Contents

Preface

Part one: The Kronecker – Duval Philosophy

1 Euclid
   1.1 The Division Algorithm
   1.2 Euclidean Algorithm
   1.3 Bezout’s Identity and Extended Euclidean Algorithm
   1.4 Roots of Polynomials
   1.5 Factorization of Polynomials
   1.6* Computing a gcd
      1.6.1* Coefficient explosion
      1.6.2* Modular Algorithm
      1.6.3* Hensel Lifting Algorithm
      1.6.4* Heuristic gcd

2 Intermezzo: Chinese Remainder Theorems
   2.1 Chinese Remainder Theorems
   2.2 Chinese Remainder Theorem for a Principal Ideal Domain
   2.3 A Structure Theorem (1)
   2.4 Nilpotents
   2.5 Idempotents
   2.6 A Structure Theorem (2)
   2.7 Lagrange Formula

3 Cardano
   3.1 A Tautology?
   3.2 The Imaginary Number
   3.3 An Impasse
   3.4 A Tautology!
Contents

4 Intermezzo: Multiplicity of Roots  53
  4.1 Characteristic of a Field  54
  4.2 Finite Fields  55
  4.3 Derivatives  57
  4.4 Multiplicity  58
  4.5 Separability  62
  4.6 Perfect Fields  64
  4.7 Squarefree Decomposition  68

5 Kronecker I: Kronecker’s Philosophy  74
  5.1 Quotients of Polynomial Rings  75
  5.2 The Invention of the Roots  76
  5.3 Transcendental and Algebraic Field Extensions  81
  5.4 Finite Algebraic Extensions  84
  5.5 Splitting Fields  86

6 Intermezzo: Sylvester  91
  6.1 Gauss Lemma  92
  6.2 Symmetric Functions  96
  6.3* Newton’s Theorem  100
  6.4 The Method of Indeterminate Coefficients  106
  6.5 Discriminant  108
  6.6 Resultants  112
  6.7 Resultants and Roots  115

7 Galois I: Finite Fields  119
  7.1 Galois Fields  120
  7.2 Roots of Polynomials over Finite Fields  123
  7.3 Distinct Degree Factorization  125
  7.4 Roots of Unity and Primitive Roots  127
  7.5 Representation and Arithmetics of Finite Fields  133
  7.6* Cyclotomic Polynomials  135
  7.7* Cycles, Roots and Idempotents  141
  7.8 Deterministic Polynomial-time Primality Test  148

8 Kronecker II: Kronecker’s Model  156
  8.1 Kronecker’s Philosophy  156
  8.2 Explicitly Given Fields  159
  8.3 Representation and Arithmetics  164
     8.3.1 Representation  164
     8.3.2 Vector space arithmetics  165
     8.3.3 Canonical representation  165
     8.3.4 Multiplication  167
     8.3.5 Inverse and division  167
Contents ix

8.3.6 Polynomial factorization 168
8.3.7 Solving polynomial equations 169
8.3.8 Monic polynomials 169
8.4 Primitive Element Theorems 170

9 Steinitz 175
9.1 Algebraic Closure 176
9.2 Algebraic Dependence and Transcendency Degree 180
9.3 The Structure of Field Extensions 184
9.4 Universal Field 186
9.5* Lüroth’s Theorem 187

10 Lagrange 191
10.1 Conjugates 192
10.2 Normal Extension Fields 193
10.3 Isomorphisms 196
10.4 Splitting Fields 203
10.5 Trace and Norm 206
10.6 Discriminant 212
10.7* Normal Bases 216

11 Duval 221
11.1 Explicit Representation of Rings 221
11.2 Ring Operations in a Non-unique Representation 223
11.3 Duval Representation 224
11.4 Duval’s Model 228

12 Gauss 232
12.1 The Fundamental Theorem of Algebra 232
12.2 Cyclotomic Equations 237

13 Sturm 263
13.1* Real Closed Fields 264
13.2 Definitions 272
13.3 Sturm 275
13.4 Sturm Representation of Algebraic Reals 280
13.5 Hermite’s Method 284
13.6 Thom Codification of Algebraic Reals (1) 288
13.7 Ben-Or, Kozen and Reif Algorithm 290
13.8 Thom Codification of Algebraic Reals (2) 294

14 Galois II 297
14.1 Galois Extension 298
14.2 Galois Correspondence 300
14.3 Solvability by Radicals 305
14.4 Abel–Ruffini Theorem 314
Contents

14.5* Constructions with Ruler and Compass 318

Part two: Factorization 327

15 Prelude 329
15.1 A Computation 329
15.2 An Exercise 338

16 Kronecker III: factorization 346
16.1 Von Schubert Factorization Algorithm over the Integers 347
16.2 Factorization of Multivariate Polynomials 350
16.3 Factorization over a Simple Algebraic Extension 352

17 Berlekamp 361
17.1 Berlekamp’s Algorithm 361
17.2 The Cantor–Zassenhaus Algorithm 369

18 Zassenhaus 380
18.1 Hensel’s Lemma 381
18.2 The Zassenhaus Algorithm 389
18.3 Factorization Over a Simple Transcendental Extension 391
18.4 Cauchy Bounds 395
18.5 Factorization over the Rationals 398
18.6 Swinnerton-Dyer Polynomials 402
18.7 $L^3$ Algorithm 405

19 Finale 415
19.1 Kronecker’s Dream 415
19.2 Van der Waerden’s Example 415

Bibliography 420

Index 422
This preliminary chapter is just devoted to recalling the Euclidean Algorithms over a univariate polynomial ring and its elementary applications: roughly speaking they are essentially the obvious generalization of those over integers.

The fundamental tool related to the Euclidean Algorithms and to solving univariate polynomials is nothing more than the elementary Division Algorithm (Section 1.1), whose iterative application produces the Euclidean Algorithm (Section 1.2), which can be extended to prove and compute Bezout’s Identity (Section 1.3).

The Division- and Euclidean Algorithms and theorems have many important consequences for solving polynomial equations: they relate roots and linear factors of a polynomial (Section 1.4) allowing them, at least, to be counted, and are the basis for the theory (not the practice) of polynomial factorization (Section 1.5).

They also have another, more important, consequence which is a crucial tool in solving: they allow a computational system to be developed within quotients of polynomial rings; the discussion of this is postponed to Section 5.1.

