = \sum_{i=0}^{m+n} c_i x^i$ their product. Suppose, for contradiction, that h is not primitive, and p is a prime[19] dividing $\mathrm{cont}(h)$. Suppose that p divides all the coefficients of f up to, *but not including*, a_k, and similarly for g up to but not including b_l. Now consider

$$c_{k+l} = a_k b_l + \sum_{i=0}^{k-1} a_i b_{k+l-i} + \sum_{i=k+1}^{k+l} a_i b_{k+l-i} \tag{2.6}$$

(where any indices out of range are deemed to correspond to zero coefficients).

Since p divides $\mathrm{cont}(h)$, p divides c_{k+l}. By the definition of k, p divides every a_i in $\sum_{i=0}^{k-1} a_i b_{k+l-i}$, and hence the whole sum. Similarly, by definition of l, p divides every b_{k+l-i} in $\sum_{i=k+1}^{k+l} a_i b_{k+l-i}$, and hence the whole sum. Hence p divides every term in equation (2.6) except $a_k b_l$, and hence has to divide $a_k b_l$. But, by definition of k and l, it does not divide either a_k or b_l, and hence cannot divide the product. Hence the hypothesis, that $\mathrm{cont}(h)$ could be divisible by a prime, is false.

Corollary 2 $\mathrm{cont}(fg) = \mathrm{cont}(f)\mathrm{cont}(g)$.

Theorem 5 ("Gauss' Lemma") *If R is a g.c.d. domain, and $f, g \in R[x]$, then* $\gcd(f,g)$ *exists, and is* $\gcd(\mathrm{cont}(f), \mathrm{cont}(g))\gcd(\mathrm{pp}(f), \mathrm{pp}(g))$.

Proof. Since R is an integral domain, its field of fractions, say K is a field. Hence, in $K[x]$ where theorem 4 is applicable, $\mathrm{pp}(f)$ and $\mathrm{pp}(g)$ have a greatest common divisor, say h. If c is any non-zero element of R, then ch is also a greatest common divisor of f and g. Hence we can assume that h is in $R[x]$ and, as a polynomial of $R[x]$, is primitive. In $K[x]$, f is a multiple of h, say $f = hk$ for $k \in K[x]$. We can write $k = dk'$, where $k' \in R[x]$ and is primitive. Then $d^{-1}\mathrm{pp}(f) = hk'$. But h and k' are primitive, so, by the Lemma, their product is primitive, and d is a unit. Hence h is, in $R[x]$, *a* common divisor of $\mathrm{pp}(f)$ and $\mathrm{pp}(g)$.

But, if $\bar{h}$ is a common divisor of $\mathrm{pp}(f)$ and $\mathrm{pp}(g)$ in $R[x]$, it is certainly a common divisor of f and g in $K[x]$, hence divides h in $K[x]$, and so $\mathrm{pp}(\bar{h})$ divides h in R[x]. Hence h is a *greatest* common divisor of $\mathrm{pp}(f)$ and $\mathrm{pp}(g)$ in $R[x]$, and the rest of the theorem is obvious.

This gives us one obvious means of computing g.c.d.s in $R[x]$, which can be described as "compute in $K[x]$ and sort out the contents afterwards". Certainly this is an algorithm, but is it a good one? Consider the computation of the g.c.d. of the following two polynomials (this analysis is mostly taken from [Bro71]):

$$\begin{aligned} A(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5; \\ B(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21. \end{aligned}$$

[19] The reader may complain that, in note 18, we said that the ability to compute g.c.d.s was *not* equivalent to the ability to compute unique factors, and hence primes. But we are not asking to factorise $\mathrm{cont}(f)$, merely supposing, *for the sake of contradiction* that it is non-trivial, and therefore has a prime divisor.

The first elimination gives $A - (\frac{x^2}{3} - \frac{2}{9})B$, that is

$$\frac{-5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3},$$

and the subsequent eliminations give

$$\frac{-117}{25}x^2 - 9x + \frac{441}{25},$$

$$\frac{233150}{19773}x - \frac{102500}{6591},$$

and, finally,

$$-\frac{1288744821}{543589225}.$$

Since this is a number, it follows that no *polynomial* can divide both A and B, i.e. that $\gcd(A, B) = 1$.

It is obvious that these calculations on polynomials with rational coefficients require several g.c.d. calculations on integers, and that the integers in these calculations are not always small.

We can eliminate these g.c.d. calculations by working all the time with polynomials with integer coefficients, and this gives a generalisation of the a_i of algorithm 1, known as *polynomial remainder sequences* or *p.r.s.*, by extending the definition of division.

Definition 27 *Instead of dividing f by g in $K[x]$, we can multiply f by a suitable power of the leading coefficient of g, so that the divisions stay in R. The* pseudo-remainder *of dividing f by g, written* $\mathrm{prem}(f, g)$*, is the remainder when one divides* $\mathrm{lc}(g)^{\deg(f)-\deg(g)+1} f$ *by g, conceptually in $K[x]$, but in fact all the calculations can be performed in R, i.e. all divisions are exact in R. This is denoted*[20] *by* $\mathrm{prem}(f, g)$.

In some applications (section 3.1.7) it is necessary to keep track of the signs: we define a signed polynomial remainder sequence *or s.p.r.s. of $f_0 = f$ and $f_1 = g$ to have f_i proportional by a* positive *constant to* $-\mathrm{rem}(f_{i-2}, f_{i-1})$.

This gives us a *pseudo-euclidean algorithm*, analogous to algorithm 1 where we replace rem by prem, and fix up the contents afterwards. In the above example, we deduce the following sequence:

$$-15x^4 + 3x^2 - 9,$$

$$15795x^2 + 30375x - 59535,$$

$$1254542875143750x - 1654608338437500$$

and

$$12593338795500743100931141992187500.$$

Again, this is a number, so $\gcd(A, B) = 1$. We have eliminated the fractions, but at a cost of even larger numbers. Can we do better?

[20] This definition agrees with Maple, but not with all software systems, which often use `prem` to denote what Maple calls `sprem`, i.e. only raising $\mathrm{lc}(g)$ to the smallest power necessary.

2.3.2 Subresultant sequences

One option would be to make the a_i primitive at each step, since we are going to fix up the content terms later: giving the so-called *primitive p.r.s.* algorithm. This is perfectly reasonable when it is a question of polynomials in one variable, and is essentially equivalent to calculating with rational numbers, but over a common denominator. However, if we come to polynomials in several variables, every step of the g.c.d. for polynomials in n variables would involve the g.c.d. of several polynomials in $n-1$ variables, each step of each of which would involve the g.c.d. of several polynomials in $n-2$ variables, and so on.

The following, slightly mysterious[21], algorithm will do the trick. By the Subresultant Theorem [Loo82], all divisions involved are exact, i.e. we always stay in $R[x]$. Furthermore, the factors β_i that are cancelled are *generically* **as large as possible**, where by "generically" we mean that, if the coefficients of f and g were all independent, nothing more could be cancelled[22].

Algorithm 2 (General p.r.s.)
Input: $f, g \in K[x]$.
Output: $h \in K[x]$ *a greatest common divisor of* $\mathrm{pp}(f)$ *and* $\mathrm{pp}(g)$
Comment: *If $f, g \in R[x]$, where R is an integral domain and K is the field of fractions of R, then all computations are exact in $R[x]$.*

$i := 1$;
if $\deg(f) < \deg(g)$
  then $a_0 := \mathrm{pp}(g)$; $a_1 := \mathrm{pp}(f)$;
  else $a_0 := \mathrm{pp}(f)$; $a_1 := \mathrm{pp}(g)$;
$\delta_0 := \deg(a_0) - deg(a_1)$;
$\beta_2 := (-1)^{\delta_0+1}$;
$\psi_2 := -1$;
while $a_i \neq 0$ **do**
  $a_{i+1} = \mathrm{prem}(a_{i-1}, a_i)/\beta_{i+1}$;
  $\#q_i :=$the corresponding quotient: $a_{i+1} = a_{i-1} - q_i a_i$
  $\delta_i := \deg(a_i) - \deg(a_{i+1})$;
  $i := i + 1$;
  $\psi_{i+1} := -\mathrm{lc}(a_{i-1})^{\delta_{i-2}} \psi_i^{1-\delta_{i-2}}$;
  $\beta_{i+1} := -\mathrm{lc}(a_{i-1}) \psi_{i+1}^{\delta_{i-1}}$;
return $\mathrm{pp}(a_{i-1})$;

In the same example as before, we get the following:

$$a_2 \quad = \quad 15x^4 - 3x^2 + 9,$$

[21] Some of the mystery is explained by corollary 3 on page 55.

[22] The reader may comment that the example, repeated below with this algorithm, shows a consistent factor of 3 in a_2, and this is true however the coefficients are perturbed. Indeed, if the leading coefficient of a_1 is changed to, say, 4, we get a consistent factor of 4. However, if the coefficient of x^7 in a_0 is made non-zero, then the common factor will generally go away, and that is what we mean by "generically".

$$\begin{aligned} a_3 &= 65x^2 + 125x - 245, \\ a_4 &= 9326x - 12300, \\ a_5 &= 260708. \end{aligned}$$

Here the numbers are much smaller, and indeed it can be proved that the coefficient growth is only linear in the step number. a_2 has a content of 3, which the primitive p.r.s. would eliminate, but this content in fact disappears later. Similarly a_3 has a content of 5, which again disappears later. Hence we have the following result.

Theorem 6 *Let R be a g.c.d. domain. Then there is an algorithm to calculate the g.c.d. of polynomials in $R[x]$. If the original coefficients have length bounded by B, the length at the i-th step is bounded by iB.*

This algorithm is the best method known for calculating the g.c.d., of all those based on Euclid's algorithm applied to polynomials with integer coefficients. In chapter 4 we shall see that if we go beyond these limits, it is possible to find better algorithms for this calculation.

2.3.3 Polynomials in several variables

Here it is best to regard the polynomials as recursive, so that $R[x, y]$ is regarded as $R[y][x]$. In this case, we now know how to compute the greatest common divisor of two bivariate polynomials.

Algorithm 3 (Bivariate g.c.d.)
Input: f, $g \in R[y][x]$.
Output: $h \in R[y][x]$ *a greatest common divisor of f and g*

$h_c :=$ the g.c.d. of $\mathrm{cont}_x(f)$ and $\mathrm{cont}_x(g)$
\# this is a g.c.d. computation in $R[y]$.
$h_p :=$ algorithm 2 $(\mathrm{pp}_x(f), \mathrm{pp}_x(g))$
\# replacing R by $R[y]$, which we know, by theorem 6, is a g.c.d. domain.
return $h_c h_p$
\# which by theorem 5 is a g.c.d. of f and g.

This process generalises.

Theorem 7 *If R is a g.c.d. domain, then $R[x_1, \ldots, x_n]$ is also a g.c.d. domain.*

Proof. Induction on n, with theorem 6 as the building block.

What can we say about the complexity of this process? It is easier to analyse if we split up the division process $\mathrm{rem}(a_{i-1}, a_i)$ into a series of repeated subtractions of shifted scaled copies of a_i from a_{i-1}. Each such subtraction reduces $\deg(a_{i-1})$, in general by 1. For simplicity, we shall assume that the

reduction is precisely by 1, and that[23] $\deg(a_{i+1}) = \deg(a_i) - 1$. It also turns out that the polynomial manipulation in x is the major cost (this is not the case for the primitive p.r.s, where the recursive costs of the content computations dominates), so we will skip all the other operations (the proof of this is more tedious than enlightening). Let us assume that $\deg_x(f) + \deg_x(g) = k$, and that the coefficients have maximum degree d. Then the first subtraction will reduce k by 1, and replace d by $2d$, and involve k operations on the coefficients. The next step will involve $k-1$ operations on coefficients of size $2d$, and so on, giving $\sum_{i=0}^{k}(k-i)F(id)$, where $F(d)$ is the cost of operating on coefficients of degree d. Let us suppose that there are v variables *in all*: x and $v-1$ in the coefficients.

$v = 2$ Here the coefficients are univariate polynomials. If we assume classic multiplication on dense polynomials, $F(d) = cd^2 + O(d)$. We are then looking at

$$\begin{aligned}
\sum_{i=0}^{k}(k-i)F(id) &\leq c\sum_{i=0}^{k}(k-i)i^2d^2 + \sum_{i=0}^{k}kO(id) \\
&\leq ck\sum_{i=0}^{k}i^2d^2 - c\sum_{i=0}^{k}i^3d^2 + k^3O(d) \\
&= c\left(\frac{1}{3}k^4 + \frac{1}{2}k^3 + \frac{1}{6}k^2\right)d^2 - c\left(\frac{1}{4}k^4 + \frac{1}{2}k^3 + \frac{1}{4}k^2\right)d^2 + k^3O(d) \\
&= c\left(\frac{1}{12}k^4 - \frac{1}{12}k^2\right)d^2 + k^3O(d)
\end{aligned}$$

which we can write as $O(k^4d^2)$. We should note the asymmetry here: this means that we should choose the principal variable (i.e. the x in algorithm 3) to be whichever of x and y minimises

$$\min(\max(\deg_x(f), \deg_x(g)), \max(\deg_y(f), \deg_y(g))).$$

$v = 3$ Here the coefficients are bivariate polynomials. If we assume classic multiplication on dense polynomials, $F(d) = cd^4 + O(d^3)$. We are then looking at

$$\begin{aligned}
\sum_{i=0}^{k}(k-i)F(id) &\leq c\sum_{i=0}^{k}(k-i)i^4d^42 + \sum_{i=0}^{k}kO(i^3d^3) \\
&\leq ck\sum_{i=0}^{k}i^4d^4 - c\sum_{i=0}^{k}i^5d^4 + k^5O(d^3)
\end{aligned}$$

[23] This assumption is known as assuming that the remainder sequence is *normal*. Note that our example is distinctly non-normal, and that, in the case of a normal p.r.s., $\psi_i = \pm 1$. In fact, the sub-resultant algorithm was first developed for normal p.r.s., where it can be seen as a consequence of the Dodgson–Bareiss Theorem (theorem 12).

$$\begin{aligned} &= c\left(\frac{1}{5}k^6 + \cdots\right)d^2 - c\left(\frac{1}{6}k^6 + \cdots\right)d^2 + k^5 O(d^3) \\ &= c\left(\frac{1}{30}k^6 + \cdots\right)d^4 + k^5 O(d^3) \end{aligned}$$

which we can write as $O(k^6 d^4)$. The asymmetry is again obvious.

general v The same analysis produces $O(k^{2v} d^{2v-2})$.

We see that the cost is exponential in v, even though it is polynomial in d and k. This is not a purely theoretical observation: any experiment with several variables will bear this out, even when the inputs (being sparse) are quite small: the reader need merely use his favourite algebra system on

$$a_0 := ax^4 + bx^3 + cx^2 + dx + e; \quad a_1 := fx^4 + gx^3 + hx^2 + ix + j,$$

treating x as the main variable (which of course one would not do in practice), to see the enormous growth of the coefficients involved.

2.3.4 Square-free decomposition

Let us revert to the case of polynomials in one variable, x, over a field K, and let us assume that $\mathrm{char}(K) = 0$ (see definition 13 — the case of characteristic non-zero is more complicated [DT81], and we really ought to talk about 'separable decomposition' [Lec08]).

Definition 28 *The* formal derivative *of* $f(x) = \sum_{i=0}^{n} a_i x^i$ *is written* $f'(x)$ *and computed as* $f'(x) = \sum_{i=1}^{n} i a_i x^{i-1}$.

This is what is usually referred to as the derivative of a polynomial in calculus texts, but we are making no appeal to the theory of differentiation here: merely defining a new polynomial whose coefficients are the old ones (except that a_0 disappears) multiplied by the exponents, and where the exponents are decreased by 1.

Proposition 14 *The formal derivative satisfies the usual laws:*

$$(f+g)' = f' + g' \qquad (fg)' = f'g + fg'.$$

Proof. by algebra from the definition. This is taken up in more generality in Proposition 43.

Let us consider the case $f = g^n h$, where g and h have no common factors. Then $f' = g^n h' + n g^{n-1} g' h$ and is clearly divisible by g^{n-1}. n is not zero in K (by the assumption on $\mathrm{char}(K)$), so g does not divide $f'/g^{n-1} = gh' + ng'h$. Hence $\gcd(f, f')$ is divisible by g^{n-1} but not by g^n. These considerations lead to the following result.

Proposition 15 *Let* $f = \prod_{i=1}^{k} f_i^{n_i}$ *where the* f_i *are relatively prime and have no repeated factors. Then*

$$\gcd(f, f') = \prod_{i=1}^{k} f_i^{n_i - 1}.$$

Definition 29 *The* square-free decomposition *of a polynomial* f *is an expression*

$$f = \prod_{i=1}^{n} f_i^i \tag{2.7}$$

where the f_i *are relatively prime and have no repeated factors.* f *is said to be* square-free *if* $n = 1$.

Note that some of the f_i may be 1.

Lemma 3 *Such a decomposition exists for any non-zero* f*, and can be calculated by means of* gcd *computations and divisions.*

Proof. Let $g = \gcd(f, f') = \prod_{i=1}^{n} f_i^{i-1}$ by the previous proposition. Then $f/g = \prod_{i=1}^{n} f_i$ and $\gcd(g, f/g) = \prod_{i=2}^{n} f_i$. Hence

$$\frac{f/g}{\gcd(g, f/g)} = f_1.$$

Applying the same process to g will compute f_2, and so on.

This is not in fact the most efficient way of computing such a decomposition: a better method was given by Yun [Yun76].

2.4 Non-commutative polynomials

The label "non-commutative polynomials" in fact covers three cases.

1. The indeterminates commute, but the coefficients do not. For definiteness, we will refer to this case as *polynomials with non-commuting coefficients*. In this case, rule 5 of definition 16 has to be replaced by

   5′ $x * y = y * x$;

   where x and y are indeterminates, not general polynomials. This means that some of the traditional laws of algebra cease to operate: for example

   $$(ax + b)(ax - b) = a^2x^2 - b^2$$

   becomes

   $$(ax + b)(ax - b) = a^2x^2 + (-a * b + b * a)x - b^2$$

2. The coefficients commute, but the indeterminates do not. For definiteness, we will refer to this case as *polynomials with non-commuting indeterminates*. In this case, rule 5 of definition 16 has to be replaced by the assumption

   - $m \otimes n = n \otimes m$.

   At this point, many of the traditional laws of algebra cease to operate: even the Binomial Theorem in the form

   $$(x+y)^2 = x^2 + 2xy + y^2$$

   has to be replaced by

   $$(x+y)^2 = x^2 + (xy + yx) + y^2.$$

   A common case of non-commuting indeterminates is in differential algebra, where the variable x and the differentiation operator $\frac{\mathrm{d}}{\mathrm{d}x}$ do not commute, but rather satisfy the equation

   $$\frac{\mathrm{d}}{\mathrm{d}x}(xa) = x\frac{\mathrm{d}a}{\mathrm{d}x} + a. \tag{2.8}$$

3. Neither can be assumed to commute, in which case rule 5 of definition 16 is just deleted, with no replacement.

Notation 6 *If the variables do not commute, it is usual to use the notation $R\langle x_1, \ldots, x_n\rangle$ for the ring of polynomials with coefficients in R and the non-commuting variables $x_1, \ldots, x_n$.*

Chapter 3

Polynomial Equations

In the first parts of this chapter, we will deal with polynomial equations, either singly or as sets of equations. A preliminary remark is in order. Any polynomial equation

$$A = B, \tag{3.1}$$

where A and B are polynomial equations, can be reduced to one whose right-hand side is zero, i.e.

$$A - B = 0. \tag{3.2}$$

Notation 7 *Henceforth, all polynomial equations will be assumed to be in the form of (3.2).*

3.1 Equations in One Variable

We may as well assume that the unknown variable is x. If the equation is linear in x then, by the notation above, it takes the form

$$ax + b = 0. \tag{3.3}$$

The solution is then obvious: $x = -b/a$.

3.1.1 Quadratic Equations

Again, by the notation above, our equation takes the form

$$ax^2 + bx + c = 0. \tag{3.4}$$

The solutions are well-known to most schoolchildren: there are two of them, of the form

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{3.5}$$

However, if $b^2 - 4ac = 0$, i.e. $c = b^2/4a$ then there is only one solution: $x = \frac{-b}{2a}$. In this case, the equation becomes $ax^2 + bx + \frac{b^2}{4a} = 0$, which can be re-written as $a\left(x + \frac{b}{2a}\right)^2 = 0$, making it more obvious that there is a repeated root, and that the polynomial is not square-free (definition 29).

Mathematicians dislike the sort of anomaly in "this equations has two solutions except when $c = b^2/4a$", especially as there are two roots as c tends to the value $b^2/4a$. We therefore say that, in this special case, $x = \frac{-b}{2a}$ is *a double root* of the equation. This can be generalised, and made more formal.

Definition 30 *If, in the equation $f = 0$, f has a square-free decomposition $f = \prod_{i=1}^{n} f_i^i$, and $x = \alpha$ is a root of f_i, we say that $x = \alpha$ is a root of f* of multiplicity *i. When we say we are counting the roots of f* with multiplicity*, we mean that $x = \alpha$ should be counted i times.*

Proposition 16 *The number of roots of a polynomial equation over the complex numbers, counted with multiplicity, is equal to the degree of the polynomial.*

Proof. $\deg(f) = \sum i\deg(f_i)$, and each root of f_i is to be counted i times as a root of f. That f_i has i roots is the so-called Fundamental Theorem of Algebra.

In this case, the two roots are given by the two possible signs of the square root, and $\sqrt{0}$ is assumed to have both positive and negative signs.

3.1.2 Cubic Equations

There is a formula for the solutions of the cubic equation

$$x^3 + ax^2 + bx + c, \tag{3.6}$$

albeit less well-known to schoolchildren:

$$\frac{1}{6}\sqrt[3]{36\,ba - 108\,c - 8\,a^3 + 12\,\sqrt{12\,b^3 - 3\,b^2a^2 - 54\,bac + 81\,c^2 + 12\,ca^3}} - \frac{2b - \frac{2}{3}a^2}{\sqrt[3]{36\,ba - 108\,c - 8\,a^3 + 12\,\sqrt{12\,b^3 - 3\,b^2a^2 - 54\,bac + 81\,c^2 + 12\,ca^3}}} - \frac{1}{3}a.$$

We can simplify this by making a transformation[1] to equation (3.6): replacing x by $x - \frac{a}{3}$. This transforms it into an equation

$$x^3 + bx + c \tag{3.7}$$

(where b and c have changed). This has solutions of the form

$$\frac{1}{6}\sqrt[3]{-108\,c + 12\,\sqrt{12\,b^3 + 81\,c^2}} - \frac{2b}{\sqrt[3]{-108\,c + 12\,\sqrt{12\,b^3 + 81\,c^2}}}. \tag{3.8}$$

[1] This is the simplest case of the Tschirnhaus transformation[vT83], which can always eliminate the x^{n-1} term in a polynomial of degree n.

$$\begin{array}{rcl} S & := & \sqrt{12\,b^3 + 81\,c^2}; \\ T & := & \sqrt[3]{-108\,c + 12\,S}; \\ \textbf{return} & & \frac{1}{6}T - \frac{2b}{T}; \end{array}$$

Figure 3.1: Program for computing solutions to a cubic

Now a cubic is meant to have three roots, but a naïve look as equation (3.8) shows two cube roots, each with three values, and two square roots, each with two values, apparently giving a total of $3 \times 3 \times 2 \times 2 = 36$ values. Even if we decide that the two occurrences of the square root should have the same sign, and similarly the cube root should have the same value, i.e. we effectively execute the program in figure 3.1, we would still seem to have six possibilities. In fact, however, the choice in the first line is only apparent, since

$$\frac{1}{6}\sqrt[3]{-108\,c - 12\,\sqrt{12\,b^3 + 81\,c^2}} = \frac{2b}{\sqrt[3]{-108\,c + 12\,\sqrt{12\,b^3 + 81\,c^2}}}. \tag{3.9}$$

In the case of the quadratic with real coefficients, there were two real solutions if $b^2 - 4ac > 0$, and complex solutions otherwise. However, the case of the cubic is more challenging. If we consider $x^3 - 1 = 0$, we compute (in figure 3.1)

$$S := 9; \qquad T := 6; \qquad \textbf{return } 1;$$

(or either of the complex cube roots of unity if we choose different values of T). If we consider $x^3 + 1 = 0$, we get

$$S := 9; \qquad T := 0; \qquad \textbf{return } \text{“}\tfrac{0}{0}\text{”};$$

but we can (and must!) take advantage of equation (3.9) and compute

$$S := -9; \qquad T := -6; \qquad \textbf{return } -1;$$

(or either of the complex variants).

For $x^3 + x$, we compute

$$S := \sqrt{12}; \qquad T := \sqrt{12}; \qquad \textbf{return } 0;$$

and the two complex roots come from choosing the complex roots in the computation of T, which is really $\sqrt[3]{12\sqrt{12}}$. $x^3 - x$ is more challenging: we compute

$$S := \sqrt{-12}; \qquad T := \sqrt{-12}; \qquad \textbf{return } \{-1, 0, 1\}; \tag{3.10}$$

i.e. three real roots which can only be computed (at least via this formula) by means of complex numbers. In fact it is clear that any other formula must have the same problem, since the only choices of ambiguity lie in the square and cube roots, and with the cube root, the ambiguity involves complex cube roots of unity.

3.1.3 Quartic Equations

Here the equation would be $x^4 + ax^3 + bx^2 + cx + d$, but after the Tschirnhaus transformation $x \to x - \frac{a}{4}$, analogous to that which took equation (3.6) to (3.7), we can assume that $a = 0$. A truly marvellous solution then looks as follows (but the page is too small to contain it!).

$$\frac{\sqrt{6}}{12}\sqrt{\frac{-4\,b\sqrt[3]{-288\,db+108\,c^2+8\,b^3+12\sqrt{-768\,d^3+384\,d^2b^2-48\,db^4-432\,dbc^2+81\,c^4+12\,c^2b^3}}+\left(-\right.}{\sqrt[3]{-288\,db+108\,c^2+8\,b^3+12\sqrt{-768\,d^3+}}}} \tag{3.11}$$

We can adopt the same formulation as in figure 3.1, as shown in figure 3.2. Here

$$\begin{array}{rcl} S & := & \sqrt{-768\,d^3+384\,d^2b^2-48\,db^4-432\,dbc^2+81\,c^4+12\,c^2b^3} \\ T & := & \sqrt[3]{-288\,db+108\,c^2+8\,b^3+12\,S} \\ U & := & \sqrt{\dfrac{-4\,bT+T^2+48\,d+4\,b^2}{T}} \\ \textbf{return} & & \dfrac{\sqrt{6}}{12}U+\dfrac{\sqrt{6}}{12}\sqrt{\dfrac{-\left(8\,bTU+UT^2+48\,Ud+4\,Ub^2+12\,c\sqrt{6}T\right)}{TU}} \end{array}$$

Figure 3.2: Program for computing solutions to a quartic

the problem of multiple choices is even more apparent, but in this formulation it turns out that choices cancel, much as in the case of the cubic. We have the same problem as in the case of the cubic, that real solutions can arise from complex intermediates, but also that the answer apparently involves $\sqrt{6}$, even though it clearly need not do so in reality. For example, with $x^4 - 5x^2 + 4$, whose solutions are $\pm 1, \pm 2$, we can evaluate

$$S := 72\sqrt{-3}; \qquad T := 17 + \sqrt{-3}; \qquad U := 3\sqrt{6}; \qquad \textbf{return } 2; \tag{3.12}$$

taking the other square root at the end gives 1, and taking the other square root when computing U gives -1 or -2. We should also note that T was evaluated as $\sqrt[3]{4760 + 864\sqrt{-3}}$: not entirely obvious.

3.1.4 Higher Degree Equations

When it comes to higher degree equations, the situation is very different.

Theorem 8 (Abel, Galois [Gal79]) *The general polynomial equation of degree 5 or more is not soluble in radicals (i.e. in terms of k-th roots).*

In fact,[2] if we choose such a polynomial "at random", the probability of its having a solution that can be expressed in terms of radicals is zero. Of course, any particular quintic, or higher degree equation, may have solutions expressible in radicals, such as $x^5 - 2$, whose solutions are $\sqrt[5]{2}$, but this is the exception rather than the rule.

Hence algebra systems, if they handle such concepts, can only regard the roots of such equations as being defined by the polynomial of which they are a root. A Maple example[3] is given in figure 1.5.1, where the Maple operator `RootOf` is generated. It is normal to insist that the argument to `RootOf` (or its equivalent) is square-free: then different rooots are genuinely different. Then α, the first root of $f(x)$, satisfies $f(\alpha) = 0$, the second root β satisfies $f(x)/(x-\alpha) = 0$, and so on. Even if f is irreducible, these later polynomials may not be, but determining the factorisations if they exist is a piece of Galois theory which would take us too far out of our way [FM89]. It is, however, comparatively easy to determine the Monte Carlo question: "such factorisations definitely do not exist"/"they probably do exist" [DS00].

It should be noted that handling such constructs when the defining polynomial is not irreducible can give rise to unexpected results. For example, in Maple, if α is `RootOf(x^2-1,x)`, then $\frac{1}{\alpha-1}$ returns that, but attempting to evaluate this numerically gives infinity, which is right if $\alpha = 1$, but wrong if $\alpha = -1$, the other, equally valid, root of $x^2 - 1$. In this case, the mathematical answer to "is $\alpha - 1$ zero?" is neither 'yes' nor 'no', but rather 'it depends which α you mean', and Maple is choosing the 1 value (as we can see from $\frac{1}{\alpha+1}$, which evaluates to 0.5). However, the ability to use polynomials not guaranteed to be irreducible can be useful in some cases — see section 3.3.6. In particular, algorithm 7 asks if certain expressions are invertible, and a 'no' answer here entrains a splitting into cases, just as asking "is $\alpha - 1$ zero?" entrains a splitting of `RootOf(x^2-1,x)`.

3.1.5 Solutions in Real Radicals

We have seen above, both in the case of the cubic, equation (3.10), and the quartic, equation (3.12), that real roots may need to be expressed via complex radicals, even if all the root are real. Indeed, in the case of the cubic, this is necessary. However, the quartic $x^4 + 4\,x^3 + x^2 - 6\,x + 2$, whose roots are

$$\left\{-1 + \sqrt{3}, -1 - \sqrt{3}, -1 + \sqrt{2}, -1 - \sqrt{2}\right\}$$

shows that polynomials *can* have real roots expressible in terms of real radicals ,and a slightly less obvious example is given by $x^4 + 4\,x^3 - 44\,x^2 - 96\,x + 552$,

[2] The precise statement is as follows. For all $n \geq 5$, the fraction of polynomials of degree n and coefficients at most H which have a root expressible in radicals tends to zero as H tends to infinity.

[3] By default, Maple will also use this formulation for roots of most quartics, and the expression in figure 3.2 is obtained by `convert(%,radical)` and then locating the sub-expressions by hand. This can be seen as an application of Carette's view of simplification (page 14), though historically Carette's paper is a retrospective justification.

whose roots are

$$\left\{-1-\sqrt{25+2\sqrt{6}}, -1+\sqrt{25+2\sqrt{6}}, -1-\sqrt{25-2\sqrt{6}}, -1+\sqrt{25-2\sqrt{6}}\right\}.$$

There is a little-known theorem in this area.

Theorem 9 ([Isa85]) *Suppose that all the roots of an irreducible polynomial $f(x)$ over $\mathbf{Q}$ are real. Then if any root of the polynomial is expressible in radicals, the degree of the polynomial must be a power of two.*

3.1.6 Equations of curves

For a fuller description of this topic, see [Ful69]. In particular, we only consider the affine case, whereas the projective case (i.e. allowing for "points at infinity") is in many ways more general.

Definition 31 *An (affine)* algebraic curve $C(x_1, \ldots, x_n)$ *in n dimensions over a field K is the set of solutions $(x_1, \ldots, x_n)$ to $n-1$ independent algebraic equations, i.e. polynomials $g_i(x_1, \ldots, x_n) = 0$.*

If $n = 2$ we say that we have a *plane* algebraic curve.

Of course, the precise curve and equations are often not very interesting: for instance we would like to think that the parabola $x_1^2 - x_2$ was "the same" as $y_1 - y_2^2$, and so on.

Definition 32 *Two curves $C(x_1, \ldots, x_n)$ and $C'(y_1, \ldots, y_m)$ are said to be* birationally equivalent *if there are two families of rational functions*

$$F = (f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$$

and

$$G = (g_1(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m))$$

such that:

1. *for almost all $(x_1, \ldots, x_n) \in C$, $(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$ is defined and $\in C'$;*
2. *for almost all $(y_1, \ldots, y_m) \in C'$, $g_1(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m)$ is defined and $\in C$;*
3. *almost everywhere, F and G are mutually inverse, i.e.*

$$f_i(g_1(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m)) = y_i$$

and

$$g_j(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)) = x_j.$$

"Almost everywhere" means "on a non-empty Zariski open set" [Ful69,], and can be thought of as "except where we get $\frac{0}{0}$ behaviour".

Theorem 10 *Every algebraic curve is birationally equivalent to a plane curve.*

Proof. If there are more than two variables, there is more than one equation, and we can use resultants to eliminate one variable and one equation.

We then have the concept [Sen08] of a curve being *soluble by radicals*. In this case, the generic curve of degree greater than six is not soluble by radicals [Zar26]. However, many "interesting" curves are soluble by radicals.

Proposition 17 *[Sen08, Corollary 3.2] Every irreducible plane curve of degree at most five is soluble by radicals.*

Proposition 18 *[Sen08, Corollary 3.3] Every irreducible singular plane curve of degree at most six is soluble by radicals.*

It is also the case[4] that the *offset*, i.e. the curve defined as the set of points a fixed distance d from the original curve, to a curve soluble by radicals is also soluble by radicals.

3.1.7 How many real roots?

We have seen that it is not obvious how many *real* roots a polynomial has: can we answer that question, or more formally the following?

Problem 1 *Given a square-free polynomial f, determine how many real roots f has, and describe each real root sufficiently precisely.* Many authors have asked the same question about non-square-free polynomials, and have laboured to produce better theoretical complexity bounds, since the square-free part of a polynomial may have larger coefficients than the original polynomial. However, in practice it is always better to compute the square-free part first.

Definition 27 introduced the concept of a signed polynomial remainder sequence, also called a Sturm–Habicht sequence: f_i is proportional by a *positive* constant to $-\mathrm{rem}(f_{i-2}, f_{i-1})$. The positive constant is normally chosen to keep the coefficients integral and as small as possible.

