Universidad de La Laguna

Algunos algoritmos de factorización de polinomios con coeficientes enteros

Some algorithms of factorization of polynomials over the integers

Raquel Padrón García

Trabajo de Fin de Grado

Sección de Matemáticas

Facultad de Ciencias

Universidad de La Laguna

La Laguna, 16 de septiembre de 2015

Dr. Dña. **Margarita Rivero Álvarez**, con N.I.F. 42.773.834-K profesora Titular de Álgebra adscrita al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

"*Algunos algoritmos de factorización de polinomios con coeficientes enteros.*"

ha sido realizada bajo su dirección por Dña. **Raquel Padrón García**, con N.I.F. 78.643.392-K.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 16 de septiembre de 2015

Agradecimientos

A mi familia por todo.

A mis amigos por los pequeños detalles.

A Margarita por el apoyo, la paciencia y
las luchas hasta el último momento con Hensel.

A Adrián por las muchas horas de más.

Resumen

El objetivo de esta memoria es estudiar algunos algoritmos que permiten obtener la descomposición de un polinomio en una variable con coeficientes enteros como producto de factores irreducibles sobre $\mathbb{Z}$.

En primer lugar se introducen ciertas magnitudes asociadas a los polinomios que permiten acotar el tamaño de los factores. A continuación se estudian el algoritmo de Berlekamp (1970) para la factorización de polinomios con coeficientes en cuerpos finitos y el método de elevación lineal dado por Hensel (1918). Estos algoritmos basados en la aritmética modular son de gran utilidad para la factorización de polinomios enteros.

En cuanto a la factorización de polinomios sobre $\mathbb{Z}$, se estudian los algoritmos de Kronecker (1882) y el de Berlekamp-Zassenhaus (1969). Además se describe el algoritmo dado por Lenstra-Lenstra y Lovász (LLL) (1981), sin entrar en la justificación del mismo ya que sobrepasa los objetivos del trabajo. Todos los algoritmos han sido implementados en lenguaje SageMath.

Palabras Clave Factorización de polinomios enteros, Factorización de polinomios sobre cuerpos finitos, Implementación en SageMath.

Abstract

The purpose of this work is to study some algorithms to obtain the factorization of a polynomial in one variable with integer coefficients as a product of irreducible factors over $\mathbb{Z}$.

First, certain magnitudes associated with the polynomials allows to constrain the size of the factors are introduced. Then we study the Berlekamp's algorithm (1970) for factoring polynomials with coefficients in finite fields and the linear lift method given by Hensel (1918). These algorithms based on modular arithmetic are useful for integer factorization polynomials.

For the factorization of integer polynomials, on the one hand we study the Kronecker's (1882) and the Berlekamp-Zassenhaus (1969) algorithms. On the other hand, we describe the Lenstra-Lentra and Lovász algorithm (LLL) (1981) without showing his justification as it exceeds the objectives of this work. All algorithms are implemented in SageMath.

Keywords *Factorization of integers polynomials, Factorization of polynomials over integers fields, Implementación en SageMath.*

Índice general

Introducción **1**

1. Algunos resultados sobre polinomios **3**
1.1. Definiciones básicas . . . 3
1.2. Magnitudes relacionadas con los polinomios . . . 4
1.3. Cotas de los factores para polinomios enteros . . . 6
1.4. Interpolación polinómica . . . 9
1.5. Diferencias finitas . . . 10
1.6. Polinomios estables . . . 12
1.7. Resultante . . . 12

2. Polinomios con coeficientes en un cuerpo finito. Lema de Hensel **15**
2.1. Cuerpos finitos . . . 15
2.2. Algoritmo de Berlekamp . . . 16
2.3. Lema de Hensel . . . 19

3. Polinomios enteros **25**
3.1. Método de Factorización de Kronecker . . . 25
3.2. El algoritmo de Kronecker-Hausmann . . . 27
3.3. El algoritmo de Berlekamp-Zassenhaus-Cantor . . . 30
3.4. El algoritmo de factorización LLL . . . 32

Conclusiones **35**

A. Algoritmos **37**
A.1. Algoritmo de Berlekamp sobre cuerpos finitos . . . 37
A.2. Algoritmo para el levantamiento lineal . . . 38
A.3. Algoritmo de Kronecker . . . 39
A.4. Algoritmo de Kronecker-Hausmann . . . 40
A.5. Algoritmo de Berlekamp-Zassenhaus-Cantor . . . 41
A.6. Algoritmo de Lenstra, Lenstra y Lovász . . . 42

Bibliografía **45**

Introducción

La obtención de fórmulas para el cálculo de las raíces de un polinomio fue durante mucho tiempo un tema fundamental dentro del Álgebra. El trabajo de Abel (1824), completado por Galois, puso de manifiesto la imposibilidad de obtener una fórmula que implicara sumas, restas, multiplicaciones, divisiones y radicales, de los coeficientes del polinomio, que permitiera calcular dichas raíces para polinomios de grado mayor que 4. Una consecuencia de este trabajo fue el interés que cobró el problema de factorizar polinomios, es decir descomponerlos como producto de factores de menor grado.

Hoy en día este interés se ha incrementado dada la importancia que los polinomios irreducibles, es decir los no factorizables en el sentido antes expuesto, desempeñan en la Teoría de Códigos y su relación con la seguridad en la transferencia de información sensible por medio de la red, y la necesidad de algoritmos cada vez más eficientes motivado por el incremento de la potencia de cálculo de los ordenadores actuales.

Newton en su obra Arithmetica Universalis (1707) dio un método para encontrar factores lineales y cuadráticos de polinomios sobre los enteros. El método de Newton fue extendido por Schubert (1793) para el cálculo de factores de grado n en un número finito de pasos. El método de Schubert fue redescubierto de forma independiente por Kronecker 90 años más tarde. Este método, en la práctica, es poco usado ya que el coste computacional es exponencial respecto al grado del polinomio.

Se obtienen muchos mejores resultados haciendo uso de los métodos de factorización módulo p (p primo), y utilizando las técnicas de elevación de Hensel para llegar a conseguir la factorización sobre los enteros. Este planteamiento mejora los métodos clásicos y funciona bastante bien en la mayoría de los casos, pero en los polinomios más complicados tiene aún un coste de tiempo exponencial.

A. K. Lenstra, H. W. Lenstra y L. Lovász en 1982 aplicaron la teoría de retículos a la factorización de polinomios enteros, obteniendo un algoritmo mucho más eficiente denominado LLL o L^3, cuyo coste es polinomial.

Este trabajo aborda el estudio de algunos algoritmos para la factorización de polinomios en una variable con coeficientes enteros en factores irreducibles sobre $\mathbb{Z}$.

En el capítulo primero se introducen algunos conceptos relativos a polinomios que serán de utilidad posteriormente. Se definen conceptos como la norma, la media, la longitud, altura, etc., y se estudian algunas cotas y relaciones entre dichas medidas.

En el capítulo segundo se estudia el algoritmo dado por Berlekamp (1967) para la factorización de polinomios sobre cuerpos finitos así como el método de la elevación lineal de Hensel.

En el tercer capítulo se estudian los algoritmos de factorización de Kronecker-Hausmann (1937) y el de Berlekamp, Zassenhaus y Cantor (1969) para polinomios sobre los enteros. Se hace además una descripción del algoritmo LLL, si bien no se estudian sus fundamentos ya que exceden los contenidos de este trabajo.

Todos los algoritmos estudiados han sido implementados en SageMath, aportándose el código de cada uno de ellos, lo que puede resultar de utilidad en diversas aplicaciones de los mismos. Además, cada algoritmo se acompaña de ejemplos para facilitar su comprensión.