A direct implementation of the Euclidean Algorithm provides an unexpected phenomenon, the ‘coefficient explosion’: during the application of the Euclidean Algorithm to two polynomials whose coefficients have small size, polynomials are produced with huge coefficients, even if the final output is simply 1. Finding efficient implementations of the Euclidean Algorithm was a crucial subject of research in the early days of Computer Algebra; in Section 1.6 I will briefly discuss this phenomenon and present efficient solutions to this problem.
1.1 The Division Algorithm

Throughout this chapter $k$ will be a field and $\mathcal{P} := k[X]$ the univariate polynomial ring over $k$.

If $f = \sum_{i=0}^{n} a_i X^i \in \mathcal{P}$ with $a_n \neq 0$, denote by $\text{lc}(f) := a_n$ the leading coefficient of $f$.

**Theorem 1.1.1 (Division Theorem).** Given $A(X), B(X) \in \mathcal{P}, B \neq 0$, there are unique $Q(X), R(X) \in \mathcal{P}$ such that

1. $A(X) = Q(X)B(X) + R(X)$;
2. $R \neq 0 \implies \deg(R) < \deg(B)$.

We call $Q$ the quotient and $R$ the remainder of $A$ modulo $B$ in $\mathcal{P}$.

**Proof**

Existence: The proof is by induction on $\deg(A)$.

If $A = 0$ or $\deg(A) < \deg(B)$, then $Q := 0$ and $R := A$ obviously satisfy the thesis.

If $\deg(A) = n \geq m = \deg(B)$, we inductively assume that the theorem is true for each polynomial $A_0$ such that $A_0 = 0$ or $\deg(A_0) < n$. We then have

\[
A(X) = a_n X^n + A_1(X), \quad B(X) = b_m X^m + B_1(X),
\]

with $a_n \neq 0, b_m \neq 0, A_1 = 0$ or $\deg(A_1) < n, B_1 = 0$ or $\deg(B_1) < m$.

Let

\[
A_0(X) := A(X) - a_n b_m^{-1} X^{n-m} B(X),
\]

which, if non-zero, has degree less than $n$; by the inductive assumption there are then $Q_0, R_0$ such that

1. $A_0(X) = Q_0(X) B(X) + R_0(X)$,
2. $R_0 \neq 0 \implies \deg(R_0) < \deg(B)$,

so that

\[
A(X) = (a_n b_m^{-1} X^{n-m} + Q_0(X)) B(X) + R_0(X)
\]

and therefore

\[
Q(X) := a_n b_m^{-1} X^{n-m} + Q_0(X), R(X) := R_0(X)
\]

satisfy the requirement.

Uniqueness: Assume that

1. $A(X) = Q_1(X) B(X) + R_1(X)$,
2. $A(X) = Q_2(X) B(X) + R_2(X)$,

Then

\[
Q_i(X) = a_n b_m^{-1} X^{n-m} + Q_0(X)
\]

for $i = 1, 2$.

Solving for $Q_0(X)$ in both equations and setting them equal yields

\[
a_n b_m^{-1} X^{n-m} + Q_0(X) = Q_1(X) - Q_2(X)
\]

or

\[
a_n b_m^{-1} X^{n-m} + Q_0(X) = R_1(X) - R_2(X)
\]

which, since $B(X)$ is non-zero, implies that $Q_0(X)$ is non-zero.

But $Q_0(X)$ is unique, so

\[
Q_0(X) = Q_1(X) - Q_2(X)
\]

and

\[
Q_0(X) = R_1(X) - R_2(X)
\]

This can only hold if $Q_1(X) = Q_2(X)$ and $R_1(X) = R_2(X)$. Therefore $Q(X)$ and $R(X)$ are unique.
1.1 The Division Algorithm

(3) \( R_i \neq 0 \implies \deg(R_i) < \deg(B), \, 1 \leq i \leq 2, \)
so that
\[
R_1(X) - R_2(X) = (Q_2(X) - Q_1(X)) B(X).
\]
If \( R_1 \neq R_2 \) then
\[
\deg(R_1 - R_2) < \deg(B) \leq \deg(Q_2 - Q_1) + \deg(B) = \deg(R_1 - R_2)
\]
giving a contradiction.
Therefore \( R_1 - R_2 = 0 \) and (since \( B \neq 0 \)) also \( Q_2 - Q_1 = 0 \).

Corollary 1.1.2. The ring \( \mathcal{P} \) is a euclidean domain.

In further applications, denote
\[
Q := \text{Quot}(A, B), \, R := \text{Rem}(A, B).
\]
Because of their uniqueness in \( \mathcal{P} \), if \( K \) is a field such that \( K \supseteq k \), the quotient and the remainder of \( A \) modulo \( B \) in \( K[X] \) are still \( Q \) and \( R \).

Algorithm 1.1.3. An inductive proof can be transformed into a recursive algorithm: If we assume \( k \) to be effective\(^1\) then the iterative algorithm in Figure 1.1 performs polynomial division.

---

\(^1\) The concept of effectiveness was first introduced as the notion of endlich vielen Schritten (finite number of steps) by Grete Hermann in 1926 for polynomial ideals in the fundamental paper


where she wrote:

Die Behauptung, eine Berechnung kann mit endlich vielen Schritten durchgeführt werden, soll dabei bedeuten, es kann eine obere Schranke für die Anzahl der zur Berechnung notwendigen Operationen angegeben werden. Es genügt also z. B. nicht, ein Verfahren anzugeben, von dem man theoretisch nachweisen kann, daß es mit endlich vielen Operationen zum Ziele führt, wenn für die Anzahl dieser Operationen keine obere Schranke bekannt ist.

The assertion that a computation can be carried through in a finite number of steps shall mean that an upper bound for the number of operations needed for the computation can be given. Thus it is not sufficient, for example, to give a procedure for which one can theoretically verify that it leads to the desired result in a finite number of operations, so long as no upper bound is known for the number of operations.