Definition 33 *If f is a square-free polynomial, let $V_f(a)$ denote the number of sign changes in the sequence $f_0(a)$, $f_1(a)$, ..., $f_n(a)$, where f_0, ..., f_n is the Sturm–Habicht sequence of f and f', also known as the Sturm sequence of f.*

If f is not square-free, we need more careful definitions [BPR06], and to be clear whether we are counting with multiplicity or not.

[4] Unpublished. Prof. Sendra has supplied this proof for plane curves.

Let (R_1, R_2) be a square parametrization of the curve and $K_0 = \mathbf{C}(t) \subset K_1 \subset \ldots \subset K_s$ be a (radical) field tower such that $R_1, R_2 \in K_s$. Considering the formal derivation with respect to t, one can deduce (for instance by induction on s) that if $R \in K_s$ then its derivative R' is also in K_s.

Now consider $a = (R_1')^2 + (R_2')^2 \in K_s$ and $K_{s+1} = K_s(\sqrt{a})$, then $(O_1, O_2) = (R_1, R_2) \pm d/\sqrt{a}(-(R_2)', (R_1)') \in (K_{s+1})^2$. So (O_1, O_2) is radical with the tower $K_0 \subset \ldots \subset K_s \subset K_{s+1}$.

Theorem 11 (Sturm) *If $a < b$ are not zeros of f, and f is square-free, then $V_f(a) - V_f(b)$ is the number of zeros of f in (a, b).*

$V_f(\infty)$ (which can be regarded as $\lim_{a\to\infty} V_f(a)$) can be computed as the number of sign changes in the sequence of leading coefficients of the Sturm sequence. Similarly, $V_f(-\infty)$ is the number of sign changes in the sequence of leading coefficients of the Sturm sequence with the signs of the odd-degree terms reversed. Hence $V_f(-\infty) - V_f(\infty)$, the total number of real roots, is easily computed from the Sturm sequence.

While the obvious way of computing $V_f(a)$ is by the definition, i.e. evaluating $f_0(a)$, ..., this turns out not to be the most efficient. Rather, while computing the Sturm sequence $f_0 \ldots$, we should also store the quotients q_i, so that $f_i(x) = -(f_{i-2}(x) - q_i(x)f_{i-1}(x))$. We then compute as follows.

Algorithm 4 (Sturm Sequence evaluation)

	a:	*A number*
Input:	$f_n(x)$:	*Last non-zero element of Sturm sequence of f*
	$q_i(x)$:	*Quotient sequence from Sturm sequence of f*

Output: *Sequence L of $f_n(a)$, $f_{n-1}(a)$, ..., $f_0(a)$.*

$L[n] := f_n(a)$;
$L[n-1] := q_{n+1}(a)L[n]$;
for $i = n \ldots 2$
  $L[i-2] := q_i(a)L[i-1] - L[i]$;
return L

If f has degree n, coefficients of bit-length at most τ, a has numerator and denominator of bit-length σ, this algorithm has asymptotic complexity $\tilde{O}(d^2 \max(\sigma, \tau))$ [LR01].

Since it is possible to say how big the roots of a polynomial can be (propositions 59, 60 and 61), we can determine, as precisely as we wish, the location of the real roots of a univariate polynomial: every time the Sturm sequence says that there are more than one root in an interval, we divide the interval in two, and re-compute $V(a) - V(b)$ for each half.

This is far from the only way of counting and locating real roots, i.e. solving problem 1: other methods are based on Descartes'[5] rule of signs (Theorem 23: the number of roots of f in $(0, \infty)$ is less than or equal to, by an even number, the number of sign changes in the coefficients of f) [CA76], its generalisation the Budan–Fourier theorem [Hur12] (Corollaries 21 and 22: the number of roots of f in[6] $[a, b]$ is less than or equal to, by an even number, the number of sign changes in the derivatives of f evaluated at a (i.e. $f(a)$, $f'(a)$, $f''(a) \ldots$) less the same evaluated at b), on continued fractions [TE07], or on numerical methods [Pan02].

[5] This rule is always called after Descartes, though the proof actually seems to be due to Gauss [BF93].

[6] We assume neither a nor b are roots.

3.2 Linear Equations in Several Variables

We now consider the case of several polynomial equations in several (not necessarily the same number) of variables.

Notation 8 *The variables will be called $x_1, \ldots, x_n$, though in specific examples we may use x, y or x, y, z etc.*

3.2.1 Linear Equations and Matrices

A typical set of 3-by-3 linear equations might look like the following.

$$\begin{aligned} 2x + 3y - 4z &= a; \\ 3x - 2y + 2z &= b; \\ 4x - 3y - 2z &= c. \end{aligned}$$

If we denote by $\mathbf{M}$ the matrix $\begin{pmatrix} 2 & 3 & -4 \\ 3 & -2 & 2 \\ 4 & -3 & -1 \end{pmatrix}$, $\mathbf{x}$ the (column) vector (x, y, z) and $\mathbf{a}$ the (column) vector (a, b, c), then this becomes the single matrix equation

$$\mathbf{M}.\mathbf{x} = \mathbf{a}, \tag{3.13}$$

which has, assuming $\mathbf{M}$ is invertible, the well-known solution

$$\mathbf{x} = \mathbf{M}^{-1}.\mathbf{a}. \tag{3.14}$$

This poses two questions: how do we store matrices, and how do we compute inverses?

3.2.2 Representations of Matrices

The first question that comes to mind here is "dense or sparse?", as in definition 19 (page 23). For a dense representation of matrices, the solution is obvious: we store a two-dimensional array (or one-dimensional array of one-dimensional arrays if our language does not support two-dimensional arrays) containing the values $m_{i,j}$ of the elements of the matrix. The algorithms for adding and multiplying dense matrices are pretty obvious, though in fact it is possible to multiply two 2×2 matrices with seven multiplications of entries rather than the obvious eight [Str69, Win71]: this leads to being able to multiply two $n \times n$ matrices with $O(n^{\log_2 7 \approx 2.807})$ element multiplications rather than $O(n^3)$. In theory one can do better than this, $O(n^{2.376})$, [CW90], but these methods requite unfeasably large[7] n to be cost-effective.

For a sparse representation of matrices, we have more choices.

[7] Even Strassen's method, with floating-point numbers, has break-even points between 400 and 2000 rows (160,000 to 4 million elements) [DN07].

row-sparse Here M is stored as a one–dimensional array of rows: the i-th row consisting of a list of pairs $(j, m_{i,j})$. This representation is equivalent to that of the sparse polynomial $\sum_j m_{i,j}x^j$, and this technique has been used in practice [CD85] and has a useful analogue in the case of non-linear equations (see section 3.3.3).

column-sparse Here M is stored as a one–dimensional array of columns: the j-th column consisting of a list of pairs $(i, m_{i,j})$.

totally sparse Here M is stored as a list of triples $(i, j, m_{i,j})$.

structured There are a large variety of special structures of matrices familar to numerical analysts, such as 'banded', 'Toeplitz' etc. Each of these can be stored efficiently according to their special form. In fact, Toeplitz matrices, of the form

$$\begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_{n-1} & a_n \\ a_n & a_1 & a_2 & a_3 & \dots & a_{n-1} \\ \vdots & \ddots & \dots & \dots & \ddots & \vdots \\ a_2 & a_3 & \dots & a_{n-1} & a_n & a_1 \end{pmatrix},$$

are, strictly speaking, dense, but only have n distinct entries, so are "information-sparse".

Clearly, if the matrix is structured, we should use the corresponding representation. For randomly sparse matrices, the choice depends on what we are doing with the matrix: if it is row operations, then row-sparse is best, and so on. One big issue with sparse matrices is known as *fill-in* — the tendency for operations on sparse matrices to yield less sparse matrices. For example, if we multiply two $n \times n$ matrices, each with e non-zero elements per row, with $n \gg e$, we would expect, assuming the non-zero elements are scattered at random, the resulting matrix to have e^2 non-zero elements per row. This has some apparently paradoxical consequences. Suppose M and N are two such matrices, and we wish to compute MNv for many vectors v. Clearly, we compute MN once and for all, and multiply this by v, and for dense M and N, this is right if there are more than n such vectors v. But, if $n \gg e > 2$, this is not optimal, since, once MN is computed, computing $(MN)v$ requires ne^2 operations, while compluting Nv requires ne, as does computing $M(Nv)$, totalling $2ne < ne^2$.

3.2.3 Matrix Inverses: not a good idea!

The first response to the question "how do we compute matrix inverses" ought to be "are you *sure* you want to?" Solving (as opposed to thinking about the solution of) equation (3.13) via equation (3.14) is often not the best way to proceed. Gaussian elimination (possibly using some of the techniques described later in this section) directly on equation (3.13) is generally the best way. This is particularly true if $\mathbf{M}$ is sparse, since $\mathbf{M}^{-1}$ is generally not sparse — an extreme

example of fill-in. Indeed, special techniques are generally used for the solution of large sparse systems, particularly those arising in integer factorisation or other cryptographic applications [HD03].

The usual method of solving linear equations, or computing the inverse of a matrix, is via Gaussian elimination, i.e. transforming equation (3.13) into one in which $\mathbf{M}$ is upper triangular, and then back-substituting. This transformation is done by row operations, which amount to adding/subtracting multiples of one row from another, since

$$P = Q \quad \& \quad R = S \quad \text{implies} \quad P + \lambda R = Q + \lambda S. \tag{3.15}$$

If we try this on the above example, we deduce successively that $z = a - 18b + 13c$, $y = a - 14b + 10c$ and $x = a - 15b + 11c$. Emboldened by this we might try a larger matrix:

$$M = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}.$$

After clearing out the first column, we get the matrix

$$\begin{pmatrix} a & b & c & d \\ 0 & -\frac{eb}{a} + f & -\frac{ec}{a} + g & -\frac{ed}{a} + h \\ 0 & -\frac{ib}{a} + j & -\frac{ic}{a} + k & -\frac{id}{a} + l \\ 0 & -\frac{mb}{a} + n & -\frac{mc}{a} + o & -\frac{md}{a} + p \end{pmatrix}.$$

Clearing the second column gives us

$$\begin{pmatrix} a & b & c & d \\ 0 & -\frac{eb}{a} + f & -\frac{ec}{a} + g & -\frac{ed}{a} + h \\ 0 & 0 & -\frac{\left(-\frac{ib}{a}+j\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{ic}{a} + k & \frac{-\left(-\frac{ib}{a}+j\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{id}{a} + l \\ 0 & 0 & -\frac{\left(-\frac{mb}{a}+n\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{mc}{a} + o & \frac{\left(\frac{mb}{a}-n\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)} - \frac{md}{a} + p \end{pmatrix},$$

which we can "simplify" to

$$\begin{pmatrix} a & b & c & d \\ 0 & \frac{-eb+af}{a} & \frac{-ec+ag}{a} & \frac{-ed+ah}{a} \\ 0 & 0 & \frac{afk-agj-ebk+ecj+ibg-icf}{-eb+af} & \frac{afl-ahj-ebl+edj+ibh-idf}{-eb+af} \\ 0 & 0 & \frac{afo-agn-ebo+ecn+mbg-mcf}{-eb+af} & \frac{afp-ahn-ebp+edn+mbh-mdf}{-eb+af} \end{pmatrix}. \tag{3.16}$$

After clearing the third column, the last element of the matrix is

$$-\left(\frac{-\left(-\frac{ib}{a}+j\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{id}{a}+l\right)\left(\frac{-\left(-\frac{mb}{a}+n\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{mc}{a}+o\right)\times$$

$$\left(\frac{-\left(-\frac{ib}{a}+j\right)\left(-\frac{ec}{a}+g\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{ic}{a}+k\right)^{-1}-\frac{\left(-\frac{mb}{a}+n\right)\left(-\frac{ed}{a}+h\right)}{\left(-\frac{eb}{a}+f\right)}-\frac{md}{a}+p.$$

This simplifies to

$$-\frac{\begin{array}{c}-afkp+aflo+ajgp-ajho-angl+anhk+ebkp-eblo\\-ejcp+ejdo+encl-endk-ibgp+ibho+ifcp-ifdo\\-inch+indg+mbgl-mbhk-mfcl+mfdk+mjch-mjdg\end{array}}{afk-agj-ebk+ecj+ibg-icf}. \quad (3.17)$$

The numerator of this expression is in fact the determinant of the original matrix, $|M|$.

In general, for an $n \times n$ matrix, we would perform $O(n^3)$ computations with rational functions, which would, if we were to simplify, involve g.c.d. computations, often costly.

Can we do better? We could take a leaf out of the calculation on page 36, and not introduce fractions, but rather cross-multiply. If we do this while clearing column one, we get

$$\begin{pmatrix} a & b & c & d \\ 0 & -eb+af & -ec+ag & -ed+ah \\ 0 & aj-ib & ak-ic & al-id \\ 0 & -mb+an & ao-mc & ap-md \end{pmatrix}.$$

After clearing column two,we get

$$\begin{pmatrix} a & b & c & d \\ 0 & -eb+af & -ec+ag & -ed+ah \\ 0 & 0 & \begin{array}{c}(-aj+ib)(-ec+ag)+\\(-eb+af)(ak-ic)\end{array} & \begin{array}{c}(-aj+ib)(-ed+ah)+\\(-eb+af)(al-id)\end{array} \\ 0 & 0 & \begin{array}{c}(-an+mb)(-ec+ag)+\\(-eb+af)(ao-mc)\end{array} & \begin{array}{c}(-an+mb)(-ed+ah)+\\(-eb+af)(ap-md)\end{array} \end{pmatrix}.$$

The result of the next step is better contemplated than printed!

However, if we do contemplate the result printed above, we see that rows 3 and 4 contain polynomials of degree four, whereas in the "simplified" form (3.16) we only have polynomials of degree three in the numerators. Indeed, if we were to expand the matrix above, we would observe that rows three and four each had a common factor of a. Similarly, if we were to (or were to get a

computer algebra system to) expand and then factor the last step, we would get $a^2(af-eb)|M|$, as in equation (3.17). Such common factors are not a coïncidence (indeed, they cannot be, since M is the most general 4×4 matrix possible).

Theorem 12 (Dodgson–Bareiss [Bar68, Dod66]) [8] *Let $a_{i,j}^{(k)}$ be the determinant*

$$\begin{vmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,k} & a_{1,j} \\ a_{2,1} & a_{2,2} & \dots & a_{2,k} & a_{2,j} \\ \dots & \dots & \dots & \dots & \dots \\ a_{k,1} & a_{k,2} & \dots & a_{k,k} & a_{k,j} \\ a_{i,1} & a_{i,2} & \dots & a_{i,k} & a_{i,j} \end{vmatrix},$$

i.e. that of rows $1 \dots k$ and i, with columns $1 \dots k$ and j. In particular, the determinant of the matrix of size n whose elements are $(a_{i,j})$ is $a_{n,n}^{(n-1)}$ and $a_{i,j} = a_{i,j}^{(0)}$. Then (assuming $a_{0,0}^{(-1)} = 1$):

$$a_{i,j}^{(k)} = \frac{1}{a_{k-1,k-1}^{(k-2)}} \begin{vmatrix} a_{k,k}^{(k-1)} & a_{k,j}^{(k-1)} \\ a_{i,k}^{(k-1)} & a_{i,j}^{(k-1)} \end{vmatrix}.$$

Proof. By fairly tedious induction on k.

Corollary 3 (Bareiss' algorithm) *When doing fraction-free Gaussian elimination, after clearing column k, every elements of rows $k+1 \dots n$ is divisible by $a_{k-1,k-1}^{(k-2)}$.*

This is actually the 'one-step' variant of Bareiss [Bar68]: there are other variants with more advanced look-ahead, but they do not (and can not) cancel any more in general. This result accounts for the factor of a observed in rows 3 and 4 above, and for the factors of a^2 and $af - eb$ in the last step. Cancelling the a in rows 3 and 4 would in fact automatically prevent the a^2 from being generated — far better than generating it and then cancelling it!

Corollary 4 *If the initial entries are integers of length m (resp. polynomials of degree m), then after k steps, the entries will have length (resp. degree) $O(km)$.*

This is to be contrasted with the $O(2^k m)$ of the naïve fraction-free approach.

It is possible to view Euclid's algorithm for polynomials as Gaussian elimination in a matrix (Sylvester's matrix — definition 65) of coefficients, and the factors β_i that are cancelled by the sub-resultant variant for normal polynomial remainder sequences (footnote 23 on page 39) are those predicted by corollary 3 above.

3.2.4 Over/under-determined Systems

So far we have implicitly assumed that there are as many equations as there are unknowns, and that the equations determine the unknowns precisely (in

[8] The Oxford logician Charles Dodgson was better known as Lewis Carroll.

other words, that the determinant of the corrsponding matrix is non-zero). What happens if these assumptions do not hold? There are several cases to be distinguished.

Over-determined and consistent Here the 'extra' equations are consistent with those that determine the solution. A trivial example in one variable would be the pair $2x = 4$, $3x = 6$.

Over-determined and inconsistent Here the 'extra' equations are not consistent with those that determine the solution. A trivial example in one variable would be the pair $2x = 4$, $3x = 9$, where the first implies that $x = 2$, but the second that $x = 3$.

Spuriously over-determined This is a generalisation of "over-determined and consistent" when, after deleting the 'extra' equations that convery no new information, we are left with an under-determined system.

Under-determined and consistent Here there are not enough equations (possibly after deleting spurious ones) to determine all the variables. An example would be $x+y = 3$. Here x can be anything, but, once x is chosen, y is fixed as $3-x$. Equally, we could say that y can be anything, but, once y is chosen, x is fixed as $3-y$. The solutions form a k-dimensional hyper-plane, where k is the number of variables minus the number of (non-spurious) equations.

Under-determined yet inconsistent Here the equations (possibly after deleting spurious ones) are still inconsistent. One example would be $x + 2y + 3z = 1$, $2x + 4x + 6z = 3$.

We are then left with three possibilities for the solutions, which can be categorised in terms of the dimension ('dim').

dim $= -1$ This is the conventional 'dimension' assigned when there are no solutions, i.e. the equations are inconsistent.

dim $= 0$ Precisely one solution.

dim > 0 An infinite number of solutions, forming a hyperplane of dimension dim.

3.3 Nonlinear Equations in Several Variables

Most of the section has its origin in the pioneering work of Buchberger [Buc70]. Some good modern texts are [AL94, BW93, CLO06].

If the equations are nonlinear, equation (3.15) is still available to us. So, given the three equations

$$x^2 - y = 0 \qquad x^2 - z = 0 \qquad y + z = 0,$$

we can subtract the first from the second to get $y-z=0$, hence $y=0$ and $z=0$, and we are left with $x^2=0$, so $x=0$, albeit with multiplicity 2 (definition 30). However, we can do more than this. Given the two equations

$$x^2-1=0 \qquad xy-1=0, \tag{3.18}$$

there might seem to be no row operation available. But in fact we can subtract x times the second equation from y times the first, to get $x-y=0$. Hence the solutions are $x=\pm 1$, $y=x$.

We can generalise equation (3.15) to read as follows: for all polynomials f and g,

$$P=Q \quad \& \quad R=S \quad \text{implies} \quad fP+gR=fQ+gS. \tag{3.19}$$

Lemma 4 *In equation (3.19), it suffices to consider terms (monomials with leading coefficients) for f and g rather than general polynomials.*

Proof. Let f be $\sum a_i m_i$ and g be $\sum b_i m_i$, where the m_i are monomials and the a_i and b_i coefficients (possibly zero, but for a given i, both a_i and b_i should not be zero, since then m_i would be redundant). Then for each i, the monomial version of equation (3.19) gives

$$P=Q \quad \& \quad R=S \quad \text{implies} \quad a_i m_i P + b_i m_i R = a_i m_i Q + b_i m_i S.$$

Then we can use equation (3.15) repeatedly, with $\lambda=1$, to add these together to get the general form of equation (3.19).

Because of equation (3.2), we can regard equations as synonymous with polynomials. Equation (3.19) then motivates the following definition.

Definition 34 *Let S be a set of polynomials in the variables $x_1,\ldots,x_n$, with coefficients from R. The* ideal generated by *S, denoted (S), is the set of all finite sums $\sum f_i s_i$: $s_i \in S$, $f_i \in R[x_1,\ldots,x_n]$. If S generates I, we say that S is a* basis *for I.*

Proposition 19 *This is indeed an ideal in the sense of definition 5.*

Strictly speaking, what we have defined here is the *left ideal*: there are also concepts of *right ideal* and *two-sided ideal*, but all concepts agree in the case of commutative polynomials, which we will assume until section 3.3.13.

Proposition 20 $((S))=(S)$.

Definition 35 *Two sets of polynomial equations are* equivalent *if the polynomials defining the left-hand sides generate the same ideal. We will see how to test this in corollary 5.*

Just as an upper triangular matrix is a nice formulation of a set of linear equations, allowing us to "read off", the solutions, so we would like a similarly 'nice' basis for an ideal generated by non-linear equations. In order to do this, we will regard our polynomials in a distributed format, with the terms sorted in some admissible (page 27) ordering $>$. Note that we are *not* requiring that the polynomials are stored this way in an algebra system, though in fact most algebra systems specialising in this area will do so: we are merely discussing the mathematics of such polynomials. Having fixed such an ordering $>$, we can define the following concepts.

Definition 36 *If f is a non-zero polynomial, the* leading term *of f, denoted* $\mathrm{lt}(f)$*, is that term greatest with respect to $>$. The corresponding monomial is called the* leading monomial *of f,* $\mathrm{lm}(f)$*. We will sometimes apply* lm *to sets, where* $\mathrm{lm}(S) = \{\mathrm{lm}(s) | s \in S\}$.

"Monomial algebra" is a particularly simple form of polynomial algebra: in particular

$$\begin{aligned} \gcd\left(\prod_{i=1}^{n} x_i^{a_i}, \prod_{i=1}^{n} x_i^{b_i}\right) &= \prod_{i=1}^{n} x_i^{\min(a_i,b_i)}, \\ \mathrm{lcm}\left(\prod_{i=1}^{n} x_i^{a_i}, \prod_{i=1}^{n} x_i^{b_i}\right) &= \prod_{i=1}^{n} x_i^{\max(a_i,b_i)}. \end{aligned}$$

Definition 37 *If* $\mathrm{lm}(g)$ *divides* $\mathrm{lm}(f)$*, then we say that g reduces f to $h = \mathrm{lc}(g)f - (\mathrm{lt}(f)/\mathrm{lm}(g))g$, written $f \to^g h$. Otherwise we say that f is* reduced *with respect to g.* The Maple user should note that Maple's `Reduce` command actually implements *complete* reduction — see Definition 38.

If R is a field, division is possible, and so it is more usual to reduce f to $f - (\mathrm{lt}(f)/\mathrm{lt}(g))g$. In the construction of h, the leading terms of both $\mathrm{lc}(g)f$ and $(\mathrm{lt}(f)/\mathrm{lm}(g))g$ are $\mathrm{lc}(f)\mathrm{lc}(g)\mathrm{lm}(f)$, and so cancel. Hence $\mathrm{lm}(h) < \mathrm{lm}(f)$. This observation and theorem 3 give us the following result.

Proposition 21 *Any chain $f_1 \to^g f_2 \to^g f_3 \cdots$ is finite, i.e. terminates in a polynomial h reduced with respect to g. We write $f_1 \overset{*}{\to}{}^{g} h$.*

These concepts and results extend to reduction by a set G of polynomials, where $f \to^G h$ means $\exists g \in G : f \to^g h$. We must note that a polynomial can have several reductions with respect to G (one for each element of G whose leading monomial divides the leading monomial of f). For example, let $G = \{g_1 = x - 1, g_2 = y - 2\}$ and $f = xy$. Then there are two possible reductions of f: $f \to^{g_1} h_1 = f - yg_1 = y$, and $f \to^{g_2} h_2 = f - xg_2 = 2x$. In this case $h_1 \to^{g_2} 2$ and $h_2 \to^{g_1} 2$, so that $f \overset{*}{\to}{}^{G} 2$ uniquely, but even this need not always be the case. If we let $G = \{g_1 = x-1, g_2 = x^2\}$ and $f = x^2-1$, then $f \to^{g_2} h_2 = f-g_2 = -1$, whereas $f \to^{g_1} f - xg_1 = x - 1 \to^{g_1} 0$: so $f \overset{*}{\to}{}^{G}$ 0 or -1.

This definition deals with reduction of the leading monomial of f by g, but it might be that other monomials are reducible. For simplicity we consider the case when R is a field.

Definition 38 *If any term cm of f is reducible by g, i.e. the leading monomial of g divides m, we say that g part-reduces f, and write $f \Rightarrow^g f - (cm/\mathrm{lt}(g))g$.*

We can continue this process (only finitely often, by repeated application of theorem 3), until no monomial of f is reducible by g, when we write $f \overset{*}{\Rightarrow}^g h$, and say that f is *completely reduced* by g to h. Again, this extends to reduction by a set of polynomials.

In section 3.2.1, we performed row operations: subtracting a multiple of one row from another, which is essentially what reduction does, except that the 'multiple' can include a monomial factor. It turns out that we require a more general concept, given in the next definition.

Definition 39 *Let $f, g \in R[x_1, \ldots, x_n]$. The S-polynomial of f and g, written $S(f, g)$ is defined as*

$$S(f,g) = \frac{\mathrm{lt}(g)}{\gcd(\mathrm{lm}(f), \mathrm{lm}(g))} f - \frac{\mathrm{lt}(f)}{\gcd(\mathrm{lm}(f), \mathrm{lm}(g))} g. \tag{3.20}$$

We note that the divisions concerned are exact, and that this generalises reduction in the sense that, if $\mathrm{lm}(g)$ divides $\mathrm{lm}(f)$, then $f \rightarrow^g S(f, g)$. As with reduction, the leading monomials in the two components on the righthand side of equation (3.20) cancel. Another way of thinking of the S-polynomial (when R is a field) is that it is the difference between what you get by reducing $\mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))$ by f and by g.

Proposition 22 $S(f, g) = -S(g, f)$.

Proposition 23 $S(f, g) \in (\{f, g\})$.

3.3.1 Gröbner Bases

From now until section 3.3.12, we will assume that R is a field. However, we will continue to use R, and not gratuitously make polynomials monic, since this can be expensive.

Theorem 13 *[BW93, Proposition 5.38, Theorem 5.48] The following conditions on a set $G \in R[x_1, \ldots, x_n]$, with a fixed ordering $>$ on monomials, are equivalent.*

1. $\forall f, g \in G, S(f, g) \overset{*}{\rightarrow}^G 0$.
2. *If $f \overset{*}{\rightarrow}^G g_1$ and $f \overset{*}{\rightarrow}^G g_2$, then g_1 and g_2 differ at most by a multiple in R, i.e. $\overset{*}{\rightarrow}^G$ is essentially well-defined.*
3. $\forall f \in (G), f \overset{*}{\rightarrow}^G 0$.

4. $(\mathrm{lm}(G)) = (\mathrm{lm}((G)))$, *i.e. the leading monomials of* G *generate the same ideals as the leading monomials of the whole of* (G).

If G *satisfies these conditions,* G *is called a* Gröbner base (or standard basis).

These are very different kinds of conditions, and the strength of Gröbner theory lies in their interplay. Condition 2 underpins the others: $\stackrel{*}{\rightarrow}^{G}$ is well-defined. Condition 1 looks technical, but has the great advantage that, for finite G, it is finitely checkable: if G has k elements, we take the $k(k-1)/2$ unordered (by proposition 22) pairs from G, compute the S-polynomials, and check that they reduce to zero. This gives us either a proof or an explicit counter-example (which is the key to algorithm 5). Since $f \stackrel{*}{\rightarrow}^{G} 0$ means that $f \in (G)$, condition 3 means that ideal membership is testable if we have a Gröbner base for the ideal. Condition 4 can be seen as a generalisation of "upper triangular" — see section 3.3.3.

Now let G and H be Gröbner bases, possibly with respect to different orderings.

Proposition 24 *If* $\forall g \in G, g \stackrel{*}{\rightarrow}^{H} 0$, *then* $(G) \subseteq (H)$.

Proof. Let $f \in (G)$. Then $f \stackrel{*}{\rightarrow}^{G} 0$, so $f = \sum c_i g_i$. But $g_i \stackrel{*}{\rightarrow}^{H} 0$, so $g_i = sum d_{ij} h_j$. Therefore $f = sum_j \left(\sum_i c_i d_{ij}\right) h_j$, and so $f \in (H)$.

Corollary 5 *If* $\forall g \in G, g \stackrel{*}{\rightarrow}^{H} 0$, *and* $\forall h \in H, h \stackrel{*}{\rightarrow}^{G} 0$, *then* $(G) = (H)$.

Over a field, a particularly useful Gröbner base is a *completely reduced Gröbner base* (abbreviated *crGb*) G, i.e. one where every element is completely reduced with respect to all the others: in symbols

$$\forall g \in G \qquad g \stackrel{*}{\Rightarrow}^{G \setminus \{g\}} g.$$

For a consistent set of linear polynomials, the crGb would be a set of linear polynomials in one variable each, e.g. $\{x-1, y-2, z-3\}$, effectively the solution. In general, a crGb is a canonical (definition 3) form for an ideal: two ideals are equal if, and only if, they have the same crGb (with respect to the same ordering, of course).

Every polynomial ideal has a Gröbner base: we will show this constructively for finitely-generated[9] ideals over noetherian (definition 6) rings.

Algorithm 5 (Buchberger)
Input: *finite* $G_0 \subset R[x_1, \ldots, x_n]$*; monomial ordering* $>$.
Output: G *a Gröbner base for* (G_0) *with respect to* $>$.

[9] In fact, every polynomial ideal over a noetherian ring *is* finitely generated. However, it is possible to encode undecidability results in infinite descriptions of ideals, hence we say "finitely generated" to avoid this paradox.

$G := G_0$; $n := |G|$;
we consider G as $\{g_1, \ldots, g_n\}$
$P := \{(i,j) : 1 \leq i < j \leq n\}$
while $P \neq \emptyset$ **do**
    Pick $(i,j) \in P$;
    $P := P \setminus \{(i,j)\}$;
    Let $S(g_i, g_j) \overset{*}{\to}^{G} h$
    If $h \neq 0$ **then**
        # $\mathrm{lm}(h) \notin (\mathrm{lm}(G))$
        $g_{n+1} := h$; $G := G \cup \{h\}$;
        $P := P \cup \{(i, n+1) : 1 \leq i \leq n\}$;
        $n := n + 1$;

Proof. The polynomials added to G are reductions of S-polynomials of members of G, and hence are in the same ideal as G, and therefore of G_0. If this process terminates, then the result satisfies condition 1, and so is a Gröbner base for some ideal, and therefore the ideal of G. By proposition 23 and the properties of $\overset{*}{\to}^{G}$, $h \in (G)$, so (G) is constant throughout this process and G has to be a Gröbner base for (G_0). Is it possible for the process of adding new h to G, which implies increasing $(\mathrm{lm}(G))$, to go on for ever? No: corollary 1 says that the increasing chain of $(\mathrm{lm}(G))$ is finite, so at some point we cannot increase $(\mathrm{lm}(G))$ any further, i.e. we cannot add a new h.

Proposition 25 *Every finitely generated polynomial ideal over a field K has a completely reduced Gröbner base with respect to any given ordering, and this is unique up to order of elements and multiplication by elements of K^*.*

Hence, for a fixed ordering, a crGb is a "fingerprint" of an ideal, uniquely identifying it. This makes definition 35 algorithmic. It also allows ideal arithmetic.

Proposition 26 *Let G_1 and G_2 be Gröbner bases of the ideals I_1 and I_2 with respect to a fixed ordering. Then:*

1. $I_1 \lhd I_2$ *iff* $\forall g \in G_1 \qquad g \overset{*}{\to}^{G_2} 0$;
2. $I_1 + I_2 = (G_1 \cup G_2)$;
3. $I_1 I_2 = (\{g_1 g_2 \mid g_1 \in G_1, g_2 \in G_2\})$.

Furthermore, all these processes are algorithmic.

We have proved nothing about the running time of Buchberger's algorithm. Indeed, "algorithm" is almost an over-statement: we have not specified the choice of $(i,j) \in P$ at all. It turns out that the complexity, though *not* the correctness, of the algorithm is strongly dependent on this choice.

Proposition 27 (Buchberger's gcd (or First) Criterion [Buc79]) *If*

$$\gcd(\mathrm{lm}(f), \mathrm{lm}(g)) = 1, \tag{3.21}$$

then $S(f,g) \overset{*}{\to}^{\{f,g\}} 0$.

In practice, this is implemented by not even adding to P, either in the initial construction or when augmenting it due to a new h, pairs for which equation (3.21) is satisfied. This proposition also explains why the more general construct of an S-polynomial is not relevant to linear equations: when f and g are linear, if they have the same leading variable, one can reduce the other, and if they do not, then the S-polynomial reduces to zero.

Proposition 28 (Buchberger's lcm (or Third) Criterion [Buc79]) *If* $I = (B)$ *contains* f, g, h *and the reductions under* $\overset{*}{\to}^{B}$ *of* $S(f,g)$ *and* $S(f,h)$, *and if both* $\mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))$ *and* $\mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(h))$ *divide* $\mathrm{lcm}(\mathrm{lm}(g), \mathrm{lm}(h))$, *then* $S(g,h) \overset{*}{\to}^{B} 0$, *and hence need not be computed.*

This has been generalised to a chain of polynomials f_i connecting g and h: see [BF91].

Propositions 27 and 28 are therefore *sufficient* to say that we need not compute an S-polynomial: the question of whether they are *necessary* is discussed by [HP07]. Terminology varies in this area, and some refer to Buchberger's *Second* Criterion as well. The more descriptive gcd/lcm terminology is taken from [Per09].

3.3.2 How many Solutions?

Here we will try to give an analysis of the various possibilities for the number of solutions of a set of polynomial equations. We will assume that a crGb for the polynomials has been computed, which therefore cannot be overdetermined in the sense of having redundant equations. However, we may still need more equations than variables — see the examples at the start of section 3.3.6.

Unlike section 3.2.4 however, we have to ask ourselves "in which domain are the solutions?" We saw in theorem 8 that, even for an equation in one variable, the 'solutions' may have no simpler formulation than 'this is a root of $p(x)$'. Fortunately, this is all that we need. We will assume that K is the algebraic closure (definition 15) of (the field of fractions of) R.

Definition 40 *The set of solutions over* K *of an ideal* I *is called the* variety *of* I, *written* $V(I)$. *If* S *is a set of polynomials which generates* I, *so* $I = \langle S \rangle$, *we will write* $V(S)$ *as shorthand for* $V(\langle S \rangle)$.