Capítulo 1

Algunos resultados sobre polinomios

En este capítulo se introducen algunas magnitudes y resultados relacionados con el tamaño de los polinomios en una variable. Se muestran también varias desigualdades cruciales en la factorización de los polinomios relativas al tamaño de los factores de los mismos. Se introducen además los polinomios estables, las diferencias finitas de una sucesión, la interpolación polinómica y la resultante. Todos los resultados expuestos serán de utilidad en los capítulos siguientes.

1.1. Definiciones básicas

Recogemos en primer lugar algunas nociones básicas sobre la factorización.

Definición 1.1.1. *Un elemento $a \neq 0$ de un dominio de integridad A se dice que es* irreducible *si no es una unidad (elemento inversible) y no se puede expresar como producto de dos elementos que no sean unidades.*

Ejemplo 1.1.2. 5 *es irreducible en* $\mathbb{Z}$ *pero* 6 *no.*

Definición 1.1.3. *Un dominio A se denomina* dominio de factorización única *si todo elemento no nulo se puede expresar como producto de una unidad y un número finito de elementos irreducibles y dicha representación es única salvo unidades y el orden de los factores.*

Ejemplo 1.1.4. *Si K es un cuerpo, el anillo de polinomios $K[X]$ es un dominio de factorización única, en particular $\mathbb{Q}[X]$ es uno de ellos.*

Nota 1.1.5. *La aplicación $\varphi : \mathbb{Z}[X] \longrightarrow \mathbb{Z}_q[X]$ que hace corresponder a $F(X) = a_0 + \cdots + a_d X^d$ el polinomio $\varphi(F) = \overline{F}(X) = \overline{a_0} + \cdots + \overline{a_d} X^d$ es un homomorfismo de anillos. Si $\varphi(F) = \varphi(F')$ escribimos $F \equiv F' \bmod q$.*

Usamos la notación $F \equiv G \bmod H$ para indicar que $H \mid F - G$ con $F, G, H \in \mathbb{Z}_q$.

Definición 1.1.6. *Sea A un dominio de factorización única y $F(X) \in \mathbb{A}[X]$ se dice que es primitivo si el máximo común divisor de sus coeficientes es* 1.

Si d es el mcd de los coeficientes de $F(X)$, se le puede asociar un polinomio $F_1(X) = \dfrac{1}{d}F(X) \in \mathbb{Z}[X]$ que es primitivo.

Lema 1.1.7 (Lema de Gauss)**.** *El producto de dos polinomios primitivos es primitivo.*

Corolario 1.1.8. *Sea A un dominio de factorización única y K su cuerpo de fracciones. Si $F(X) \in A[X]$ tiene grado positivo y es irreducible en $A[X]$ entonces $F(X)$ es irreducible en $K[X]$.*

El polinomio $F(X) = 5X + 10$ es irreducible sobre $\mathbb{Q}$ pero no lo es sobre $\mathbb{Z}$ ya que el polinomio $F(X) = 5(X + 2)$, siendo 5 y $x + 2$ no unidades en $\mathbb{Z}[X]$, mientras que 5 si es una unidad de $\mathbb{Q}$.

En esta memoria diremos que un polinomio $F(X) \in \mathbb{Z}[X]$ es irreducible si no se puede expresar como producto de dos polinomios de grado mayor o igual a 1 con coeficientes en $\mathbb{Z}$. Así diremos que el polinomio $F(X) = 5X + 10$ es irreducible sobre $\mathbb{Z}$.

1.2. Magnitudes relacionadas con los polinomios

Definición 1.2.1. *Sea $F(X) = \sum_{j=0}^{d} a_j X^j \in \mathbb{C}[X]$. La* norma cuadrática *del polinomio F es*

$$\|F\| = \sqrt{\sum_{j=0}^{d} |a_j|^2},$$

y la norma *de F es $|F| = \max_{|z|=1} |F(z)|$.*

Lema 1.2.2. *Si $F \in \mathbb{C}[X]$ y $\alpha \in \mathbb{C}$, entonces*

$$\|(X - \alpha)F(X)\| = \|(\overline{\alpha}X - 1)F(X)\|$$

y

$$|(X - \alpha)F(X)| = |(\overline{\alpha}X - 1)F(X)|.$$

Demostración. Se supone que $F(X) = \sum_{j=0}^{d} c_j X^j$. Desarrollando el cuadrado del lado izquierdo de la primera igualdad se obtiene

$$\sum_{j=0}^{d+1} (c_{j-1} - \alpha c_j)(\overline{c}_{j-1} - \overline{\alpha}\,\overline{c}_j) = (1 + \|\alpha\|^2)\|F\|^2 - \sum_{j=0}^{d} (\alpha c_j \overline{c}_{j-1} + \overline{\alpha}\,\overline{c}_j c_{j-1}),$$

donde $c_{-1} = c_{d+1} = 0$. Y el desarrollo del lado derecho de la igualdad da el mismo resultado.

La segunda relación se sigue de la siguiente identidad

$$|\overline{\alpha}z - 1| = |z - \alpha|, \text{ para todo } z \in \mathbb{C}, |z| = 1. \tag{1.1}$$

Para verificar (1.1) se toma $z = e^{i\theta}$. Se tiene

$$|\overline{\alpha}z - 1|^2 = |\alpha|^2 - \overline{\alpha}e^{i\theta} + 1 \text{ y } |z - \alpha|^2 = |\alpha|^2 - \overline{\alpha}e^{i\theta} + 1. \qquad \square$$

Definición 1.2.3. *Sea $F \in \mathbb{C}[X]$. Se supone que*

$$F(X) = a_d X^d + a_{d-1}X^{d-1} + \cdots + a_0 = a_d(X - z_1)\dots(X - z_d),$$

con $a_d \neq 0$. La medida de F *se define por la fórmula*

$$M(F) = |a_d| \prod_{j=1}^{d} \text{máx}\{1, |z_j|\}.$$

Proposición 1.2.4. *Si $F(X) = a_d X^d + a_{d-1}X^{d-1} + \cdots + a_0 \in \mathbb{C}[X] \setminus \mathbb{C}$, entonces la medida de F satisface la desigualdad*

$$M(F)^2 + |a_0 a_d|^2 M(F)^{-2} \leq \|F\|^2.$$

Demostración. Sean $z_1, \dots, z_k$ las raíces de F que están por fuera del disco unidad. Entonces la medida de F es

$$M(F) = |a_d||z_1 \dots z_k|.$$

Se considera el polinomio

$$R(X) = a_d \prod_{j=1}^{k} (\overline{z}_j X - 1) \cdot \prod_{j=k+1}^{d} (X - z_j) = b_d X^d + \cdots + b_0.$$

Aplicando k veces el Lema (1.2.2) se obtiene $\|F\| = \|R\|$. Por lo que

$$\|R\|^2 \geq |b_d|^2 + |b_0|^2 = M(F)^2 + |a_d a_0|^2 M(F)^{-2},$$

se tiene la desigualdad deseada. $\square$

Corolario 1.2.5. *Si el polinomio F no es un monomio, entonces $M(F) < \|F\|$.*

Definición 1.2.6. *Si $F(X) = a_d X^d + \cdots + a_0$ es un polinomio con coeficientes complejos, se define su* longitud *mediante la fórmula*

$$L(F) = \sum_{k=0}^{d} |a_k|$$

y se define la altura *de F como*

$$H(F) = \text{máx}\{|a_0|, |a_1|, \dots, |a_d|\}.$$

Los siguientes resultados aportan algunas desigualdades que relacionan las medidas asociadas anteriormente a los polinomios sobre $\mathbb{C}$.