To this, van der Waerden in

6

Euclid

Fig. 1.1. Polynomial Division Algorithm

\[(Q,R) := \text{PolynomialDivision}(A,B)\]

where

- \(A, B \in k[X], B \neq 0\)
- \(Q, R \in k[X]\) are such that
  - \(A = QB + R\)
  - \(R \neq 0 \implies \deg(R) < \deg(B)\)

\[b := \text{lc}(B), m := \deg(B)\]
\[A_0 := A, Q := 0\]
\[\text{While } A_0 \neq 0 \text{ and } \deg(A_0) \geq \deg(B) \text{ do}\]
  \[a := \text{lc}(A_0), n := \deg(A_0)\]
  \[Q := Q + ab^{-1}x^{n-m}\]
  \[A_0 := A_0 - ab^{-1}x^{n-m}B\]
  \[R := A_0\]

1.2 Euclidean Algorithm

Let \(P_0, P_1 \in \mathcal{P}\), with \(P_1 \neq 0\) (and, to dispose of the trivial cases, assume also that \(P_0 \neq 0\)). Let \(P_2 := \text{Rem}(P_0, P_1)\) and inductively, define

\[P_{i+1} := \text{Rem}(P_i, P_{i+1})\]

while \(P_i \neq 0\). It is clear that the sequence \(P_0, P_1, \ldots, P_i, \ldots\) (which is called the polynomial remainder sequence (PRS) of \(P_0, P_1\)) is finite since, otherwise,

---

Ein Körper \(K\) soll explizite-bekannt heißen, wenn seine Elemente Symbole aus einem bekannten abzählbaren Vorrat von unterscheidbaren Symbolen sind, deren Addition, Multiplikation, Subtraktion und Division sich in endlichvielen Schritten ausführen lassen.

A field \(K\) is called explicitly given when its elements are symbols from a known numerable set of distinguishable symbols, whose addition, multiplication, subtraction and division can be performed in a finite number of steps.

In this book I will happily drop Hermann’s requirement that an algorithm must be provided with its complexity evaluation, and will mainly follow Macaulay’s opinion in


Macaulay considered the practical feasibility of an algorithm to be more crucial:

[The theory of polynomial ideals] might be regarded as in some measure complete if it were admitted that a problem is solved when its solution has been reduced to a finite number of feasible operations. If, however, the operations are too numerous or too involved to be carried out in practice the solution is only a theoretical one.
each $P_i$ must be non-zero which would give an infinite decreasing sequence of natural numbers:

$$\deg(P_1) > \deg(P_2) > \cdots > \deg(P_i) > \cdots.$$  

Let $D(X)$ denote the last non-zero element $P_r$ of the sequence, and note that $r \leq \min(\deg(P_0), \deg(P_1))$. Also denote $Q_i := \text{Quot}(P_{r-i}, P_r)$.

**Proposition 1.2.1.** $D(X) = \gcd(P_0, P_1)$.

**Proof** Since $P_{r-1} = Q_r P_r$, then $P_r$ divides $P_{r-1}$. So let us assume that $P_r$ divides $P_i$ for $i > k$ and prove that it divides $P_k$: this is obvious from the identity

$$P_k = Q_{k+1} P_{k+1} + P_{k+2}.$$  

Therefore $D = P_r$ is a common divisor of $P_0$ and $P_1$.

If $S(X)$ divides both $P_0$ and $P_1$, then since

$$P_2 = P_0 - Q_1 P_1,$$

it divides $P_2$. Assuming that $S$ divides $P_i$, for $i < k$, then by the identity

$$P_k = P_{k-2} - Q_{k-1} P_{k-1},$$

it also divides $P_k$, therefore it divides $P_r$.

Greatest common divisors in $\mathcal{P}$ are obviously not unique, but they are associate (cf. Definition 1.5.1).

Again if $K$ is a field such that $K \supseteq k$, $\gcd(A, B)$ and the PRS of $A$ and $B$ are the same in $K[\![X]\!]$ as in $\mathcal{P}$.

**Algorithm 1.2.2.** If $k$ is effective, the algorithm in Figure 1.2 computes the gcd of two polynomials; it actually computes the PRS of the two polynomials and also computes all the intermediate quotients $Q_j$.

---

**Fig. 1.2. Euclidean Algorithm**

\[
\begin{align*}
D & := \text{GCD}(A, B) \\
\text{where} & \\
& A, B \in \mathcal{P}, A \neq 0, B \neq 0 \\
& D \text{ is a } \gcd(A, B) \\
D & := A, U := B \\
\text{While } U \neq 0 \text{ do} & \\
& (Q, V) := \text{PolynomialDivision}(D, U) \\
& D := U, U := V
\end{align*}
\]
1.3 Bezout’s Identity and Extended Euclidean Algorithm

Proposition 1.3.1 (Bezout’s Identity). Let $P_0, P_1 \in \mathcal{P} \setminus k$, and let us denote $D := \gcd(P_0, P_1)$. Then there are $S, T \in \mathcal{P} \setminus \{0\}$ such that

(i) $P_0S + P_1T = D$

(ii) $\deg(S) < \deg(P_1), \deg(T) < \deg(P_0)$

Proof. Let $P_0, P_1, \ldots, P_r = D$ be the PRS of $P_0$ and $P_1$. Also, for $i = 0, \ldots, r - 1$, let $Q_i := \text{Quot}(P_i - 1, P_i)$. Inductively define:

- $S_0 := 1, T_0 := 0$
- $S_1 := 0, T_1 := 1$
- $S_i := S_{i-2} - Q_{i-1}S_{i-1}, T_i := T_{i-2} - Q_{i-1}T_{i-1}, \quad 2 \leq i \leq r$
- $S_i := \text{Rem}(S'_i, P_i), T_i := T'_i + \text{Quot}(S'_i, P_i)P_0, \quad 2 \leq i \leq r$

We claim that for $i = 0, \ldots, r$:

(i) $P_0S_i + P_1T_i = P_i$

(ii) $\deg(S_i) < \deg(P_1), \deg(T_i) < \deg(P_0)$

In fact the claims are trivial for $i = 0, 1$, and so, inductively assuming them to be true for $i < k$, and denoting $U_k := \text{Quot}(S'_k, P_1)$, so that

$$S'_k = U_kP_1 + S_k, \quad T_k = T'_k + U_kP_0$$

we have

$$P_k = P_{k-2} - Q_{k-1}P_{k-1}$$

$$= P_0S_{k-2} + P_1T_{k-2} - Q_{k-1}P_0S_{k-1} - Q_{k-1}P_1T_{k-1}$$

$$= P_0(S_{k-2} - Q_{k-1}S_{k-1}) + P_1(T_{k-2} - Q_{k-1}T_{k-1})$$

$$= P_0S'_k + P_1T'_k$$

$$= P_0U_kP_1 + P_0S_k + P_1T_k - P_1U_kP_0$$

$$= P_0S_k + P_1T_k.$$

Clearly $\deg(S_k) < \deg(P_1)$ and therefore also $\deg(T_k) < \deg(P_0)$, otherwise $\deg(P_1T_k) \geq \deg(P_1P_0) > \deg(S_kP_0)$ and $\deg(P_1T_k) > \deg(P_1) \geq \deg(P_k)$ would lead to an obvious contradiction.