We should note that two different ideals can have the same variety, e.g. (x) and (x^2) both have the variety $x = 0$, but the solution has different multiplicity.

Definition 41 *The* radical *of an ideal* I, *denoted* $\sqrt{I}$, *is defined as*

$$\sqrt{I} = \{p | \forall \mathbf{x} \in V(I), p(\mathbf{x}) = 0\}.$$

If I is generated by a single polynomial p, $\sqrt{I}$ is generated by the square-free part of p.

Proposition 29 *$\sqrt{I}$ is itself an ideal.*

Proposition 30 $V(I_1 \cdot I_2) = V(I_1) \cup V(I_2)$.

Proposition 31 $V(I_1 \cup I_2) = V(I_1) \cap V(I_2)$.

Definition 42 *The* dimension *of an ideal I in $S = k[x_1, \ldots, x_n]$ is the maximum number of algebraically independent, over k, elements of the quotient S/I. We can also talk about the dimension of a variety.*

No solutions in K Here the crGb will be $\{1\}$, or more generally $\{c\}$ for some non-zero constant c. The existence of a solution would imply that this constant was zero, so there are no solutions. The dimension is undefined, but normally written as -1.

A finite number of solutions in K There is a neat generalisation of the result that a polynomial of degree n has n roots.

> **Proposition 32** *The number (with multiplicity) of solutions of a system with Gröbner basis G is equal to the number of monomials which are not reducible by G.*
>
> It follows from this that, if (and only if) there are finitely many solutions, every variable x_i must appear alone, to some power, as the leading monomial of some element of G. In this case, the dimension is zero. We return to this case in section 3.3.6.

An infinite number of solutions in K Then some variables do not occur alone, to some power, as the leading monomial of any element of G. In this case, the dimension is greater than zero.

While 'dimension', as defined above, is a convenient generalisation of the linear case, many more things can happen in the non-linear case. If the dimension of the ideal is d, there must be at least d variables which do not occur alone, to some power, as the leading monomial of any element of G. However, if $d > 0$, there may be more. Consider the ideal $(xy - 1) \lhd k[x, y]$. $\{xy - 1\}$ is already a Gröbner base, and neither x nor y occur alone, to any power, in a leading term (the only leading term is xy). However, the dimension is 1, not 2, because fixing x determines y, and *vice versa*, so there is only one independent variable. In the case of a triangular set (definition 44), we can do much better, as in proposition 33.

3.3.3 A Matrix Formulation

Equation (3.13) showed how a family of linear equations can be represented as a matrix equation. We can do the same with nonlinear equations: (3.18) can be written as

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x^2 \\ xy \\ x \\ y \\ 1 \end{pmatrix} = \mathbf{0} \tag{3.22}$$

However, this does not give us an obvious solution. Rather, we need to extend the system, allowing not just the original equations, but also y times the first and x times the second, to give the following.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x^2y \\ x^2 \\ xy \\ x \\ y \\ 1 \end{pmatrix} = \mathbf{0}. \tag{3.23}$$

Elimination in this gives us

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x^2y \\ x^2 \\ xy \\ x \\ y \\ 1 \end{pmatrix} = \mathbf{0}, \tag{3.24}$$

which produces, as the third row, the equation $y - x$, as we do (up to a change of sign) after (3.18). In pure linear algebra, we can do no further, since we really require y times this equation. This means considering

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x^2y^2 \\ x^2y \\ x^2 \\ xy^2 \\ xy \\ x \\ y^2 \\ y \\ 1 \end{pmatrix} = \mathbf{0}. \tag{3.25}$$

Eliminating here (using row 1 to kill the leading term in row 4, and the same with row 2 against row 5) gives

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x^2y^2 \\ x^2y \\ x^2 \\ xy^2 \\ xy \\ x \\ y^2 \\ y \\ 1 \end{pmatrix} = \mathbf{0}, \tag{3.26}$$

and now row 4 can kill the leading term in row 6, to give

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x^2y^2 \\ x^2y \\ x^2 \\ xy^2 \\ xy \\ x \\ y^2 \\ y \\ 1 \end{pmatrix} = \mathbf{0}. \tag{3.27}$$

The last line of this corresponds to $y^2 - 1$. To use this to deduce an equation for x, we would need to consider x times this equation, which would mean addig further rows and columns to the matrix.

No-one would actually suggest doing this in practice, any more than any-one would compute a g.c.d. in practice by building the Sylvester matrix (which is actually the univariate case of this process), but the fact that it exists can be useful in theory, as we will find that the Sylvester matrix formulation of g.c.d. computation is useful in section 4.1.

3.3.4 Orderings

In section 2.1.4, we defined an admissible ordering on monomials, and the theory so far is valid for all orderings. What sort of orderings are admissible? We first need an ordering on the variables themselves, which we will also denote $>$, and we will assume that $x_1 > \cdots > x_n$ (in examples, $x > y > z$). Suppose the two monomials to be compared are $A = x_1^{a_1} \dots x_n^{a_n}$ and $B = x_1^{b_1} \dots x_n^{b_n}$. These monomials have total degree $a = \sum_{i=1}^n a_i$ and $b = \sum_{i=1}^n b_i$.

purely lexicographic — `plex` in Maple We first compare a_1 and b_1. If they differ, this tells us whether $A > B$ ($a_1 > b_1$) or $A < B$ ($a_1 < b_1$). If they are the same, we go on to look at a_2 versus b_2 and so on. The order is similar to looking up words in a dictionary/lexicon — we look at the first letter, and after finding this, look at the second letter, and so on. In this order x^2 is more important than xy^{10}.

total degree, then lexicographic — `grlex` in Maple We first look at the total degrees: if $a > b$, then $A > B$, and $a < b$ means $A < B$. If $a = b$, then we look at lexicographic comparison. In this order xy^{10} is more important than x^2, and x^2y more important than xy^2.

total degree, then reverse lexicographic — `tdeg` in Maple This order is the same as the previous, except that, if the total degrees are equal, we look lexicographically, then *take the opposite*. Many systems, in particular Maple and Mathematica[10], reverse the order of the variables first. The reader may ask "if the order of the variables is reversed, and we then reverse the sense of the answer, what's the difference?". Indeed, for two variables, there is no difference. However, with more variables it does indeed make a difference. For three variables, the monomials of degree three are ordered as

$$x^3 > x^2y > x^2z > xy^2 > xyz > xz^2 > y^3 > y^2z > yz^2 > z^3$$

under `grlex`, but as

$$x^3 > x^2y > xy^2 > y^3 > x^2z > xyz > y^2z > xz^2 > yz^2 > z^3$$

under `tdeg`. One way of seeing the difference is to say that `grlex` with $x > y > z$ discriminates *in favour of* x, whereas `tdeg` with $z > y > x$ discriminates *against* z. This metaphor reinforces the fact that there is no difference with two variables.

It seems that `tdeg` is, in general, the most efficient order.

k-elimination Here we choose any order $>'$ on $x_1, \ldots, x_k$, and use that. If this cannot decide, we then use a second order $>''$ on $x_{k+1}, \ldots, x_n$. Since $>'$ is admissible, the least monomial is $x_1^0 \ldots x_k^0$, so this order will eliminate $x_1, \ldots, x_k$ as far as possible, in the sense that the polynomials in only $x_{k+1}, \ldots, x_n$ in a Gröbner base computed with such an order are all that can be deduced about these variables. It is common, but by no means required, to use `tdeg` for both $>'$ and $>''$. Note that this is not the same as simply using `tdeg`, since the exponents of $x_{k+1}, \ldots, x_n$ are not considered unless $x_1, \ldots, x_k$ gives a tie.

weighted orderings Here we compute the total degree with a weighting factor, e.g. we may weight x twice as much as y, so that the total degree of x^iy^j would be $2i + j$. This can come in lexicographic or reverse lexicographic variants.

matrix orderings These are in fact the most general form of orderings [Rob85]. Let $\mathbf{M}$ be a fixed $n \times n$ matrix of reals, and regard the exponents of A as an n-vector $\mathbf{a}$. Then we compare A and B by computing the two vectors $\mathbf{M}.\mathbf{a}$ and $\mathbf{M}.\mathbf{b}$, and comparing these lexicographically.

[10] `http://reference.wolfram.com/mathematica/tutorial/PolynomialOrderings.html`

lexicographic $\mathbf{M}$ is the identity matrix.

`grlex` $\mathbf{M} = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}$.

`tdeg` It would be tempting to say, by analogy with `grlex`, that the matrix is $\begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 0 & 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & \dots & 0 \end{pmatrix}$. However, this is actually `grlex` with the variable order reversed, not genuine reverse lexicographic. To get that, we need the matrix $\begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ -1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 0 \end{pmatrix}$, or, if we are adopting the Maple convention of reversing the variables as well, $\begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 0 & 0 & \dots & 0 & -1 \\ 0 & \dots & 0 & -1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & -1 & 0 & \dots & 0 \end{pmatrix}$.

k-elimination If the matrices are $\mathbf{M}_k$ for $>'$ and $\mathbf{M}_{n-k}$ for $>''$, then $\mathbf{M} = \begin{pmatrix} \mathbf{M}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{n-k} \end{pmatrix}$.

weighted orderings Here the first row of $\mathbf{M}$ corresponds to the weights, instead of being uniformly 1.

Most "serious" Gröbner systems[11] implement matrix orderings, but have special case implementations for the more common ones listed above, often storing the (weighted) total degree as well as the individual degrees to minimise recomputation.

3.3.5 Example

Consider the three polynomials below.

$$\begin{aligned} g_1 &= x^3yz - xz^2, \\ g_2 &= xy^2z - xyz, \\ g_3 &= x^2y^2 - z. \end{aligned}$$

[11] Such as SINGULAR [Sch03], CoCoA [Abb04] or Macauley [BS86].

The S-polynomials to be considered are $S(g_1, g_2)$, $S(g_1, g_3)$ and $S(g_2, g_3)$. We use a purely lexicographical ordering with $x > y > z$. The leading terms of $g_2 = xy^2z - xyz$ and $g_3 = x^2y^2 - z$ are xy^2z and x^2y^2, whose l.c.m. is x^2y^2z. Therefore

$$S(g_2, g_3) = xg_2 - zg_3 = (x^2y^2z - x^2yz) - (x^2y^2z - z^2) = -x^2yz + z^2.$$

This polynomial is non-zero and reduced with respect to G, and therefore G is not a Gröbner basis. Therefore we can add this polynomial (or, to make the calculations more readable, its negative) to G — call it g_4. This means that the S-polynomials to be considered are $S(g_1, g_2)$, $S(g_1, g_3)$, $S(g_1, g_4)$, $S(g_2, g_4)$ and $S(g_3, g_4)$.

Fortunately, we can make a simplification, by observing that $g_1 = xg_4$, and therefore the ideal generated by G does not change if we suppress g_1. This simplification leaves us with two S-polynomials to consider: $S(g_2, g_4)$ and $S(g_3, g_4)$.

$$S(g_2, g_4) = xg_2 - yg_4 = -x^2yz + yz^2,$$

and this last polynomial can be reduced (by adding g_4), which gives us $yz^2 - z^2$. As it is not zero, the basis is not Gröbner, and we must enlarge G by adding this new generator, which we call g_5. The S-polynomials to be considered are $S(g_3, g_4)$, $S(g_2, g_5)$, $S(g_3, g_5)$ and $S(g_4, g_5)$.

$$S(g_3, g_4) = zg_3 - yg_4 = -z^2 + yz^2,$$

and this last one can be reduced to zero (by adding g_5) In fact, this reduction follows from Buchberger's lcm (third) criterion, proposition 28.

$$S(g_2, g_5) = zg_2 - xyg_5 = -xyz^2 + xyz^2 = 0.$$

$$S(g_4, g_5) = zg_4 - x^2g_5 = -z^3 + x^2z^2 = x^2z^2 - z^3,$$

where the last rewriting arranges the monomials in decreasing order (with respect to $<$). This polynomial is already reduced with respect to G, G is therefore not a Gröbner basis, and we must add this new polynomial to G — let us call it g_6. The S-polynomials to be considered are $S(g_3, g_5)$, $S(g_2, g_6)$, $S(g_3, g_6)$, $S(g_4, g_6)$ and $S(g_5, g_6)$. The reader can check that G reduces all these S-polynomials to zero, and that G *is* therefore a Gröbner basis of the ideal, viz.

$$\begin{array}{rcl} g_2 & = & xy^2z - xyz, \\ g_3 & = & x^2y^2 - z, \\ g_4 & = & x^2yz - z^2, \\ g_5 & = & yz^2 - z^2, \\ g_6 & = & x^2z^2 - z^3. \end{array}$$

No power of x, y or z occurs alone, so we see that the variety is certainly not zero-dimensional, even though we have three equations in three variables, and z is undetermined. If $z \neq 0$, then g_5 can be divided by z^2 to give $y = 1$ and then

g_3 becomes $x^2 - z$, hence this part of the solution variety is a parabola. But if $z = 0$, all equations except g_3 collapse, and we have $x^2y^2 = 0$. Hence this part of the solution variety is two straight lines $x = z = 0$ and $y = z = 0$, each in fact of multiplicity four. Hence the solution is in fact of dimension one, a fact that was not evident when we started.

3.3.6 The Gianni–Kalkbrener Theorem

In this section, we will consider the case of dimension 0, i.e. finitely many solutions over K. We first remark that the situation can be distinctly more challenging than in the case of linear equations, which we illustrate by means of two examples.

1. $G = \{x^2 - 1, y^2 - 1\}$. This is a Gröbner base with respect to any ordering. There are four irreducible monomials $\{1, x, y, xy\}$, and hence four solutions, $x = \pm 1, y = \pm 1$.
2. $G = \{x^2 - 1, y^2 - 1, (x - 1)(y - 1)\}$. This is also a Gröbner base with respect to any ordering. There are three irreducible monomials $\{1, x, y\}$, and hence three solutions. There are $x = 1, y = \pm 1$, but when $x = -1$, we only have $y = 1$. The additional polynomial $(x-1)(y-1)$, which rules out the monomial xy, rules out the solution $x = y = -1$. Another way of looking at this is that, when $x = 1$, the polynomial $(x-1)(y-1)$ vanishes, but when $x = -1$, it adds an extra constraint.

Can we generalise this? The answer is 'yes', at least for *purely lexicographical* Gröbner bases of *zero-dimensional* ideals. If the order is $x_n < x_{n-1} < \cdots < x_1$ then such a Gröbner base G must have the form

$$\begin{gathered} p_n(x_n) \\ p_{n-1,1}(x_{n-1}, x_n), \ldots, p_{n-1,k_{n-1}}(x_{n-1}, x_n), \\ p_{n-2,1}(x_{n-2}, x_{n-1}, x_n), \ldots, p_{n-2,k_{n-2}}(x_{n-2}, x_{n-1}, x_n), \\ \cdots \\ p_{1,1}(x_1, \cdots, x_{n-1}, x_n), \ldots, p_{1,k_1}(x_1, \cdots, x_{n-1}, x_n), \end{gathered}$$

where $\deg_{x_i}(p_{i,j}) < \deg_{x_i}(p_{i,j+1})$ and p_{i,k_i} is monic in x_i. Let $G_k = G \cap k[x_k, \ldots, x_n]$, i.e. those polynomials in $x_k, \ldots, x_n$ only.

Theorem 14 (Gianni–Kalkbrener [Gia89, Kal89]) *Let α be a solution of G_{k+1}. Then if $\mathrm{lc}_{x_k}(p_{k,i})$ vanishes at α, then $(p_{k,i})$ vanishes at α. Furthermore, the lowest degree (in x_k) of the $p_{k,i}$ not to vanish at α, say p_{k,m_α}, divides all of the other $p_{k,j}$ at α. Hence we can extend α to solutions of G_k by adding $x_k = \mathrm{RootOf}(p_{k,m_\alpha})$.*

This gives us an algorithm to describe the solutions of a zero-dimensional ideal from such a Gröbner base G. This is essentially a generalisation of back-substitution into triangularised linear equations, except that there may be more than solution, since the equations are non-linear, and possibly more than one equation to substitute into.

Algorithm 6 (Gianni–Kalkbrener) $GK(G, n, A)$
Input: *A Gröbner base G for a zero-dimensional ideal I in n variables with respect to lexicographic order.*
Output: *A list of solutions of G.*

$S := \{x_n = \text{RootOf}(p_n)\}$
for $k = n-1, \ldots, 1$ **do**
    $S := GKS(G, k, S)$
return S

In practice, particularly if we are interested in keeping track of multiplicities of solutions, it may be more efficient to perform a square-free decomposition (Definition 29) of p_n, and initialise S to a list of the RootOf of each of its square-free factors, and also associate the multiplicity to each solution.

Algorithm 7 (Gianni–Kalkbrener Step) $GKS(G, k, A)$
Input: *A Gröbner base G for a zero-dimensional ideal I with respect to lexicographic order, an integer k, and A a list of solutions of G_{k+1}.*
Output: *A list of solutions of G_k.*

$B := \emptyset$
for each $\alpha \in A$
        i:=1
        while $(L := \text{lc}_{x_k}(p_{k,i}(\alpha))) = 0$ **do** $i := i + 1$
        if L is invertible
        # see last paragraph of section 3.1.4, page 47
          then $B := B \cup \{(\alpha \cup \{x_k = \text{RootOf}(p_{k,i}(\alpha))\})\}$
          else # α is split as $\alpha_1 \cup \alpha_2$
                $B := B \cup GKS(G, k, \{\alpha_1\}) \cup GKS(G, k, \{\alpha_2\})$
return B

In case 2. above, the three equations are $G = \{x^2 - 1, y^2 - 1, (x-1)(y-1)\}$. Taking $x_n = x$, we start off with $S = \{x = \text{RootOf}(x^2 - 1)\}$, and we call GKS on this. The initial value of L is $\text{RootOf}(x^2 - 1) - 1$, and we ask whether this is invertible. Adopting a common-sense (i.e. heuristic) approach for the moment, we see that this depends on *which* root we take: for $+1$ it is not invertible, and for -1 it is. Hence GKS makes two recursive calls to itself, on $x = 1$ and $x = -1$.

$GKS(G, 1, \{x = 1\})$ Here $L := \text{lc}_{x_1}(p_{1,1}(x = 1))$ is 0, so we consider $p_{1,2}$, whose leading coefficient is 1, so $y = \text{RootOf}(y^2 - 1)$.

$GKS(G, 1, \{x = -1\})$ Here $L := \text{lc}_{x_1}(p_{1,1}(x = -1))$ is -2, and $y = 1$.

There is a larger worked example of this later, at equation (3.34).

The theorem can be generalised in several ways to non-zero-dimensional ideals, but not completely [FGT01, Example 3.11].

3.3.7 The Faugère–Gianni–Lazard–Mora Algorithm

We have seen in the previous section that, for a zero-dimensional ideal, a *purely lexicographical* Gröbner base is a very useful concept. But these are generally the most expensive to compute, with a worst-case complexity of $O(d^{n^3})$ for polynomials of degree d in n variables [CGH88]. A *total degree, reverse lexicographic* Gröbner base, on the other hand, has complexity $O(d^{n^2})$, or $O(d^n)$ if the number of solutions at infinity is also finite [Laz83]. Hence the following algorithm [FGLM93] can be very useful, with $>'$ being *total degree, reverse lexicographic* and $>''$ being *purely lexicographical*, though it does have uses in other settings as well.

Algorithm 8 (FGLM)
Input: *A Gröbner base G for a zero-dimensional ideal I with respect to $>'$; an ordering $>''$.*
Output: *A Gröbner base H for I with respect to $>''$.*

$H := \emptyset$; $i := j := 0$
Enumerate the monomials irreducible under H in increasing order for $>''$
#This is finite by proposition 32
for each such m
    Let $m \xrightarrow{*}{}^{G} v$
    if $v = \sum_{k=1}^{j} c_k v_k$
        then $h_{i+1} := m - \sum_{k=1}^{j} c_k m_k$
            $H := H \cup \{h_{i+1}\}$; $i := i + 1$
        else $j := j + 1$; $m_j := m$; $v_j := v$
return H
#It is not totally trivial that H is a Gröbner base, but it is [FGLM93].

Since this algorithm is basically doing linear algebra in the space spanned by the irreducible monomials under G, whose dimension D is the number of solutions (proposition 32), it is not surprising that the running time seems to be $O(D^3)$, whose worst case is $O(d^{3n})$.

An an example of the FGLM algorithm, we take the system `Aux` from their paper[12], with three polynomials

$$abc + a^2bc + ab^2c + abc^2 + ab + ac + bc$$
$$a^2bc + a^2b^2c + b^2c^2a + abc + a + c + bc$$
$$a^2b^2c + a^2b^2c^2 + ab^2c + ac + 1 + c + abc$$

The total degree Gröbner basis has fifteen polynomials, whose leading monomials are

$$c^4, bc^3, ac^3, b^2c^2, abc^2, a^2c^2, b^3c, ab^2c, a^2bc, a^3c, b^4, ab^3, a^2b^2, a^3b, a^4.$$

[12]The system is obtained as they describe, except that the substitutions are $x_5 = 1/c$, $x_7 = 1/a$.

This defines a zero-dimensional ideal (c^4, b^4 and a^4 occur in this list), and we can see that the irreducible monomials are

$$1, c, c^2, c^3, b, bc, bc^2, b^2, b^2c, b^3, a, ac, ac^2, ab, abc, ab^2, a^2, a^2c, a^2b, a^3 :$$

twenty in number (as opposed to the 64 we would have if the basis only had the polynomials $a^4 + \cdots, b^4 + \cdots, c^4 + \cdots$). If we wanted a purely lexicographic base to which to apply Gianni-Kalkbrener, we would enumerate the monomials in lexicographic order as

1 (irreducble)

c (irreducble)

c^2 (irreducble)

c^3 (irreducble)

c^4 which reduces to $-\frac{185}{14} - \frac{293}{42}\,a^3 - \frac{1153}{42}\,a^2b + \frac{509}{7}\,ab^2 - \frac{323}{42}\,b^3 - \frac{2035}{42}\,a^2c - \frac{821}{21}\,abc + \frac{173}{6}\,b^2c - \frac{751}{14}\,ac^2 + \frac{626}{21}\,bc^2 + \frac{31}{42}\,c^3 - \frac{449}{14}\,a^2 + \frac{1165}{14}\,ab - \frac{772}{21}\,b^2 + \frac{550}{21}\,ac - \frac{429}{7}\,bc + \frac{184}{21}\,c^2 - \frac{407}{6}\,a - \frac{281}{42}\,b - \frac{4799}{42}\,c$

$\vdots$

c^{20} which reduces to $-\frac{156473200555876438}{7} + \frac{1355257348062243268}{21}\,bc^2 - \frac{2435043982608847426}{21}\,a^2c - \frac{455474473888607327}{3}\,a - \frac{87303768951017165}{21}\,b - \frac{5210093087753678597}{21}\,c + \frac{1264966801336921700}{7}\,ab - \frac{9959773482858358 22}{7}\,bc - \frac{2106129034377806827}{21}\,abc + \frac{136959771343895855}{3}\,b^2c + \frac{1119856342658748374}{21}\,ac + \frac{629351724586787780}{21}\,c^2 - \frac{774120922299216564}{7}\,ac^2 - \frac{1416003666295496227}{21}\,a^2b + \frac{1196637352769448957}{7}\,ab^2 - \frac{706526575918247673}{7}\,a^2 - \frac{1536916645521260147}{21}\,b^2 - \frac{417871285415094524}{21}\,a^3 - \frac{356286659366988974}{21}\,b^3 + \frac{3738195275477521 63}{21}\,c^3$, which can be expressed in terms of the previous ones as $-1 + 6\,c + 41\,c^2 - 71\,c^3 + 41\,c^{18} - 197\,c^{14} - 106\,c^{16} + 6\,c^{19} - 106\,c^4 - 71\,c^{17} - 92\,c^5 - 197\,c^6 - 145\,c^7 - 257\,c^8 - 278\,c^9 - 201\,c^{10} - 278\,c^{11} - 257\,c^{12} - 145\,c^{13} - 92\,c^{15}$; (and all higher powers of c are therefore expressible)

b which can be expressed in terms of the previous ones as $-\frac{9741532}{1645371} - \frac{8270}{343}\,c + \frac{32325724}{548457}\,c^2 + \frac{140671876}{1645371}\,c^3 - \frac{2335702}{548457}\,c^{18} + \frac{13420192}{182819}\,c^{14} + \frac{79900378}{1645371}\,c^{16} + \frac{1184459}{1645371}\,c^{19} + \frac{3378002}{42189}\,c^4 - \frac{5460230}{182819}\,c^{17} + \frac{688291}{4459}\,c^5 + \frac{1389370}{11193}\,c^6 + \frac{337505020}{1645371}\,c^7 + \frac{118784873}{548457}\,c^8 + \frac{271667666}{1645371}\,c^9 + \frac{358660781}{1645371}\,c^{10} + \frac{35978916}{182819}\,c^{11} + \frac{193381378}{1645371}\,c^{12} + \frac{553986}{3731}\,c^{13} + \frac{43953929}{548457}\,c^{15}$; (and all multiples of b are therefore expressible)

a which can be expressed in terms of the previous ones as $\frac{487915}{705159}\,c^{18} - \frac{4406102}{705159}\,c - \frac{16292173}{705159}\,c^{14} - \frac{17206178}{705159}\,c^2 - \frac{1276987}{235053}\,c^{16} - \frac{91729}{705159}\,c^{19} + \frac{377534}{705159} - \frac{801511}{26117}\,c^3 - \frac{26686318}{705159}\,c^4 + \frac{4114333}{705159}\,c^{17} - \frac{34893715}{705159}\,c^5 - \frac{37340389}{705159}\,c^6 - \frac{409930}{6027}\,c^7 - \frac{6603890}{100737}\,c^8 - \frac{14279770}{235053}\,c^9 - \frac{15449995}{235053}\,c^{10} - \frac{5382578}{100737}\,c^{11} - \frac{722714}{18081}\,c^{12} - \frac{26536060}{705159}\,c^{13} - \frac{13243117}{705159}\,c^{15}$.

These last three give us the Gröbner base in a purely lexicographical order, which looks like $\left\{c^{20}+\cdots, b+\cdots, a+\cdots\right\}$. As there are twenty solutions in reasonably general position (the polynomial in c alone *does* factor, but is square-free), we only need one polynomial per variable, as is often the case.

The existence of this algorithm leads to the following process for 'solving' a set of polynomial equations.

Algorithm 9
Input: *A set S of polynomials*
Output: *A 'description of solutions'*

G :=Buchberger$(S, >_{\texttt{tdeg}})$
if G is not zero-dimensional (Proposition 32)
  then return "not zero-dimensional"
  else H :=FGLM$(G, >_{\texttt{plex}})$
    Use Gianni–Kalkbrener to solve H

3.3.8 The Shape Lemma

Let us look again at example 2 of section 3.3.6. Here we needed three equations to define an ideal in two variables. We note that interchanging the rôles of x and y does not help (in this case, it might in others). However, using other coordinates than x and y definitely does. If we write the equations in terms of $u = x + y, v = x - y$ instead, we get the basis

$$[-4\,v + v^3, v^2 - 4 + 2\,u] : \tag{3.28}$$

three values for v (0, 2 and -2), each with one value of u (2, 0 and 0 respectively), from which the solutions in x, y can be read off. Note that ordering v before u would give the basis

$$[u^2 - 2\,u, uv, v^2 - 4 + 2\,u], \tag{3.29}$$

which is not triangular.

This kind of operation is called, for obvious reasons, a *rotation*. Almost all rotations will place the equations "in general position": and many theoretical approaches to these problems assume a "generic rotation" has been performed. In practice, this is a disaster, since sparsity is lost.

Definition 43 ([BMMT94]) *A basis for a zero-dimensional ideal is a* shape basis *if it is of the form*

$$\{g_1(x_1), x_2 - g_2(x_1), \ldots, x_n - g_n(x_1)\}\,.$$

This is a Gröbner basis for any ordering in which 'degree in x_1' is the first criterion: in the terminology of matrix orderings (page 66), any ordering where the first row of the matrix is $(\lambda, 0 \ldots, 0)$.

For a shape basis, the Gianni–Kalkbrener process is particularly simple: "determine x_1 and the rest follows". Almost all zero-dimensional ideals have shape bases. The precise criterion (•) in the theorem below is somewhat technical, but is satisfied if there are no repeated components.

Theorem 15 (Shape lemma) *[BMMT94, Corollary 3] After a generic rotation, a zero-dimensional ideal has a shape basis if, and only if,*

- *each primary component is simple or of local dimension 1.*

Furthermore [BMMT94, Lemma 2], such a rotation need only be 1-generic, i.e. have matrix $\begin{pmatrix} 1 & \mathbf{v} \\ \mathbf{0} & I \end{pmatrix}$ *for some generic vector* $\mathbf{v}$.

Their paper generalises this to ideals of higher dimension, but the complexity in notation is not worth it for our purposes.

3.3.9 Triangular Sets

An alternative approach to polynomial equation solving is that of characteristic [Rit32, Wu86] or triangular [Laz91] sets. See [ALMM99] for a reconciliation of the various theories, and [AMM99] for a practical comparison. We can regard triangular sets as an approach based on recursive views of polynomials, while Gröbner bases are based on a distributed view. We assume that the variables $x_1, \ldots, x_n$ are ordered as $x_1 < \ldots < x_n$, and define the *main variable* of p, $\mathrm{mvar}(p)$, to be the most important variable occurring in p.

Definition 44 *A set T of polynomials is said to be* triangular *if different polynomials have different main variables.*

Example 2 of section 3.3.6 shows that there may not always be a triangular set generating a particular ideal. *If* we have a triangular set, then the structure of the ideal, and the variety, is relatively obvious.

Definition 45 *Let T be a triangular set generating an ideal I in $k[x_1, \ldots, x_n]$. Then every variable x_i which occurs as a main variable is called* algebraic*, and the set of such variables is denoted* $\mathrm{AlgVar}(T)$.

Proposition 33 *For a triangular set T, the dimension of $I(T)$ is $n - |\mathrm{AlgVar}(T)|$.*

Much of the theory applies to positive dimension as well, but we will only consider in this section the case of zero-dimensional ideals/varieties. Let V be a zero-dimensional variety, and V_k be its *projection* onto $x_1, \ldots, x_k$, i.e.

$$V_k = \{(\alpha_1, \ldots, \alpha_k) : \exists (\alpha_1, \ldots, \alpha_n) \in V\}.$$

Definition 46 *A zero-dimensional variety V is* equiprojectable *iff, for all k, the projection $V_k \to V_{k-1}$ is an $n_k : 1$ mapping for some fixed n_k.* Note that this definition depends on the order of the x_i: a variety might be equiprojectable with respect to one order, but not another, as in (3.28) versus (3.29).

Such an equiprojectable variety will have $\prod n_k$ points (i.e. solutions, not counting multiplicity, to the equations).

Proposition 34 *Every equiprojectable variety corresponds to a triangular set, and vice versa.*

The variety V of Example 2 of section 3.3.6 is $\{(x = -1, y = 1), (x = 1, y = \pm 1)\}$ and is not equiprojectable. However, it can be written as $V = V_1 \cup V_2$ where $V_1 = \{(x = -1, y = 1)\}$ and $V_2 = \{(x = 1, y = \pm 1)\}$, each of which is equiprojectable. The corresponding triangular sets are $T_1 = \{x + 1, y - 1\}$ and $T_2 = \{x - 1, y^2 - 1\}$.

Theorem 16 (Gianni–Kalkbrener (triangular variant)) *Every zero-dimensional variety can be written as a union of disjoint equiprojectable varieties — an* equiprojectable decomposition*.*

In fact, each solution description in Algorithm 6 is a description of an equiprojectable variety.

This theorem can be, and was, proved independently, and the decomposition into triangular sets (the union of whose varieties is the original variety) can be computed directly. This gives us an alternative to algorithm 9: compute the triangular sets corresponding to the equiprojectable decomposition, and solve each one separately [Laz92].

It appears that the triangular set approach is more suitable to modular methods (section 4.1) than the Gröbner-base approach, but this is an area of active research.

3.3.10 Positive Dimension

Here we consider the case of solution sets of positive dimension (over the algebraic closure, e.g. over the complexes). As in Theorem 16, the ultimate aim is to express a variety as a union (preferably a disjoint union) of "nicer" varieties, or other sets.

Definition 47 *If P and Q are two (finite) sets of polynomials, we call the ordered pair (P, Q) a* quasi-algebraic system*, and we write $Z(P, Q)$, the zeros of the quasi-algebraic system, for $V(P) \setminus V(\prod Q)$, with the convention that if Q is empty, $\prod Q = 1$, so $V(Q) = \emptyset$.*

$$Z(P, Q) = \{\mathbf{x} \in K^n | (\forall p \in P \quad p(\mathbf{x}) = 0) \wedge (\forall q \in Q \quad q(\mathbf{x}) \neq 0)\}.$$

We say that (P, Q) is consistent *if $Z(P, Q) \neq \emptyset$.*

In Definition 40, we defined the *variety* of a set of polynomials, but we need some more concepts, all of which depend on having fixed an order of the variables.

Definition 48 *Let p be a polynomial. The* main variable *of p, denoted* $\mathrm{mvar}(p)$*, is the most important variable of p. The* initial *of p, written* $\mathrm{init}(p)$*, is its leading coefficient, when regarded as a univariate polynomial in* $\mathrm{mvar}(p)$*.*

If $S = \{p_1, \ldots, p_k\}$ is a finite set of polynomials, we write

$$\text{init}(S) = \text{lcm}_{1 \leq i \leq k} \text{init}(p_i).$$

It should be noted that many authors define $\text{init}(S)$ as $\prod_{1 \leq i \leq k} \text{init}(p_i)$. Since we are normally concerned with the zeros of $\text{init}(S)$, the two definitions have the same consequences, and ours leads to smaller polynomials.

Definition 49 *If T is a triangular system, we define the* pseudo-remainder *of p by T to be the pseudo-remainder of dividing p by each $q_i \in T$ in turn (turn defined by decreasing order of* $\text{mvar}(q_i)$*), regarded as univariate polynomials in* $\text{mvar}(q_i)$.

This is a generalization of Definition 27 (page 36).

Definition 50 *Let S be a finite set of polynomials. The set of* regular zeros *of S, written $W(S)$, is $Z(P, \{\text{init}(P)\}) = V(S) \setminus V(\{\text{init}(S)\})$. For $(a_1, \ldots, a_n)$ to be in $W(S)$, where $S = \{p_1, \ldots, p_k\}$, we are insisting that* all *of the p_i vanish at this point, but* none *of the* $\text{init}(p_i)$.