Lema 1.2.7. *Si F es un polinomio no constante con coeficientes complejos, entonces*

$$|a_j| \leq \binom{d}{j} M(F), \quad H(F) \leq \binom{d}{\lfloor d/2 \rfloor} M(F),$$

y

$$L(F) \leq 2^d M(F).$$

Demostración. Sean $z_1, \ldots, z_d$ las raíces del polinomio $F \in \mathbb{C}[X]$, contadas con sus multiplicidades. La fórmula $F(X) = a_d(X - z_1) \ldots (X - z_d)$ muestra que

$$\frac{a_{d-j}}{a_d} = (-1)^{d-j} \sum_{i_1 < \ldots < i_j} z_{i_1} \ldots z_{i_j} \quad \text{para} \quad j = 1, 2, \ldots, d.$$

Por lo tanto $|a_j| \leq \binom{d}{j} M(F)$, para $j = 1, 2, \ldots, d$.

Las otras dos desigualdades se siguen de $\binom{d}{j} \leq \binom{d}{\lfloor d/2 \rfloor}$, para todo j, y $\sum_{j=0}^{d} \binom{d}{j} = 2^d$ respectivamente. □

1.3. Cotas de los factores para polinomios enteros

Será de utilidad en los capítulos siguientes conocer algunas cotas relativas a los factores de los polinomios con coeficientes en $\mathbb{Z}$.

Lema 1.3.1. *Si F es un polinomio de grado d sobre $\mathbb{Z}$, entonces*

$$H(F) \leq L(F) \leq \sqrt{d+1}\|F\| \leq (d+1)H(F).$$

Demostración. La desigualdad $L(F) \leq \sqrt{d+1}\|F\|$ se sigue de la desigualdad de Schwartz:

$$\left(\sum_{j=0}^{d} |a_j| \right)^2 \leq \left(\sum_{j=0}^{d} |a_j|^2 \right) \cdot \left(\sum_{j=0}^{d} 1 \right).$$

Mientras que las otras dos desigualdades se siguen de la definición. □

Dado un polinomio no constante $F \in \mathbb{Z}[X]$, se pueden calcular algunas cotas de los factores de F.

Teorema 1.3.2. *Sea $F \in \mathbb{Z}[X]$ un polinomio no constante que admite la factorización*

$$F = G_1 \cdots G_m, \quad G_1, \ldots, G_m \in \mathbb{Z}[X]$$

Entonces

$$\prod_{j=1}^{m} L(G_j) \leq 2^D M(F), \quad \text{donde} \quad D = \sum_{j=1}^{m} \mathrm{gr}(G_j),$$

$$\text{y} \prod_{j=1}^{m} L(G_j) \leq 2^D \|F\|.$$

Demostración. Se tiene

$$\prod_{j=1}^{m} L(G_j) \leq 2^D \prod_{j=1}^{m} M(G_j).$$

Entonces las desigualdades $M(G_1) \cdots M(G_m) \leq M(F) \leq \|F\|$ conducen a la conclusión. □

Proposición 1.3.3. *Sea $F \in \mathbb{Z}[X]$ un polinomio no constante y sea G un divisor de F en $\mathbb{Z}[X]$. Supóngase que $G(X) = \sum_{i=0}^{m} b_i X^i$. Entonces*

$$|b_i| \leq \binom{m}{i} \cdot \|F\| \quad \text{para} \quad i = 0, 1, \ldots, m.$$

Demostración. Sean $z_1, \ldots, z_s$ las raíces de G y sea a el coeficiente principal de F, entonces

$$|b_i| \leq \binom{m}{i} |z_1 \ldots z_s| \cdot |b_m| \leq \binom{m}{i} |z_1 \ldots z_s| \cdot |a| \leq \binom{m}{i} \|F\|.$$ □

Proposición 1.3.4. *Si $F \in \mathbb{Z}[X]$ y G es un divisor de grado m de F en $\mathbb{Z}[X]$, entonces*

i. $M(G) \leq M(F)$,

ii. $L(G) \leq 2^m M(F)$,

iii. $L(G) \leq 2^m \|F\|$.

Demostración. Puesto que todas las raíces de G son raíces de F y el coeficiente principal de G divide a F, se tiene que $M(G) \leq M(F)$.

Por el Lema 1.2.7 se sabe que $L(G) \leq 2^m M(G)$, por tanto se tiene **ii.**

La última desigualdad se sigue de **ii.** y el Corolario 1.2.5. □

Un paso clave en los algoritmos de factorización de polinomios enteros es la determinación de un primo conveniente p y un entero positivo n tal que los valores absolutos de los coeficientes de todos los posibles divisores del polinomio dado estén acotados por p^n. En este sentido, si

$$F(X) = a_d X^d + a_{d-1} X^{d-1} + \cdots + a_0 \in \mathbb{Z}[X]$$

y

$$G(X) = b_m X^m + b_{m-1} X^{m-1} + \cdots + b_m \in \mathbb{Z}[X]$$

un divisor de F en $\mathbb{Z}[X]$, por la Proposición 1.3.2 y el Lema 1.3.1 se tiene que

$$H(G) \leq 2^m M(F) \qquad \text{y} \qquad H(G) \leq 2^m \sqrt{d+1} H(F),$$

Tomando p de forma que $p^n \geq 2H(G)$ podemos considerar que los coeficientes del divisor G están en el intervalo

$$\left[-\frac{p^n-1}{2}, \frac{p^n-1}{2}\right].$$

Se denota por $B(F)$ el menor entero positivo tal que $B(F) \geq 2H(G)$ para cualquier divisor G de F, $\mathrm{gr}(G) \leq \mathrm{gr}(F)/2$. Se tiene entonces que

Lema 1.3.5. *Si $F \in \mathbb{Z}[X]$, entonces*

$$B(F) < 2^{1+d/2} \cdot \sqrt{d+1} H(F) \qquad \text{y} \qquad B(F) < 2^{1+d/2} \cdot M(F).$$

Demostración. Sea G un divisor del polinomio F en $\mathbb{Z}[X]$. Se puede suponer $\mathrm{gr}(G) \leq d/2$. Luego

$$2H(G) \leq 2 \cdot 2^{d/2} \sqrt{d+1} H(F) = 2^{1+d/2} \cdot \sqrt{d+1} H(F).$$

Por tanto $B(F) \leq 2^{1+d/2} \cdot \sqrt{d+1} H(F)$.

La otra estimación se obtiene de $H(G) \leq 2^{\mathrm{gr}(G)} M(F)$. □

Proposición 1.3.6. *Si $F(X) = \sum_{i=0}^{d} a_i X^i \in \mathbb{Z}[X]$, entonces*

$$B(F) < 2 \binom{\lfloor d/2 \rfloor}{\lfloor d/4 \rfloor} \cdot M(F) \qquad \text{y} \qquad B(F) < 2 \binom{\lfloor d/2 \rfloor}{\lfloor d/4 \rfloor} \cdot \|F\|.$$

Demostración. Sea $F(X) = \sum_{i=0}^{d} a_i X^i$ y sea $G(X) = \sum_{j=0}^{m} b_j X^j$ un divisor de F en $\mathbb{Z}[X]$. Por el Lema 1.2.7 se tiene

$$H(G) \leq \binom{m}{\lfloor m/2 \rfloor} \cdot M(F).$$

La primera desigualdad se sigue de la siguiente relación

$$\operatorname*{máx}_{m \leq d/2} \binom{m}{\lfloor m/2 \rfloor} = \binom{\lfloor d/2 \rfloor}{\lfloor d/4 \rfloor}.$$

La segunda desigualdad se obtiene de la primera por la desigualdad de Landau (Corolario 1.2.5). □

1.4. Interpolación polinómica

Definición 1.4.1. *Sea A un anillo y sea $\Phi : A \longrightarrow A$ una función. Se supone que la función $\Phi(x) = \Phi(x; a_0, a_1, \ldots, a_n)$ depende de los $n+1$ parámetros $a_0, a_1, \ldots, a_n \in A$. Se tiene un* problema de interpolación *para Φ si los parámetros a_i son tales que, para $n+1$ parejas dadas $(x_i, y_i) \in A^2$, $i = 0, 1, \ldots, n$, $x_i \neq x_j$ para $i \neq j$, se tiene*

$$\Phi(x_i; a_0, a_1, \ldots, a_n) = y_i, \quad i = 0, 1, \ldots, n.$$

Las parejas (x_i, y_i) se denominan puntos base.