Corollary 1.3.2. The ring $\mathcal{P}$ is a principal ideal domain.
1.4 Roots of Polynomials

Fig. 1.3. Extended Euclidean Algorithm

\[(D, S, T) := \text{ExtGCD}(A, B)\]

where

- \(A, B \in \mathbb{P}, A \neq 0, B \neq 0\)
- \(D\) is a \(\text{gcd}(A, B)\)
- \(SA + BT = D\)
- \(\deg(S) < \deg(B), \deg(T) < \deg(A)\)

\[D := A, U := B\]
\[S_0 := 1, S_1 := 0\]
\[\rightarrow T_0 := 0, T_1 := 1\]

\[\text{While } U \neq 0 \text{ do}\]

\[(Q, V) := \text{PolynomialDivision}(D, U)\]
\[D := U, U := V\]
\[S := S_0 - QS_1, \rightarrow T := T_0 - QT_1\]
\[(Q, S) := \text{PolynomialDivision}(S, B)\]
\[\rightarrow T := T + QA\]
\[S_0 := S_1, S_1 := S\]
\[\rightarrow T_0 := T_1, T_1 := T\]
\[S := S_0, \rightarrow T := T_0\]

Algorithm 1.3.3. Again, on an effective field, \(S\) and \(T\) can be computed by the algorithm in Figure 1.3.

Algorithm 1.3.4. The so-called Half-extended Euclidean Algorithm allows us to compute \(S\), without having to compute \(T\); it simply involves removing the lines marked by \(\rightarrow\) in the algorithm in Figure 1.3. It is useful to compute inverses of field elements (see Remark 5.1.4).

1.4 Roots of Polynomials

The Division Theorem also has an obvious but important consequence on the solving of polynomial equations:

**Corollary 1.4.1.** For \(f(X) \in \mathbb{P}\), and \(\alpha \in k\) we have:

\[f(\alpha) = 0 \iff (X - \alpha) \text{ divides } P(X).\]

**Proof** Let

\[Q(X) := \text{Quot}(f(X), X - \alpha), \quad R(X) := \text{Rem}(f(X), X - \alpha);\]

since \((X - \alpha)\) is linear, either \(R(X) = 0\) or \(\deg(R) = 0\), i.e. \(R(X)\) is a constant \(r \in k\).
Therefore,

\[ f(X) = Q(X)(X - \alpha) + r, \]

and evaluating in \( \alpha \) obtains \( f(\alpha) = r \), from which the proof follows.

As a consequence a polynomial cannot have more roots than its degree.

1.5 Factorization of Polynomials

**Definition 1.5.1.** In a domain \( D \):

(i) two elements \( a \) and \( b \) are called associate if there exists \( c \in D \), with \( c \) invertible, such that \( a = bc \);

(ii) a non-zero and non-invertible element \( a \) is called irreducible if it is divisible only by invertible elements and by its associates, i.e.

\[ a = bc, \text{ and } b \text{ non-invertible } \implies c \text{ is invertible and so } b \text{ is associate to } a. \]

**Definition 1.5.2.** A domain \( D \) is a unique factorization domain if for each non-invertible \( a \in D \setminus \{0\} \)

(i) there is a factorization \( a = p_1 \ldots p_r \) where each \( p_i \) is irreducible;

(ii) the factorization is unique in the following sense:

if \( a = q_1 \ldots q_s \) is another factorization with \( q_i \) irreducible, then

- \( r = s \),
- each \( p_i \) is associate to some \( q_j \),
- each \( q_j \) is associate to some \( p_i \).

**Lemma 1.5.3.** If \( p(X) \in k[X] \) is irreducible, \( p \) divides \( q_1q_2 \) and \( p \) does not divide \( q_2 \), then \( p \) divides \( q_1 \).

**Proof** Since \( \gcd(p, q_2) \) divides \( p \), it either is associate to \( p \) or is a unit; since \( p \) does not divide \( q_2 \), we can then conclude that \( \gcd(p, q_2) = 1 \).

By Bezout’s Identity, there are \( s, t \in k[X] \), such that \( sp + tq_2 = 1 \) and therefore \( spq_1 + tq_1q_2 = q_1 \), so that \( p \) divides \( q_1 \).

**Lemma 1.5.4.** Let \( f \in k[X] \): Let \( f = p_1 \ldots p_r, f = q_1 \ldots q_s \) be two factorizations in irreducible factors. Then

(i) \( r = s \),

(ii) each \( p_i \) is associate to some \( q_j \),

(iii) each \( q_j \) is associate to some \( p_i \).
1.5 Factorization of Polynomials

Proof The proof is by induction on \( r \). If \( r = 1 \), then \( p_1 = f = q_1 \ldots q_s \), so that \( s = 1 \) and \( p_1 = q_1 \) because \( p_1 \) is irreducible.

Assume therefore that each polynomial that has a factorization with less than \( r \) irreducible factors, has a unique factorization and let \( f = p_1 \ldots p_r, f = q_1 \ldots q_s \) be two factorizations of \( f \) in irreducible factors. Then \( p_1 \) divides \( q_1 \ldots q_s \) and therefore, by Lemma 1.5.3, it must divide one among the \( q_i \)’s, say \( q_j \).

Since \( q_j \) is irreducible, we have \( p_1 = uq_j \) for some \( u \in k \setminus \{0\} \). We then have
\[
f = uq_jp_2 \ldots p_r = q_1 \ldots q_s,
\]
and, dividing out \( q_j \),
\[
(up_2)p_3 \ldots p_r = q_1 \ldots q_{j-1}q_{j+1} \ldots q_s.
\]
The proof can then be completed using the inductive assumption.