For Example 2 of section 3.3.6, the variety is $\{(x = -1, y = 1), (x = 1, y = \pm 1)\}$. If we take $y > x$, then the inital of the set of polynomials is $\text{lcm}(1, x - 1, 1) = x - 1$, so only the zero with $x = -1, y = 1$ is regular. Conversely, if we take $x > y$, the initial is $y - 1$ and only the zero with $y = -1, x = 1$ is regular. This emphasises that W depends on the variable ordering. It is also a property of the precise set S, not just the ideal $\langle S \rangle$.

In this case, $W(S)$ was in fact a variety (as always happens in dimension 0). In general, this is not guaranteed to happen: consider the (trivial) triangular system $S = \{(x-1)y - x + 1\}$ with $y > x$. Since this polynomial is $(x-1)(y-1)$, $V(S)$ is the two lines $x = 1$ and $y = 1$. However, $W(S)$ is the line $y = 1$ *except* for the point $(1, 1)$. In fact this is the only direct description we can give, though we could say that $W(S)$ is "almost" the line $y = 1$. This "almost" is made precise as follows.

Definition 51 *If W is any subset of K^n, the* Zariski closure *of W, written*[13] *$\overline{W}$, is the smallest variety containing it:*

$$\overline{W} = \bigcap \{V(F) \mid W \subseteq V(F)\},$$

which is itself a variety by Proposition 31.

In the example above, $\overline{W(S)} = V(y - 1)$.

3.3.10.1 An example

This example is from [AMM99, p. 126][14]. Suppose we have, in two dimensions, a manipulator consisting of an arm of length 1 fixed at the origin, and with

[13] Note that we use the same notation for algebraic closure and Zariski closure.

[14] The author is grateful to Russell Bradford for explaining the geometric context.

another arm, also of length 1, at its other end. We wish the far end of the manipulator to reach the point (a, b) in the plane. Let θ_1 be the angle that the first arm makes with the x axis, and write $c_1 = \cos\theta_1$, $s_1 = \sin\theta_1$. Let θ_2 be the angle that the second arm makes with the first. Then we have the following equations

$$\begin{aligned} c_1 + \cos(\theta_1 + \theta_2) &= a && (3.30)\\ s_1 + \sin(\theta_1 + \theta_2) &= b && (3.31)\\ s_1^2 + c_1^2 &= 1 && (3.32)\\ s_2^2 + c_2^2 &= 1, && (3.33) \end{aligned}$$

where the last two equations state that the arms have length 1. We can apply the addition formulae for trigonometric functions to (3.30) and (3.31) to get

$$c_1 + c_1c_2 - s_1s_2 = a \qquad 3.30',$$

$$s_1 + c_1s_2 + c_2s_1 = b \qquad 3.31'.$$

Rewriting these equations as polynomials, assumed to be zero, and using the order

$$c_2 > s_2 > c_1 > s_1 > b > a,$$

we get

$$S = \{c_2c_1 - s_2s_1 + c_1 - a, c_2s_1 + s_2c_1 + s_1 - b, c_1^2 + s_1^2 - 1, c_2^2 + s_2^2 - 1\},$$

which is not triangular since c_2 is the main variable of three different equations.

[AMM99] implement the method of [Laz91] to express $V(S)$ as a (disjoint) union

$$W(T_1) \cup W(T_2) \cup W(T_3),$$

where

$$\begin{aligned} T_1 = \{&(b^2 + a^2)(4s_1^2 - 4bs_1 + b^2 + a^2) - 4a^2, 2ac_1 + 2bs_1 - b^2 - a^2,\\ &2as_2 + 2(b^2 + a^2)s_1 - b^2 - a^2b, 2c_2 - b^2 - a^2 + 2\},\\ T_2 = \{&a, 2s_1 - b, 4c_1^2 + b^2 - 4, s_2 - bc_1, 2c_2 - b^2 + 2\},\\ T_3 = \{&a, b, c_1^2 + s_1^2 - 1, s_2, c_2 + 1\}. \end{aligned}$$

3.3.10.2 Regular Zeros and Saturated Ideals

Definition 52 *If T is a triangular system, define the* saturated ideal *of T to be*

$$\begin{aligned} \mathrm{sat}(T) &= \{p \in K[x_1, \ldots, x_n] | \exists n \in \mathbf{N} \quad \mathrm{init}(T)^n p \in (T)\}\\ &= \{p \in K[x_1, \ldots, x_n] | \mathrm{prem}(p, T) = 0\}. \end{aligned}$$

In other words it is the set of polynomials which can be reduced to zero by T after multiplying by enough of the initials of T so that division works. In terms of the more general concept of saturation $I : S^\infty$ of an ideal ([Bou61, p. 90]), this is $(T) : (\mathrm{init}(T))^\infty$.

Theorem 17 ([ALMM99, Theorem 2.1]) *For any non-empty triangular set T,*

$$\overline{W(T)} = V(\mathrm{sat}(T)).$$

In the example motivating Definition 51, $\mathrm{sat}(S)$ is generated by $y-1$, and indeed $\overline{W(S)} = V(\mathrm{sat}(S))$.

3.3.11 Conclusion

Whether we follow the Gianni–Kalkbrener approach directly (algorithm 9) or go via triangular sets, the solutions to a zero-dimensional family of polynomial equations can be expressed as a union of (equiprojectable) sets, each of which can be expressed as a generalised `RootOf` construct. For example, if we take the ideal

$$\{-3\,x - 6 + x^2 - y^2 + 2\,x^3 + x^4, -x^3 + x^2 y + 2\,x - 2\,y, -6 + 2\,x^2 - 2 \\ y^2 + x^3 + y^2 x, -6 + 3\,x^2 - xy - 2\,y^2 + x^3 + y^3\},$$

its Gröbner basis (purely lexicographic, $y > x$) is

$$[6 - 3\,x^2 - 2\,x^3 + x^5, -x^3 + x^2 y + 2\,x - 2\,y, 3\,x + 6 - x^2 + y^2 - 2\,x^3 - x^4]. \quad (3.34)$$

There are seven irreducible monomials: 1, x, x^2, x^3, x^4, y and xy. We know that x satisfies a quintic, and y then satisfies $\left(x^2 - 2\right) y - x^3 + 2\,x$. When $x^2 = 2$, this vanishes, so our quintic for x decomposes into $(x^2 - 2)(x^3 - 3)$, and the whole solution reduces to

$$\left\langle x^2 - 2, y^2 - x \right\rangle \cup \left\langle x^3 - 3, y - x \right\rangle. \quad (3.35)$$

Unfortunately, we do not have a convenient syntax to express this other than via the language of ideals. We are also very liable to fall into the 'too many solutions' trap, as in equation (3.8): Maple resolves the first component (in radical form) to

$$\left\{y = \sqrt[4]{2}, x = \sqrt{2}\right\}, \quad (3.36)$$

and the second one to

$$\left\{y = \sqrt[3]{3}, x = \sqrt[3]{3}\right\}, \quad (3.37)$$

both of which lose the connections between x and y ($x = y^2$ in the first case, $x = y$ in the second).

We are also dependent on the choice of order, since with $x > y$ the Gröbner basis is

$$[6 - 3\,y^4 - 2\,y^3 + y^7, 18 - 69\,y^2 - 9\,y^4 - 46\,y + 23\,y^5 - 2\,y^6 + 73\,x], \quad (3.38)$$

and no simplification comes to mind, short of factoring the degree seven polynomial in y, which of course is $(y^3-3)(y^4-2)$, and using the choice here to simplify the equation for x.

Maple's `RegularChains` package, using the technology of section 3.3.9, produces essentially equation (3.35) for the order $y > x$, and for $x > y$ produces

$$[[\left(2\,y+y^3+4\,y^2+2\right)x-8-2\,y^2-2\,y^3-2\,y, y^4-2],$$
$$[\left(5\,y+3+4\,y^2\right)x-12-5\,y^2-3\,y, -3+y^3]],$$

essentially the factored form of 3.38.

3.3.12 Coefficients other than fields

Most of the theory of this section, notably theorem 13, goes over to the case when R is a P.I.D. (definition 10) rather than a field, as described in [BW93, section 10.1]. However, when it comes to computation, things are not quite so obvious. What is the Gröbner base of $\{2x, 3y\}$ [Pau07]? There are two possible answers to the question.

- $\{2x, 3y\}$ [Tri78].
- $\{2x, 3y, xy\}$ [Buc84].

We note that $xy = x(3y) - y(2x) \in (2x, 3y)$, so we need xy to reduce to zero. We therefore modify definition 39 as follows

Definition 53 *Let $f, g \in R[x_1, \ldots, x_n]$. Suppose the leading coefficients of f and g are a_f and a_g, and the leading monomials m_f and m_g. Let a be a least common multiple of a_f and a_g, and write $a = a_f b_f = a_g b_g$. Let m be the least common multiple of m_f and m_g. The S-polynomial of f and g, written $S(f, g)$ is defined as*

$$S(f,g) = b_f \frac{m}{m_f} f - b_g \frac{m}{m_g} g. \tag{3.39}$$

Let $c_f a_f + c_g a_g = \gcd(a_f, a_g)$, and define the G-polynomial of f and g, written $G(f, g)$, as

$$G(f,g) = c_f \frac{m}{m_f} f + c_g \frac{m}{m_g} g. \tag{3.40}$$

Note that (3.39) is the same as (3.20) up to a factor of $\gcd(a_f, a_g)$. The S-polynomial is defined up to unit factors, whereas the G-polynomial is much less well-defined, but it turns out not to matter.

For the example quoted above, $S(2x, 3y) = 0$ (which follows from Proposition 27), while $G(2x, 3y) = xy$. Algorithm 5 goes over to this setting, execpt that we have to add G-polynomials as well as S-polynomials, and some care has to be taken to eliminate G-polynomials first — see [BW93, table 10.1].

3.3.13 Non-commutative Ideals

Much of the general mechanism of ideals generalises to the case of non-commutative ideals, provided we are careful to distinguish left, right or two-sided ideals. However, the theory is notably weaker. In particular we have the following opposite of theorem 1 and its corollary.

Proposition 35 *$K\langle x_1, \ldots, x_n\rangle$ is not noetherian for $n \geq 2$.*

Hence Buchberger's algorithm 5 might not terminate, and in general it does not [Mor86].

In fact, not only does this approach not work, but no approach can, as demonstrated by this result.

Proposition 36 ([KRW90]) *Ideal membership is insoluble in $\mathbf{Q}\langle x_1, x_2\rangle$.*

One case of great interest is when R is some field of (expressions representing) functions, and the "indeterminates" are differential or difference operators.

Example 1 *R is $\mathbf{Q}(x, y)$ and the indeterminates are $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$, so that we are working in $R[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}]$. Here the "indeterminates" commute with each other, but not with R, since $\frac{\partial}{\partial x}(xf) = f + x\frac{\partial}{\partial x}f$, i.e. $\frac{\partial}{\partial x}x = 1 + x\frac{\partial}{\partial x}$.*

We should note that the result of multiplying a term by an indeterminate is not necessarily a term, e.g. $\frac{\partial}{\partial x}\left(x\frac{\partial}{\partial x}\right) = \frac{\partial}{\partial x} + x\frac{\partial^2}{\partial x^2}$. This makes characterising a Gröbner base harder, but the following definition is an appropriate generalisation of the last clause of theorem 13 in the setting where the indeterminates commute with each other.

Definition 54 *[Pau07, Definition 4] A finite subset G of $I \setminus \{0\}$, where I is a left-ideal, is a Gröbner basis of I iff, for all monomials m, the R-ideal $(\mathrm{lc}(f)|f \in I \wedge \mathrm{lm}(f) = m)$ is generated by $\{\mathrm{lc}(g)|g \in G \wedge \mathrm{lm}(g)$ divides $m\}$.*

If R is a principal ideal domain, it is possible to define S-polynomials and G-polynomials as in the previous section, but in general we need to consider more complicated (but still finitely many) combinations [Pau07]. This leads to an *effective* test for a Gröbner base in this setting, i.e. we need to check that finitely many combinations reduce to zero. We also get a generalisation of Buchberger's algorithm [Pau07, Proposition 10]: if the combination does not reduce to zero, add it. Termination is non-trivial, however.

3.4 Equations and Inequalities

While it is possible to work in more general settings (real closed fields), we will restrict our attention to solving systems over $\mathbf{R}$. Consider the two equations

$$x^2 + y^2 = 1 \tag{3.41}$$
$$x^2 + y^2 = -1. \tag{3.42}$$

Over the complexes, there is little to choose between these two equations, both define a one-dimensional variety. Over $\mathbf{R}$, the situation is very different: (3.41) still defines a one-dimensional variety (a circle), while (3.42) defines the empty set, even though we have only one equation in two variables.

Hence we can essentially introduce the constraint $x \geq 0$ by adding a new variable y and the equation $y^2 - x = 0$. We can also introduce the constraint $x \neq 0$ by adding a new variable z and $xz - 1 = 0$ (essentially insisting that x be invertible). Hence $x > 0$ can be introduced. Having seen that $\geq$ and $>$ can creep in through the back door, we might as well admit them properly, and deal with the language of *real closed fields*, i.e. the language of fields (definition 11) augmented with the binary predicate $>$ and the additional laws:

1. Precisely one of $a = b$, $a > b$ and $b > a$ holds;
2. $a > b$ and $b > c$ imply $a > c$;
3. $a > b$ implies $a + c > b + c$;
4. $a > b$ and $c > 0$ imply $ac > bc$.

This is the domain of *real algebraic geometry*, a lesser-known, but very important, variant of classical algebraic geometry. Suitable texts on the subject are [BPR06, BCR98]. However, we will reserve the word 'algebraic' to mean a set defined by equalities only, and reserve *semi-algebraic* for the case when inequalities (or inequations[15]) are in use. More formally:

Definition 55 *An algebraic proposition is one built up from expressions of the form $p_i(x_1, \ldots, x_n) = 0$, where the p_i are polynomials with integer coefficients, by the logical connectives $\neg$ (not), $\wedge$ (and) and $\vee$ (or). A semi-algebraic proposition is the same, except that the building blocks are expressions of the form $p_i(x_1, \ldots, x_n)\sigma 0$ where σ is one of $=, \neq, >, \geq, <, \leq$.*

This language is in fact redundant, since $\neq, \geq, \leq$ can be replaced with the help of $\neg$, but corresponds more closely to natural usage. The reader will also notice that it is not quite the language of real closed fields described above, since we do not allow division. This is partly for ease of subsequent development, but also allows us to sidestep "division by zero" questions, as raised in problem 1 of section 1.2.2. Hence the proposition $\frac{p}{q} > 0$ has to be translated as $(q > 0 \wedge p > 0) \vee (q < 0 \wedge p < 0)$, which is not true when $q = 0$. If this is not what we mean, e.g. when p and q have a common factor, we need to say so.

3.4.1 Applications

It runs out that many of the problems one wishes to apply computer algebra to can be expressed in terms of real semi-algebraic geometry. This is not totally surprising, since after all, the "real world" is largely real in the sense of $\mathbf{R}$.

[15] Everyone agrees that an *equation* $a = b$ is an *equality*. $a > b$ and its variants are referred to an *inequalities*. This only leaves the unfamiliar *inequation* for $a \neq b$.

Furthermore, even if problems are posed purely in terms of equations, there may well be implicit inequalities as well. For example, it may be implicit that quanitites are non-negative, or that concentrations is biochemistry lie in the range $[0, 1]$.

Robot motion planning ...

It is also often important to prove *unsatisfiability*, i.e. that a semi-algebraic formula has *no* solutions. [Mon09] gives several examples, ranging from program proving to biological systems. The program proving one is as follows. One wishes to prove that I is an invariant (i.e. if it was true at the start, it is true at the end) of a program which moves from one state to another by a transition relation τ. More formally, one wishes to prove that there do *not* exist two states s, s' such that $s \in I$, $s' \notin I$, but $s \rightarrow_\tau s'$. Such a pair (s, s') would be where "the program breaks down", so a proof of unsatisfiability becomes a proof of program correctness. This places stress on the concept of 'proof' — "I can prove that there are no bad cases" is much better than "I couldn't find any bad cases".

3.4.2 Quantifier Elimination

A fundamental result of algebraic geometry is the following, which follows from the existence of resultants (section A.1).

Theorem 18 *A projection of an algebraic set is itself an algebraic set.*

For example, the projection of the set defined by

$$\left\{(x-1)^2 + (y-1)^2 + (z-1)^2 - 4, x^2 + y^2 + z^2 - 4\right\} \tag{3.43}$$

on the x, y-plane is the ellipse

$$8\,x^2 + 8\,y^2 - 7 - 12\,x + 8\,xy - 12\,y. \tag{3.44}$$

We can regard equation (3.43) as defining the set

$$\exists z \left((x-1)^2 + (y-1)^2 + (z-1)^2 = 4 \wedge x^2 + y^2 + z^2 = 4\right) \tag{3.45}$$

and equation (3.44) as the *quantifier-free* equivalent

$$8\,x^2 + 8\,y^2 - 12\,x + 8\,xy - 12\,y = 7. \tag{3.46}$$

Is the same true in real algebraic geometry? If P is a projection operator, and $\Re$ denotes the real part, then clearly

$$P(\Re(U) \cap \Re(V)) \subseteq \Re(P(U \cap V)). \tag{3.47}$$

However, the following example shows that the inclusion can be strict. Consider

$$\left\{(x-3)^2 + (y-1)^2 + z^2 - 1, x^2 + y^2 + z^2 - 1\right\}$$

Its projection is $(10 - 6\,x - 2\,y)^2$, i.e. a straight line (with multiplicity 2). If we substitute in the equation for y in terms of x, we get $z = \sqrt{-10\,x^2 + 30\,x - 24}$, which is never real for real x. In fact $\Re(U) \cap \Re(V) = \emptyset$, as is obvious from the geometric interpretation of two spheres of radius 1 centred at $(0, 0, 0)$ and $(3, 1, 0)$. Hence the methods we used for (complex) algebraic geometry will not translate to real algebraic geometry.

The example of $y^2 - x$, whose projection is $x \geq 0$, shows that the projection of an algebraic set need not be an algebraic set, but might be a semi-algebraic set. Is even this guaranteed? What about the projection of a semi-algebraic set? In the language of quantified propositions, we are asking whether, when F is an algebraic or semi-algebraic proposition, the proposition

$$\exists y_1 \ldots \exists y_m F(y_1, \ldots, y_m, x_1, \ldots, x_n) \tag{3.48}$$

has a quantifier-free equivalent $G(x_1, \ldots, x_n)$, where G is a semi-algebraic proposition. We can generalise this.

Problem 2 (Quantifier Elimination) *Given a quantified proposition*[16]

$$Q_1 y_1 \ldots Q_m y_m F(y_1, \ldots, y_m, x_1, \ldots, x_n), \tag{3.49}$$

where F is a semi-algebraic proposition and the Q_1 are each either $\exists$ or $\forall$, does there exist a quantifier-free equivalent $G(x_1, \ldots, x_n)$. If so, can we compute it?

The fact that there is a quantifier-free equivalent is known as the *Tarski–Seidenberg Principle* [Sei54, Tar51]. The first constructive answer to the question was given by Tarski [Tar51], but the complexity of his solution was indescribable[17]. A better (but nevertheless doubly exponential) solution had to await the concept of cylindrical algebraic decomposition (CAD) [Col75] described in the next section.

Notation 9 *Since $\exists x \exists y$ is equivalent to $\exists y \exists x$, and similarly for $\forall$, we extend $\exists$ and $\forall$ to operate on blocks of variables, so that, if $\mathbf{x} = (x_1, \ldots, x_n)$, $\exists \mathbf{x}$ is equivalent to $\exists x_1 \ldots \exists x_n$. If we use this notation to rewrite equation 3.49 with the fewest number of quantifiers, the quantifiers then have to alternate, so the formula is (where the $\mathbf{y}_i$ are sets of variables)*

$$\forall \mathbf{y}_1 \exists \mathbf{y}_2 \forall \mathbf{y}_3 \ldots F(\mathbf{y}_1, \mathbf{y}_2, \ldots, x_1, \ldots, x_n), \tag{3.50}$$

or

$$\exists \mathbf{y}_1 \forall \mathbf{y}_2 \exists \mathbf{y}_3 \ldots F(\mathbf{y}_1, \mathbf{y}_2, \ldots, x_1, \ldots, x_n). \tag{3.51}$$

In either form, the number of quantifiers is one more than the number of alternations.

[16] Any proposition with quantified variables can be converted into one in this form, so-called *prenex normal form* — see any standard logic text.

[17] In the formal sense, that there was no elementary function which could describe it, i.e. no tower of exponentials of fixed height would suffice!

3.4.3 Algebraic Decomposition

Definition 56 *An* algebraic decomposition *of* $\mathbf{R}^n$ *is an expression of* $\mathbf{R}^n$ *as the disjoint union of non-empty connected sets, known as* cells*, each defined as*

$$p_1(x_1, \ldots, x_n)\sigma 0 \wedge \cdots \wedge p_m(x_1, \ldots, x_n)\sigma 0, \tag{3.52}$$

where the σ *are one of* $=, >, <$*. Equation (3.52) is known as the* defining formula *of the cell* C*, and denoted* $\mathrm{Def}(C)$*.*

These should properly be called *semi-algebraic decompositions*, but this terminology has stuck. Note that (3.52) need not define a non-empty connected set — external information is required to show this. We should note that these definitions are a very restricted form of definition 55. Here are some examples.

1. $\mathbf{R}^1$ can be decomposed as $\{x < 0\} \cup \{x = 0\} \cup \{x > 0\}$.
2. $\mathbf{R}^1$ cannot be decomposed as $\{x^2 = 0\} \wedge \{x^2 > 0\}$, as the second set is not connected. Rather, we need the previous decomposition.
3. $\mathbf{R}^1$ cannot be decomposed as

   $$\{(x^2-3)^2 - 2 = 0\} \wedge \{(x^2-3)^2 - 2 > 0\} \wedge \{(x^2-3)^2 - 2 < 0\},$$

   as the sets are not connected. Rather, we need the decomposition (writing $(x^2-3)^2 - 2$ as f)

   $$\begin{gathered}\{f > 0 \wedge x < -2\} \cup \{f = 0 \wedge x < -2\} \cup \{f < 0 \wedge x < 0\} \cup \\ \{f = 0 \wedge x > -2 \wedge x < 0\} \cup \{f > 0 \wedge x > -2 \wedge x < 2\} \cup \\ \{f = 0 \wedge x > 0 \wedge x < 2\} \cup \{f < 0 \wedge x > 0\} \cup \\ \{f = 0 \wedge x > 2\} \cup \{f > 0 \wedge x > 2\}.\end{gathered}$$

4. $\mathbf{R}^2$ can be decomposed as $\{(x^2+y^2) < 0\} \cup \{(x^2+y^2) = 0\} \cup \{(x^2+y^2) > 0\}$.
5. $\mathbf{R}^2$ cannot be decomposed as $\{xy < 1\} \cup \{xy = 1\} \cup \{xy > 1\}$, as the last two sets are not connected. Rather, we need the more complicated

   $$\begin{gathered}\{xy < 1\} \cup \\ \{xy = 1 \wedge x > 0\} \cup \{xy = 1 \wedge x < 0\} \cup \\ \{xy > 1 \wedge x < 0\} \cup \{xy > 1 \wedge x > 0\}.\end{gathered}$$

6. $\mathbf{R}^2$ cannot be decomposed as $\{f < 0\} \cup \{f = 0\} \cup \{f > 0\}$, where $f = \left(x^2 + y^2 - 1\right)\left((x-3)^2 + y^2 - 1\right)$, as the first two sets are not connected. Rather, we need the more complicated

   $$\begin{gathered}\{f < 0 \wedge x < \tfrac{3}{2}\} \cup \{f < 0 \wedge x > \tfrac{3}{2}\} \cup \{f = 0 \wedge x < \tfrac{3}{2}\} \\ \cup \{f = 0 \wedge x > \tfrac{3}{2}\} \cup \{f > 0\}\end{gathered}$$

The reader may complain that example 3 is overly complex: can't we just write

$$\{f > 0 \wedge x < -2\} \cup \{x = -\sqrt{3+\sqrt{2}}\} \cup \{f < 0 \wedge x < 0\} \cup$$
$$\{x = -\sqrt{3-\sqrt{2}} < 0\} \cup \{f > 0 \wedge x > -2 \wedge x < 2\} \cup$$
$$\{x = \sqrt{3-\sqrt{2}}\} \cup \{f < 0 \wedge x > 0\} \cup \{x = \sqrt{3+\sqrt{2}}\} \cup \{f > 0 \wedge x > 2\}?$$

In this case we could, but in general theorem 8 means that we cannot[18]: we need `RootOf` constructs, and the question then is "which root of ...". In example 3, we chose to use numeric inequalities (and we were lucky that they could be chosen with integer end-points). It is also possible [CR88] to describe the roots in terms of the signs of the derivatives of f, i.e.

$$\{f > 0 \wedge x < -2\} \cup \{f = 0 \wedge f' < 0 \wedge f''' < 0\} \cup \{f < 0 \wedge x < 0\} \cup$$
$$\{f = 0 \wedge f' > 0 \wedge f''' < 0\} \cup \{f > 0 \wedge x > -2 \wedge x < 2\} \cup$$
$$\{f = 0 \wedge f' < 0 \wedge f''' > 0\} \cup \{f < 0 \wedge x > 0\} \cup$$
$$\{f = 0 \wedge f' > 0 \wedge f''' > 0\} \cup \{f > 0 \wedge x > 2\}$$

(as it happens, the sign of f'' is irrelevant here). This methodology can also be applied to the one-dimensional regions, e.g. the first can also be defined as $\{f > 0 \wedge f' > 0 \wedge f'' < 0 \wedge f''' < 0\}$.

We may ask how we know that we have a decomposition, and where these extra constraints (such as $x > 0$ in example 5 or $x < \frac{3}{2}$ in example 6) come from. This will be addressed in the next section, but the brief answers are:

- we know something is a decomposition because we have constructed it that way;
- $x = 0$ came from the leading coefficient (with respect to y) of $xy - 1$, whereas $\frac{3}{2}$ in example 6 is a root of $\mathrm{Disc}_y(f)$.

We stated in definition 56 that the cells must be non-empty. How do we know this? For the zero-dimensional cells $\{f = 0 \wedge x > a \wedge x < b\}$, we can rely on the fact that if f changes sign between a and b, there must be at least one zero, and if f' does not[19], there cannot be more than one: such an interval can be called an *isolating interval*. In general, we are interested in the following concept.

Definition 57 *A* sampled algebraic decomposition *of* $\mathbf{R}^n$ *is an algebraic decomposition together with, for each cell* C*, an explicit point* $\mathrm{Sample}(C)$ *in that cell.*

By 'explicit point' we mean a point each of whose coordinates is either a rational number, or a precise algebraic number: i.e. a defining polynomial[20] together with an indication of which root is meant, an isolating interval, a sufficiently

[18] And equation (3.11) means that we probably wouldn't want to even when we could!
[19] Which will involve looking at f'' and so on.
[20] Not necessarily irreducible, though it is normal to insist that it be square-free.

exact[21] numerical approximation or a Thom's Lemma [CR88] list of signs of derivatives.

Definition 58 *A decomposition D of $\mathbf{R}^n$ is said to the sign-invariant for a polynomial $p(x_1, \ldots, x_n)$ if and if only if, for each cell $C \in D$, precisely one of the following is true:*

1. $\forall \mathbf{x} \in C \quad p(\mathbf{x}) > 0$;
2. $\forall \mathbf{x} \in C \quad p(\mathbf{x}) < 0$;
3. $\forall \mathbf{x} \in C \quad p(\mathbf{x}) = 0$;

It is sign-invariant for a set of polynomials if, and only if, for each polynomial, one of the above conditions is true for each cell.

It therefore follows that, for a sampled decomposition, the sign throughout the cell is that at the sample point.

3.4.4 Cylindrical Algebraic Decomposition

Notation 10 *Let $n > m$ be positive natural numbers, and let $\mathbf{R}^n$ have coordinates $x_1, \ldots, x_n$, with $\mathbf{R}^m$ having coordinates $x_1, \ldots, x_m$.*

Definition 59 *An algebraic decomposition D of $\mathbf{R}^n$ is said to be* cylindrical *over a decomposition D' of $\mathbf{R}^m$ if the projection onto $\mathbf{R}^m$ of every cell of D is a cell of D'. The cells of D which project to $C \in D'$ are said to form the* cylinder *over C, denoted* $\mathrm{Cyl}(C)$. *For a sampled algebraic decomosition, we also insist that the sample point in C be the projection of the sample points of all the cells in the cylinder over C.*

Cylindricity is by no means trivial.

Example 2 *Consider the decomposition of $\mathbf{R}^2 = S_1 \cup S_2 \cup S_3$ where*

$$\begin{aligned} S_1 &= \{(x,y) \mid x^2 + y^2 - 1 > 0\}, \\ S_2 &= \{(x,y) \mid x^2 + y^2 - 1 < 0\}, \\ S_3 &= \{(x,y) \mid x^2 + y^2 - 1 = 0\}. \end{aligned}$$

This is an algebraic decomposition, and is sign-invariant for $x^2 + y^2 - 1$. However, it is not cylindrical over any decomposition of the x-axis $\mathbf{R}^1$. The projection of S_2 is $(-1, 1)$, so we need to decompose $\mathbf{R}^1$ as

$$(-\infty, -1) \cup \{-1\} \cup (-1, 1) \cup \{1\} \cup (1, \infty). \tag{3.53}$$

[21] By this, we mean an approximation such that the root cannot be confused with any other, which generally means at least an approximation close enough that Newton's iteration will converge to the indicated root. Maple's `RootOf` supports such a concept.

S_3 projects onto $[-1,1]$, which is the union of three sets in (3.53). We have to decompose S_3 into four sets:

$$\begin{aligned} S_{3,1} &= \{(-1,0)\}, \quad S_{3,2} = \{(1,0)\}, \\ S_{3,3} &= \{(x,y) \mid x^2+y^2-1=0 \wedge y>0\}, \\ S_{3,4} &= \{(x,y) \mid x^2+y^2-1=0 \wedge y<0\}. \end{aligned}$$

S_1 splits into eight sets, one above each of $(-\infty,-1)$ and $(1,\infty)$ and two above each of the other components of (3.53). It is obvious that this is the minimal refinement of the original decomposition to possess a cylindric decomposition. Furthermore in this case no linear transformation of the axes can reduce this. If we wanted a sampled decomposition, we could choose x-coordinates of -2, -1, 0, 1 and 2, and y-coordinates to match, from $\{0, \pm 1, \pm 2\}$.

Cylindricity is fundamental to solving problem 2 via the following two propositions.

Proposition 37 *Let*

$$\exists x_n \ldots \exists x_{m+1} P(x_1, \ldots, x_n) \tag{3.54}$$

be an existentially quantified formula, D be a sampled algebraic decomposition of $\mathbf{R}^n$ which is sign-invariant for all the polynomials occurring in P, and D' be a sampled algebraic decomposition of $\mathbf{R}^m$ such that D is cylindrical over D'. Then a quantifier-free form of (3.54) is

$$\bigvee_{C' \in D' \exists C \in \mathrm{Cyl}(C') P(\mathrm{Sample}(C))} \mathrm{Def}(C'). \tag{3.55}$$

Proposition 38 *Let*

$$\forall x_n \ldots \forall x_{m+1} P(x_1, \ldots, x_n) \tag{3.56}$$

be a universally quantified formula, D be a sampled algebraic decomposition of $\mathbf{R}^n$ which is sign-invariant for all the polynomials occurring in P, and D' be a sampled algebraic decomposition of $\mathbf{R}^m$ such that D is cylindrical over D'. Then a quantifier-free form of (3.56) is

$$\bigvee_{C' \in D' \forall C \in \mathrm{Cyl}(C') P(\mathrm{Sample}(C))} \mathrm{Def}(C'). \tag{3.57}$$

These two propositions lead to a solution of problem 2.

Theorem 19 ([Col75]) *Let $\mathbf{x}_0, \ldots, \mathbf{x}_k$ be sets of variables, with $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n_i})$, and let $N_i = \sum_{j=0}^{i} n_j$. Let $P(\mathbf{x}_0, \ldots, \mathbf{x}_k)$ be a semi-algebraic proposition, D_i be an algebraic decomposition of $\mathbf{R}^{N_i}$ such that each D_i is cylindric over D_{i-1} and D_k is sign-invariant for all the polynomials in P. Then a quantifier-free form of*

$$Q_k \mathbf{x}_k \ldots Q_1 \mathbf{x}_1 P(\mathbf{x}_0, \ldots, \mathbf{x}_k) \tag{3.58}$$

(where the Q_i are $\forall$ or $\exists$) is

$$\bigvee_{C' \in D_0 \forall \exists C \in \mathrm{Cyl}^k(C') P(\mathrm{Sample}(C))} \mathrm{Def}(C'), \tag{3.59}$$

where by $\forall\exists$ we mean that we are quantifying across the coordinates of $\mathrm{Sample}(C)$ *according to the quantifiers in (3.58).*

We can use the (sampled) cylindrical algebraic decomposition in example 2 to answer various questions.

Example 3 $\forall y \quad x^2+y^2-1 > 0$. *For the sampled cells $\langle(-\infty,-1),(x=-2,y=0)\rangle$ and $\langle(1,\infty),(x=2,y=0)\rangle$, the proposition is true at the sample points, hence true everywhere in the cell. For all the other cells in (3.53), there is a sample point for which it is false (in fact, $y=0$ always works). So the answer is $(-\infty,-1)\cup(1,\infty)$.*

Example 4 $\exists y \quad x^2+y^2-1>0$. *For every cell in (3.53), there is a sample point above it for which the proposition is true, hence we deduce that the answer is (3.53), which can be simplified to* **true**.

We should note (and this is both one of the strengths and weaknesses of this approach) that the same cylindrical algebraic decomposition can be used to answer all questions of this form with the same order of (blocks of) quantified variables, irrespective of what the quantifiers actually are.

Example 5 $(\exists y \quad x^2+y^2-1>0) \wedge (\exists y \quad x^2+y^2-1<0)$. *This formula is not directly amenable to this approach, since it is not in prenex form. In prenex form, it is $\exists y_1 \exists y_2 \left((x^2+y_1^2-1>0)\wedge(x^2+y_2^2<0)\right)$ and we need an analogous*[22] *decomposition of $\mathbf{R}^3$ cylindric over $\mathbf{R}^1$. Fortunately, (3.53) suffices for our decomposition of $\mathbf{R}^1$, and the answer is $(-1,1)$, shown by the sample point $(x=0,y_1=2,y_2=0)$, and by the fact that at other sample points of $\mathbf{R}^1$, we do not have y_1,y_2 satisfying the conditions.*

However, it could be argued that all we have done is reduce problem 2 to the following one.