Si Φ es lineal con respecto a los parámetros a_i, es decir, existen funciones $\Phi_0, \Phi_1, \ldots, \Phi_n$ tales que

$$\Phi(x; a_0, a_1, \ldots, a_n) = a_0\Phi_0(x) + a_1\Phi_1(x) + \cdots + a_n\Phi_n(x), \tag{1.2}$$

para todo $x \in A$, entonces se trata un problema de interpolación lineal.

Si las funciones Φ_i en (1.2) son funciones polinomiales, entonces se tiene un problema de interpolación polinómica.

Se supone que A es un dominio de integridad y sean K el cuerpo de fracciones y $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n\}$ una base de un K-espacio vectorial E_n de polinomios en $K[X]$ de grado al menos n. Si $F \in E_n$, existen $a_0, a_1, \ldots, a_n \in K$ tales que

$$F = a_0\mathbf{b}_0 + a_1\mathbf{b}_1 + a_n\mathbf{b}_n.$$

Obsérvese que la representación corresponde a la interpolación polinómica de la función polinómica asociada a F.

Teorema 1.4.2. *Sean (x_0, y_0), $(x_1, y_1), \ldots, (x_n, y_n) \in K^2$, con x_i distintos. Entonces existe un único polinomio $F \in E_n[X]$ de grado n tal que*

$$F(x_i) = y_i, \quad \text{para} \quad i = 0, 1, \ldots, n.$$

Demostración. Consideremos

$$\mathbf{b}_i(X) = \frac{W(X)}{X - x_i}, \quad \text{donde} \quad W(X) = \prod_{i=0}^{n}(X - x_i).$$

Si

$$\sum_{i=0}^{n} \lambda_i \mathbf{b}_i(X) = 0 \quad \text{para todo} \quad x \in K,$$

entonces $\lambda_i\mathbf{b}_i(x_i) = 0$, por tanto $\lambda_i = 0$ para $i = 0, 1, \ldots, n$. Con lo cual los polinomios

$$\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n$$

forman una base de E_n. Por otra parte los coeficientes a_i, $< 0 \leq i \leq n$, vienen dados por

$$a_i = y_i \frac{1}{\mathbf{b}_i(x_i)}.$$

Si hubiera dos soluciones distintas F_1 y F_2 entonces $F_1 - F_2$ sería un polinomio de grado n a lo sumo con $n+1$ raíces, lo que es absurdo. $\square$

Interpolación de Lagrange

El teorema anterior da el siguiente desarrollo conocido como la fórmula de interpolación de Lagrange

$$F(X) = \sum_{i=0}^{n} \frac{F(x_i)}{\mathbf{b}_i(x_i)} \cdot b_i(X),$$

donde $\mathbf{b}_i(X) = (X - x_0)\dots(X - x_{i-1})(X - x_{i+1})\dots(X - x_n)$.

Interpolación de Newton

Se consideran $x_0, x_1, \dots, x_n \in K$ distintos y sea

$$\begin{cases} \mathbf{b}_0 = & 1 \\ \mathbf{b}_1(X) = & X - x_0 \\ \mathbf{b}_2(X) = & (X - x_0)(X - x_1) \\ & \dots\dots \\ \mathbf{b}_n = & (X - x_0)(X - x_1)\dots(X - x_{n-1}) \end{cases}$$

Entonces $\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n$ es una base de E_n. Por tanto, dado $F(X) \in A[X]$ se puede escribir como

$$\begin{cases} F(X) = & \displaystyle\sum_{i=0}^{n} \Delta_i(y_i; x_o, \dots, x_i) \prod_{j=0}^{i+1} (X - x_j) \\ = & \Delta_0 + \Delta_1(X - x_0) + \dots + \Delta_n(X - x_0)\dots(X - x_{n-1}). \end{cases} \tag{1.3}$$

donde se sabe que los coeficientes Δ_i se calculan por recursividad

$$\Delta_{i+k}(y; x_i, \dots, x_{i+k}) = \frac{\Delta_{i+k-1}(y; x_{i+1}, \dots, x_{i+k}) - \Delta_{i+k-1}(y; x_i, \dots, x_{i+k-1})}{x_{i+k} - x_i}$$

El desarrollo (1.3) es conocido como la fórmula de interpolación de Newton.

1.5. Diferencias finitas

Definición 1.5.1. *Sea A un anillo y sean $y_0, \dots, y_n \in A$. Se establecen*

$$\begin{aligned} \Delta_1(y_0, \dots, y_n) &= \{y_1 - y_0, y_2 - y_1, \dots, y_n - y_{n-1}\} \\ &= \{y_0^{(1)}, y_1^{(1)}, \dots, y_{n-1}^{(1)}\}, \\ \Delta_2(y_0, \dots, y_n) &= \Delta_1(y_0^{(1)}, \dots, y_{n-1}^{(1)}) \\ &= \{y_1^{(1)} - y_0^{(1)}, y_2^{(1)} - y_1^{(1)}, \dots, y_{n-1}^{(1)} - y_{n-2}^{(1)}\} \\ &= \{y_0^{(2)}, y_1^{(2)}, \dots, y_{n-2}^{(2)}\}, \\ \dots & \qquad \dots \\ \Delta_i(y_0, \dots, y_n) &= \Delta_1(y_0^{(i-1)}, \dots, y_{n-i+1}^{(i-1)}). \end{aligned}$$

La sucesión $\Delta_i(y_0, \ldots, y_n)$ *se denomina* diferencia de orden i *de la sucesión* $y_0, \ldots, y_n$ *y* tabla de diferencias *a*

$$\Delta_1(y_0, \ldots, y_n),\ \Delta_2(y_0, \ldots, y_n), \ldots,\ \Delta_i(y_0, \ldots, y_n), \ldots.$$

Lema 1.5.2. *Sea A un dominio de integridad y F un polinomio en $A[X]$ de grado $d \leq n$. Se considera la sucesión $x_0, x_1 = x_0 + a, \ldots, x_n = x_0 + na$ de elementos distintos de A y sea $y_i = F(x_i)$. Entonces $\Delta_d(y_d, \ldots, y_n)$ es una constante y d es el menor entero con esta propiedad.*

Demostración. Sea d' el menor entero positivo tal que $\Delta_{d'}(F)$ es una constante. Por inducción sobre el grado de F veamos que $d' \leq d$. Para $d = 1$ es obvio.

Si $gr(F) = d > 1$, entonces el primer orden diferencial $\Delta_1(y_0, \ldots, y_n)$ es la sucesión asociada a $F(X + a) - F(X)$, que es un polinomio de grado $d - 1$. Entonces, por recurrencia

$$\Delta_{d-1}(\Delta_1(y_0, \ldots, y_n)) = \Delta_d(F) \quad \text{esconstante.}$$

por lo tanto $d' \leq d$.