Lemma 1.5.5. Each non-constant polynomial \( f \in k[X] \) has a factorization into irreducible factors.

Proof The proof is by induction on \( \deg(f) \).

Since linear polynomials are obviously irreducible, the result is true for polynomials of degree 1.

Assume next that it is true for polynomials \( g \in k[X], \deg(g) < n \), and let \( f \in k[X] \) be such that \( \deg(f) = n \). Either \( f \) is irreducible, so that \( f \) satisfies the lemma, or \( f \) is not irreducible, so that \( f = f_1f_2 \) where neither \( f_1 \) nor \( f_2 \) is a constant and each has degree less than \( n \); therefore there are factorizations \( f_1 = p_1 \ldots p_r \) and \( f_2 = q_1 \ldots q_s \) in irreducible factors, and
\[
f = p_1 \ldots p_rq_1 \ldots q_s
\]
is then a factorization of \( f \).

Theorem 1.5.6. \( k[X] \) is a unique factorization domain.

Proof Existence of a factorization is guaranteed by Lemma 1.5.5, uniqueness by Lemma 1.5.4.
Remark 1.5.7. It is important to note that, unlike the other results of this chapter, Theorem 1.5.6 does not give any way of computing a factorization. In fact the argument of Lemma 1.5.5, that either $f$ is irreducible or it has a proper factorization, does not give any hint of how to decide which is the case, nor how to find proper divisors. We will show in Part II that there are factorization algorithms for polynomials over all fields which are important for our theory (namely all finite fields and all finite extensions of the rationals).

However, there exist effective fields $k$ such that it is undecidable whether the polynomial $X^2 + 1 \in k[X]$ is irreducible or not, the reason being that it is undecidable whether the imaginary number $i$ is in $k$ (see Section 19.2).

1.6 Computing a gcd

1.6.1 Coefficient explosion

Example 1.6.1. Let us assume that we need to compute the gcd of the two polynomials

\[
P_0 \ := \ X^8 + X^6 - 3X^4 - 3X^3 + 8X^2 + 2X - 5,
\]
\[
P_1 \ := \ 3X^6 + 5X^3 - 4X^2 - 9X + 21,
\]
in $\mathbb{Z}[X]$; we need of course to apply the Euclidean Algorithm; let us even assume that we have available nothing more than a pocket calculator, so that we can compute only in $\mathbb{Z}$ but not in $\mathbb{Q}$.

Well, that is not a serious problem: in fact, since the gcd is stable under associate elements, it is clear that by substituting the line of the algorithm of Figure 1.1

\[
A_0 := A_0 - ab^{-1}X^{n-m}B
\]

by

\[
A_0 := bA_0 - aX^{n-m}B;
\]

the answer is correct.

In this way we obtain the following PRS:

\[
P_2 \ := \ -15X^4 + 3X^2 - 9,
\]
\[
P_3 \ := \ -15795X^2 - 30375X + 59535,
\]
\[
P_4 \ := \ 1254542875143750X - 1654608338437500,
\]
\[
P_5 \ := \ 12593338795500743100931141992187500,
\]

from which, provided we are able to complete this computation, we deduce that

\[
\gcd(P_0, P_1) = 1.
\]
1.6 Computing a gcd

Clearly, we can perform rational arithmetic, even if it is not available on our pocket calculator, using simply the Euclidean Algorithm for the integers; the computation is of course more complex and the answer is

\[ \begin{align*}
P_2 &:= -\frac{5}{9}X^4 + \frac{1}{9}X^2 - \frac{1}{3}, \\
P_3 &:= -\frac{117}{25}X^2 - 9X + \frac{441}{25}, \\
P_4 &:= \frac{233150}{6591}X - \frac{102500}{2197}, \\
P_5 &:= \frac{1288744821}{543589225}.
\end{align*} \]

Having already used stability under associate elements, we could, at each step, force each \( P_i \) to become monic; this requires more integer Euclidean Algorithms, but we could hope to do it with small size elements; in fact we get:

\[ \begin{align*}
P_2 &:= X^4 - \frac{1}{5}X^2 + \frac{3}{5}, \\
P_3 &:= X^2 + \frac{25}{13}X - \frac{49}{13}, \\
P_4 &:= X - \frac{6150}{4663}, \\
P_5 &:= 1.
\end{align*} \]

Historical Remark 1.6.2. The amusing assumption of having just a pocket calculator, while not realistic, has a meaning. In fact, the above example is taken from the second volume of Knuth’s book *The Art of Computer Programming*.

That book was published in 1969, when programs were input via punched cards . . . and computer algebra was being born. In fact, an analysis of the unexpected phenomenon of coefficient growth explosion, and the first tentative steps taken for solving it, marked the beginning of the unexpected phenomenon of computer algebra’s rapid growth.

Independently Collins and Brown\(^2\), applying subresultant theory, showed that in computing the PRS over \( \mathbb{Z} \) it was possible at each step, while producing an element \( P_i \), to predict an integer \( c_i \) dividing each coefficient of \( P_i \), and thereby, performing the substitution \( P_i \leftarrow P_i / c_i \), get smaller size coefficients;

\(^2\) See


The discussion (and the computations) of the example are taken from Brown’s paper.
Euclid

for instance, in the example above we get:

\[ P_2 := 15X^4 - 3X^2 + 9, \]
\[ P_3 := 65X^2 + 125X - 245, \]
\[ P_4 := 9326X - 12300, \]
\[ P_5 := 260708. \]

Research on how to compute the polynomial gcd continues; on the basis of
general knowledge, there are three competing approaches\(^3\):  

- **modular algorithm** based on the Chinese Remainder Theorem (Brown, 1971);
- the **Hensel Lifting Algorithm** (Moses–Yun, 1973; Wang, 1980) based on
  Hensel’s Lemma (cf. Section 18.1);
- the **Heuristic GCD** (Char–Geddes–Gonnet, 1984; Davenport–Padget, 1985).

In the following sections we will briefly discuss these three algorithms\(^4\),
using freely some facts that will be proved later:

**Fact 1.6.3.** Let \( f \in \mathbb{Z}[X] \) be a polynomial. Then:

1. there is a computable integer \( B \in \mathbb{N} \) such that for each factor \( \sum a_i X^i \) of
   \( f \), we have \( -B < a_i \leq B \);
2. there is a computable integer \( r \in \mathbb{N} \) such that for each root \( \rho \in \mathbb{C} \) of \( f \),
   we have \( |\rho| < r \).