Problem 3 *Given a quantified semi-algebraic proposition as in theorem 19, produce a sign-invariant decomposition D_k cylindrical over the appropriate D_i such that theorem 19 is applicable.* Furthermore, since theorem 19 only talks about "a" quantifier-free form, we would like the simplest possible such D_k (see [Laz88]).

There is no common term for such a decomposition: we will call it *block-cylindrical*.

[22] Easier said than done. Above $x=-1$ we have nine cells: $\{y_1<0, y_1=0, y_1>0\}\times\{y_2<0, y_2=0, y_2>0\}$, and the same for $x=1$, whereas above $(-1,1)$ we have 25, totalling 45.

3.4.5 Computing Algebraic Decompositions

Though many improvements have been made to it since, the basic strategy for computing algebraic decompositions is still that due to Collins [Col75], and is to compute them cylindrically, as illustrated in the following diagram.

$$
\begin{array}{ccccc|ccc}
S_n & \in & \mathbf{R}[x_1,\ldots,x_n] & \mathbf{R}^n & & \mathbf{R}^n & \text{decomposed by} & D_n \\
\downarrow & & & \downarrow & & \uparrow & & \text{Cyl} \\
S_{n-1} & \in & \mathbf{R}[x_1,\ldots,x_{n-1}] & \mathbf{R}^{n-1} & & \mathbf{R}^{n-1} & \text{decomposed by} & D_{n-1} \\
\downarrow & & & \downarrow & & \uparrow & & \text{Cyl} \\
\cdots & & \cdots & \downarrow & & \uparrow & & \cdots \\
S_2 & \in & \mathbf{R}[x_1,x_2] & \mathbf{R}^2 & & \mathbf{R}^2 & \text{decomposed by} & D_2 \\
\downarrow & & & \downarrow & & \uparrow & & \text{Cyl} \\
S_1 & \in & \mathbf{R}[x_1] & \mathbf{R}^1 & \underbrace{\longrightarrow}_{\text{Problem 1}} & \mathbf{R}^1 & \text{decomposed by} & D_1.
\end{array}
$$

From the original proposition, we extract the set of polynomials S_n. We then project this set into S_{n-1} in $n-1$ variables, and so on, until we have a set of univariates S_1. We then isolate, or otherwise describe, the roots of these polynomials, as described in problem 1, to produce a decomposition D_1 of $\mathbf{R}^1$, and then successively lift this to a decomposition D_2 of $\mathbf{R}^2$ and so on, each D_i being sign-invariant for S_i and cylindrical over D_{i-1}.

Note that the projection from S_{i+1} to S_i must be such that a decomposition D_i sign-invariant for S_i can be lifted to a decomposition D_{i+1} sign-invariant for S_{i+1}. Note also that the decomposition thus produced will be block-cylindric for every possible blocking of the variables, since it is block-cylindric for the finest such.

Projection turns out to be a trickier problem than might be expected. One's immediate thought is that one needs the discriminants (with respect to the variable being projected) of all the polynomials in S_{i+1}, since this will give all the critical points. Then one sees that one needs the resultants of all pairs of such polynomials. Example 5 (page 84) shows that one might need leading coefficients. Then there are issues of what happens when leading coefficients vanish. This led Collins [Col75] to consider the following projection operation.

3.4.6 Complexity

Let us suppose that there are s polynomials involved in the input formula (3.49), of maximal degree d. Then such a cylindrical algebraic decomposition can be computed in time $O\left((sd)^{2^{O(k)}}\right)$.

There are examples [BD07, DH88], which shows that this behaviour is best-possible, indeed the projection onto $\mathbf{R}^1$ might have a number of components doubly-exponential in k.

3.5 Conclusions

1. The `RootOf` construct is inevitable (theorem 8), so should be used, as described in footnote 3 (page 47). Such a notation can avoid the "too many solutions" trap — see equations (3.36) and (3.37). We should find a way of extending it to situations such as equation (3.35).

2. While matrix inversion is a valuable concept, it should generally be avoided in practice.

3. Real algebraic geometry is not simply "algebraic geometry writ real": it has different problems and needs different techniques.

3.5.1 Open Problems

Open Problem 4 *The FGLM algorithm is $O(D^3)$ where D is the number of solutions. Can faster matrix algorithms such as Strassen–Winograd [Str69, Win71] speed this up?*

Chapter 4

Advanced Algorithms

4.1 Modular Methods

In chapter 2, describing the subresultant method of computing greatest common divisors, we said the following.

> This algorithm is the best method known for calculating the g.c.d., of all those based on Euclid's algorithm applied to polynomials with integer coefficients. In chapter 4 we shall see that if we go beyond these limits, it is possible to find better algorithms for this calculation.

Now is the time to fulfil that promise, which we do by describing the historically-first "advanced" algorithm, and its greatest success, g.c.d. calculation.

4.1.1 Gcd in one variable

Let us consider Brown's example from page 35 again:

$$A(x) = x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5; \quad (4.1)$$

$$B(x) = 3x^6 + 5x^4 - 4x^2 - 9x + 21. \quad (4.2)$$

Let us suppose that these two polynomials have a common factor, that is a polynomial P (of non-zero degree) which divides A and B. Then there is a polynomial Q such that $A = PQ$. This equation still holds if we take each coefficient as an integer modulo 5. If we write P_5 to signify the polynomial P considered as a polynomial with coefficients modulo 5, this equation implies that P_5 divides A_5. Similarly, P_5 divides B_5, and therefore it is a common factor[1] of A_5 and B_5. But calculating the g.c.d. of A_5 and B_5 is fairly easy:

$$A_5(x) = x^8 + x^6 + 2x^4 + 2x^3 + 3x^2 + 2x;$$

[1] Note that we cannot deduce that $P_5 = \gcd(A_5, B_5)$: a counter-example is $A = x - 3$, $B = x + 2$, where $P = 1$, but $A_5 = B_5 = x + 2$, and so $\gcd(A_5, B_5) = x + 2$, whereas $P_5 = 1$.

$$\begin{aligned} B_5(x) &= 3x^6 + x^2 + x + 1; \\ C_5(x) &= \text{remainder}(A_5(x), B_5(x)) = A_5(x) + 3(x^2+1)B_5(x) = 4x^2 + 3; \\ D_5(x) &= \text{remainder}(B_5(x), C_5(x)) = B_5(x) + (x^4 + 4x^2 + 3)C_5(x) = x; \\ E_5(x) &= \text{remainder}(C_5(x), D_5(x)) = C_5(x) + xD_5(x) = 3. \end{aligned}$$

Thus A_5 and B_5 are relatively prime, which implies that $P_5 = 1$. As the leading coefficient of P has to be one, we deduce that $P = 1$.

The concept of modular methods is inspired by this calculation, where there is no possibility of intermediate expression swell, for the integers modulo 5 are bounded (by 4). Obviously, there is no need to use the integers modulo 5: any prime number p will suffice (we chose 5 because the calculation does not work modulo 2, for reasons to be described later, and 3 divides one of the leading coefficients). In this example, the result was that the polynomials are relatively prime. This raises several questions about generalising this calculation to an algorithm capable of calculating the g.c.d. of any pair of polynomials:

1. how do we calculate a non-trivial g.c.d.?

2. what do we do if the modular g.c.d. is not the modular image of the g.c.d. (as in the example in the footnote[1])?

3. how much does this method cost?

4.1.1.1 Bounds on divisors

Before we can answer these questions, we have to be able to bound the coefficients of the g.c.d. of two polynomials.

Theorem 20 (Landau–Mignotte Inequality [Lan05, Mig74, Mig82]) *Let $Q = \sum_{i=0}^{q} b_i x^i$ be a divisor of the polynomial $P = \sum_{i=0}^{p} a_i x^i$ (where a_i and b_i are integers). Then*

$$\max_{i=0}^{q} |b_i| \leq \sum_{i=0}^{q} |b_i| \leq 2^q \left| \frac{b_q}{a_p} \right| \sqrt{\sum_{i=0}^{p} a_i^2}.$$

These results are corollaries of statements in Appendix A.2.2.

If we regard P as known and Q as unknown, this formulation does not quite tell us about the unknowns in terms of the knowns, since there is some dependence on Q on the right, but we can use a weaker form:

$$\sum_{i=0}^{q} |b_i| \leq 2^p \sqrt{\sum_{i=0}^{p} a_i^2}.$$

When it comes to greatest common divisors, we have the following result.

Corollary 6 *Every coefficient of the g.c.d. of* $A = \sum_{i=0}^{\alpha} a_i x^i$ *and* $B = \sum_{i=0}^{\beta} b_i x^i$ *(with* a_i *and* b_i *integers) is bounded by*

$$2^{\min(\alpha,\beta)} \gcd(a_\alpha, b_\beta) \min\left(\frac{1}{|a_\alpha|} \sqrt{\sum_{i=0}^{\alpha} a_i^2}, \frac{1}{|b_\beta|} \sqrt{\sum_{i=0}^{\beta} b_i^2} \right).$$

Proof. The g.c.d. is a factor of A and of B, the degree of which is, at most, the minimum of the degrees of the two polynomials. Moreover, the leading coefficient of the g.c.d. has to divide the two leading coefficients of A and B, and therefore has to divide their g.c.d.

A slight variation of this corollary is provided by the following result.

Corollary 7 *Every coefficient of the g.c.d. of* $A = \sum_{i=0}^{\alpha} a_i x^i$ *and* $B = \sum_{i=0}^{\beta} b_i x^i$ *(where* a_i b_i *are integers) is bounded by*

$$2^{\min(\alpha,\beta)} \gcd(a_0, b_0) \min\left(\frac{1}{|a_0|} \sqrt{\sum_{i=0}^{\alpha} a_i^2}, \frac{1}{|b_0|} \sqrt{\sum_{i=0}^{\beta} b_i^2} \right).$$

Proof. If $C = \sum_{i=0}^{\gamma} c_i x^i$ is a divisor of A, then $\hat{C} = \sum_{i=0}^{\gamma} c_{\gamma-i} x^i$ is a divisor of $\hat{A} = \sum_{i=0}^{\alpha} a_{\alpha-i} x^i$, and conversely. Therefore, the last corollary can be applied to $\hat{A}$ and $\hat{B}$, and this yields the bound stated.

It may seem strange that the coefficients of a g.c.d. of two polynomials can be greater than the coefficients of the polynomials themselves. One example which shows this is the following (due to Davenport and Trager):

$$\begin{aligned} A = &\quad x^3 + x^2 - x - 1 &&= (x+1)^2(x-1); \\ B = &\quad x^4 + x^3 + x + 1 &&= (x+1)^2(x^2 - x + 1); \\ \gcd(A,B) = &\quad x^2 + 2x + 1 &&= (x+1)^2. \end{aligned}$$

This example can be generalised, as say

$$\begin{aligned} A = &\quad x^5 + 3x^4 + 2x^3 - 2x^2 - 3x - 1 &&= (x+1)^4(x-1); \\ B = &\quad x^6 + 3x^5 + 3x^4 + 2x^3 + 3x^2 + 3x + 1 &&= (x+1)^4(x^2 - x + 1); \\ \gcd(A,B) = &\quad x^4 + 4x^3 + 6x^2 + 4x + 1 &&= (x+1)^4. \end{aligned}$$

In fact, Mignotte [Mig81] has shown that the number 2 in corollaries 6 and 7 is asymptotically the best possible, i.e. it cannot be replaced by any smaller c.

4.1.1.2 The modular – integer relationship

In this sub-section, we answer the question raised above: what do we do if the modular g.c.d. is not the modular image of the g.c.d. calculated over the integers?

Lemma 5 *If* p *does not divide the leading coefficient of* $\gcd(A,B)$*, the degree of* $\gcd(A_p, B_p)$ *is greater than or equal to that of* $\gcd(A,B)$*.*

Proof. Since $\gcd(A, B)$ divides A, then $(\gcd(A, B))_p$ divides A_p. Similarly, it divides B_p, and therefore it divides $\gcd(A_p, B_p)$. This implies that the degree of $\gcd(A_p, B_p)$ is greater than or equal to that of $\gcd(A, B)_p$. But the degree of $\gcd(A, B)_p$ is equal to that of $\gcd(A, B)$, for the leading coefficient of $\gcd(A, B)$ does not cancel when it is reduced modulo p.

This lemma is not very easy to use on its own, for it supposes that we know the g.c.d. (or at least its leading coefficient) before we are able to check whether the modular reduction has the same degree. But this leading coefficient has to divide the two leading coefficients of A and B, and this gives a formulation which is easier to use.

Corollary 8 *If p does not divide the leading coefficients of A and of B (it may divide one, but not both), then the degree of $\gcd(A_p, B_p)$ is greater than or equal to that of $\gcd(A, B)$.*

As the g.c.d. is the only polynomial (to within an integer multiple) of its degree which divides A and B, we can test the correctness of our calculations of the g.c.d.: if the result has the degree of $\gcd(A_p, B_p)$ (where p satisfies the hypothesis of this corollary) and if it divides A and B, then it is the g.c.d. (to within an integer multiple).

It is quite possible that we could find a $\gcd(A_p, B_p)$ of too high a degree. For example, in the case cited above, $\gcd(A_2, B_2) = x + 1$ (it is obvious that $x + 1$ divides the two polynomials modulo 2, because the sum of the coefficients of each polynomial is even). The following lemma shows that this possibility can only arise for a finite number of p.

Lemma 6 *Let $C = \gcd(A, B)$. If p satisfies the condition of the corollary above, and if p does not divide $\mathrm{Res}_x(A/C, B/C)$, then $\gcd(A_p, B_p) = C_p$.*

Proof. A/C and B/C are relatively prime, for otherwise C would not be the g.c.d. of A and B. By the corollary, C_p does not vanish. Therefore

$$\gcd(A_p, B_p) = C_p \gcd(A_p/C_p, B_p/C_p).$$

For the lemma to be false, the last g.c.d. has to be non-trivial. This implies that the resultant $\mathrm{Res}_x(A_p/C_p, B_p/C_p)$ vanishes, by proposition 47 of the Appendix. This resultant is the determinant of a Sylvester matrix, and $|M_p| = (|M|)_p$, for the determinant is only a sum of products of the coefficients. In the present case, this amounts to saying that $\mathrm{Res}_x(A/C, B/C)_p$ vanishes, that is that p divides $\mathrm{Res}_x(A/C, B/C)$. But the hypotheses of the lemma exclude this possibility.

Definition 60 *If $\gcd(A_p, B_p) = \gcd(A, B)_p$, we say that the reduction of this problem modulo p is* good*, or that p is* of good reduction*. If not, we say that p is* of bad reduction*.*

This lemma implies, in particular, that there are only a finite number of values of p such that $\gcd(A_p, B_p)$ does not have the same degree as that of $\gcd(A, B)$, that is the p which divide the g.c.d. of the leading coefficients and the p which divide the resultant of the lemma (the resultant is non-zero, and therefore has only a finite number of divisors). In particular, if A and B are relatively prime, we can always find a p such that A_p and B_p are relatively prime.

4.1.1.3 Computing the g.c.d.: one large prime

In this section we answer the question posed earlier: how do we calculate a non-trivial g.c.d.? One obvious method is to use the Landau-Mignotte inequality, which can determine an M such that all the coefficients of the g.c.d. are bounded by M, and to calculate modulo a prime number greater than $2M$. This method translates into the following algorithm (where `Landau_Mignotte_bound`(A, B) applies corollary 6 and/or corollary 7, and `find_large_prime`$(2M)$ produces a different prime $> 2M$ each tine it is called within a given invocation of the algorithm). We restrict ourselves to momic polynomials, and assume `modular_gcd` gives a monic result, to avoid the problem that a modular g.c.d. is only defined up to a constant multiple.

Algorithm 10 (Modular GCD (Large prime version))
Input: A, B *monic polynomials in* $\mathbf{Z}[x]$.
Output: $\gcd(A, B)$

M :=`Landau_Mignotte_bound`(A, B);
do p :=`find_large_prime`$(2M)$;
  if p does not divide $\gcd(\mathrm{lc}(A), \mathrm{lc}(B))$
    then C :=`modular_gcd`(A, B, p);
      if C divides A and C divides B
        then return C
  forever #Lemma 6 guarantees termination

If the inputs are not monic, we still know that the leading coefficient of the g.c.d. divides each leading coefficient $\mathrm{lc}(A)$ and $\mathrm{lc}(B)$, and therefore their g.c.d. $g = \gcd(\mathrm{lc}(A), \mathrm{lc}(B))$. We therefore compute

$$C := \mathrm{pp}(g\texttt{modular\_gcd}(A, B) \pmod M)$$

instead, where the pp is needed in case the leading coefficient of the g.c.d. is a propoer factor of g.

It is tempting to argue that this algorithm will only handle numbers of the size of twice the Landau–Mignotte bound, but this belief has two flaws.

- While we have proved that there are only finitely many bad primes, we have said nothing about how many there are. The arguments can in fact be made effective, but the results tend to be unduly pessimistic, since it is extremely likely that all the bad primes would be clsutered just above $2M$.

- In theory, the division tests could yield very large numbers if done as tests of the remainder being zero: for example the remainder on dividing x^{100} by $x - 10$ is 10^{100}. This can be solved by a technique known as "early abort" trial division.

  Proposition 39 *If h, of degree m, is a factor of f of degree n, the coefficient of x^{n-m-i} in the quotient is bounded by $\binom{n-m}{i}\frac{1}{\mathrm{lc}(h)}||f||$.*

This is basically Corollary 17. Hence, as we are doing the trial division, we can give up as soon as we find a coefficient in the quotient that exceeds this bound, which *is* closely related to M (the difference relates to the leading coefficient terms).

For example, if we take $p = 7$ in the example at the start of this chapter, we find that the g.c.d. of A_7 and B_7 is $x + 10$ (it could equally well be $x + 3$, but $x + 10$ makes the point better). Does $x + 10$ divide B? We note that $||B|| \approx 23.92$. Successive terms in the quotient are $3x^5$ (and 3 is a permissible coefficient), $-30x^4$ (and $30 < \binom{5}{1} \times 23.92$) and $305x^3$, at which point we observe that $305 > \binom{5}{2} \times 23.92 = 239.2$, so this *cannot* be a divisor of B. hence 7 was definitely unlucky.

With this refinement, it is possible to state that the numbers dealt with in this algorithm are "not much larger" than $2M$, thugh considerable ingenuity is needed to make this statement more precise.

If we apply this algorithm to the polynomials at the start of this section, we deduce that $\sqrt{\sum_{i=0}^{8} a_i^2} = \sqrt{113}$, $\sqrt{\sum_{i=0}^{6} b_i^2} = 2\sqrt{143}$, and hence corollary 6 gives a bound of

$$2^6 \min\left(\sqrt{113}, \frac{2}{3}\sqrt{143}\right) \approx 510.2, \tag{4.3}$$

so our first prime would be 1021, which is indeed of good reduction. In this case, corollary 7 gives a bound of

$$2^6 \min\left(\frac{1}{5}\sqrt{113}, \frac{2}{21}\sqrt{143}\right) \approx 72.8, \tag{4.4}$$

so our first prime would be 149. In general, we cannot tell which gives us the best bound, and it is normal to take the minimum.

Open Problem 5 *A significant factor in the Landau–Mignotte bound here, whether (4.3) or (4.4), was the $2^{\min(8,6)}$ contribution from the degree of the putative g.c.d. But in fact the exponent is at most 4, not 6, since the g.c.d. cannot have leading coefficient divisible by 3 (since A does not). Hence the g.c.d. must have at most the degree of the g.c.d. modulo 3, and modulo 3 B has degree 4.*

Can this be generalised, and is it worth it? In view of the 'early success' strategies discussed later, the answer to the last part is probably negative.

4.1.1.4 Computing the g.c.d.: several small primes

While algorithm 10 does give us some control on the size of the numbers being considered, we are still *often* using numbers larger than those which hindsight would show to be necessary. For example, in (4.1), (4.2) we could deduce coprimeness using the prime 5, rather than 1021 from (4.3) or 149 from (4.4). If instead we consider $(x-1)A$ and $(x-1)B$, the norms change, giving 812.35 in (4.3) (a prime of 1627) and 116.05 in (4.4) (a prime of 239). Yet primes such

Figure 4.1: Algorithm 11

Algorithm 11 (Modular GCD (Small prime version))
Input: A, B *polynomials in* $\mathbf{Z}[x]$.
Output: $\gcd(A, B)$

M :=`Landau_Mignotte_bound`(A, B);
$g := \gcd(\mathrm{lc}(A), \mathrm{lc}(B))$;
$p :=$ `find_prime`(g);
$C :=$ `gmodular_gcd`(A, B, p);
if $\deg(C) = 0$ **then return** 1
$N := p$; # N is the modulus we will be constructing
while $N < 2M$ **repeat**
    $p :=$ `find_prime`(g);
    $C :=$ `gmodular_gcd`(A, B, p);
    if $\deg(C) = \deg(D)$
      then $D :=$ Algorithm 17(C, D, p, N);
           $N := pN$;
      else if $\deg(C) < \deg(D)$
           # C proves that D is based on primes of bad reduction
           if $\deg(C) = 0$ **then return** 1
           $D := C$;
           $N := p$;
      else #D proves that p is of bad reduction, so we ignore it
$D := \mathrm{pp}(D)$; # In case multiplying by g was overkill
Check that D divides A and B, and return it
If not, all primes must have been bad, and we start again

as 5, 11, 13 etc. will easily show that the result is $x - 1$. Before we leap ahead and use such primes, though, we should reflect that, had we taken $(x - 10)A$ and $(x - 10)B$, 5 would have suggested x as the gcd, 11 would have suggested $x + 1$, 13 would have suggested $x + 3$ and so on.

The answer to this comes in observing that the smallest polynomial (in terms of coefficient size) which is congruent to x modulo 5 *and* to $x + 1$ modulo 11 is $x-10$ (it could be computed by algorithm 17). More generally, we can apply the Chinese Remainder Theorem (Theorem 22) to enough primes of good reduction, as follows. We assume that `find_prime`(g) returns a prime not dividing g, a different one each time. The algorithms is given in Figure 4.1 We should note the heavy reliance on Corollary 8 to detect bad reduction. We impose g as the leading coefficient throughout, and make the result primitive at the end as in the large prime variant.

4.1.1.5 Computing the g.c.d.: early success

While Algorithm 11 will detect a g.c.d. of 1 early, it will otherwise compute as far as the Landau–Mignotte bound if the g.c.d. is not 1. While this may be encessary, it would be desirable to terminate earlier if we have already found the g.c.d. This is easily done by replacing the line

then $D :=$ Algorithm 17(C, D, p, N);

by

then $D' := D$
$D :=$ Algorithm 17(C, D, p, N);
if $D = D'$ #We *may* have found the answer
then $E :=$ pp(D);
if E divides A and B
then return E;
Otherwise this was a false alert, and we continue as normal.

We should note that we return an E which divides the inputs, and is derived from modular images, and therefore has to be the *greatest* common divisor by Corollary 8.

4.1.2 Polynomials in several variables

The same techniques can be used to compute the greatest common divisor of polynomials in several variables. This is even more important than in the case of univariate polynomials, since the coeffcient growth observed on page 36 becomes degree growth in the other variables. As a trivial example of this, we can consider

$$\begin{aligned} A &= (y^2 - y - 1)x^2 - (y^2 - 2)x + (2y^2 + y + 1); \\ B &= (y^2 - y + 1)x^2 - (y^2 + 2)x + (y^2 + y + 2). \end{aligned}$$

The first elimination gives

$$C = (2y^2 - 4y)x + (y^4 - y^3 + 2y^2 + 3y + 3),$$

and the second gives

$$D = -y^{10} + 3\,y^9 - 10\,y^8 + 11\,y^7 - 23\,y^6 + 22\,y^5 - 37\,y^4 + 29\,y^3 - 32\,y^2 + 15\,y - 9.$$

The algorithm is based on Algorithm 11, except that evaluating a variable at a value replaces working modulo a prime.

4.1.3 Conclusions

4.2 p-adic Methods

In this section, we wish to consider a different problem, that of factoring polynomials.

4.2.1 Introduction to the factorization problem

For simplicity, we will begin with the case of factoring a univariate polynomial over $\mathbf{Z}$. More precisely, we consider the following.

Problem 6 *Given $f \in \mathbf{Z}[x]$, compute polynomials $f_i \in \mathbf{Z}[x]$ $(1 \leq i \leq k)$ such that:*

1. *$f = \prod_{i=1}^{k} f_i$;*
2. *each f_i is irreducible in $\mathbf{Z}[x]$, i.e. any polynomial g that divides f_i is either an integer or has the same degree as f_i.*

We might wonder whether we wouldn't be better off considering $f_i \in \mathbf{Q}[x]$, but in fact the answers are the same.

Proposition 40 *Any factorization over $\mathbf{Q}[x]$ of a polynomial $f \in \mathbf{Z}[x]$ is (up to rational multiples) a factorization over $\mathbf{Z}[x]$.*

Proof. Let $f = \prod_{i=1}^{k} f_i$ with $f_i \in \mathbf{Q}[x]$. By clearing denoninators and removing contents, we can write $f_i = c_i g_i$ with $g_i \in \mathbf{Z}[x]$ and primitive and $c_i \in \mathbf{Q}$. Hence $f = \left(\prod_{i=1}^{k} c_i\right)\left(\prod_{i=1}^{k} g_i\right)$, and, since the product of primitive polynomials is primitive (Lemma 2), $\prod_{i=1}^{k} c_i$ is an integer, and can be absorbed into, say, g_1.

Even knowing that we have only to consider integer coefficients does not seem to help much — we still seem to have an infinite number of possibilities to consider. In fact this is not quite the case.

Notation 11 *Let the polynomial $f = \sum_{i=0}^{n} a_i x^i$ to be factored have degree n, and coefficients bounded by H. Let us suppose we are looking for factors of degree at most d.*

Corollary 9 (to Theorem 20) *It is sufficient to look for factors of degree $d \leq n/2$, whose coefficients are bounded by $2^d H$.*

One might have hoped that it was sufficient to look for factors whose coefficients are bounded by H, but this is not the case. [Abb09] gives the example of

$$\begin{aligned} f \;=\; & x^{80} - 2x^{78} + x^{76} + 2x^{74} + 2x^{70} + x^{68} + 2x^{66} + x^{64} + x^{62} + 2x^{60} + 2x^{58} \\ & -2x^{54} + 2x^{52} + 2x^{50} + 2x^{48} - x^{44} - x^{36} + 2x^{32} + 2x^{30} + 2x^{28} - 2x^{26} \\ & +2x^{22} + 2x^{20} + x^{18} + x^{16} + 2x^{14} + x^{12} + 2x^{10} + 2x^6 + x^4 - 2x^2 + 1 \end{aligned}$$

whose factors have coefficients as large as 36, i.e. 18 times as large as the coefficients of f.

Corollary 10 *To detect all irreducible factors of f (except possibly for the last one, which is f divided by all the factors of degree $\leq n/2$), it suffices to consider $\left(2^d H\right)^{d+1}$ polynomials.*

We can in fact do better, since the leading coefficient of the factor must divide a_n, and similarly the trailing coefficient must divide a_0, so we get $2^{d(d-1)} H^{d+1}$, and in practice such "brute force" methods[2] can easily factor low-degree polynomials, but the asymptotics are still very poor.

[2]Sometimes known as Newton's algorithm.

4.2.2 Modular methods

Hence we might want to use modular methods. Assuming always that p does not divide a_n, we know that, *if* f is irreducible modulo p, it is irreducible over the integers. Since, modulo p, we can assume our polynomials are monic, there are only p^d polynomials of degree d, this gives us a bound that is exponential in d rather than d^2.

In fact, we can do better by comparing results of factorizations modulo different primes. For example, if f is quartic, and factors modulo p into a linear and a cubic, and modulo q into two quadratics (all such factors being irreducible), we can deduce that it is irreducible over the integers, since no factorization over the integers is compatible with both pieces of information.

It is an experimental observation [Mus78] that, *if* a polynomial can be proved irreducible this way, five primes are nearly always sufficient to prove it. This seems to be the case for polynomials chosen "at random"[3]: more might be required if the shape of the polynomial is not random [Dav97].

4.2.3 Factoring modulo a prime

Throughout this section, we let p be an odd[4] prime, and f a polynomial of degree n which is square-free modulo p.

4.2.3.1 Berlekamp's small p method

4.2.3.2 Berlekamp's large p method

4.2.3.3 The Cantor–Zassenhaus method

This method is generally attributed to [CZ81][5]. It is based on the following generalization of Fermat's (Little Theorem).

Proposition 41 *All irreducible polynomials of degree d with coefficients modulo p divide $x^{p^d} - x$.*

Corollary 11 *All irreducible polynomials of degree d with coefficients modulo p divide $x^{p^d-1} - 1$, except for x itself in the case $d = 1$. Furthermore, no irreducible polynomials of degree more than d divide $x^{p^d-1} - 1$.*

Corollary 12 *Half of the irreducible polynomials of degree d (except for x itself in the case $d = 1$) with coefficients modulo p divide $x^{(p^d-1)/2} - 1$.*

Algorithm 12 (Distinct Degree Factorization)
Input: *$f(x)$ a square-free polynomial modulo p, not divisible by x; a prime p*
Output: *A decomposition $f = \prod f_i$, where each f_i is the product of irreducibles of degree i*

[3] Which means that the Galois gorup of the polynomial is almost always S_n.

[4] It is possible to generalise these methods to the case $p = 2$, but the cost, combined with the unlikelihood of 2 being a good prime, means that computer algebraists rarely do so.

[5] Though [Zip93] credits algorithm 12 to [Arw18].

```
i:=1
while 2i ≤ deg(f)
        g := x^{p^i} - 1   (mod f)          (*)
        f_i := gcd(g - 1, f)
        f := f/f_i
        i := i + 1
if      f ≠ 1
        then f_{deg(f)} := f
```

Note that the computation in line (*) should be done by the repeated squaring method, reducing modulo f at each stage. We can save time in practice by re-using the previous g.

If f_i has degree i, then it is clearly irreducible: otherwise we have to split it. This is the purpose of the next algorithm, which relies on a generalization of Corollary 11.

Proposition 42 ([CZ81, p. 589]) *Let f be a product of $r > 1$ irreducible polynomials of degree d modulo p, and g a random (non-constant) polynomial of degree $< d$. Then the probability that $\gcd(g^{(p^d-1)/2} - 1, f)$ is either 1 or f is at most 2^{1-r}.*

Algorithm 13 (Split a Distinct Degree Factorization)
Input: *A prime p, a degree d and a polynomial $f(x) \pmod p$ known to be the product of irreducibles of degree d (and not divisible by x)*
Output: *The factorization of f (modulo p) into irreducibles of degree d.*

```
if d = deg(f)
  then return f            # degree d so is irreducible
W := {f}
ans := ∅
while W ≠ ∅                # factors of degree d found
  h :=RandomPoly(p,d)
  h := h^{(p^i-1)/2}  (mod g)          (*)
  V := ∅                   # list of polynomials to be processed
  for g ∈ W do
      h_1 := gcd(h - 1, g)
      if deg(h_1) = 0 ∨ deg(h_1) = deg(g)
        then V := V ∪ {g}      # we made no progress
        else process(h_1)
             process(g/h_1)
  W := V
return ans
```

Here RandomPoly(p,d) returns a random non-constant polynomial modulo p of degree less than d, and the sub-function process(g_1) takes a polynomial g_1 which is a result of splitting g by h_1, and adds it to *ans* if it has degree d, otherwise adds it to V.

4.2.4 The recombination problem

However, as pointed out above, the fact that we have a factorization of f modulo p^k, where $p^k > 2M$, M being the Landau–Mignotte bound (Theorem 20), or some alternative (see page 120) on the size of any coefficients in any factor of f, does not solve our factorization problem. It is perfectly possibly that f is irreducible, or that f does factor, but less than it does modulo p.

Assuming always that p does not divide $\mathrm{lc}(f)$, all that can happen when we reduce f modulo p is that a polynomial that was irreducible over the integers now factors modulo p (and hence modulo p^k). This gives us the following algorithm, first suggested in [Zas69].

Algorithm 14 (Combine Modular Factors)
Input: *A prime power p^k, a monic polynomial $f(x)$ over the integers, a factorisation $f = \prod_{i=1}^{r} f_i$ modulo p^k*
Output: *The factorization of f into monic irreducible polynomials over the integers.*

$ans := \emptyset$
$M :=$LandauMignotteBound(f)
$S := \{f_1, \ldots, f_r\}$
for subsets T of S
    $h := \prod_{g \in T} g$
    if h divides f
        then $ans := ans \cup \{h\}$
            $f := f/h$
            $S := S \setminus T$
            $M :=$ min(M,LandauMignotteBound(f))

To guarantee irreducibility of the factors found, the loop must try all subsets of T before trying T itself, but this still leaves a lot of choices for the loop: see 1 below.

The running time of this algorithm is, in the worst case, exponential in r since 2^{r-1} subsets of S have to be considered (2^{r-1} since considering T effectively also considers $S \setminus T$). Let n be the degree of f, and H a bound on the coefficients of f, so the Landau–Mignotte bound is at most $2^n(n+1)H$, and $k \leq \log_p(2^{n+1}(n+1)H)$.

Many improvements to, or alternatives to, this basic algorithm have been suggested since. In essentially chronological order, the most significant ones are as follows.

1. [Col79] pointed out that there were two obvious ways to code the "**for** subsets T of S" loop: increasing cardinality of T and increasing degree of $\prod_{g \in T} g$. He showed that, subject to two very plausible conjectures, the average number of products actually formed with the cardinality ordering was $O(n^2)$, thus the *average* running time would be polynomial in n.

2. [LLJL82] had a completely different approach to algorithm 14. They asked, for each $d < n$, "given $f_1 \in S$, what is the polynomial g of degree d which divides f over the integers and is divisible by f_1 modulo p^k. Unfortunately, answering this question needed a k far larger than that implied by the Landau–Mignotte bound, and the complexity, while polynomial in n, was $O(n^{12})$, at least while using classical arithmetic. This paper introduced the 'LLL' lattice reduction algorithm, which has many applications in computer algebra and far beyond.

3. [ABD85] showed that,by a combination of simple divisibility tests and "early abort" trial division (Proposition 39) it was possible to make dramatic reductions, at the time up to four orders of magnitude, in the constant implied in the statement "exponential in r".

4. [ASZ00] much improved this, and the authors were able to eliminate whole swathes of possible T at one go.