Recíprocamente, se supone que $\Delta_d(F)$ es una constante. Entonces, por la interpolación de Newton (1.3), el polinomio F queda unívocamente determinado por la sucesión $(\Delta_i(y_o, \ldots, y_n))_{i \geq 1})$. Puesto que

$$\begin{aligned} F(X) = F(x_0) &+ \Delta_1(F; x_0, x_1)(X - x_0) + \ldots \\ &+ \Delta_k(F; x_0, \ldots, x_n)(X - x_0) \ldots (X - x_{k-1}) \\ &+ \cdots + \Delta_{d'}(F; x_0, \ldots, x_n)(X - x_0) \ldots (X - x_{d'-1}), \end{aligned}$$

se sigue que $d \leq d'$ y por tanto $d = d'$. □

Definición 1.5.3. *Si $a = (a_0, a_1, \ldots, a_n)$, entonces el* orden diferencial *de la sucesión es el menor entero positivo s tal que los miembros de orden s de la tabla de diferencias de a son todos iguales a una constante distinta de cero.*

Lema 1.5.4. *Sean $m_0, m_1 = m_0 + a, \ldots, m_n = m_0 + na \in A$ y $a_0, a_1, \ldots, a_n \in A$ donde A es un dominio de integridad. Entonces existe un polinomio $Q \in A[X]$ de grado $d \leq n$ tal que*

$$Q(m_i) = a_i \text{ para todo } i = 0, 1, \ldots, n$$

si y sólo si el orden diferencial de la sucesión $(a_0, a_1, \ldots, a_n)$ es d.

Demostración. Por la interpolación de Newton, un polinomio Q tal que $Q(m_i) = a_i$ para todo

$i = 0, 1, \ldots, n$ está representado únicamente por

$$Q(X) = Q(0) + \Delta_1(X - m_0) + \Delta_2(X - m_0)(X - m_0 - a) +$$

$$\cdots + \Delta_n(X - m_o)(X - m_0 - a)\ldots(X - m_0 - (n-1)a).$$

Si $\Delta_d \neq 0$ y $\Delta_{d+1} = \cdots = \Delta_n = 0$, entonces $\mathrm{gr}(Q) = d$.

El recíproco es el Lema 1.5.2. □

1.6. Polinomios estables

Definición 1.6.1. *$F \in \mathbb{C}[X]$ se dice que es un* polinomio estable *si cualquier raíz de F tiene una parte real negativa.*

Nota 1.6.2. *Todos los coeficientes de un polinomio estable F con coeficientes reales tienen el mismo signo.*

Los polinomios estables son importantes en la localización de las raíces del polinomio y la factorización de polinomios enteros. A. Hurwitz en [14] da un criterio de estabilidad para polinomios con coeficientes reales que simplifica la comprobación.

Teorema 1.6.3 (A. Hurwitz)**.** *Sea $F(X) = a_d X^d + \cdots + a_1 X + a_0 \in \mathbb{R}[X]$, $a_0 > 0$ y*

$$\Lambda_k = \begin{vmatrix} a_1 & a_0 & 0 & \ldots & 0 \\ a_3 & a_2 & a_1 & \ldots & 0 \\ a_5 & a_4 & a_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{2k-1} & a_{2k-2} & a_{2k-3} & \ldots & a_k \end{vmatrix},$$

para $k = 1, 2, \ldots, d$, con $a_j = 0$ si $j > d$. Entonces el polinomio F es estable si y sólo si

$$\Lambda_1 > 0, \Lambda_2 > 0, \ldots, \Lambda_d > 0.$$

Nota 1.6.4. *Para convertir cualquier polinomio entero en un polinomio estable, se considera $F \in \mathbb{Z}[X]$ no estable y sea $M \in (0, \infty)$ tal que*

$$|F(z)| > 0 \text{ si } |z| \geq M.$$

Entonces el polinomio $S(X) = F(X + |M| + 1)$ es estable.

Los coeficientes del polinomio S pueden llegar a ser muy grandes, por lo que es recomendable utilizar una cota $M = \max |a_k/a_0|$, $k = 1, 2, \ldots, d$.

1.7. Resultante

La resultante de dos polinomios en una variable F y G con coeficientes en un dominio de integridad A, que se denominará $\mathrm{Res}(F, G)$, fue introducida tratando de encontrar condiciones necesarias y

suficientes para que F y G tengan raíces comunes en una extensión del dominio A.

Definición 1.7.1. *Sean* $F(X) = \sum_{i=0}^{m} a_i X^i, \quad G(X) = \sum_{i=0}^{n} b_i X^i$ *polinomios no constantes en una variable con coeficientes en* $\mathbb{Q}$. *La* matriz de Sylvester *asociada a ellos es la matriz* $(m+n)\times(m+n)$ *siguiente:*

$$S(F,G) = \begin{pmatrix} a_0 & a_1 & \dots & \dots & a_{m-1} & a_m & 0 & \dots & 0 & 0 \\ 0 & a_0 & a_1 & \dots & \dots & a_{m-1} & a_m & \dots & 0 & 0 \\ 0 & 0 & a_0 & a_1 & \dots & \dots & a_{m-1} & a_m & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & a_0 & a_1 & \dots & a_{m-1} & a_m & 0 \\ 0 & 0 & \dots & \dots & 0 & a_0 & a_1 & \dots & a_{m-1} & a_m \\ b_0 & b_1 & \dots & \dots & b_{n-1} & b_n & 0 & \dots & 0 & 0 \\ 0 & b_0 & b_1 & \dots & \dots & b_{n-1} & b_n & \dots & 0 & 0 \\ 0 & 0 & b_0 & b_1 & \dots & \dots & b_{n-1} & b_n & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & b_0 & b_1 & \dots & b_{n-1} & b_n & 0 \\ 0 & 0 & \dots & \dots & 0 & b_0 & b_1 & \dots & b_{n-1} & b_n \end{pmatrix}.$$

La resultante de F y G *es el determinante de la matriz de Sylvester*

$$\mathrm{Res}(F,G) = det(S(F,G)).$$

Las propiedades básicas de esta resultante son las siguientes:

Proposición 1.7.2.

1. $\mathrm{Res}(F,G) = 0$ *si y sólo si* $F(x)$ *y* $G(x)$ *tienen un factor común en* $\mathbb{Q}[x]$.
2. $\mathrm{Res}(F,G)$ *es un polinomio en* $a_0, ..., a_m, b_0, ..., b_n$ *con coeficientes enteros.*
3. *Existen polinomios* $A, B \in \mathbb{Q}[x]$ *tales que* $AF + BG = \mathrm{Res}(F,G)$.

Demostración. La demostración puede verse en [Cox, Little y O'Shea [5], apartado 3.5] □

Definición 1.7.3. *Sea* $F(X) = \sum_{i=0}^{m} a_i X^i$ *un polinomio no constante con coeficientes en* $\mathbb{Q}$. *Se denomina* discriminante de F, *y se denota por* $D(F)$, *a*

$$D(F) = (-1)^{\frac{m(m-1)}{2}} \frac{1}{a_m} Res(F, F')$$

siendo F' *la derivada de* F.

Se demuestra que si $\alpha_1, ..., \alpha_m$ son las raíces de F en el cuerpo de los números complejos, contando cada raíz tantas veces como indica su multiplicidad, entonces

$$D(F) = a_m^{(2m-2)} \prod_{i<j} (\alpha_i - \alpha_j)$$

y por tanto el discriminante se anula si y solo si F tiene raíces múltiples.

Capítulo 2

Polinomios con coeficientes en un cuerpo finito. Lema de Hensel

En este capítulo se estudia el algoritmo de factorización de polinomios en una variable sobre cuerpos finitos dado por E. R. Berlekamp ([2], [3]) y el lema de Hensel en la versión constructiva de las elevaciones lineales.

2.1. Cuerpos finitos

Los cuerpos finitos son importantes no sólo por su lugar clave en la teoría de cuerpos, sino por sus fructíferas aplicaciones a campos como las comunicaciones electrónicas, combinatoria, teoría de codificación y criptografía. Algunos algoritmos algebraicos recurren a cálculos realizados sobre cuerpos finitos, como ocurre por ejemplo con la factorización de polinomios con coeficientes enteros.