**Proof** cf. Section 18.4.

For each \( p \in \mathbb{N} \) let us denote the canonical projection morphism as
\(-p : \mathbb{Z}[X] \mapsto \mathbb{Z}_p[X]\); conversely, we can consider the (implicit) immersion
\( \mathbb{Z}_p[X] \subset \mathbb{Z}[X] \), where each polynomial \( f(X) \in \mathbb{Z}_p[X] \) can be interpreted,

\(^3\) See

W.S. Brown, On Euclid’s Algorithm and the Computation of Polynomial and Greatest Common
159–166;
B.W. Char, K.O. Geddes, G. H. Gonnet, GCDHEU: Heuristic Polynomial GCD Algorithm
J. Davenport, J. Padget, HEUGCD: How Elementary Upperbounds Generate Cheaper Data, *L.

\(^4\) The presentation of modular algorithm depends freely on the results discussed in Section 2.1
and the presentation of the Hensel Lifting Algorithm in Section 18.1. It is suggested that the
interested reader go to those sections first.
1.6 Computing a gcd

with a slight abuse of notation, as a polynomial \( f(X) := \sum_{i=0}^{n} a_i X^i \in \mathbb{Z}[X] \) such that

\[
f(X) = f_p(X),
\]
\[-p/2 < a_i \leq p/2,
\]

from which we can readily identify \( f \) and \( f_p \).

Let \( f, g \in \mathbb{Z}[X] \), \( h := \gcd(f, g) \) and let \( p \in \mathbb{N} \) be a prime. Then:

Lemma 1.6.4. With the above notation:

1. \( h_p \) divides \( \gcd(f_p, g_p) \);
2. if \( \text{lc}(f) \not\equiv 0 \not\equiv \text{lc}(g) \pmod{p} \), then

\[
\deg(\gcd(f_p, g_p)) \geq \deg(h_p) = \deg(h).
\]

Proof. Part 1 is obvious and implies \( \deg(h_p) \leq \deg(\gcd(f_p, g_p)) \). The assumption of Part 2 implies that \( \text{lc}(h) \not\equiv 0 \pmod{p} \) so that

\[
\deg(h) = \deg(h_p) \leq \deg(\gcd(f_p, g_p)).
\]

Fact 1.6.5. If \( \text{lc}(f) \not\equiv 0 \not\equiv \text{lc}(g) \pmod{p} \), then there exists \( R \in \mathbb{Z} \) such that \( p \) does not divide \( R \Rightarrow h_p = \gcd(f_p, g_p) \).

Proof (sketch). Corollary 6.6.6 will show that, given \( f', g' \in \mathbb{Z}[X] \), there is \( R \in \mathbb{Z} \) such that the following are equivalent

\[
R \not\equiv 0 \pmod{p};
\]
\[
\gcd(f'_p, g'_p) = 1.
\]

Therefore we only have to apply this result to \( f' := f/h \) and \( g' := g/h \) since

\[
\gcd(f_p, g_p) = h_p \gcd(f'_p, g'_p).
\]

Corollary 1.6.6. There are only finitely many primes \( p \in \mathbb{N} \) for which

\[
\gcd(f_p, g_p) = h_p
\]

does not hold.

Proof. We only need to discard those primes which divide either \( \text{lc}(f) \), \( \text{lc}(g) \) or \( R \).
1.6.2 Modular Algorithm

On the basis of the above result, denoting by \( P \) the set of integer primes, the modular algorithm consists of computing

\[ h(p) := \gcd(f_p, g_p) \]

for several primes \( p \in P \subset \mathbb{N} \) until we obtain a subset \( P' \subset P \) such that

- \( p \) does not divide \( \text{lc}(f) \text{lc}(g) \), for all \( p \in P' \);
- \( \deg(h(p)) \leq \deg(h(q)) \), for all \( p \in P' \), for all \( q \in P' \);
- \( \prod_{p \in P'} p \geq \mathfrak{B} \),

where \( \mathfrak{B} \) satisfies Fact 1.6.3.1, for both \( f \) and \( g \).

Then,

either for all \( p \in P' \), \( \deg(h(p)) = \deg(h) \) and so \( h(p) = h_p \), in which case we can apply the Chinese Remainder Theorem (Corollary 2.1.5) in order to compute the single element \( h = \sum a_i X^i \in \mathbb{Z}[X] \) such that

\[ -\mathfrak{B} < a_i \leq \mathfrak{B}, \text{ for all } i; \]
\[ h_p = h(p) = h_p, \]

from which

\[ h = h = \gcd(f, g); \]

or for all \( p \in P' \), we have \( \deg(h(p)) > \deg(h) \), which happens with low probability; in this case the above computation gives a wrong answer, but this can be detected by checking whether \( h \) divides \( f \) and \( g \); in fact, if the answer is positive then we can deduce that \( h \) divides \( h = \gcd(f, g) \) and since \( \deg(h) \geq \deg(h) \) we can deduce that \( h = h = \gcd(f, g) \).

Algorithm 1.6.7. This approach leads to the algorithm presented in Figure 1.4.

1.6.3 Hensel Lifting Algorithm

The algorithm is based on the following

**Fact 1.6.8.** Let \( p \in \mathbb{N} \) be a prime and let \( f(X) \in \mathbb{Z}[X] \) satisfy

\[ \text{lc}(f) \not\equiv 0 \pmod{p}. \]

Let \( f, h \in \mathbb{Z}[X] \) satisfy

1. \( f \equiv fh \pmod{p} \),
2. \( \deg(f) = \deg(f) + \deg(h) \),
3. \( \gcd(f_p, h_p) = 1 \).
1.6 Computing a gcd

Fig. 1.4. Modular GCD

\[ h := \text{GCD}(f, g) \]

where

\[ f, g \in \mathbb{Z}[X], \]
\[ h := \text{gcd}(f, g) \]

Repeat

choose a prime \( p \in \mathbb{N} \) such that \( p \) does not divide \( \text{lc}(f) \text{lc}(g) \)

\[ h^{(p)} := \text{gcd}(f_p, g_p) \]
\[ p := p, h := h^{(p)}, d := \text{deg}(h) \]