5. [vH02] reduces the problem to a 'knapsack' problem, which is also solved by LLL, but the lattices involved are much smaller — of dimension r rather than n. At the time of writing, this seems to be the best known method. His paper quoted a polynomial of degree $n = 180$, with $r = 36$ factors of degree 5 modulo $p = 19$, but factoring as two polynomials of degree 90 over the integers. This took 152 seconds to factor.

4.2.5 Conclusions

1. Although no system to the author's knowledge implements it, it would be possible to implement an efficient procedure to find all factors of limited *total* degree d. This would be efficient for three reasons:

   - the Cantor–Zassenhaus algorithm (section 4.2.3.3) need only run up to maximum degree d;
   - it should be possible to use smaller bounds on the lifting
   - The potentially-exponential recombination process need only run up to total degree d. In particular, if $d = 1$, no recombination is needed.

Chapter 5

Calculus

Throughout this chapter we shall assume that we are in characteristic zero, and therefore all rings contains **Z**, and all fields contain **Q**. The emphasis in this chapter will be on *algorithms* for integration. Historically, the earliest attempts at integration in computer algebra [Sla61] were based on rules, and attempts to "do it like a human would". These were rapidly replaced by algorithmic methods, based on the systematisation in [Ris69] of work going back to Liouville. Recently, attempts to get 'neat' answers have revived interest in rule-based approaches, which can be surprisingly powerful on *known* integrals — see [JR10] for one recent approach. In general, though, only the algorithmic approach is capable of *proving* that an expression in unintegrable.

5.1 Introduction

We defined (Definition 28) the *formal derivative* of a polynomial purely algebraically, and observed (Proposition 2.3.4) that it satisfied the sum and product laws. We can actually make the whole theory of differentiation algebraic as follows.

Definition 61 *A* differential ring *is a ring equipped with an additional unary operation, referred to as differentiation and normally written with a postfix* $'$*, which satisfies two additional laws:*

1. $(f+g)' = f' + g'$;
2. $(fg)' = fg' + f'g$.

A differential ring which is also a field is referred to as a differential field.

Definition 28 and Proposition 2.3.4 can then be restated as the following result.

Proposition 43 *If R is any ring, we can make $R[x]$ into a differential ring by defining $r' = 0 \quad \forall r \in R$ and $x' = 1$.*

Definition 62 *In any differential ring, an element α with $\alpha' = 0$ is referred to as a* constant.

We will revisit this definition in section 6.2.

Proposition 44 *In any differential field*

$$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}. \tag{5.1}$$

The proof is by differentiating $f = g\left(\frac{f}{g}\right)$. (5.1) is generally referred to as the *quotient rule* and can be given a fairly tedious analytic proof in terms of ϵ/δ, but from our present point of view it is an algebraic corollary of the product rule. It therefore follows that $K(x)$ can be made into a differential field.

Notation 12 (Fundamental Theorem of Calculus) *(Indefinite) integration is the inverse of differentiation, i.e.*

$$F = \int f \Leftrightarrow F' = f. \tag{5.2}$$

The reader may be surprised to see a "Fundamental Theorem" reduced to the status of a piece of notation, but from the present point of view, that is what it is. We shall return to this point in section 6.1. The reader may also wonder "where did $\mathrm{d}x$ go?", but x is, from this point of view, merely that object such that $x' = 1$, i.e. $x = \int 1$.

We should also note that we have no choice over the derivative of algebraic functions.

Proposition 45 *Let K be a differential field, and θ be algebaric over K with $p(\theta) = 0$ for some polynomial $p = \sum_{i=0}^{n} a_i z^i \in K[z]$. Then $K(\theta)$ can be made into a differential field in only one way: by defining*

$$\theta' = -\frac{\sum_{i=0}^{n} a_i' \theta^i}{\sum_{i=0}^{n} i a_i \theta^{i-1}}. \tag{5.3}$$

In particular, if the coefficients of p are all constants, so is θ.

The proof is by formal differentiation of $p(\theta) = 0$.

5.2 Integration of Rational Functions

The integration of polynomials is trivial:

$$\int \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} \frac{1}{i+1} a_i x^{i+1}. \tag{5.4}$$

Since any rational function $f(x) \in K(x)$ can be written as

$$f = p + \frac{q}{r} \text{ with } \left\{ \begin{array}{l} p, q, r \in K[x] \\ \deg(q) < \deg(r) \end{array} \right. , \tag{5.5}$$

and p is always integrable by (5.4), we have proved the following (trivial) result.

Proposition 46 (Decomposition Lemma (rational functions)) *In the notation of (5.5), f is integrable if, and only if, q/r is.*

q/r with $\deg(q) < \deg(r)$ is generally termed a *proper rational function*.

5.2.1 Integration of Proper Rational Functions

In fact, the integration of proper rational functions is conceptually trivial (we may as well assume r is monic, absorbing any constant factor in q):

1. perform a square-free decomposition (Definition 29) of $r = \prod_{i=1}^{n} r_i^i$;
2. factorize each r_i completely, as $r_i(x) = \prod_{j=1}^{n_i}(x - \alpha_{i,j})$;
3. perform a partial fraction decomposition of q/r as

$$\frac{q}{r} = \frac{q}{\prod_{i=1}^{n} r_i^i} = \sum_{i=1}^{n} \frac{q_i}{r_i^i} = \sum_{i=1}^{n} \sum_{j=i}^{n_i} \sum_{k=1}^{i} \frac{\beta_{i,j,k}}{(x - \alpha_{i,j})^k}; \tag{5.6}$$

4. integrate this term-by-term, obtaining

$$\int \frac{q}{r} = \sum_{i=1}^{n} \sum_{j=i}^{n_i} \sum_{k=2}^{i} \frac{-\beta_{i,j,k}}{(k-1)(x - \alpha_{i,j})^{k-1}} + \sum_{i=1}^{n} \sum_{j=i}^{n_i} \beta_{i,j,1} \log(x - \alpha_{i,j}). \tag{5.7}$$

From a practical point of view, this approach has several snags:

1. we have to factor r, and even the best algorithms from the previous chapter can be expensive;
2. we have to factor each r_i *into linear factors*, which might necessitate the introduction of algebraic numbers to represent the roots of polynomials;
3. These steps might result in a complicated expression of what is otherwise a simple answer.

To illustrate these points, consider the following examples.

$$\begin{array}{c} \displaystyle\int \frac{5x^4 + 60x^3 + 255x^2 + 450x + 274}{x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120} \mathrm{d}x \\ = \log(x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120) \\ = \log(x+1) + \log(x+2) + \log(x+3) + \log(x+4) + \log(x+5) \end{array} \tag{5.8}$$

is pretty straightforward, but adding 1 to the numerator gives

$$\begin{gathered}\int \frac{5x^4+60x^3+255x^2+450x+275}{x^5+15x^4+85x^3+225x^2+274x+120}\mathrm{d}x \\ = \tfrac{5}{24}\log(x^{24}+72x^{23}+\cdots+102643200000x+9331200000) \\ = \tfrac{25}{24}\log(x+1)+\tfrac{5}{6}\log(x+2)+\tfrac{5}{4}\log(x+3)+\tfrac{5}{6}\log(x+4)+\tfrac{25}{24}\log(x+5)\end{gathered} \tag{5.9}$$

Adding 1 to the denominator is pretty straightforward,

$$\begin{gathered}\int \frac{5x^4+60x^3+255x^2+450x+274}{x^5+15x^4+85x^3+225x^2+274x+121}\mathrm{d}x \\ = \log(x^5+15x^4+85x^3+225x^2+274x+121),\end{gathered} \tag{5.10}$$

but adding 1 to both gives

$$\begin{gathered}\int \frac{5x^4+60x^3+255x^2+450x+275}{x^5+15x^4+85x^3+225x^2+274x+121}\mathrm{d}x \\ = 5\sum_{\alpha}\alpha\,\ln\Big(x+\frac{2632025698}{289}\,\alpha^4-\frac{2086891452}{289}\,\alpha^3+ \\ \frac{608708804}{289}\,\alpha^2-\frac{4556915}{17}\,\alpha+\frac{3632420}{289}\Big),\end{gathered} \tag{5.11}$$

where

$$\alpha = \mathrm{RootOf}\left(38569\,z^5-38569\,z^4+15251\,z^3-2981\,z^2+288\,z-11\right). \tag{5.12}$$

Hence the challenge is to produce an algorithm that achieves (5.8) and (5.10) simply, preferably gives us the second form of the answer in (5.9), *but* is still capable of solving (5.11). We might also wonder where (5.11) came from.

5.2.2 Hermite's Algorithm

The key to the method (usually attributed to Hermite [Her72], though informally known earlier) is to rewrite equation (5.7) as

$$\int \frac{q}{r} = \frac{s_1}{t_1} + \int \frac{s_2}{t_2}, \tag{5.13}$$

where the integral on the right-hand resolves itself as *purely* a sum of logarithms, i.e. is the $\sum_{i=1}^{n}\sum_{j=i}^{n_i}\beta_{i,j,1}\log(x-\alpha_{i,j})$ term. Then a standard argument from Galois theory shows that s_1, t_1, s_2 and t_2 do not involve any of the α_i, i.e. that the decomposition (5.13) can be obtained without introducting any algebraic numbers. If we could actually obtain this decomposition without introducing these algebraic numbers, we would have gone a long way to solving objection 2 above.

We can perform a square-free decomposition (Definition 29) of r as $\prod r_i^i$, and then a partial fraction decomposition to write

$$\frac{q}{r} = \sum \frac{q_i}{r_i^i} \tag{5.14}$$

and, since each term is a rational function and therefore integrable, it suffices to integrate (5.14) term-by-term.

Now r_i and r_i' are relatively prime, and hence there are an a and a b satisfying $ar_i + br_i' = 1$. Therefore

$$\begin{aligned}
\int \frac{q_i}{r_i^i} &= \int \frac{q_i(ar_i + br_i')}{r_i^i} && (5.15)\\
&= \int \frac{q_i a}{r_i^{i-1}} + \int \frac{q_i b r_i'}{r_i^i} && (5.16)\\
&= \int \frac{q_i a}{r_i^{i-1}} + \int \frac{(q_i b/(i-1))'}{r_i^{i-1}} - \left(\frac{q_i b/(i-1)}{r_i^{i-1}}\right)' && (5.17)\\
&= -\left(\frac{q_i b/(i-1)}{r_i^{i-1}}\right) + \int \frac{q_i a + (q_i b/(i-1))'}{r_i^{i-1}}, && (5.18)
\end{aligned}$$

and we have reduced the exponent of r_i by one. When programming this method one may need to take care of the fact that, while $\frac{q_i}{r_i^i}$ is a proper rational function, $\frac{q_i b}{r_i^{i-1}}$ may not be, but the excess is precisely conpensated for by the other term in (5.18).

Hence, at the cost of a square-free decomposition and a partial fraction decomposition, but *not* a factorization, we have found the rational part of the integral, i.e. performed the decomposition of (5.14). In fact,w ehave done somewhat better, since the $\int \frac{s_2}{t_2}$ term will have been split into summands corresponding to the different r_i.

5.2.3 The Horowitz–Ostrogradski Algorithm

Although quite simple, Hermite's method still needs square-free decomposition and partial fractions. Horowitz [Hor69, Hor71] therefore proposed to computer algebraists the following method, which was in fact already known [Ost45], but forgotten. It follows from (5.18) that, in the notation of (5.14) $t_1 = \prod r_i^{i-1}$. Furthermore evenry factor of t_2 arises from the r_i, and is not repeated. Hence we can choose

$$t_1 = \gcd(r, r') \text{ and } t_2 = r/t_1. \tag{5.19}$$

Having done this, we can solve for the coefficients in s_1 and s_2, and the resulting equations are linear in the unknown coefficients. More precisely, the equations become

$$q = s_1'(r/t_i) - s_1\frac{t_1' t_2}{t_1} + s_2 t_1, \tag{5.20}$$

where the linearity is obvious. The programmer should note that s_1/t_1 is guaranteed to be in lowest terms, but s_2/t_2 is not (and indeed will be 0 if there is no logarithmic term).

5.2.4 The Trager–Rothstein Algorithm

Whether we use the method of section 5.2.2 or 5.2.3, we have to integrate the logarithmic part. (5.8)–(5.11) shows that this may, but need not, require algebraic numbers. How do we tell? The answer is provided by the following observation[1] [Rot76, Tra76]: if we write the integral of the logarithmic part as $\sum c_i \log v_i$, we can determine the equation satisfied by the c_i, i.e. the analogue of (5.12), by purely rational computations.

So write

$$\int \frac{s_2}{t_2} = \sum c_i \log v_i, \tag{5.21}$$

where we can assume:

1. the v_i are polynomials (using $\log \frac{f}{g} = \log f - \log g$);
2. the v_i are square-free (using $\log \prod f_i^i = sumi \log f_i$);
3. the v_i are relatively prime (using $c \log pq + d \log pr = (c+d) \log p + c \log q + d \log r$);
4. the c_i are all different (using $c \log p + c \log q = c \log pq$);
5. the c_i generate the smallest possible extension of the original field of coefficients.

(5.21) can be rewritten as

$$\frac{s_2}{t_2} = \sum c_i \frac{v_i'}{v_i}. \tag{5.22}$$

Hence $t_2 = \prod v_i$ and, writing $u_j = \prod_{i \neq j} v_i$, we can write (5.22) as

$$s_2 = \sum c_i v_i' u_i. \tag{5.23}$$

Furthermore, since $t_2 = \prod v_i$, $t_2' = \sum v_i' u_i$. Hence

$$\begin{aligned}
v_k &= \gcd(0, v_k) \\
&= \gcd\left(s_2 - \sum c_i v_i' u_i, v_k\right) \\
&= \gcd\left(s_2 - c_k v_k' u_k, v_k\right) \\
&\qquad \text{since all the other } u_i \text{ are divisible by } v_k \\
&= \gcd\left(s_2 - c_k \sum v_i' u_i, v_k\right) \\
&\qquad \text{for the same reason} \\
&= \gcd\left(s_2 - c_k t_2', v_k\right).
\end{aligned}$$

[1] As happens surprisingly often in computer algebra, this is a case of simultaneous discovery.

But if $l \neq k$,

$$\begin{aligned} \gcd\left(s_2 - c_k t_2', v_l\right) &= \gcd\left(\sum c_i v_i' u_i - c_k \sum v_i' u_i, v_l\right) \\ &= \gcd\left(c_l v_l' u_l - c_k v_l' u_l, v_l\right) \\ &\qquad \text{since all the other } u_i \text{ are divisible by } v_l \\ &= 1. \end{aligned}$$

Since $t_2 = \prod v_l$, we can put these together to deduce that

$$v_k = \gcd(s_2 - c_k t_2', t_2). \tag{5.24}$$

5.3 Liouville's Theorem

Definition 63 *Let K be a field of functions. The function θ is an elementary generator over K if one of the following is satisfied:*

(a) *θ is algebraic over K, i.e. θ satisfies a polynomial equation with coefficients in K;*

(b) *θ is an exponential over K, i.e. there is an η in K such that $\theta' = \eta'\theta$, which is only an algebraic way of saying that $\theta = \exp \eta$;*

(c) *θ is a logarithm over K, i.e. there is an η in K such that $\theta' = \eta'/\eta$, which is only an algebraic way of saying that $\theta = \log \eta$.*

Definition 64 *Let K be a field of functions. An overfield $K(\theta_1, \ldots, \theta_n)$ of K is called a* field of elementary functions over K *if every θ_i is an elementary generator over K. A function is* elementary over K *if it belongs to a field of elementary functions over K.*
If K is omitted, we understand $\mathbf{C}(x)$: the field of rational functions.

Theorem 21 (Liouville's Principle) *Let f be a function from some function field K. If f has an elementary integral over K, it has an integral of the following form:*

$$\int f = v_0 + \sum_{i=1}^{n} c_i \log v_i, \tag{5.25}$$

where v_0 belongs to K, the v_i belong to $\hat{K}$, an extension of K by a finite number of constants algebraic over K, and the c_i belong to $\hat{K}$ and are constant.

Another way of putting this is to say that, if f has an elementary integral over K, then f has the following form:

$$f = v_0' + \sum_{i=1}^{n} \frac{c_i v_i'}{v_i}. \tag{5.26}$$

5.4 Integration of Logarithmic Functions

Let θ be a (transcendental) logarithm over K.

Lemma 7 (Decomposition Lemma (logarithmic)) *$f \in K(\theta)$ can be written uniquely as $p + q/r$, where p, q and r are polynomials of $K[\theta]$, q and r are relatively prime, and the degree of q is less than that of r. If f has an elementary integral over K, then p and q/r each possess an elementary integral over K.*

5.5 Integration of Exponential Functions

Let θ be a (transcendental) exponential over K.

Lemma 8 (Decomposition Lemma (exponential)) *$f \in K(\theta)$ can be written uniquely as $p + q/r$, where p is a* generalised (or Laurent) polynomial *(that is $\sum_{i=-m}^{n} a_i \theta^i$), q and r are polynomials of $K[\theta]$ such that θ does not divide r, q and r are relatively prime, and the degree of q is less than that of r. If f has an elementary integral over K, then each of the terms of p, and also q/r, have an elementary integral over K.*

Chapter 6

Algebra versus Analysis

We have seen in the previous chapter how we can construct an *algebraic* theory of mathematical objects such as 'exp', 'log' and 'atan'. From an algebraic point of view, they seem to behave like the mathematical objects we are familiar with from analysis. Are they the same? If not, what are the differences? This is perhaps one of the less-discussed topics[1] in computer algebra, and indeed possibly in mathematics more generally.

6.1 Fundamental Theorem of Calculus Revisited

In the previous chapter we reduced the Fundamental Theorem of Calculus to the status of Notation 12.

6.2 Constants Revisited

In the previous chapter we defined (Definition 62) a constant as being an element whose derivative was zero. How well does this compare with our usual intuition, which is of a *constant function*?

[1] But see [Dav10].

Appendix A

Algebraic Background

A.1 The resultant

It quite often happens that we have to consider whether two polynomials, which are usually relatively prime, can have a common factor in certain special cases. The basic algebraic tool for solving this problem is called the resultant. In this section we shall define this object and we shall give some properties of it. We take the case of two polynomials f and g in one variable x and with coefficients in a ring R.

We write $f = \sum_{i=0}^{n} a_i x^i$ and $g = \sum_{i=0}^{m} b_i x^i$.

Definition 65 *The* Sylvester matrix *of f and g is the matrix*

$$\begin{pmatrix}
a_n & a_{n-1} & \dots & a_1 & a_0 & 0 & 0 & \dots & 0 \\
0 & a_n & a_{n-1} & \dots & a_1 & a_0 & 0 & \dots & 0 \\
\vdots & \ddots & \ddots & \ddots & \dots & \ddots & \ddots & \ddots & \vdots \\
0 & \dots & 0 & a_n & a_{n-1} & \dots & a_1 & a_0 & 0 \\
0 & \dots & 0 & 0 & a_n & a_{n-1} & \dots & a_1 & a_0 \\
b_m & b_{m-1} & \dots & b_1 & b_0 & 0 & 0 & \dots & 0 \\
0 & b_m & b_{m-1} & \dots & b_1 & b_0 & 0 & \dots & 0 \\
\vdots & \ddots & \ddots & \ddots & \dots & \ddots & \ddots & \ddots & \vdots \\
0 & \dots & 0 & b_m & b_{m-1} & \dots & b_1 & b_0 & 0 \\
0 & \dots & 0 & 0 & b_m & b_{m-1} & \dots & b_1 & b_0
\end{pmatrix}$$

where there are m lines constructed with the a_i, n lines constructed with the b_i.

Definition 66 *The* resultant *of f and g, written* $\mathrm{Res}(f,g)$*, or* $\mathrm{Res}_x(f,g)$ *if there is doubt about the variable, is the determinant of this matrix.*

Well-known properties of determinants imply that the resultant belongs to R, and that $\mathrm{Res}(f,g)$ and $\mathrm{Res}(g,f)$ are equal, to within a sign. We must note that, although the resultant is defined by a determinant, this is not the best way

of calculating it. Because of the special structure of the Sylvester matrix, we can consider Euclid's algorithm as Gaussian elimination in this matrix (hence the connection betwen the resultant and the g.c.d.). One can also consider the sub-resultant method as an application of the Sylvester identity (theorem 12) to this elimination. It is not very difficult to adapt advanced methods (such as the method of sub-resultants described in section 2.3.2, or the modular and p-adic methods described in chapter 4) to the calculation of the resultant. Collins [1971] and Loos [1982] discuss this problem. We now give a version of Euclid's algorithm for calculating the resultant. We denote by $lc(p)$ the leading coefficient of the polynomial $p(x)$,by $degree(p)$ its degree, and by $remainder(p, q)$ the remainder from the division of $p(x)$ by $q(x)$. We give the algorithm in a recursive form.

Algorithm 15 (resultant)
Input: f, g;
Output: $r = \mathrm{Res}(f, g)$.

$n := degree(f)$;
$m := degree(g)$;
if $n > m$ **then** $r := (-1)^{nm} resultant(g, f)$
else $a_n := lc(f)$;
if $n = 0$ **then** $r := a_n^m$
else $h := remainder(g, f)$;
if $h = 0$ **then** $r := 0$
else $p := degree(h)$;
$r := a_n^{m-p} resultant(f, h)$;
return r;

We write $h = \sum_{i=0}^{p} c_i x^i$ and $c_i = 0$ for $i > p$. This algorithm does indeed give the resultant of f and g for, when $n \leq m$ and $n \neq 0$, the polynomials $x^i g - x^i h$ (for $0 \leq i < n$) are linear combinations of the $x^j f$ (for $0 \leq j < m$), and therefore we are not changing the determinant of the Sylvester matrix of f and g by replacing b_i by c_i for $0 \leq i < m$. Now this new matrix has the form $\begin{pmatrix} A & * \\ 0 & B \end{pmatrix}$ where A is a triangular matrix with determinant a_n^{m-p} and B is the Sylvester matrix of f and h. From this algorithm we immediately get

Proposition 47 $\mathrm{Res}(f, g) = 0$ *if and only if* f *and* g *have a factor in common.*

It is now easy to prove the following propositions:

Proposition 48 *If the* α_i *are the roots of* f*, then*

$$\mathrm{Res}(f, g) = a_n^m \prod_{i=1}^{n} g(\alpha_i).$$

Proposition 49 *If the β_i are the roots of g, then*

$$\mathrm{Res}(f,g) = (-1)^{mn} b_m^n \prod_{i=1}^{m} f(\beta_i).$$

Proposition 50 $\mathrm{Res}(f,g) = a_n^m b_m^n \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j)$.

Proof [Duv87]. We write the right hand sides of the three propositions as

$$\begin{aligned} R_2(f,g) &= a_n^m \prod_{i=1}^{n} g(\alpha_i), \\ R_3(f,g) &= (-1)^{mn} b_m^n \prod_{i=1}^{m} f(\beta_i), \\ R_4(f,g) &= a_n^m b_m^n \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j). \end{aligned}$$

It is clear that $R_2(f,g)$ and $R_3(f,g)$ are equal to $R_4(f,g)$. The three propositions are proved simultaneously, by induction on the integer $\min(n,m)$. If f and g are swapped, their resultant is multiplied by $(-1)^{nm}$, and gives $R_4(f,g) = (-1)^{nm} R_4(g,f)$, while $R_2(f,g) = (-1)^{nm} R_3(g,f)$. We can therefore suppose that $n \leq m$. Moreover $R_2(f,g)$ is equal to a_n^m when $n = 0$, as is the resultant of f and g, and $R_4(f,g)$ is zero when $n > 0$ and $h = 0$, as is the resultant. It only remains to consider the case when $m \geq n > 0$ and $h \neq 0$. But then $R_2(f,g) = a_n^{m-p} R_2(f,h)$ for $g(\alpha_i) = h(\alpha_i)$ for each root α_i of f, and the algorithm shows that $\mathrm{Res}(f,g) = a_n^{m-p} \mathrm{Res}(f,h)$, from which we get the desired result.

Corollary 13 $\mathrm{Res}(fg,h) = \mathrm{Res}(f,h)\,\mathrm{Res}(g,h)$.

Definition 67 *The* discriminant *of f,* $\mathrm{Disc}(f)$ *or* $\mathrm{Disc}_x(f)$*, is*

$$a_n^{2n-2} \prod_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} (\alpha_i - \alpha_j).$$

Proposition 51 $\mathrm{Disc}(f) = 0$ *if, and only if, f has a repeated root, i.e. is not square-free.*

Proposition 52 $\mathrm{Res}(f,f') = a_n \mathrm{Disc}(f)$. *Moreover* $\mathrm{Disc}(f) \in R$.

Corollary 14 $\mathrm{Disc}(fg) = \mathrm{Disc}(f)\,\mathrm{Disc}(g)\,\mathrm{Res}(f,g)^2$.

Whichever way they are calculated, the resultants are often quite large. For example, if the a_i and b_i are integers, bounded by A and B respectively, the resultant is less than $(n+1)^{m/2}(m+1)^{n/2} A^m B^n$, but it is very often of this

order of magnitude (see section A.2). Similarly, if the a_i and b_i are polynomials of degree α and β respectively, the degree of the resultant is bounded by $m\alpha+n\beta$. A case in which this swell often matters is the use of resultants to calculate primitive elements, which uses the following result.

Proposition 53 *If α is a root of $p(x) = 0$, and β is a root of $q(x, \alpha) = 0$, then β is a root of* $\mathrm{Res}_y(y - p(x), q(x, y))$.

A.2 Useful Estimates

Estimates of the sizes of various things crop up throughout computer algebra. They can be essentially of three kinds.

- **Upper bounds:** $X \leq B$.
- **Lower bounds:** $X \geq B$.
- **Average sizes:** X "is typically" B. This may be a definitive average result (but even here we have to be careful what we are averaging over: the average number of factors of a polynomial is one, for example, but this does not mean we can ignore factorisation), or some heuristic "typically we find".

Estimates are used throughout complexity theory, of course. "Worst case" complexity is driven by upper bound results, while "average case" complexity is driven by average results. They are also used in algorithms: most algorithms of the 'Chinese Remainder' variety (section 4.1) rely on upper bounds to tell when they have used enough primes/evaluation points. In this case, a better upper bound generally translates into a faster algorithm *in practice*.

A.2.1 Matrices

How big is the determinant $|M|$ of an $n \times n$ matrix M?

Notation 13 *If $\mathbf{v}$ is a vector, then $|\mathbf{v}|$ denotes the* Euclidean norm *of $\mathbf{v}$, $\sqrt{\sum |v_i^2|}$. If f is a polynomial, $|f|$ denotes the Euclidean norm of its vector of ceofficients.*

Proposition 54 *If M is an $n \times n$ matrix with entries $\leq B$, $|M| \leq n!B^n$.*

This bound is frequently used in analysis, but we can do better.

Proposition 55 (Hadamard bound H_r) *If M is an $n \times n$ matrix whose rows are the vectors $\mathbf{v}_i$, then $|M| \leq H_r = \prod |\mathbf{v}_i|$, which in turn is $\leq n^{n/2}B^n$.*

Corollary 15 *If f and g are polynomials of degrees n and m respectively, then* $\mathrm{Res}(f, g) \leq |f|^m |g|^n$.

Corollary 16 *If f is a polynomial of degree n, then* $\mathrm{Disc}(f) \leq n^{n-1}|f|^{2n-1}$.

In practice, especially if f is not monic, it is worth taking rather more care over this estimation.

Proposition 56 (Hadamard bound (columns)) *If M is an $n \times n$ matrix whose columns are the vectors $\mathbf{v}_i$, then $|M| \leq H_c = \prod |\mathbf{v}_i| \leq n^{n/2} B^n$.*

In practice, for general matrices one computes $\min(H_r, H_c)$. While there are clearly bad cases (e.g. matrices of determinant 0), the Hadamard bounds are "not too bad". As pointed out in [AM01], $\log(\min(H_r, H_c)/|M|)$ is a measure of the "wasted effort" in a modular algorithm for computing the determinant, and "on average" this is $O(n)$, with a variance of $O(\log n)$. It is worth noting that this is independent of the size of the entries.

A.2.2 Coefficients of a polynomial

Here we are working implicitly with polynomials with complex coefficients, though the bounds will be most useful in the case of integer coefficients.

Notation 14 *Let*

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_n \prod_{i=1}^{n} (x - \alpha_i)$$

be a polynomial of degree n (i.e. $a_n \neq 0$). Define the following measures of the size of the polynomial f:

$H(f)$ (often written $||f||_\infty$), the height or 1-norm, is $\max_{i=0}^{n} |a_i|$;

$||f||$ (often written $||f||_2$), the 2-norm, is $\sqrt{\sum_{i=0}^{n} |a_i|^2}$;

$L(f)$ (often written $||f||_1$), the length or 1-norm, is $\sum_{i=0}^{n} |a_i|$;

$M(f)$, the Mahler measure of f, is $|a_n| \prod_{|\alpha_i|>1} |\alpha_i|$.

Proposition 57

$$H(f) \leq ||f|| \leq L(f) \leq (n+1) H(f),$$

where the first inequality is strict unless f is a monomial, the second is strict unless all the a_i are equal or zero, and the third is strict unless all the a_i are equal.

We should note that the last inequality could be replaced by $\leq cH(f)$, where c is the number of nonzero monomials in f, but this seems not to be much exploited.

Proposition 58 (Landau's Inequality [Lan05], [Mig89, §4.3]) .

$$M(f) \leq ||f||$$

and the inequality is strict unless f is a monomial.

Corollary 17 $|a_{n-i}| \leq \binom{n}{i} M(f) \leq \binom{n}{i} M(f)$.

The first part of this follows from the fact that a_{n-i} is $\pm a_n$ times a sum of $\binom{n}{i}$ products of roots, and products of roots are bounded by Proposition 58. For some applications, e.g. Theorem 20, we often bound $\binom{n}{i}$ by 2^n, but for others, such as Proposition 39, the more precise value is needed. 2^n might seem like overkill, but in fact, both in general and in the application to factoring [Mig81], 2 cannot be replaced by any smaller number.

It should be noted that the Landau–Mignotte bound is not the only way to bound the coefficients of a polynomial: [Abb09] give four methods that depend on knowing the degree of the factor being searched for, and two others that will bound the *least* height of a factor. Depressingly, he gives a family of exampls that shows that no bound is superior to any other, and indeed [Abb09, 3.3.5] it may be necessary to "mix-and-match" using different bounds for different coefficients.

A.2.3 Roots of a polynomial

Several questions can be asked about the roots of a univariate polynomial. The most obvious ones are how large/small can they be, but one is often interested in how close they can be. These questions are often asked of the *real* roots (section 3.4.4), but we actually need to study all roots, real and complex. The distinction between real and complex roots is pretty fine, as shown in the following example.

Example 6 (Wilkinson Polynomial [Wil59]) *Let W_{20} have roots at -1 , -2 , ..., -20, so that $W_{20} = (x+1)(x+2)\dots(x+20) = x^{20}+210x^{19}+\dots+20!$. Consider now the polynomial $W_{20}(x) + 2^{-23}x^{19}$. One might expect this to have twenty real roots close to the original ones, but in fact it has ten real roots, at approximately -1, -2, ... -7, -8.007, -8.917 and -20.847, and five pairs of complex conjugate roots, $-10.095 \pm 0.6435i$, $-11.794 \pm 1.652i$, $-13.992 \pm 2.519i$, $-16.731 \pm 2.813i$ and $-19.502 \pm 1.940i$.*

The discriminant of W_{20} is 2.74×10^{275}, which would seem well away from zero. However, the largest coefficient of W_{20} is 1.38×10^{19}, and of W'_{20} is -3.86×10^{19}. The norms of W_{20} and W'_{20} are 2.27×10^{19} and 6.11×10^{19}, so corollary 16 gives a bound of 4.43×10^{779} for $\mathrm{Disc}(W_{20})$, and a direct application of corollary 15 gives 3.31×10^{763}. Hence the discriminant of W_{20} is "much smaller than it ought to be", and W_{20} is "nearly not square-free". Put another way, the Sylvester matrix for the discriminant is very illconditioned (this was in fact Wilkinson's original motivation for constructing W_{20}): the discrepancy between the actual determinant and corollary 15 is 489 decimal digits, whereas [AM01] would lead us to expect about 17.

Notation 15 *Let $f \in \mathbf{C}[x] = \sum_{i=0}^{n} a_i x^i$, and let the roots of f be α_1, ..., α_n, and define*

$$\mathrm{rb}(f) = \max_{1 \leq i \leq n} |\alpha_i|,$$

$$\operatorname{sep}(f) = \min_{1 \le i < j \le n} |\alpha_i - \alpha_j|.$$

$\operatorname{sep}(f)$ *is zero if, and only if,* f *has a repeated factor.*

Proposition 59 [Cau29, p. 122] $\operatorname{rb}(f) \le 1 + \max(|a_i|)/|a_n|$.

Corollary 18 *If the polynomial* f *does not take the value* 0 *at* $x = 0$*, then every root of* f *has absolute value at least* $|a_0|/(|a_0| + \max(|a_i|))$.

Proposition 60 [Cau29, p. 123]

$$\operatorname{rb}(f) \le \max\left(\frac{n|a_{n-1}|}{a_n}, \sqrt{\frac{n|a_{n-2}|}{a_n}}, \ldots, \sqrt[n-1]{\frac{n|a_1|}{a_n}}, \sqrt[n]{\frac{n|a_0|}{a_n}}\right).$$

Proposition 61 [Knu98, 4.6.2 exercise 20]

$$\operatorname{rb}(f) \le B = 2\max\left(\frac{|a_{n-1}|}{a_n}, \sqrt{\frac{|a_{n-2}|}{a_n}}, \ldots, \sqrt[n-1]{\frac{|a_1|}{a_n}}, \sqrt[n]{\frac{|a_0|}{a_n}}\right).$$

Furthermore, there is at least one root of absolute value $\ge B/(2n)$.

Applied to W_{20}, these propositions give respectively 1.38×10^{19}, 4200 and 420. If we centre the roots, to be $-9.5, \ldots, 9.5$, the three propositions give respectively 2.17×10^{12}, 400 and 40 (and hence the roots of W_{20} are bounded by 2.17×10^{12}, 409.5 and 49.5). While the advantages of centrality are most prominent in the case of proposition 59, they are present for all of them. There are in fact improved bounds available in this ($a_{n-1} = 0$) case [Mig00].

If instead we re-balance W_{20} so that the leading and trailing coefficients are both 1, by replacing x by $\sqrt[20]{20!}x$, then the bounds for this new polynomial are 6961419, 505.76 and 50.58 (and hence the roots of W_{20} are bounded by 838284.7, 4200 and 420).