Recogemos algunos resultados básicos relativos a los cuerpos finitos que pueden encontrarse por ejemplo en [8].

Definición 2.1.1. *Un cuerpo $\mathbb{F}$ es un cuerpo finito si tiene un número finito de elementos.*

Ejemplo 2.1.2. *Si p es un número primo el anillo cociente $\frac{\mathbb{Z}}{p\mathbb{Z}}$ es un cuerpo finito con p elementos que denotaremos por $\mathbb{F}_p$.*

Definición 2.1.3. *La característica de un cuerpo finito $\mathbb{F}$ es el menor entero positivo m tal que $m \cdot 1 = 0$.*

La característica es siempre un número primo $p = car(\mathbb{F})$, y el número de elementos de $\mathbb{F}$ es $q = p^n$ para algún n entero positivo.

Proposición 2.1.4. *Todos los cuerpos con $q = p^n$ elementos son isomorfos al cuerpo de descomposición del polinomio $F(X) = X^q - X$ sobre $\mathbb{F}_p$. Por tanto podemos decir que existe un único cuerpo, salvo isomorfismo, con q elementos y los denotaremos por $\mathbb{F}_q$.*

2.2. Algoritmo de Berlekamp

Este algoritmo de factorización sobre cuerpos finitos se basa esencialmente en métodos de álgebra lineal.

Teorema 2.2.1. *Si $F \in \mathbb{F}_q[X]$ es un polinomio mónico y $H \in \mathbb{F}_q[X]$ es tal que $H^q \equiv H \bmod F$, entonces*

$$F(X) = \prod_{c \in \mathbb{F}_q} \operatorname{mcd}(F(X), H(X) - c). \tag{2.1}$$

Demostración. Dado que los polinomios $H(X) - c$, con $c \in \mathbb{F}_q$ son primos dos a dos también lo son los $\operatorname{mcd}(F(X), H(X) - c)$ y todos dividen a F. Por tanto el segundo miembro de la igualdad divide a F.

Por otra parte $H^q(X) - H = \prod_{c \in \mathbb{F}_q} (H(X) - c) = F \cdot G$, con $G \in \mathbb{F}_q[X]$.

De aquí se obtiene que $F(X)$ divide al segundo miembro. Como ambos lados de la igualdad son mónicos, se tiene la igualdad. □

Nota 2.2.2. *El teorema anterior da una factorización de $F(X)$ pero no tiene que ser en factores irreducibles pues el* $\operatorname{mcd}(F(X), H(X) - c)$ *puede ser reducible.*

Más aún, si $H \equiv c \bmod F$ para algún $c \in \mathbb{F}_q$ el teorema anterior da una factorización trivial de F.

Definición 2.2.3. *Un polinomio $H \in \mathbb{F}_q[X]$ se dice que es un* polinomio reductor de F *si la factorización dada en* (2.1) *no es trivial.*

Si un polinomio $H \in \mathbb{F}_q[X]$ verifica que $H^q \equiv H \bmod F$ y que $0 < \operatorname{gr}(H) < \operatorname{gr}(F)$, la factorización (2.1) no es trivial y H es un polinomio reductor de F.

Se supone ahora que $F(X) \in \mathbb{F}_q[X]$ es mónico y no tiene factores irreducibles repetidos, es decir $F = F_1 \cdots F_k, F_i \in \mathbb{F}_q[X]$ irreducibles, mónicos y distintos. Esto siempre se puede conseguir ya que bastaría tomar $F_1 = \dfrac{F}{lc(F)}$ para tener un polinomio mónico y luego $F_2 = \dfrac{F_1}{\operatorname{mcd}(F_1, F_1')}$.

Si $\operatorname{gr}(F) = d$ entonces $A = \dfrac{\mathbb{F}_q[X]}{(F)}$ es un $\mathbb{F}_q$-espacio vectorial de dimensión d siendo una base que se denota por $\mathcal{B}$ la formada por las clases de los $\left\{X^i\right\}_{i=0,\ldots,d-1}$.

Definición 2.2.4. *Se llamará* matriz de Berlekamp asociada a F, *a la matriz*

$$B_q(F) = (b_{ij})_{\substack{1 \le i \le d-1 \\ 1 \le j \le d-1}}$$

siendo, $\displaystyle\sum_{j=0}^{d-1} b_{ij} X^j$ *el resto de dividir* $(X^i)^q$ *entre* F, $\forall i, 0 \le i \le d-1$

En estas condiciones se tiene que

Teorema 2.2.5. *La aplicación lineal $\varphi : A \longrightarrow A$ que tiene como matriz asociada respecto de $\mathcal{B}$ la matriz $B_q^\tau(F) - I \in M_d(\mathbb{F}_q)$ tiene rango $d-k$, siendo k el número de factores irreducibles distintos de F. Además*

$$\mathrm{Ker}(\varphi) = \left\{\overline{H} \in A \diagup H^q \equiv H \bmod (F) \wedge \mathrm{gr}(H) < \mathrm{gr}(F)\right\}$$

Demostración. Sea $(c_1, \ldots, c_k) \in \mathbb{F}_q^k$ una k-upla cualquiera de elementos de $\mathbb{F}_q$. Por el teorema chino de los restos para polinomios (véase [18]) existe un único $H \in \mathbb{F}_q[X]$ tal que

$$H(X) \equiv c_i \bmod F_i(X),\ 1 \leq i \leq k$$

con $\mathrm{gr}(H) < \mathrm{gr}(F)$. Este polinomio $H(X)$ verifica que

$$H(X)^q \equiv c_i^q = c_i \equiv H(X) \bmod F_i(X),\ 1 \leq i \leq k$$

y por tanto

$$H(X)^q \equiv H(X) \bmod F(X),\ \mathrm{gr}(H) < \mathrm{gr}(F) \tag{2.2}$$

Por otra parte, si H es solución de (2.2), por el teorema anterior, cada factor irreducible de F divide a algún polinomio $H(X) - c$, $c \in \mathbb{F}_q$ y por tanto $H(X) \equiv c_i \bmod F_i, 1 \leq i \leq k$, para alguna k-upla $(c_1, \ldots, c_k) \in \mathbb{F}_q^k$.

Al poder formar q^k-uplas de elementos de $\mathbb{F}_q$ se tiene que la ecuación (2.2) tiene q^k soluciones.

Por otra parte, $\varphi(\overline{X^i}) = (\overline{X^i})^q - \overline{X^i}$ y de aquí que $\forall\ \overline{H} \in A \quad \varphi(\overline{H}) = H^q - H$ y

$$\mathrm{Ker}(\varphi) = \{H^q \notin A \diagup H^q \equiv H \bmod (F) \wedge \mathrm{gr}(H) < \mathrm{gr}(F)\}$$

es decir, las soluciones de (2.2).

Al ser $\mathrm{Ker}(\varphi)$ un espacio vectorial con q^k elementos sobre un cuerpo de q elementos se tiene que $\dim(\mathrm{Ker}(\varphi)) = k$ y por tanto $\dim(\mathrm{Im}(\varphi)) = rang(B_q^\tau - I) = d - k$. □

Nota 2.2.6. *Si $k = 1$, es decir, F es irreducible, entonces* $\dim(\mathrm{Ker}(\varphi)) = 1$ *y dado que $H_1(X) = 1 \in \mathrm{Ker}(\varphi)$ siempre, se tendría que el* $\mathrm{Ker}(\varphi)$ *estará formado por las clases de los polinomios constantes.*

Si $k \geq 2$ existirán polinomios $H_2, \ldots, H_k$ tales que $0 < \mathrm{gr}(H_i) < \mathrm{gr}(F)$ y tales que $H_i \equiv H \bmod F$. Es decir son polinomios reductores de F.