Repeat

If \( \text{deg}(h^{(p)}) < d \) then

\[ p := p, h := h^{(p)}, d := \text{deg}(h) \]

else

If \( d = 0 \) then

\[ h := 1 \]

else

choose a prime \( p \in \mathbb{N} \) such that \( p \) does not divide \( \text{plc}(f) \text{lc}(g) \)

\[ h^{(p)} := \text{gcd}(f_p, g_p) \]

If \( \text{deg}(h^{(p)}) = \text{deg}(h) \) then

Compute by the Chinese Remainder Theorem \( h' \) such that

\[ h' \equiv \begin{cases} 
  h \pmod{p} \\
  h^{(p)} \pmod{p}
\end{cases} \]

\[ h := h', p := p^p \]

until \( p \geq 2^3 \)

until \( h \) divides \( f \) and \( g \)

Then for each \( n \in \mathbb{N} \), denoting \( q := p^n \), it is possible to compute

\[ f', h' \in \mathbb{Z}[X] \]

such that

1. \( f \equiv f' h' \pmod{q} \),
2. \( f' \equiv f \pmod{p} \), \( h' \equiv h \pmod{p} \),
3. \( \text{deg}(f') = \text{deg}(f), \text{deg}(h') = \text{deg}(h) \).

Moreover there is an algorithm (the Hensel Lifting Algorithm) for computing them.

Proof Compare with Theorem 18.1.2.

Let \( f, g \in \mathbb{Z}[X] \), and \( h := \text{gcd}(f, g) \). After computing \( \text{gcd}(f_p, g_p) \) for several primes \( p \in \mathbb{N} \), we will probabilistically obtain an element \( h := \text{gcd}(f_p, g_p) \).
Euclid

for a suitable prime \( p \in \mathbb{N} \), such that \( \text{deg}(h) = \text{deg}(h) \), choosing only the one for which \( \text{deg}(h) \) is minimal.

Denoting \( f := f/h \), then \( f \) and \( h \) satisfy the assumptions of the above Fact. Therefore choosing \( n \in \mathbb{N} \) such that \( q := p^n \geq B \), we can obtain the polynomials \( f', h' = \sum a_i X^i \) satisfying the above condition.

Therefore

\[ \text{deg}(h') = \text{deg}(h) \geq \text{deg}(h), \quad -B < a_i \leq B, \quad \text{for all } i, \]

so that if \( h' \) divides \( f \) and \( g \) then \( h' = \gcd(f, g) \).

1.6.4 Heuristic gcd

As both the modular and the Hensel lifting gcds are based on restricting the mapping

\[ -p : \mathbb{Z}[X] \mapsto \mathbb{Z}_p[X] \]

to the suitable subset

\[ S := \left\{ \sum_{i=0}^{n} a_i X^i : -\frac{p}{2} < a_i \leq \frac{p}{2}, \text{ for all } i \right\} \subset \mathbb{Z}[X] \]

so that the restriction of \( -p \) to \( S \) is an isomorphism, the heuristic gcd is based on the restriction of a different projection to a subset in order to make it invertible.

Let us just consider, for each \( \xi \in \mathbb{Z} \), the evaluation map \( \text{ev}_\xi : \mathbb{Z}[X] \mapsto \mathbb{Z} \)
defined by \( \text{ev}_\xi (h) := h(\xi) \), for all \( h(X) \in \mathbb{Z}[X] \).

**Lemma 1.6.9.** Let

\[ S := \left\{ h(X) = \sum_{i=0}^{n} a_i X^i \in \mathbb{Z}[X] : -\frac{\xi}{2} < a_i \leq \frac{\xi}{2}, \text{ for all } i \right\} \subset \mathbb{Z}[X]. \]

Then the restriction of \( \text{ev}_\xi \) to \( S \) is an isomorphism between it and \( \mathbb{Z} \).

It is clear how to compute \( \text{ev}_\xi^{-1}(\gamma) \) for each integer \( \gamma \) (cf. Fig. 1.5).

**Theorem 1.6.10.** Let \( f, g \in \mathbb{Z}[X] \) and let \( \tau \in \mathbb{N} \) be a bound for all the roots of both \( f \) and \( g \) (cf. Fact 1.6.3).

Let \( \xi \in \mathbb{Z} \) be such that \( |\xi| > 1 + \tau \); let \( m := f(\xi), n := g(\xi), \gamma := \gcd(m, n) \) and \( h(X) := \text{ev}_\xi^{-1}(\gamma) \).
1.6 Computing a gcd

Fig. 1.5. Computation of \( ev^{-1}_\xi \)

\[ h := ev^{-1}_\xi(y) \]

where
\[ \xi \in \mathbb{Z}, \]
\[ \gamma \in \mathbb{Z}, \]
\[ h(X) \in S \subset \mathbb{Z}[X], \]
\[ h(\xi) = \gamma \]
\[ h := \gamma, h := 0, i := 0. \]

While \( h \neq 0 \) do

Let \( a \in \mathbb{Z} \) be the unique element such that
\[ a \equiv h \pmod{\xi}, \]
\[ \frac{-\xi}{2} < a \leq \frac{\xi}{2} \]
\[ h := (h - a)/\xi, h := h + aX, i := i + 1 \]

Then the following conditions are equivalent:

\( h(X) \) divides both \( f(X) \) and \( g(X) \);
\[ h(X) = \gcd(f, g). \]

Proof If \( h \) divides both \( f(X) \) and \( g(X) \) and therefore \( \gcd(f, g) \), then there exists \( H \in \mathbb{Z}[X] \) such that \( \gcd(f, g) = hH \); then we have
\[ h(\xi) = \gamma = \gcd(m, n) = \gcd(f(\xi), g(\xi)) \geq \gcd(f, g)(\xi) = h(\xi)H(\xi), \]
so that \( H(\xi) = \pm 1 \).

Since, by the Fundamental Theorem of Algebra, \( H(X) = \prod_{i}(X - \alpha_i) \) for suitable \( \alpha_i \in \mathbb{C} \), we can deduce that \( \prod_{i}(\xi - \alpha_i) = H(\xi) = \pm 1 \) and that there is an \( \alpha \) such that \( |\xi - \alpha| \leq 1 \), giving the contradiction
\[ |\xi| > 1 + r \geq 1 + |\alpha| \geq |\xi|. \]

This leads to the probabilistic algorithm presented in Figure 1.6.