Proposition 62 ([Gra37]) *If* $p(x) = p_e(x^2) + xp_o(x^2)$*, i.e.* $p_e(x^2)$ *is the even terms of* $p(x)$*, then the roots of* $q(x) = p_e^2(x) - xp_o^2(x)$ *are the squares of the roots of* p.[1]

Applying this process to W_{20}, then computing the three bounds, and square-rooting the results, gives us bounds of 3.07×10^{18}, 239.58 and 75.76. Repeating the process gives 2.48×10^{18}, 61.66 and 34.67. On the centred polynomial, we get 2.73×10^{12}, 117.05 and 37.01, and a second application gives 1.83×10^{18}, 30.57 and 17.19 (and hence root bounds for W_{20} as 1.83×10^{18}, 40.07 and 26.69). This last figure is reasonably close to the true value of 20 [DM90].

[1] For the history of the attribution to Graeffe, see [Hou59].

A.2.4 Root separation

The key result is the following.

Proposition 63 ([Mah64]) $\operatorname{sep}(f) > \sqrt{3|\operatorname{Disc}(f)|}n^{-(n+2)/2}|f|^{1-n}$.

We note that the bound is zero if, and only if, the discriminant is zero, as it should be, and this bound is unchanged if we multiply the polynomial by a constant. The bound for W_{20} is 7.27×10^{-245}, but for the centred variant it becomes 1.38×10^{-113}. Should we ever need a root separation bound in practice, centring the polynomial first is almost always a good idea. Similarly re-balancing changes the separation bound to 5.42×10^{-112}, which means 6.52×10^{-113} for the original polynomial.

The monic polynomials of degree 7 with maximum root separation and all roots in the unit circle are:

six complex $x^7 - 1$, with discriminant -823543, norm $\sqrt{2}$ and root bounds

$$[2.0, 1.383087554, 2.0]$$

and root separation bound (proposition 63) $\frac{\sqrt{3}}{56} \approx 3.09 \times 10^{-2}$ (true value 0.868);

four complex $x^7 - x$, with discriminant $6^6 = 46656$, norm $\sqrt{2}$ and root bounds

$$[2.0, 1.383087554, 2.0]$$

and root separation bound (proposition 63) $\frac{27\sqrt{21}}{18607} \approx 7.361786385 \times 10^{-3}$ (true value 1);

two complex $x^7 - \frac{1}{4}x^5 - x^3 + \frac{1}{4}x$, with discriminant $\frac{-50625}{4096} \approx -12.36$, norm $\frac{1}{4}\sqrt{34} \approx 1.457737974$ and root bounds

$$[2, 1.626576562, 2.0]$$

and root separation bound (proposition 63) $\frac{1880\sqrt{21}}{82572791} \approx 9.99 \times 10^{-5}$ (true value 1/2);

all real $x^7 - \frac{14}{9}x^5 + \frac{49}{81}x^3 - \frac{4}{81}x$, with discriminant[2]

$$\frac{10485760000}{1853020188851841} \approx 5.66 \times 10^{-6}, \tag{A.1}$$

norm $\frac{17}{81}\sqrt{86} \approx 1.946314993$ and root bounds

$$[\frac{23}{9}, 3.299831645, 2.494438258]$$

and root separation bound (proposition 63) $\frac{83980800}{32254409474403781}\sqrt{3}\sqrt{7} \approx 1.19 \times 10^{-8}$ (true value 1/3).

[2] We note how the constraint that all the roots be real forces the discriminant to be small.

If there are $n+1$ equally-spaced real roots (call the spacing 1 for the time being), then the first root contributes $n!$ to $\prod_i \prod_{j \neq i} (\alpha_i - \alpha_j)$, the second root $1!(n-1)!$ and so on, so we have a total product of

$$\prod_{i=0}^{n} i!(n-i)! = \prod_{i=0}^{n} i!^2 = \left(\prod_{i=0}^{n} i! \right)^2 . \tag{A.2}$$

This is sequence `A055209` [Slo07], which is the square of `A000178`.

If we now assume that the roots are equally-spaced in $[-1, 1]$, then the spacing is $2/n$, we need to correct equation (A.2) by dividing by $(n/2)^{n(n-1)}$: call the result C_n. C is initially greater than one, with $C_3 = \frac{65536}{59049} \approx 1.11$, but $C_4 = \frac{81}{1024}$, $C_5 = \frac{51298814505517056}{37252902984619140625} \approx 0.001377042066$, and C_6 as in (A.1).

While assuming equal spacing might seem natural, it does not, in fact, lead to the largest values of the discriminant. Consider polynomials with all real roots $\in [-1, 1]$, so that we may assume the extreme roots are at ± 1.

degree 4 Equally spaced roots, at $\pm\frac{1}{3}$, give a discriminant of $\frac{65536}{59049} \approx 1.11$, whereas $\pm\frac{1}{\sqrt{5}}$ gives $\frac{4096}{3125} \approx 1.31$, the optimum. The norms are respectively $\frac{\sqrt{182}}{9} \approx 1.4999$ and $\frac{\sqrt{62}}{5} \approx 1.575$.

degree 5 Equally spaced roots, at $\pm\frac{1}{2}$ and 0, give a discriminant of $\frac{81}{1024} \approx 0.079$, whereas $\pm\sqrt{\frac{3}{7}}$ and 0 gives $\frac{12288}{16807} \approx 0.73$, the optimum. The norms are respectively $\frac{\sqrt{42}}{4} \approx 1.62$ and $\frac{\sqrt{158}}{7} \approx 1.796$.

degree 6 Equally spaced roots, at $\pm\frac{3}{5}$ and $\pm\frac{1}{5}$, give a discriminant of $\frac{51298814505517056}{37252902984619140625} \approx 0.00138$. Unconstrained solving for the maximum of the discriminant, using Maple's `Groebner,Solve`, starts becoming expensive, but if we assume symmetry, we are led to choose roots at $\frac{\pm\sqrt{147 \pm 42\sqrt{7}}}{21}$, with a discriminant of $\frac{67108864}{16209796869} \approx 0.0041$. The norms are respectively $\frac{2\sqrt{305853}}{625} \approx 1.77$ and $\frac{2\sqrt{473}}{21} \approx 2.07$.

degree 7 Equally spaced roots, at $\pm\frac{2}{3}$, $\pm\frac{1}{3}$ and 0, give a discriminant of $\frac{209715200000}{5615789612636313} \approx 5.66 \cdot 10^{-6}$. Again assuming symmetry, we are led to choose roots at $\frac{\pm\sqrt{495 \pm 66\sqrt{15}}}{33}$ and 0, which gives $\frac{209715200000}{5615789612636313} \approx 3.73 \cdot 10^{-5}$. The norms are respectively $\frac{17\sqrt{86}}{81} \approx 1.95$ and $\frac{2\sqrt{1577}}{33} \approx 2.41$,

degree 8 Equally spaced roots, at $\pm\frac{5}{7}$, $\pm\frac{3}{7}$ and $\pm\frac{1}{7}$, give a discriminant of $\approx 5.37 \cdot 10^{-9}$. Assuming symmetry, we get roots at $\pm \approx 0.87$, $\pm \approx 0.59$ and $\pm \approx 0.21$, with a discriminant of $\approx 9.65 \cdot 10^{-8}$. The norms are respectively ≈ 2.15 and $\frac{\sqrt{2}\sqrt{727171}}{429} \approx 2.81$.

degree 9 Equally spaced roots, at $\pm\frac{3}{4}$, $\pm\frac{1}{2}$, $\pm\frac{1}{4}$ and 0, give a discriminant of $1.15 \cdot 10^{-12}$. Assuming symmetry, we get roots at ± 0.8998, ± 0.677, ± 0.363 and zero, with a discriminant of $\approx 7.03 \cdot 10^{-11}$. The norms are respectively $\frac{\sqrt{5969546}}{1024} \approx 2.39$ and ≈ 3.296.

If we now consider he case with two complex root, which may as well be at $x = \pm i$, we have the following behaviour.

degree 4 The maximal polynomial is $x^4 - 1$, with discriminant -256 and norm $\sqrt{2}$. The bound is $\frac{\sqrt{6}}{216} \approx 0.153$.

degree 5 The maximal polynomial is $x^5 - x$, with discriminant -256 and norm $\sqrt{2}$. The bound is $\frac{4\sqrt{15}}{625} \approx 0.0248$.

degree 6 Equally spaced roots, at $\frac{\pm 1}{3}$, gives a discriminant of -108.26 and a norm of $\frac{2\sqrt{41}}{9}$. The bound is $\frac{400\sqrt{132}}{1860867} \approx 2.38 \cdot 10^{-3}$. The maximal discriminant is attained with roots at $\frac{\pm 1}{\sqrt{3}}$, with discriminant $\frac{-4194304}{19683} \approx -213.09$ and norm $\frac{2\sqrt{5}}{3}$. The bound is $\frac{4\sqrt{5}}{3375} \approx 2.65 \cdot 10^{-3}$.

degree 7 Equally spaced roots, at $\frac{\pm 1}{2}$ and 0, gives a discriminant of ≈ -12.36 and norm of $\frac{\sqrt{34}}{4} \approx 1.46$. The bound is $\frac{1800\sqrt{21}}{82572791} \approx 9.99 \cdot 10^{-5}$. The maximal discriminant is attained with roots at $\sqrt[4]{\frac{3}{11}}$, with discriminant ≈ 40.8 and norm of $\frac{2\sqrt{77}}{11} \approx 1.596$. The bound is $\approx 1.06 \cdot 10^{-4}$.

A.3 Chinese Remainder Theorem

In this section we review the result of the title, which is key to the methods in section 4.1.1.4, and hence to much of computer algebra.

Theorem 22 (Chinese Remainder Theorem (coprime form)) *Two congruences*

$$X \equiv a \pmod{M} \tag{A.3}$$

and

$$X \equiv b \pmod{N} \tag{A.4}$$

, where M and N are relatively prime, are precisely equivalent to one congruence

$$X \equiv c \pmod{MN}. \tag{A.5}$$

By this we mean that, given any a, b, M and N, we can find such a c that satisfaction of (A.3) and (A.4) is precisely equivalent to satisfying (A.5). The converse direction, finding (A.3) and (A.4) given (A.5), is trivial: one takes a to be $c \pmod{M}$ and b to be $d \pmod{N}$.

Algorithm 16 (Chinese Remainder)
Input: *a, b, M and N (with $\gcd(M, N) = 1$).*
Output: *c satisfying Theorem 22.*

Compute λ, μ such that $\lambda M + \mu N = 1$;
#The process is analogous to Lemma 1 (page 34)
c:=$a + \lambda M(b - a)$;

Clearly $c \equiv a \pmod{M}$, so satisfying (A.5) means that (A.3) is satisfied. What about (A.4)?

$$\begin{aligned} c &= a + \lambda M(b-a) \\ &= a + (1-\mu N)(b-a) \\ &\equiv a + (b-a) \pmod{N} \end{aligned}$$

so, despite the apparent asymmetry of the construction, $c \equiv b \pmod{N}$ as well.

In fact,we need not restrict ourselves to X being an integer: X may as well be polynomials (but M and N are still integers).

Algorithm 17 (Chinese Remainder (Polynomials))
Input: *Polynomials* $a = \sum_{i=0}^{n} a_i x^i$, $b = \sum_{i=0}^{n} b_i x^i$, *and integers* M *and* N *(with* $\gcd(M,N) = 1$*).*
Output: *A polynomial* $= \sum_{i=0}^{n} c_i x^i$ *satisfying Theorem 22.*

Compute λ, μ such that $\lambda M + \mu N = 1$;
#The process is analogous to Lemma 1 (page 34)
for $i := 0$ **to** n **do**
c_i:=$a_i + \lambda M(b_i - a_i)$;

Appendix B

Excursus

This appendix includes various topics on computer algebra that do not seem to be well-treated in the literature.

B.1 The Budan–Fourier Theorem

For the sake of simplicity, we will consider only square-free polynomails in this section: the results generalise fairly easily to non square-free ones.

Definition 68 *Given a sequence A of non-zero numbers $a_0, \ldots, a_n$, the* number of sign variations *of A, witten $V(A)$, is the number of times two consecutive elements have different signs, i.e. $a_i a_{i+1} < 0$. If A does contain zeros, we erase them before doing this computation, or equivalent we count the number of (i, j) with $a_i a_j < 0$ and all intermediate $a_{i+1}, \ldots, a_{j-1} = 0$. If f is the polynomial $a_n x^n + \cdots a_0$, we write $V(f)$ rather than $V(A)$ where A is the sequence of coefficients of f.*

Proposition 64 *$V(f) = V(f(0), f'(0), \ldots, f^{(n)}(0)$ for a polynomial f of degree n.*

The reader should note that the definition of variation is numerically unstable. $V(1) = 0$, and therefore (by the erasure rule) $V(1, 0) = 0$. For positive ϵ, $V(1, \epsilon) = 0$, but $V(1, -\epsilon) = 1$. This is related to the fact that $x + \epsilon$ has no positive real roots, but $x - \epsilon$ has one, as seen in the following result.

Theorem 23 (Descartes' rule of signs [CA76]) *(the number of roots of f in $(0, \infty)$ is less than or equal to, by an even number, $V(f)$.*

Corollary 19 *The number of roots of f in (a, ∞) is less than or equal to, by an even number, $V(f(x - a))$.*

Corollary 20 *The number of roots of f in (a, ∞) is less than or equal to, by an even number, $V((f(a), f'(a), \ldots, f^{(n)}(a))$.*

For dense f, there is not a great deal to choose between these two formulations, but, since the derivatives of a sparse polynomial are sparse but its translate is not, corollary 20 is greatly to be preferred to corollary 19 in the sparse case.

Let us fix f, and consider $V(y) := V(f(x-y))$ and $N(y) := |\{x > y : f(x) = 0\}|$ as functions of y. For large enough y, both are zero. As y decreases, $N(y)$ increases monotonically, by 1 at each root of f. In fact, the same monotonic behaviour is true of $V(y)$, increasing by 1 at roots of f and by 2 at certain other points. This allows us to compute the number of roots in an interval, a result known as the Budan–Fourier Theorem[1].

Corollary 21 (Budin–Fourier Theorem) *The number of roots of f in (a, b) is less than or equal to, by an even number, $V(f(x-a)) - V(f(x-b))$.*

Corollary 22 (Budin–Fourier Theorem [Hur12]) *The number of roots of f in (a, b) is less than or equal to, by an even number, $V((f(a), f'(a), \ldots, f^{(n)}(a)) - V((f(b), f'(b), \ldots, f^{(n)}(b))$.*

For the same reasons as above, corollary 22 is to be preferred in the case of sparse polynomials.

We can also deduce some results about the number of roots of sparse polynomials. If f has n non-zero terms, $V(f) \leq n-1$. We note that $V(ax^k) = 0$, and this polynomial indeed has no roots in $(0, \infty)$.

Corollary 23 *A square-free polynomial in x with n terms, not divisible by x, has at most $2(n-1)$ roots in $\mathbf{R}$. If it is divisible by x, then the answer is at most $2n-1$.*

The example of $x^3 - x$, which has three real roots $(\pm 1, 0)$ shows that the special case is necessary.

Open Problem 7 *Show that the same is true for non-square-free polynomials, with the addition that only distinct roots are counted. This would be trivial if we knew that the square-free part of a polynomial with n terms had at most n terms, but alas that is not the case [CD91].*

B.2 Equality of factored polynomials

This section treats the following problem.

Problem 8 *Given two polynomials in factored form (section 2.1.3), are they equal? More precisely, if*

$$f = \prod_{i=1}^{n} f_i^{a_i} \qquad g = \prod_{j=1}^{m} g_j^{b_j},$$

[1] See [BdB22, Fou31]. The question of precedence was hotly disputed at the time: see [Akr82] and `http://www-history.mcs.st-andrews.ac.uk/Biographies/Budan_de_Boislaurent.html`.

with the f_i and g_j square-free and relatively prime, i.e.:w $\gcd(f_i, f_{i'}) = 1$, $\gcd(g_j, g_{j'}) = 1$), *is* $f \stackrel{?}{=} g$.

The obvious solution is to expand f and g, and check that the expanded forms (which *are* canonical) are equal. Can we do better?

One important preliminary remark is that the square-free representation of a polynomial is unique. This leads to the following result.

Proposition 65 *If $f = g$, then every a_i has to be a b_j, and vice versa. For each such occurring value k, we must verify*

$$f_k = \prod_{\substack{i=1 \\ a_i=k}}^{n} f_i \stackrel{?}{=} g_k = \prod_{\substack{j=1 \\ b_j=k}}^{m} g_j. \tag{B.1}$$

In particular, f_k and g_k must have the same degree, i.e.

$$\sum_{\substack{i=1 \\ a_i=k}}^{n} \deg(f_i) = \sum_{\substack{j=1 \\ b_j=k}}^{m} \deg(g_j). \tag{B.2}$$

Again, we could check $f_k \stackrel{?}{=} g_k$ by expansion, but there is a better way.

Example 7 *Let $f = x^{2^l} - 1$ and $g = (x-1)(x+1)(x^2+1)\cdots(x^{2^{l-1}}+1)$, where both are square-free, so proposition 65 does not help. f is already expanded, but expansion of g can give rise to $x^{2^l-1} + x^{2^l-2} + \cdots + x + 1$, which has length $O(2^l)$, whereas g has length $O(l)$.*

From now on we will assume that we are working over a domain that includes the integers.

Lemma 9 *If f and g have degree at most N, and agree at $N+1$ different values, then $f = g$.*

Proof. $f - g$ is a polynomial of degree at most N, but has $N+1$ zeros, which contradicts proposition 5 if it is non-zero.

Hence it suffices to evaluate f and g at $N+1$ points x_i and check that the values agree. It is not necessary to construct any polynomial, and the integers involved are bounded (if we choose $|x_i| < N$) by BN^N, where B is a function of the coefficients of the f_i, g_j. Furthermore, we can evaluate at all these points in parallel. However, it does seem that we need to evaluate at $N+1$ points, or very nearly so, even if f and g are very small.

Open Problem 9 *Produce some non-trivial bounds on the maximum number of zeros of $f-g$, where f and g have small factored representations. See [RR90].*

The best we can say is as follows. Suppose, in the notation of (B.1), each f_i has k_i non-zero terms, and g_j has l_j non-zero terms, and no f_i or g_j is x (if either was, then trivially $f \neq g$, since a common factor of x would have been

detected). Then, by Corollary 23, $f - g$, if it is not identically zero, has at most $2\left(\sum_{\substack{i=1\\a_i=k}}^{n} k_i + \sum_{\substack{j=1\\b_j=k}}^{m} l_j - 1\right)$ roots in $\mathbf{R}$, and hence, if it is zero when evaluated at more than this number of integers, is identically zero. The factor of 2 can be dropped if we use only positive evaluation points, and rely on Theorem 23.

B.3 Karatsuba's method

This method was originally introduced in [KO63] for multiplying large integers: however, it is easier to explain in the (dense) polynomial context, where issues of carrying do not arise. Consider the product of two linear polynomials

$$(aX + b)(cX + d) = acX^2 + (ad + bc)X + bd. \tag{B.3}$$

This method so patently requires four multiplications of the coefficients that the question of its optimality was never posed. However, [KO63] rewrote it as follows:

$$(aX + b)(cX + d) = acX^2 + [(a + b)(c + d) - ac - bd]\,X + bd, \tag{B.4}$$

which only requires three *distinct* multiplications, ac and bd each being used twice. However, it requires four coefficients additions rather than one, so one might question the practical utility of it. For future reference, we will also express[2] equation (B.4) as

$$(aX + b)(cX + d) = ac(X^2 - X) + (a + b)(c + d)X + bd(1 - X), \tag{B.5}$$

which makes the three coefficient multiplications explicit.

However, it can be cascaded. Consider a product of two polynomials of degree three (four terms):

$$(a_3Y^3 + a_2Y^2 + a_1Y + a_0)(b_3Y^3 + b_2Y^2 + b_1Y + b_0). \tag{B.6}$$

If we write $X = Y^2$, $a = a_3Y + a_2$ etc., this product looks like the left-hand side of equation (B.4), and so can be computed with three multiplications of linear polynomials, each of which can be done in three multiplications of coefficients, thus making nine such multiplications in all, rather than the classic method's 16.

If the multiplicands have 2^k terms, then this method requires $3^k = \left(2^k\right)^{\log_2 3}$ multiplications rather than the classic $\left(2^k\right)^2$. For arbitrary numbers n of terms, not necessarily a power of two, the cost of "rounding up" to a power of two is subsumed in the O notation, and we see a cost of $O(n^{\log_2 3})$ rather than the classic $O(n^2)$ coefficient multiplications. We note that $\log_2 3 \approx 1.585$, and the number of extra coefficient additions required is also $O(n^{\log_2 3})$, being three extra

[2] This formulation is due to [Mon05].

additions at each step. While the "rounding up" is not important in O-theory, it matters in practice, and [Mon05] shows various other formulae, e.g.

$$(aX^2+bX+c)(dX^2+eX+f) = \\ cf(1-X)+be(-X+2X^2-X^3)+ad(-X^3+X^4)+(b+c)(e+f)(X-X^2) \\ +(a+b)(d+e)(X^3-X^2)+(a+b+c)(d+e+f)X^2,$$

requiring six coefficient multiplications rather than the obvious nine, or eight if we write it as

$$\left(aX^2+(bX+c)\right)\left(dX^2+(eX+f)\right) = \\ adX^4+aX^2(eX+f)+dX^2(bX+c)+(bX+c)(eX+f)$$

and use (B.4) on the last summand (asymptotics would predict $3^{\log_2 3} \approx 5.7$, so we are much nearer the asymptoic behaviour). Cascading this formula rather than (B.4) gives $O(n^{\log_3 6})$, which as $\log_3 6 \approx 1.63$ is not as favorable asymptotically. His most impressive formula describes the product of two six-term polynomials in 17 coefficient multiplications, and $\log_6 17 \approx 1.581$, a slight improvement. We refer the readerto the table in [Mon05], which shows that his armoury of formulae can get us very close to the asymptotic costs.

Theorem 24 *We can multiply two (dense) polynomials with m and n terms respectively in $O\left(\max(m,n)\min(m,n)^{(\log_2 3)-1}\right)$ coefficient operations.*

Let the polynomials be $f = a_{m-1}Y^{m-1}+\cdots$ and $f = b_{n-1}Y^{n-1}+\cdots$ Without loss of generality, we may assume $m \geq n$ (so we have to prove $O(mn^{\log_2 3-1})$, and write $k = \lceil \frac{m}{n} \rceil$. We can then divide f into blocks with n terms (possibly fewer for the most significant one) each: $f = \sum_{i=1}^{k-1} f_i Y^{in}$. Then

$$fg = \sum_{i=1}^{k-1} (f_i g)\, Y^{in}.$$

Each $f_i g$ can be computed in time $O(n^{\log_2 3})$, and the addition merely takes time $O(m-n)$ (since there is one addition to be performed for each power of Y where overlap occurs). Hence the total time is $O(kn^{\log_2 3})$, and the constant factor implied by the O-notation allows us to replace $k = \lceil \frac{m}{n} \rceil$ by $\frac{m}{n}$, which gives the desired result.

B.3.1 Karatsuba's method and sparse polynomials

The use of the Karatsuba formula and its variants for sparse polynomials is less obvious. One preliminary remark is in order: in the dense case we split the multiplicands in equation (B.6) or its equivalent in two (the same point for each), and we were multiplying components of half the size. This is no longer the case for sparse polynomials, e.g. every splitting point of

$$(a_7x^7+a_6x^6+a_5x^5+a_0)(b^7+b_2x^2+b_1x+b_0) \tag{B.7}$$

gives a 3–1 split of one or the other: indeed possibly both, as when we use x^4 as the splitting point.

Worse, in equation (B.5), the component multiplications were on 'smaller' polynomials, whereas, measured in number of terms, this is no longer the case. If we split equation (B.7) at x^4, the sub-problem corresponding to $(a+b)*(c+d)$ in equation (B.5) is

$$(a_7x^3 + a_6x^2 + a_5x^1 + a_0)(b^3 + b_2x^2 + b_1x + b_0)$$

which has as many terms as the original (B.7). This difficulty led [Fat03] to conclude that Karatsuba-based methods did not apply to sparse polynomials. It is clear that they cannot *always* apply, since the product of a polynomial with m terms by one with n terms may have mn terms, but it is probaby the difficulty of deciding *when* they apply that has led system designers to shun them.

B.3.2 Karatsuba's method and multivariate polynomials

Appendix C

Systems

This appendix discusses various computer algebra systems, especially from the point of view of their internal data structures and algorithms, and how this relates to the ideas expressed in the body of this book. We do *not* discuss the user interfaces as such, nor is this intended to replace any manuals, or specialist books.

C.1 Maple

C.1.1 History

This system started in the early 1980s, a period when computer power, and particularly memory, were much less than they are today. It was designed to support multiple users, particularly classes of students, on what were, by the standards of the time, relatively modest university resources. Two important early references are [CGGG83, CFG$^+$84]. These circumstances led to three design principles.

1. The system had to be small — early hosts systems had limitations of, for example, 110K words of memory. In particular, users must not pay, in terms of memory occupancy, for features they were not using, which led to a 'kernel plus loadable modules' design, where the kernel knew basic algebraic features, and the rest of the system was in loadable modules written in the Maple langauge itself, and therefore interpreted rather than compiled. The precise definition of 'basic' has changed over time — see section C.1.3.

2. The system had to be portable — early versions ran on both 32-bit VAX computers and 36-bit Honeywell computers. Its kernel, originally some 5500 lines of code, was macro-processed into 'languages of the BCPL family', of which the most successful, and the one used today, is C.

3. Memory was scarce, and hence re-use had to be a priority.

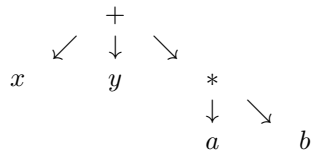
C.1.2 Data structures

These considerations led to an "expression DAG" design (see page 29). The internal form of an expression is a node of a certain type followed by an arbitrary number (hence we talk about an n-ary tree) of operands. If `A` is a Maple expression, the construct `op(0,A)` gives the type of the node, and `op(`i`,A)` gives the i-th operand. This is demonstrated by the session in table C.1, which builds the DAG shown in figure C.1. It might be assumed from table C.1 that Maple

Table C.1: A small Maple session

```
> A:=x+y+a*b;
                    A := x + y + a b
> op(0,A);
                             +
> op(3,A);
                           a b
> op(0,op(3,A));
                             *
> B:=y+b*a+x;
                    B := x + y + a b
```

Figure C.1: Tree for `A`, `B` corresponding to table C.1



had some 'preferred' order which `A` matched but `B` did not. However, if we look at table Maple:code2 (run in a fresh session of Maple), we see that `B` is now the preferred instance. The point is that, since Maple's internalising process[1] con-

Figure C.2: Tree for `A`, `B` corresponding to table C.2

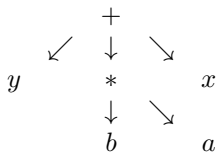


Table C.2: Another small Maple session

```
> B:=y+b*a+x;
                    B := y + b a + x
> op(0,B);
                             +
> op(2,B);
                           b a
> op(0,op(2,B));
                             *
> A:=x+y+a*b;
                    A := y + b a + x
```

structs these trees, in which `+` nodes cannot be children of `+` nodes, or `+` nodes of `*` nodes (hence implicitly enforcing associativity of these operators), and the children are unordered (hence implicitly enforcing commutativity[2]), once the subtree corresponding to `a*b` has been built, as in the first line of table C.1, an equivalent tree, such as that corresponding to `b*a`, is stored as the same tree. If it is fundamentall unordered, which way does it print? The answer is given in [CFG+84, p. 7]

> if the expression is found, the one in the table is used, and the [new] one is discarded.

Hence in table C.1, the presence of `a*b` means that `b*a` automatically becomes `a*b`. The converse behaviour occurs in table C.2.

In terms of the 10 algebraic rules on pp. 21–22, this structure automatically follows all except (8), which is implemented only in the weaker form (8').

The Maple command `expand` implements (8) fully, therefore producing, for polynomials, what we referred to (page 27) as a distributed representation[3]. This is therefore canonical (definition 3), but in a limited sense: it is canonical *within* a given Maple session, but may vary between sessions. This means that operations such as `op(`i`,A)` ($i \neq 0$) are not necessarily consistent between sessions.

[1] Referred to as the 'simplifier' in [CGGG83, CFG+84], but we do not use that word to avoid confusion with Maple's `simplify` command.

[2] We have found an example where this is not the case, but this is explicitly described as a bug by Maple's Senior Architect.

> Two simpl'd PROD DAGs containing the same entries but in a different order is a kernel bug by definition. [Vor10]

[3] But we should note that there are no guaranteed mathematical properties of the ordering. Good properties are provided by the `MonomialOrders` of the `Groebner` package.

C.1.3 Heuristic GCD

[CGG84, CGG89].

C.1.4 Conclusion

There are many books written on Maple, particularly in the educational context. A comprehensive list would be out-of-date before it was printed, but we should mention [Cor02].

Bibliography

[Abb02] J.A. Abbott. Sparse Squares of Polynomials. *Math. Comp.*, 71:407–413, 2002.

[Abb04] J.A. Abbott. CoCoA: a laboratory for computations in commutative algebra. *ACM SIGSAM Bulletin 1*, 38:18–19, 2004.

[Abb09] J.A. Abbott. Bounds on Factors in Z[x]. `http://arxiv.org/abs/0904.3057`, 2009.

[ABD85] J.A. Abbott, R.J. Bradford, and J.H. Davenport. A Remark on Factorisation. *SIGSAM Bulletin 2*, 19:31–33, 1985.

[ABD88] J.A. Abbott, R.J. Bradford, and J.H. Davenport. Factorisation of Polynomials: Old Ideas and Recent Results. In R. Janssen, editor, *Proceedings "Trends in Computer Algebra"*, pages 81–91, 1988.

[AHU74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. The Design and Analysis of Computer Algorithms. *Addison-Wesley*, 1974.

[AHU83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. Data Structures and Algorithms. *Addison-Wesley*, 1983.

[Akr82] A.G. Akritas. Reflections on a Pair of Theorems by Budan and Fourier. *Mathematics Magazine*, 55:292–298, 1982.

[AL94] W.W. Adams and P. Loustaunau. An introduction to Gröbner bases. *Amer. Math. Soc.*, 1994.

[ALMM99] P. Aubry, D. Lazard, and M. Moreno Maza. On the Theories of Triangular Sets. *J. Symbolic Comp.*, 28:105–124, 1999.

[ALSU07] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman. Compilers: Principles, Techniques and Tools. *Pearson Addison-Wesley*, 2007.

[AM01] J.A. Abbott and T. Mulders. How Tight is Hadamard's Bound? *Experimental Math.*, 10:331–336, 2001.

[AMM99] P. Aubry and M. Moreno Maza. Triangular Sets for Solving Polynomial Systems: A Comparison of Four Methods. *J. Symbolic Comp.*, 28:125–154, 1999.

[Ano07] Anonymous. MACSYMA .... `http://www.symbolicnet.org/systems/macsyma.html`, 2007.

[Arw18] A. Arwin. Über die Kongruenzen von dem fünften und höheren Graden nach einem Primzahlmodulus. *Arkiv før matematik*, 14:1–48, 1918.

[ASZ00] J.A. Abbott, V. Shoup, and P. Zimmermann. Factorization in $\mathbf{Z}[x]$: The Searching Phase. In C. Traverso, editor, *Proceedings ISSAC 2000*, pages 1–7, 2000.

[Bac94] P. Bachmann. Die analytische Zahlentheorie. *Teubner*, 1894.

[Bak75] A. Baker. Transcendental Number Theory. *Cambridge University Press*, 1975.

[Bar68] E.H. Bareiss. Sylvester's Identity and Multistep Integer-preserving Gaussian Elimination. *Math. Comp.*, 22:565–578, 1968.

[BC90] G. Butler and J. Cannon. The Design of Cayley — A Language for Modern Algebra. In *Proceedings DISCO '90*, 1990.

[BCM94] W. Bosma, J. Cannon, and G. Matthews. Programming with algebraic structures: design of the Magma language. In *Proceedings ISSAC 1994*, pages 52–57, 1994.

[BCR98] J. Bochnak, M. Coste, and M.-F. Roy. Real algebraic geometry. *Ergebnisse der Mathematik 38 (translated from the French and revised by the authors)*, 1998.

[BD89] R.J. Bradford and J.H. Davenport. Effective Tests for Cyclotomic Polynomials. In P. Gianni, editor, *Proceedings ISSAC 1988*, pages 244–251, 1989.

[BD02] R.J. Bradford and J.H. Davenport. Towards Better Simplification of Elementary Functions. In T. Mora, editor, *Proceedings ISSAC 2002*, pages 15–22, 2002.

[BD07] C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In C.W.Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.

[BdB22] F.F.D. Budan de BoisLaurent. Nouvelle méthode pour la résolution des équations numériques d'un degré quelconque. *Dondey-Dupré*, 1822.

[BF91] J. Backelin and R. Fröberg. How we proved that there are exactly 924 cyclic 7-roots. In S.M. Watt, editor, *Proceedings ISSAC 1991*, pages 103–111, 1991.

[BF93] M. Bartolozzi and R. Franci. La regola dei segni dall' enunciato di R. Descartes (1637) alla dimostrazione di C.F. Gauss (1828). *Archive for History of Exact Sciences 335-374*, 45, 1993.

[BL98] T. Breuer and S.A. Linton. The GAP4 Type System Organising Algebraic Algorithms. In O.Gloor, editor, *Proceedings ISSAC '98*, pages 38–45, 1998.

[BMMT94] E. Becker, M.G. Marinari, T. Mora, and C. Traverso. The shape of the shape lemma. In *Proceedings ISSAC 1994*, pages 129–133, 1994.

[Bou61] N. Bourbaki. Algèbre Commutative, chapter 2. *Hermann*, 1961.

[BPR06] S. Basu, R. Pollack, and M.-F. Roy. Algorithms in Real Algebraic Geometry, 2nd ed. *Springer*, 2006.

[Bro71] W.S. Brown. On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *J. ACM*, 18:478–504, 1971.

[Bro03] C.W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin 4*, 37:97–108, 2003.

[BS86] D. Bayer and M. Stillman. The Design of Macaulay: A System for Computing in Algebraic Geometry and Commutative Algebra. In *Proceedings SYMSAC 86*, pages 157–162, 1986.

[Buc70] B. Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystem. *Aequationes Mathematicae*, 4:374–383, 1970.

[Buc79] B. Buchberger. A Criterion for Detecting Unnecessary Reductions in the Construction of Groebner Bases. In *Proceedings EUROSAM 79*, pages 3–21, 1979.