Algoritmo de Berlekamp

- Sea $F \in \mathbb{F}_q[X]$ mónico y libre de cuadrados con $d := \mathrm{gr}(F)$.
- Se calcula $B_q(F)$, después se halla $rang(B_q^\tau(d) - I) = r$. Así el número de factores irreducibles de F es $d - r = k$. Si $r = d - 1$ entonces F es irreducible.

- Si $r < d-1$ se calcula una solución $(a_0, \ldots, a_d) \in \mathbb{F}_q^d$ tal que

$$(B^\tau - I)\begin{pmatrix} a_0 \\ \vdots \\ a_{d-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{y} \quad (a_0, \ldots, a_d) \neq (c, 0 \ldots, 0), \forall\, c \in \mathbb{F}_q$$

  Se tiene entonces $H_2(X) = a_0 + a_1 X + \cdots + a_{d-1}X^{d-1}$ que es un polinomio reductor de F y

$$F(X) = \prod_{c \in \mathbb{F}_q} \operatorname{mcd}(F(X), H_2(X) - c)$$

  es una factorización no trivial de F.

- Si se obtienen de esta forma los k factores, se tiene la factorización en polinomios irreducibles de F.

- En caso contrario, se factoriza por el mismo método cada uno de los factores de F que han sido obtenidos hasta llegar a la factorización en irreducibles.

Ejemplo 2.2.7. *Sea el polinomio* $F(X) = X^4 + 2X^3 + 2X^2 + X + 2 \in \mathbb{F}_3[X]$.

La derivada de F *es* $F'(X) = X^3 + X + 1$, *entonces el* $\operatorname{mcd}(F, F') = 1$ *con lo que* F *es libre de cuadrados. Se calculan ahora los restos de dividir* X^{3i} *entre* F *para* $i = 0, 1, 2, 3$ *y resulta*

$$\begin{array}{ll} X^0 & \equiv 1, \\ X^3 & \equiv X^3, \\ X^6 & \equiv 2X^3 + X^2 + 2, \\ X^9 & \equiv 2X^3 + 2X. \end{array}$$

La matriz de Berlekamp que se obtiene es:

$$\mathcal{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 2 \\ 0 & 2 & 0 & 2 \end{bmatrix} \quad y \quad \mathcal{B} - I_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 2 & 0 & 0 & 2 \\ 0 & 2 & 0 & 1 \end{bmatrix}$$

que tiene rango 2, *con lo cual el núcleo de* $\mathcal{B} - I_4$ *tiene dimensión* 2 *y una base del núcleo de la aplicación* φ *es*

$$\{\omega_1, \omega_2\} = \{(1, 0, 0, 0), (0, 1, 0, 2)\}.$$

Se escoge $G(X) = 2X^3 + X$ *el polinomio que le corresponde a* ω_2 *y se halla* $g = \operatorname{mcd}(F, G + i)$ *con* $i \in 0, 1, 2$.

- *Para* $i = 0$, $\operatorname{mcd}(F, G) = \operatorname{mcd}(X^4 + X^3 + 2X^2 + X + 2, 2X^3 + X) = X + 1$, *entonces* $X + 1$ *divide a* F. *Se cambia*

$$F(X) = \frac{F}{\operatorname{mcd}(F, G)} = X^3 + 2X + 2.$$

- *Para* $i = 1$, $\operatorname{mcd}(F, G + 1) = \operatorname{mcd}(X^3 + 2X + 2, 2X^3 + X + 1) = X^3 + 2X + 2$, *entonces* $X^3 + 2X + 2$ *divide a* F *y se modifica*

$$F(X) = \frac{F}{\operatorname{mcd}(F, G + 2)} = 1.$$

- *Para* $i = 2$, $\mathrm{mcd}(F, G+2) = \mathrm{mcd}(1, 2X^3 + X + 1) = 1$, *entonces no se añade nada y se ha terminado.*

Se han obtenido 2 *polinomios que dividen a* F *y antes se tenía que el núcleo de* φ *era de dimensión* 2, *por lo tanto ya se tiene la factorización de* F *sobre* $\mathbb{F}_3$.

$$F(X) = X^4 + 2X^3 + 2X^2 + X + 2 = (X+1) \cdot (X^3 + 2X + 2)$$

2.3. Lema de Hensel

Se considera aquí una versión constructiva del resultado obtenido por K. Hensel ([12], [13]) que hace posible refinar una congruencia de polinomios mod p^k a una congruencia mod p^{k+1}.

Teorema 2.3.1 (Lema de Hensel)**.** *Sea* $F \in \mathbb{Z}[X]$, $d = \mathrm{gr}(F) \geq 1$ *y* $p \geq 2$ *un primo. Si* $G_1, H_1 \in \mathbb{Z}[X]$ *son tales que*

$$F \equiv G_1 H_1 \bmod p \quad \text{y} \quad \mathrm{mcd}(G_1, H_1) \equiv 1 \bmod p,$$

para cualquier $k \in \mathbb{N}^*$ *entonces existen* $G_k, H_k \in \mathbb{Z}[X]$ *tales que*

$$\begin{aligned} F \equiv G_k H_k \bmod p^k, \quad & \mathrm{gr}(G_k) = \mathrm{gr}(G_1), && \mathrm{gr}(H_k) = \mathrm{gr}(H_1), \\ & G_k \equiv G_{k-1} \bmod p^{k-1}, && H_k \equiv H_{k-1} \bmod p^{k-1}, \\ & \mathrm{mcd}(G_k, H_k) \equiv 1 \bmod p. \end{aligned}$$

Los polinomios G_k, H_k *son únicos* mod p^k.

Demostración. Se supone que se tienen construidos los polinomios G_j, H_j con las propiedades necesarias para $j \leq k-1$. Sea $C \in \mathbb{Z}[X]$ tal que

$$F - G_{k-1} H_{k-1} = p^{k-1} C.$$

Se establecen

$$\begin{aligned} G_k &= G_{k-1} + p^{k-1} U, \\ H_k &= H_{k-1} + p^{k-1} V, \end{aligned}$$

donde $U, V \in \mathbb{Z}[X]$, quedan determinadas por las condiciones $\mathrm{gr}(U) \leq \mathrm{gr}(G_{k-1})$, $\mathrm{gr}(V) \leq \mathrm{gr}(H_{k-1})$ y

$$F \equiv G_k H_k \bmod p^k. \tag{2.3}$$

Obvérvese que por esta construcción

$$G_k \equiv G_{k-1} \bmod p^{k-1}, \quad H_k \equiv H_{k-1} \bmod p^{k-1}. \tag{2.4}$$

Puesto que $2k \geq k+1$, (2.3) es equivalente a

$$V G_{k-1} + U H_{k-1} \equiv \frac{1}{p^{k-1}} (F - G_{k-1} H_{k-1}) = C \bmod p. \tag{2.5}$$

Como $\mathrm{mcd}(G_{k-1}, H_{k-1}) = 1$ podemos calcular por el algoritmo euclídeo $A, B \in \mathbb{Z}[X]$ tales que

$$AH_{k-1} + BG_{k-1} \equiv 1 \bmod p, \quad \mathrm{gr}(A) < \mathrm{gr}(G_{k-1}), \quad \mathrm{gr}(B) < \mathrm{gr}(H_{k-1}).$$

Como $\mathbb{F}_p$ es un cuerpo, la solución general de (2.5) es

$$U \equiv (BC \bmod G_{k-1}) \bmod p, \quad V \equiv (AC \bmod H_{k-1}) \bmod p.$$