Example 1.6.11. An example is

\[
\begin{align*}
  f(X) & := X^3 - 3X^2 - X + 3 & = (X - 1)(X + 1)(X - 3) \\
  g(X) & := X^3 + X^2 - 9X - 9 & = (X + 1)(X - 3)(X + 3) \\
  \xi & := 10 \\
  m & := 693 & = 9 \cdot 11 \cdot 7 \\
  n & := 1001 & = 11 \cdot 7 \cdot 13 \\
  \gamma & := 77 & = 11 \cdot 7 \\
  h(X) & := X^2 - 2X - 3 & = (X + 1)(X - 3)
\end{align*}
\]
Fig. 1.6. Heuristic GCD

\[ h := \text{HEUGCD}(f, g) \]

where

\[ f, g \in \mathbb{Z}[X], \]

\[ h(X) = \gcd(f, g). \]

Choose \( e \in \mathbb{R}, e > 1 \)

Choose \( \xi \in \mathbb{Z} \)

Repeat

\[ \xi := \lfloor \xi e \rfloor \]

\[ m := f(\xi), n := g(\xi), \]

\[ \gamma := \gcd(m, n) \]

\[ h(X) := ev^{-1}_{\xi}(\gamma) \]

\[ * \ h(X) := \text{Prim}(h) \]

until \( h \) divides both \( f \) and \( g \)

---

**Example 1.6.12.** However, if you consider

\[ f(X) := (X + 1)(X + 2)(X + 3) \]

and \( g(X) := (X - 2)(X - 1)X \)

it is clear that \( \gcd(f, g) = 1 \), and \( m \equiv 0 \equiv n \pmod{6} \), for all \( \xi \in \mathbb{Z} \), so that \( h(X) \neq \gcd(f, g) \), for all \( \xi \in \mathbb{Z} \),

and the algorithm cannot terminate.

However, when \( \xi > 12 \) and \( \gcd(m, n) = 6 \), the algorithm returns \( h(X) = 6 \) which is associate to \( \gcd(f, g) \).

This suggests that we remove the content of \( h^5 \) by adding the line marked by \( * \) in Figure 1.6.

The correctness of this amended algorithm is given by

**Theorem 1.6.13.** Let \( f, g \in \mathbb{Z}[X] \) and let \( r \in \mathbb{N} \) be a bound for all the roots of \( f \) and \( g \).

Let \( \xi \in \mathbb{Z} \) be such that

\[ |\xi| \geq 1 + 2r, \]

and let \( m := f(\xi), n := g(\xi), \gamma := \gcd(m, n), h'(X) := ev^{-1}_{\xi}(\gamma), c := \text{cont}(h'), \text{ and } h := \text{Prim}(h') = c^{-1}h'. \]

\[^5\text{We recall that for a polynomial } h(X) := \sum a_i X_i, \text{ the content of } h \text{ is } \]

\[ \text{cont}(h) := c := \gcd(a_i) \]

and we will denote \( \text{Prim}(h) := c^{-1}h(X) \) (cf. Section 6.1).
1.6 Computing a gcd

Then the following conditions are equivalent:

\( h(X) \) divides both \( f(X) \) and \( g(X) \);
\( h(X) = \gcd(f, g) \).

**Proof** If \( h \) divides both \( f(X) \) and \( g(X) \) and therefore \( \gcd(f, g) \), then there exists \( H \in \mathbb{Z}[X] \) such that \( \gcd(f, g) = hH \); thus we have

\[
ch(\xi) = h'(\xi) = \gamma = \gcd(m, n) = \gcd(f(\xi), g(\xi)) \geq \gcd(f, g)(\xi) = h(\xi)H(\xi)
\]

so that \( H(\xi) \leq \pm c \). Since each coefficient of \( h' \) is bounded by \( \xi/2 \), we have \( c < \xi/2 \).

Therefore, by the same argument as in Theorem 1.6.10, there is an \( \alpha \) such that

\[
|\xi - \alpha| \leq c < \xi/2
\]

so that \( |\alpha| \geq \xi/2 > \tau \), which is a contradiction.

**Lemma 1.6.14.** Let \( f, g \in \mathbb{Z}[X] \) be such that \( \gcd(f, g) = 1 \). Then there is \( M \in \mathbb{N} \) such that

\[
\forall \xi \in \mathbb{Z}, \gcd(f(\xi), g(\xi)) \leq M.
\]

**Proof** By assumption there are \( a'(X), b'(X) \in \mathbb{Q}[X] \) such that \( a'f + b'g = 1 \); eliminating denominators, we obtain polynomials \( a(X), b(X) \in \mathbb{Z}[X] \) and an integer \( M \in \mathbb{N} \) such that

\[
a(X)f(X) + b(X)g(X) = M.
\]

Therefore for all \( \xi \in \mathbb{Z}, a(\xi)f(\xi) + b(\xi)g(\xi) = M \), from which the proof follows.

**Corollary 1.6.15.** Let \( f, g \in \mathbb{Z}[X], h(X) := \gcd(f, g) \). Then there is \( M \in \mathbb{N} \) such that

\[
\forall \xi \in \mathbb{Z}, \gcd(f(\xi), g(\xi)) \leq Mh(\xi).
\]

**Proof** Apply the above lemma to the polynomials \( f/h \) and \( g/h \).

**Corollary 1.6.16.** Let \( f, g \in \mathbb{Z}[X] \) and \( \xi > 2M\mathfrak{B} \).

Let \( m := f(\xi), n := g(\xi), \gamma := \gcd(m, n), h'(X) := ev_{\xi}^{-1}(\gamma), c := \cont(h'), \) and \( h := \text{Prim}(h') = c^{-1}h' \).
Then \( h(X) = \gcd(f, g) \).

**Proof**  Denoting \( h(X) := \sum a_i X^i \), we have

\[
2M|a_i| \leq 2M \mathcal{B} < \xi, \quad \text{for all } i,
\]

so that \( e_{\xi^{-1}}(\gamma) = Mh(X) \).

**Corollary 1.6.17.**  The algorithm of Figure 1.6 terminates.