[Buc84] B. Buchberger. A critical pair/completion algorithm for finitely generated ideals in rings. *Logic and Machines: Decision Problems and Complexity*, pages 137–155, 1984.

[BW93] T. Becker and V. (with H. Kredel) Weispfenning. Groebner Bases. A Computational Approach to Commutative Algebra. *Springer Verlag*, 1993.

[CA76] G.E. Collins and A.V. Akritas. Polynomial Real Root Isolation Using Descartes' Rule of Signs. In R.D. Jenks, editor, *Proceedings SYMSAC 76*, pages 272–275, 1976.

[Car04] J. Carette. Understanding Expression Simplification. In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 72–79, 2004.

[Cau29] A.-L. Cauchy. Exercices de Mathématiques Quatrième Année. *De Bure Frères*, 1829.

[CD85] D. Coppersmith and J.H. Davenport. An Application of Factoring. *J. Symbolic Comp.*, 1:241–243, 1985.

[CD91] D. Coppersmith and J.H. Davenport. Polynomials whose Powers are Sparse. *Acta Arithmetica*, 58:79–87, 1991.

[CDJW00] R.M. Corless, J.H. Davenport, D.J. Jeffrey, and S.M. Watt. According to Abramowitz and Stegun. *SIGSAM Bulletin 2*, 34:58–65, 2000.

[CFG+84] B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, M.B. Monagan, and S.M. Watt. On the Design and Performance of the Maple System. Technical Report CS-84-13, 1984.

[CGG84] B.W. Char, K.O. Geddes, and G.H. Gonnet. GCDHEU: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computation. In J.P. Fitch, editor, *Proceedings EUROSAM 84*, pages 285–296, 1984.

[CGG89] B.W. Char, K.O. Geddes, and G.H. Gonnet. GCDHEU: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computations. *J. Symbolic Comp.*, 7:31–48, 1989.

[CGGG83] B.W. Char, K.O. Geddes, M.W. Gentleman, and G.H. Gonnet. The Design of MAPLE: A Compact, Portable and Powerful Computer Algebra System. In *Proceedings EUROCAL 83 [Springer Lecture Notes in Computer Science 162*, pages 101–115, 1983.

[CGH88] L. Caniglia, A. Galligo, and J. Heintz. Some New Effectivity Bounds in Computational Geometry. In *Proceedings AAECC-6*, pages 131–152, 1988.

[CGH+03] D. Castro, M. Giusti, J. Heintz, G. Matera, and L.M. Pardo. The Hardness of Polynomial Equation Solving. *Foundations of Computational Mathematics*, 3:347–420, 2003.

[CH91] G.E. Collins and H. Hong. Partial Cylindrical Algebraic Decomposition for Quantifier Elimination. *J. Symbolic Comp.*, 12:299–328, 1991.

[CLO06] D.A. Cox, J.B. Little, and D.B. O'Shea. Ideals, Varieties and Algorithms. *Springer–Verlag*, 2006.

[Col71] G.E. Collins. The SAC-1 System: An Introduction and Survey. In *Proceedings SYMSAC 1971*, 1971.

[Col75] G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

[Col79] G.E. Collins. Factoring univariate integral polynomials in polynomial average time. In *Proceedings EUROSAM 79*, pages 317–329, 1979.

[Col85] G.E. Collins. The SAC-2 Computer Algebra System. In *Proceedings EUROCAL 85*, pages 34–35, 1985.

[Cor02] R.M. Corless. Essential Maple 7 : an introduction for scientific programmers. *Springer-Verlag*, 2002.

[CR88] M. Coste and M.F. Roy. Thom's Lemma, the Coding of Real Algebraic Numbers and the Computation of the Topology of Semi-Algebraic Sets. *J. Symbolic Comp.*, 5:121–129, 1988.

[CW90] D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. *J. Symbolic Comp.*, 9:251–280, 1990.

[CZ81] D.G. Cantor and H. Zassenhaus. A New Algorithm for Factoring Polynomials over Finite Fields. *Math. Comp.*, 36:587–592, 1981.

[Dav97] J.H. Davenport. Galois Groups and the Simplification of Polynomials. *Programming and Computer Software 1*, 23:31–44, 1997.

[Dav10] J.H. Davenport. The Challenges of Multivalued "Functions". In S. Autexier *et al.*, editor, *Proceedings AISC/Calculemus/MKM 2010*, pages 1–12, 2010.

[DGT91] J.H. Davenport, P. Gianni, and B.M. Trager. Scratchpad's View of Algebra II: A Categorical View of Factorization. In S.M. Watt, editor, *Proceedings ISSAC 1991*, pages 32–38, 1991.

[DH88] J.H. Davenport and J. Heintz. Real Quantifier Elimination is Doubly Exponential. *J. Symbolic Comp.*, 5:29–35, 1988.

[Dic13] L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n prime factors. *Amer. J. Math.*, 35:413–422, 1913.

[DL08] J.H. Davenport and P. Libbrecht. The Freedom to Extend OpenMath and its Utility. *Mathematics in Computer Science 2(2008/9)*, pages 379–398, 2008.

[DM90] J.H. Davenport and M. Mignotte. On Finding the Largest Root of a Polynomial. *Modélisation Mathématique et Analyse Numérique*, 24:693–696, 1990.

[DN07] P. D'Alberto and A. Nicolau. Adaptive Strassen's matrix multiplication. In *Proceedings Supercomputing 2007*, pages 284–292, 2007.

[Dod66] C.L. Dodgson. Condensation of determinants, being a new and brief method for computing their algebraic value. *Proc. Roy. Soc. Ser. A*, 15:150–155, 1866.

[DS97] A. Dolzmann and Th. Sturm. Redlog: Computer Algebra Meets Computer Logic. *ACM SIGSAM Bull. 2*, 31:2–9, 1997.

[DS00] J.H. Davenport and G.C. Smith. Fast recognition of alternating and symmetric groups. *J. Pure Appl. Algebra*, 153:17–25, 2000.

[DT81] J.H. Davenport and B.M. Trager. Factorization over finitely generated fields. In *Proceedings SYMSAC 81*, pages 200–205, 1981.

[DT90] J.H. Davenport and B.M. Trager. Scratchpad's View of Algebra I: Basic Commutative Algebra. In *Proceedings DISCO '90*, 1990.

[Duv87] D. Duval. Diverses Questions relatives au Calcul Formel avec les Nombres Algébriques. *Thèse d'Etat*, 1987.

[Fat03] R.J. Fateman. Comparing the speed of programs for sparse polynomial multiplication. *SIGSAM Bulletin 1*, 37:4–15, 2003.

[FGLM93] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *J. Symbolic Comp.*, 16:329–344, 1993.

[FGT01] E. Fortuna, P. Gianni, and B. Trager. Degree reduction under specialization. *J. Pure Appl. Algebra*, 164:153–163, 2001.

[fHN76] J.P. ffitch, P. Herbert, and A.C. Norman. Design Features of COBALG. In R.D. Jenks, editor, *Proceedings SYMSAC 76*, pages 185–188, 1976.

[FM89] D.J. Ford and J. McKay. Computation of Galois Groups from Polynomials over the Rationals. *Computer Algebra (Lecture Notes in Pure and Applied Mathematics 113*, 1989.

[Fou31] J. Fourier. Analyse des équations déterminées. *Didot*, 1831.

[FS56] A. Fröhlich and J.C. Shepherdson. Effective Procedures in Field Theory. *Phil. Trans. Roy. Soc. Ser. A 248(1955-6)*, pages 407–432, 1956.

[Ful69] W. Fulton. Algebraic Curves, An Introduction to Algebraic Geometry. *W.A. Benjamin Inc*, 1969.

[Gal79] E. Galois. Œuvres mathématiques. *Gauthier-Villars (sous l'auspices de la SMF)*, 1879.

[Gia89] P. Gianni. Properties of Gröbner bases under specializations. In *Proceedings EUROCAL 87*, pages 293–297, 1989.

[GJY75] J.H. Griesmer, R.D. Jenks, and D.Y.Y. Yun. SCRATCHPAD User's Manual. *IBM Research Publication RA70*, 1975.

[GN90] A. Giovanni and G. Niesi. CoCoA: A User-Friendly System for Commutative Algebra. In *Proceedings DISCO '90*, 1990.

[Gra37] C.H. Graeffe. Die Auflösulg der höheren numerischen Gleichungen. *F. Schulthess*, 1837.

[Ham07] S. Hammarling. Life as a developer of numerical software. *Talk at NAG Ltd AGM*, 2007.

[Has53] C.B. Haselgrove. Implementations of the Todd-Coxeter Algorithm on EDSAC-1. *Unpublished*, 1953.

[HD03] A.J. Holt and J.H. Davenport. Resolving Large Prime(s) Variants for Discrete Logarithm Computation. In P.G. Farrell, editor, *Proceedings 9th IMA Conf. Coding and Cryptography*, pages 207–222, 2003.

[Hea05] A.C. Hearn. REDUCE: The First Forty Years. In *Proceedings A3L*, pages 19–24, 2005.

[Her72] E. Hermite. Sur l'intégration des fractions rationelles. *Nouvelles Annales de Mathématiques*, 11:145–148, 1872.

[Hor69] E. Horowitz. *Algorithm for Symbolic Integration of Rational Functions*. PhD thesis, Univ. of Wisconsin, 1969.

[Hor71] E. Horowitz. Algorithms for Partial Fraction Decomposition and Rational Function Integration. In *Proceedings Second Symposium on Symbolic and Algebraic Manipulation*, pages 441–457, 1971.

[Hou59] A.S. Householder. Dandelin, Lobačevskiĭor Graeffe? *Amer. Math. Monthly*, 66:464–466, 1959.

[HP07] H. Hong and J. Perry. Are Buchberger's criteria necessary for the chain condition? *J. Symbolic Comp.*, 42:717–732, 2007.

[Hur12] A. Hurwitz. Über den Satz von Budan-Fourier. *Math. Annalen*, 71:584–591, 1912.

[IEE85] IEEE. IEEE Standard 754 for Binary Floating-Point Arithmetic. *IEEE*, 1985.

[IL80] O.H. Ibarra and B.S. Leininger. The Complexity of the Equivalence Problem for Straight-line Programs. In *Proceedings ACM STOC 1980*, pages 273–280, 1980.

[Isa85] I.M. Isaacs. Solution of polynomials by real radicals. *Amer. Math. Monthly*, 92:571–575, 1985.

[Joh71] S.C. Johnson. On the Problem of Recognizing Zero. *J. ACM*, 18:559–565, 1971.

[Joh74] S.C. Johnson. Sparse Polynomial Arithmetic. In *Proceedings EUROSAM 74*, pages 63–71, 1974.

[JR10] D.J. Jeffrey and A.D. Rich. Reducing Expression Size Using Rule-Based Integration. In S. Autexier *et al.*, editor, *Proceedings CICM 2010*, pages 234–246, 2010.

[JS92] R.D. Jenks and R.S. Sutor. AXIOM: The Scientific Computation System. *Springer-Verlag*, 1992.

[Kah53] H.G. Kahrimanian. Analytic differentiation by a digital computer. *M.A. Thesis*, 1953.

[Kal88] E. Kaltofen. Greatest Common Divisors of Polynomials given by Straight-line Programs. *J. ACM*, 35:231–264, 1988.

[Kal89] M. Kalkbrener. Solving systems of algebraic equations by using Gröbner bases. In *Proceedings EUROCAL 87*, pages 282–292, 1989.

[Knu74] D.E. Knuth. Big Omicron and big Omega and big Theta. *ACM SIGACT News 2*, 8:18–24, 1974.

[Knu98] D.E. Knuth. The Art of Computer Programming, Vol. II, Seminumerical Algorithms. *Third Edition*, 1998.

[KO63] A. Karatsuba and J. Ofman. Multiplication of multidigit numbers on automata. *Sov. Phys. Dokl.*, 7:595–596, 1963.

[KRW90] A. Kandri-Rody and V. Weispfenning. Non-commutative Gröbner bases in algebras of solvable type. *J. Symbolic Comp.*, 9:1–26, 1990.

[Lan05] E. Landau. Sur Quelques Théorèmes de M. Petrovic Relatif aux Zéros des Fonctions Analytiques. *Bull. Soc. Math. France*, 33:251–261, 1905.

[Laz83] D. Lazard. Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In *Proceedings EUROCAL 83 [Springer Lecture Notes in Computer Science 162*, pages 146–157, 1983.

[Laz88] D. Lazard. Quantifier Elimination: Optimal Solution for Two Classical Problems. *J. Symbolic Comp.*, 5:261–266, 1988.

[Laz91] D. Lazard. A New Method for Solving Algebraic Systems of Positive Dimension. *Discr. Appl. Math.*, 33:147–160, 1991.

[Laz92] D. Lazard. Solving Zero-dimensional Algebraic Systems. *J. Symbolic Comp.*, 13:117–131, 1992.

[Lec08] G. Lecerf. Fast separable factorization and applications. *AAECC*, 19:135–160, 2008.

[LLJL82] A.K. Lenstra, H.W. Lenstra, Jun., and L. Lovász. Factoring Polynomials with Rational Coefficients. *Math. Ann.*, 261:515–534, 1982.

[Loo82] R. Loos. Generalized Polynomial Remainder Sequences. *Symbolic and Algebraic Computation (Computing Supplementum 4) Springer-Verlag*, pages 115–137, 1982.

[LR01] T. Lickteig and M.-F. Roy. Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symbolic Comp.*, 31:315–341, 2001.

[Mah64] K. Mahler. An Inequality for the Discriminant of a Polynomial. *Michigan Math. J.*, 11:257–262, 1964.

[MF71] W.A. Martin and R.J. Fateman. The MACSYMA System. In *Proceedings Second Symposium on Symbolic and Algebraic Manipulation*, pages 59–75, 1971.

[MGH+03] M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron, and P. DeMarco. Maple 9 : introductory programming guide. *Maplesoft*, 2003.

[Mig74] M. Mignotte. An Inequality about Factors of Polynomials. *Math. Comp.*, 28:1153–1157, 1974.

[Mig81] M. Mignotte. Some Inequalities About Univariate Polynomials. In *Proceedings SYMSAC 81*, pages 195–199, 1981.

[Mig82] M. Mignotte. Some Useful Bounds. *Symbolic and Algebraic Computation (Computing Supplementum 4) Springer-Verlag*, pages 259–263, 1982.

[Mig89] M. Mignotte. Mathématiques pour le Calcul Formel. *PUF*, 1989.

[Mig00] M. Mignotte. Bounds for the roots of lacunary polynomials. *J. Symbolic Comp.*, 30:325–327, 2000.

[Mon05] P.L. Montgomery. Five, Six, and Seven-Term Karatsuba-Like Formulae. *IEEE Trans. Computers*, 54:362–369, 2005.

[Mon09] D. Monniaux. Fatal Degeneracy in the Semidefinite Programming Approach to the Decision of Polynomial Inequalities. `http://arxiv.org/abs/0901.4907`, 2009.

[Mor86] F. Mora. Groebner Bases for Non-commutative Polynomial Rings. In *Proceedings AAECC-3*, pages 353–362, 1986.

[Mos71] J. Moses. Algebraic Simplification — A Guide for the Perplexed. *Comm. ACM*, 14:527–537, 1971.

[Mus78] D.R. Musser. On the efficiency of a polynomial irreducibility test. *J. ACM*, 25:271–282, 1978.

[MW51] J.C.P. Miller and D.J. Wheeler. Missing Title. *Nature p. 838*, 168, 1951.

[Neu95] J. Neubüser. Re: Origins of GAP. *Message m0t5WVW-00075GC.951018.121802@astoria.math.rwth-aachen.de to GAP-Forum on 18.10.95*, 1995.

[Nol53] J. Nolan. Analytic differentiation on a digital computer. *M.A. Thesis*, 1953.

[oL43] Ada Augusta Countess of Lovelace. Sketch of the Analytical Engine invented by Charles Babbage, by L.F. Menabrea of Turin, with notes on the memoir by the translator. *Taylor's Scientific Memoirs 29*, 3:666–731, 1843.

[Ost45] M.W. Ostrogradski. De l'intégration des fractions rationelles. *Bull. Acad. Imp. Sci. St. Petersburg (Class Phys.-Math.) pp*, 4, 1845.

[Pan02] V.Y. Pan. Univariate Polynomials: Nearly Optimal Algorithms for Numerical Factorization and Root-finding. *J. Symbolic Comp.*, 33:701–733, 2002.

[Pau07] F. Pauer. Gröbner bases with coefficients in rings. *J. Symbolic Computation*, 42:1003–1011, 2007.

[Per09] J. Perry. An extension of Buchberger's criteria for Groebner basis decision. `http://arxiv.org/abs/0906.4358`, 2009.

[PW85] R. Pavelle and P.S. Wang. MACSYMA from F to G. *J. Symbolic Comp.*, 1:69–100, 1985.

[Ric97] D.S. Richardson. How to Recognize Zero. *J. Symbolic Comp.*, 24:627–645, 1997.

[Ris69] R.H. Risch. The Problem of Integration in Finite Terms. *Trans. A.M.S.*, 139:167–189, 1969.

[Ris85] J.-J. Risler. Additive Complexity of Real Polynomials. *SIAM J. Comp.*, 14:178–183, 1985.

[Ris88] J.-J. Risler. Some Aspects of Complexity in Real Algebraic Geometry. *J. Symbolic Comp.*, 5:109–119, 1988.

[Rit32] J.F. Ritt. Differential Equations from an Algebraic Standpoint. *Volume 14*, 1932.

[Rob85] L. Robbiano. Term orderings on the Polynomial Ring. In *Proceedings EUROCAL 85*, pages 513–525, 1985.

[Rot76] M. Rothstein. *Aspects of Symbolic Integration and Simplification of Exponential and Primitive Functions*. PhD thesis, Univ. of Wisconsin, 1976.

[RR90] J.-J. Risler and F. Ronga. Testing Polynomials. *J. Symbolic Comp.*, 10:1–5, 1990.

[RS79] A.D. Rich and D.R. Stoutemyer. Capabilities of the MUMATH-79 Computer Algebra System for the INTEL-8080 Microprocessor. In *Proceedings EUROSAM 79*, pages 241–248, 1979.

[RS92] A.D. Rich and D.R. Stoutemyer. DERIVE Reference Manual. *Soft-Warehouse*, 1992.

[Sch03] H. Schönemann. Singular in a Framework for Polynomial Computations. *Algebra Geometry and Software Systems*, pages 163–176, 2003.

[Sei54] A. Seidenberg. A new decision method for elementary algebra. *Ann. Math.*, 60:365–374, 1954.

[Sen08] J.R. Sendra. Algebraic Curves Soluble by Radicals. `http://arxiv.org/abs/0805.3214`, 2008.

[SGV94] A. Schönhage, A.F.W. Grotefeld, and E. Vetter. Fast Algorithms: A Multitape Turing Machine Implementation. *BI Wissenschaftsverlag*, 1994.

[Sla61] J. Slagle. *A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus*. PhD thesis, Harvard U., 1961.

[Slo07] N.J.A. Sloane. The Online Encyclopedia of Integer Sequences. `http://www.research.att.com/~njas/sequences`, 2007.

[SS06] A.P. Sexton and V. Sorge. Abstract matrices in symbolic computation. In *Proceedings ISSAC 2006*, pages 318–325, 2006.

[Sto77] D.R. Stoutemyer. sin(x)**2 + cos(x)**2 = 1. In *Proceedings 1977 MACSYMA Users' Conference*, pages 425–433, 1977.

[Str69] V. Strassen. Gaussian Elimination is not Optimal. *Numer. Math.*, 13:354–356, 1969.

[Tar51] A. Tarski. A Decision Method for Elementary Algebra and Geometry, 2nd ed. *Univ. Cal. Press*, 1951.

[TE07] E.P. Tsigaridas and I.Z. Emiris. Univariate polynomial real root isolation: Continued Fractions revisited. *Proc. 14th European Symp. Algorithms, Springer Lecture Notes in Computer Science*, 4168:817–828, 2007.

[Tra76] B.M. Trager. Algebraic Factoring and Rational Function Integration. In R.D. Jenks, editor, *Proceedings SYMSAC 76*, pages 219–226, 1976.

[Tri78] W. Trinks. Über B. Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen. *J. Number Theory*, 10:475–488, 1978.

[vH02] M. van Hoeij. Factoring polynomials and the knapsack problem. *J. Number Theory*, 95:167–189, 2002.

[Vor10] S. Vorkoetter. E-mail 4BF15DC8.7030207@maplesoft.com. *Mon*, 2010.

[vT83] E.W. von Tschirnhaus. Methodus auferendi omnes terminos intermedios ex data aeqvatione. *Acta Eruditorium*, ?:204–207, 1683.

[Wil59] J.H. Wilkinson. The Evaluation of the Zeros of Ill-conditioned Polynomials. *Num. Math.*, 1:150–166, 1959.

[Win71] S. Winograd. On multiplication of 2 x 2 matrices. *Linear algebra and its applications*, 4:381–388, 1971.

[Wu86] W.-T. Wu. On zeros of algebraic equations — an application of Ritt principle. *Kexue Tongbao*, 31:1–5, 1986.

[Yun76] D.Y.Y. Yun. On Square-free Decomposition Algorithms. In R.D. Jenks, editor, *Proceedings SYMSAC 76*, pages 26–35, 1976.

[Zar26] O. Zariski. Sull'impossibilità di risolvere parametricamente per radicali un'equazione algebrica $f(x, y) = 0$ di genere $p > 6$ a moduli generali. *Atti Accad. Naz. Lincei Rend. Cl. Sc. Fis. Mat. Natur. serie VI*, 3:660–666, 1926.

[Zas69] H. Zassenhaus. On Hensel Factorization I. *J. Number Theory*, 1:291–311, 1969.

[Zim07] P. Zimmerman. We recommend students never to use simplify inside programs. *Personal communication*, 2007.

[Zip93] R.E. Zippel. Effective Polynomial Computation. *Kluwer Academic Publishers*, 1993.

Index

Abel's theorem, 46
Additive complexity, 29
Admissible orderings, 27
Algebraic
  closure, 18
  curve, 48
  decomposition, 84
    block-cylindrical, 88
    cylindrical, 86
    sampled, 85
  proposition, 81
  variable, 74
Algorithm
  Bareiss, 55
  Buchberger, 60
  Chinese Remainder, 124
    Polynomials, 125
  Hermite's, 108
  Horowitz–Ostrogradski, 109
  Sturm sequence evaluation, 50
  Trager–Rothstein, 110
Alternations
  of quantifiers, 83
Ascending Chain Condition, 16
Associates, 17
Associativity, 16

Bad reduction, 94
Bareiss algorithm, 55
Basis
  Gröbner, 59
  shape, 73
Birational equivalence, 48
Block-cylindrical
  decomposition, 88
Bound, 118
  Cauchy, 121
  Hadamard, 118
  Knuth, 121
  Mahler, 122
Buchberger
  Algorithm, 60
  Criterion
    First, 62
    gcd, 62
    lcm, 62
    Third, 62
Budan–Fourier theorem, 50, 128

Calculus
  Fundamental Theorem of, 106, 113
Canonical representation, 12
Cauchy bound, 121
Cell, 84
Chain Condition
  Ascending, 16
  Descending, 27
Characteristic, 17
  set, 74
Chinese Remainder Theorem, 124
Closure
  Zariski, 76
Coefficient
  leading, 23
  of a polynomial, 21
Commutativity, 16
Complexity
  additive, 29
Constant
  definition of, 106
Content (of polynomials), 34
Cylinder, 86
Cylindrical algebraic decomposition, 86

Decomposition
  algebraic, 84
  block-cylindrical, 88
  cylindrical, 86
  equiprojectable, 75
  Lemma (exponential), 112
  Lemma (logarithmic), 112
  Lemma (rational functions), 107
  sign-invariant, 86
  square-free, 44
Defining formula, 84
Degree
  of polynomial, 23
Denominator, 31
  common, 31
Dense
  matrix, 51
  polynomial, 23
Descartes rule of signs, 50, 127
Descending Chain Condition, 27
Differential
  field, 105
  ring, 105
Dimension
  ideal, 63
  linear space, 56
  triangular set, 74
Directed Acyclic Graph, 29
Discriminant, 117
Distributed representation, 27
Distributivity, 16
Division, 22
Dodgson–Bareiss theorem, 55

elementary
  function, 111
  generator, 111
Equality
  algebraic, 32
Equiprojectable
  decomposition, 75
  variety, 74
Equivalent, 57
Euclid's
  Algorithm, 33
  Theorem, 33

Excel, 9
Exponentiation
  polynomial, 22
Expression
  DAG representation, 29
  tree representation, 28

Faugère–Gianni–Lazard–Mora
  algorithm, 71
Field, 17
  differential, 105
  of fractions, 17
  real closed, 81
Fill-in (loss of sparsity), 52
Formula
  defining, 84
Fundamental
  Theorem of Calculus, 106, 113

Gauss
  elimination, 53
  Lemma, 34
Generalised
  Polynomial, 112
Gianni–Kalkbrener
  algorithm, 70
  theorem, 69
Good reduction, 94
Graeffe method, 121
Greatest common divisor, 17, 32
  domain, 33
Gröbner base, 59, 80
  completely reduced, 60

Hadamard bound, 118
Hermite
  Algorithm, 108
History of Computer Algebra, 9
Horowitz–Ostrogradski
  Algorithm, 109

Ideal, 16
  polynomial, 57
  saturated, 77
Inequality
  Landau, 119

Landau–Mignotte, 92
Initial, 75
Integral domain, 16
Integtaion
indefinite, 106
Inverse
matrix, 52

Knuth
bound, 121

Landau Inequality, 119
Landau notation, 18
Landau–Mignotte Inequality, 92
Laurent
Polynomial, 112
Least common multiple, 33
Lioville's Principle, 111

Mahler bound, 122
Main variable, 75
Maple, 133
Matrix
inverse, 52
Sylvester, 115
Monic polynomial, 23
Monomial, 27
leading, 58
Multiplicity of a solution, 44

Noetherian ring, 16
Normal representation, 12
Numerator, 31

Ordering
admissible, 27
elimination, 66
matrix, 66
purely lexicographic, 65
total degree, then lexicographic, 66
total degree, then reverse lexicographic, 66
weighted, 66

Polynomial
definition, 21
factored representation, 25
generalised, 112
Laurent, 112
remainder sequence, 36
signed, 36
Wilkinson, 120
polynomial
height of, 119
length of, 119
Prenex normal form, 83
Primitive
(of polynomials), 34
p.r.s., 37
part, 34
Principal ideal domain, 17
Principle
Lioville's, 111
Projection, 89
Proposition
algebraic, 81
semi-algebraic, 81
Pseudo-euclidean algorithm, 36
Pseudo-remainder, 36, 76

Quantifier
alternation, 83
elimination, 83
Quantifier-free, 82
Quotient (polynomial), 33
Quotient rule, 106

Radical
(i.e. n-th root), 46
(of an ideal), 62
Real closed fields, 81
Recursive representation, 26
Reduction
bad, 94
good, 94
Reductum, 23
Remainder (polynomial), 33
Representation
distributed (of polynomials), 27
expression DAG, 29
expression tree, 28
factored (of polynomials), 25
of objects, 12

canonical, 12
normal, 12
recursive (of polynomials), 26
Straight-Line Program, 29
Resultant, 115–117
Ring, 16
differential, 105
RootOf, 47
Rule
quotient, 106

Saturated ideal, 77
Set
triangular, 74
Shape basis, 73
sign
variations, 127
Signed polynomial remainder sequence, 36
Sparse
matrix, 51
polynomial, 23
S-polynomial, 59, 79
Square-free, 41
decomposition, 41
Stability
numerical, 15
Straight-Line Program Representation, 29
Sturm
–Habicht sequence, 49
sequence, 49
Subresultant algorithm, 37
Sylvester matrix, 115
Systems
Maple, 133

Tarski–Seidenberg principle, 83
term
leading, 58
Theorem
Chinese Remainder, 124
Trager–Rothstein
Algorithm, 110
Triangular set, 74
Tschirnhaus transformation, 44

Unique factorisation domain, 33
Unit, 17

Variable
algebraic, 74
main, 75

Wilkinson Polynomial, 120

Zariski closure, 76
Zero-divisors, 17

The total-degree Gröbner base from page 72.

$\{42\,c^4+555+293\,a^3+1153\,a^2b-3054\,ab^2+323\,b^3+2035\,a^2c+1642\,abc-1211\,b^2c+2253\,ac^2-1252\,bc^2-31\,c^3+1347\,a^2-3495\,ab+1544\,b^2-1100\,ac+2574\,bc-368\,c^2+2849\,a+281\,b+4799\,c, 21\,bc^3-57-10\,a^3-41\,a^2b+183\,ab^2-25\,b^3-104\,a^2c-65\,abc+91\,b^2c-129\,ac^2+59\,bc^2-16\,c^3-33\,a^2+225\,ab-142\,b^2+67\,ac-174\,bc-5\,c^2-154\,a-46\,b-310\,c, 21\,ac^3-75-29\,a^3-121\,a^2b+369\,ab^2-41\,b^3-226\,a^2c-178\,abc+161\,b^2c-267\,ac^2+148\,bc^2+4\,c^3-123\,a^2+411\,ab-206\,b^2+146\,ac-324\,bc+38\,c^2-329\,a-62\,b-584\,c, 14\,b^2c^2+5+a^3+9\,a^2b+2\,ab^2-b^3+23\,a^2c+10\,abc-21\,b^2c+15\,ac^2-8\,bc^2+3\,c^3-3\,a^2-19\,ab-4\,b^2-6\,ac-26\,bc-10\,c^2+7\,a-b+3\,c, 21\,abc^2+30-a^3-2\,a^2b-51\,ab^2+8\,b^3+40\,a^2c+25\,abc-56\,b^2c+69\,ac^2-13\,bc^2+11\,c^3-18\,a^2-72\,ab+53\,b^2-8\,ac+54\,bc+10\,c^2+56\,a+29\,b+116\,c, 14\,a^2c^2+11+5\,a^3+31\,a^2b-74\,ab^2+9\,b^3+45\,a^2c+22\,abc-35\,b^2c+61\,ac^2-40\,bc^2+c^3+13\,a^2-95\,ab+50\,b^2-44\,ac+66\,bc+6\,c^2+63\,a+23\,b+127\,c, 21\,b^3c-6-4\,a^3+13\,a^2b+48\,ab^2-10\,b^3-8\,a^2c-5\,abc-14\,b^2c+3\,ac^2-10\,bc^2+2\,c^3-30\,a^2+27\,ab-40\,b^2+10\,ac-57\,bc-2\,c^2-7\,a-10\,b-40\,c, 6\,ab^2c-3-a^3-5\,a^2b+12\,ab^2-b^3-11\,a^2c-2\,abc+7\,b^2c-9\,ac^2+2\,bc^2-c^3-3\,a^2+15\,ab-4\,b^2+4\,ac-6\,bc-2\,c^2-13\,a-b-19\,c, 14\,a^2bc-13+3\,a^3+13\,a^2b+6\,ab^2-3\,b^3-a^2c+2\,abc+21\,b^2c-25\,ac^2+4\,bc^2-5\,c^3+19\,a^2+27\,ab-26\,b^2+10\,ac-8\,bc-2\,c^2-7\,a-17\,b-33\,c, 7\,a^3c+3-5\,a^3-24\,a^2b+25\,ab^2-2\,b^3-17\,a^2c-8\,abc+7\,b^2c-19\,ac^2+12\,bc^2-c^3-6\,a^2+32\,ab-8\,b^2+23\,ac-17\,bc-6\,c^2-21\,a-2\,b-36\,c, 42\,b^4-3-121\,a^3-557\,a^2b+570\,ab^2+275\,b^3-515\,a^2c-608\,abc-77\,b^2c-555\,ac^2+2\,bc^2-55\,c^3-645\,a^2+633\,ab-160\,b^2+82\,ac-690\,bc-302\,c^2-679\,a+65\,b-1147\,c, 42\,ab^3+15-11\,a^3-85\,a^2b+6\,ab^2+25\,b^3-43\,a^2c-40\,abc-49\,b^2c-39\,ac^2+4\,bc^2-5\,c^3-51\,a^2-15\,ab+58\,b^2-4\,ac+6\,bc-16\,c^2-35\,a+25\,b-5\,c, 21\,a^2b^2+3+2\,a^3+25\,a^2b-45\,ab^2+5\,b^3+25\,a^2c-29\,abc-14\,b^2c+9\,ac^2-16\,bc^2-c^3-6\,a^2-45\,ab+20\,b^2-47\,ac+18\,bc+c^2+14\,a+5\,b+41\,c, 21\,a^3b+18-16\,a^3-74\,a^2b+24\,ab^2+2\,b^3-53\,a^2c+abc-35\,b^2c+12\,ac^2+23\,bc^2+8\,c^3-36\,a^2+24\,ab+29\,b^2+40\,ac+3\,bc-8\,c^2-28\,a+23\,b-13\,c, 42\,a^4-57+431\,a^3+757\,a^2b-804\,ab^2+59\,b^3+799\,a^2c-2\,abc-119\,b^2c+417\,ac^2-340\,bc^2+5\,c^3+303\,a^2-1203\,ab+194\,b^2-752\,ac+246\,bc+184\,c^2+581\,a-67\,b+1013\,c\}$.

The equivalent lexicographic base.

$\{1-6\,c-41\,c^2+71\,c^3-41\,c^{18}+197\,c^{14}+106\,c^{16}-6\,c^{19}+106\,c^4+71\,c^{17}+c^{20}+92\,c^5+197\,c^6+145\,c^7+257\,c^8+278\,c^9+201\,c^{10}+278\,c^{11}+257\,c^{12}+145\,c^{13}+92\,c^{15}, 9741532+39671190\,c-96977172\,c^2-140671876\,c^3+7007106\,c^{18}-120781728\,c^{14}-79900378\,c^{16}-1184459\,c^{19}-131742078\,c^4+49142070\,c^{17}-253979379\,c^5-204237390\,c^6-337505020\,c^7-356354619\,c^8-271667666\,c^9-358660781\,c^{10}-323810244\,c^{11}-193381378\,c^{12}-244307826\,c^{13}-131861787\,c^{15}+1645371\,b, -487915\,c^{18}+705159\,a+4406102\,c+16292173\,c^{14}+17206178\,c^2+3830961\,c^{16}+91729\,c^{19}-377534+21640797\,c^3+26686318\,c^4-4114333\,c^{17}+34893715\,c^5+37340389\,c^6+47961810\,c^7+46227230\,c^8+42839310\,c^9+46349985\,c^{10}+37678046\,c^{11}+28185846\,c^{12}+26536060\,c^{13}+13243117\,c^{15}\}$.