Por las condiciones de los grados, los polinomios U, V que satisfacen (2.3) y (2.4) son únicos mod p, por lo tanto G_k y H_k son únicos mod p^k. □

Nota 2.3.2. *El teorema 2.3.1 proporciona un método para elevar una factorización paso a paso, es decir, de $\mathbb{Z}/p\mathbb{Z}$ a $\mathbb{Z}/p^2\mathbb{Z}$, de $\mathbb{Z}/p^2\mathbb{Z}$ a $\mathbb{Z}/p^3\mathbb{Z}$, ..., de $\mathbb{Z}/p^{k-1}\mathbb{Z}$ a $\mathbb{Z}/p^k\mathbb{Z}$. Este método se llama elevación lineal.*

Descripción del algoritmo para el levantamiento lineal

- Se tiene $F \in \mathbb{Z}[X]$, p un primo, y $G_{k-1} \in \mathbb{Z}[X]$ un divisor de $\overline{F}$ sobre $\mathbb{F}_{p^{k-1}}$.
- Se calcula $H \in \mathbb{Z}[X]$ tal que $\overline{H_{k-1}} = \overline{F}/\overline{G}$.
- Se calcula $b = 2\binom{\lfloor \mathrm{gr}(F)/2 \rfloor}{\lfloor \mathrm{gr}(F)/4 \rfloor} \|F\|$ y $n = \lfloor \log_p(b) \rfloor + 2$ que fijan los criterios de parada del algoritmo.
- Si $C = \frac{F - GH}{p^{k-1}} = 0$ ya se tiene la factorización de F.
- Si $C \neq 0$, empleando la identidad de Bézout para G_k y H_k, se debe resolver la ecuación $F = (G_{k-1} + Vp^{k-1})(H_{k-1} + Up^{k-1})$ en U y V para pasar de una factorización en $\mathbb{F}_{p^k-1}$ a una factorización en $\mathbb{F}_{p^k}$, siendo $G_k = G_{k-1} + Up^{k-1}$ y $H_k = H_{k-1} + Vp^{k-1}$.
- Los criterios de parada se aplican con cada iteración, comprobando que las alturas $H(G)$ y $H(H)$ son menores que b o que el número de iteraciones no supera n.

El algoritmo se puede aplicar incluso sin que exista una factorización en $\mathbb{Z}[X]$.

Ejemplo 2.3.3. *Se considera el polinomio mónico $F(X) = X^4 + 1 \in \mathbb{Z}[X]$ que es irreducible en los enteros. Escogemos $p = 5$, entonces $\overline{F}(X) = X^4 + 1 = (X^2 + 2)(X^2 - 2)$ siendo $G_1(X) = X^2 + 2$ y $H_1(X) = X^2 - 2$.*

Mediante el algoritmo de división euclídea para $AG_1 + BH_1 = 1$ se hallan $A = 4$ y $B = 1$.

El algoritmo requiere que se tomen negativos los coeficientes que sean mayores que $5/2$, entonces $G_1(X) = X^2 + 2$ y $H_1(X) = X^2 - 2$.

Se halla $C = (F - G_1H_1)/5 = 5$, como $C \neq 0$ se calculan U y V tales que $G_2 = G_1 + 5 \cdot U$ y $H_2 = H_1 + 5 \cdot V$ sean factorización de F en $\mathbb{F}_{5^2}$. Para ello se hace la división

$$\frac{CA}{H_1} = \frac{20}{X^2 - 2} \quad \text{sobre} \quad \mathbb{F}_5$$

y se obtiene como cociente $q = 0$ *y como resto* $V = 4$*, entonces se halla* $U = CB + qG_1 = 1$ *en* $\mathbb{F}_5$*.*

Los nuevos polinomios que factorizan a F *en* $\mathbb{F}_{5^2}$ *son*

$$G_2(X) = G_1(X) + 5 \cdot 2 = X^2 + 7 \quad \text{y} \quad H_2(X) = H_1(X) + 5(4) = X^2 - 7.$$

Se continua con el algoritmo, repitiendo los mismos pasos anteriores y los datos que se obtienen se recogen en la siguiente tabla:

k	G_k	H_k	C
1	$X^2 + 2$	$X^2 - 2$	5
2	$X^2 + 7$	$X^2 - 7$	50
3	$X^2 + 57$	$X^2 - 57$	3250
4	$X^2 + 182$	$X^2 - 182$	33125
5	$X^2 - 1068$	$X^2 + 1068$	1140625
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Al final de cada iteración se cumple que $G_k H_k \equiv F \bmod 5^k$*. Sin embargo, nunca se obtendrán dos factores tales que* $F = GH \in \mathbb{Z}[X]$*.*

Nota 2.3.4. *Si no se emplean los criterios de parada, el algoritmo de un polinomio irreducible sobre* $\mathbb{Z}[X]$ *como el anterior, seguiría calculando polinomios con coeficientes cada vez mayores.*

Ejemplo 2.3.5. *Se considera el polinomio mónico* $F(X) = X^3 + 10X^2 - 432X + 5040 \in \mathbb{Z}[X]$ *y* $p = 5$*, entonces* $\overline{F}(X) = X^3 - 2X \in \mathbb{F}_5[X]$*. La factorización en* $\mathbb{F}_5[X]$ *es* $F(X) = X(X^2 - 2)$ *y se definen* $G_1(X) = X$ *y* $H_1(X) = X^2 - 2$ *primos entre sí.*

Se aplica el algoritmo euclídeo y se hallan $A(X) = -2X$ *y* $B(X) = 2$ *tales que* $AG + BH = 1$*. Las iteraciones del algoritmo son*

k	G_k	H_k	$C = \dfrac{F - G_k H_k}{5^{k-1}}$
1	X	$X^2 - 2$	$10X^2 - 430X + 5040$
2	$X + 5$	$X^2 + 5X - 7$	$-450X + 5075$
3	$X + 30$	$X^2 - 20X + 43$	$125X + 3750$
4	$X + 30$	$X^2 + 20X + 168$	0

Las iteraciones terminan cuando $C = (F - GH)/5^k = 0$ *y* $F(X) = (X + 30)(X^2 - 20X + 168)$ *es una factorización en* $\mathbb{Z}[X]$*.*

Nota 2.3.6. *Si* F *no es mónico aparece el denominado problema del coeficiente principal.*

Por ejemplo consideramos $F(X) = 12X^3 + 10X^2 - 36X + 35 \in \mathbb{Z}[X]$ *cuya factorización en* $\mathbb{Z}[X]$ *es* $F(X) = (2X + 5) \cdot (6X^2 - 10X + 7)$*.*

Se toma $p = 5$ *y se aplica el algoritmo de Hensel*

$$F \equiv 2X^3 - X = 2X(X^2 + 2) \bmod 5$$

por tanto se eligen $G_1(X) = 2X$ *y* $H_1(X) = X^2 + 2$*.*

Aplicando el algoritmo se obtiene

k	G_k	H_k	c
1	$2X$	X^2+2	$10X^3+10X^2-40X+35$
2	$12X+5$	$X^2-10X-3$	$125X^2+50X+50$
3	$12X+30$	$X^2+40X+22$	$-500X^2-1500X-625$
4	$12X+30$	$X^2-210X-103$	$2500X^2+7500X+3125$

Como $H_{k+1}=H_k+p^kV$ y $\mathrm{gr}(V)<\mathrm{gr}(H_k)$, entonces el coeficiente principal de H_{k+1} es igual al coeficiente principal de H_k y siempre será mónico, $\forall k$.

Por tanto nunca obtendremos la factorización en $\mathbb{Z}[X]$.

Nota 2.3.7. *Sea $\alpha=\mathrm{lc}(F)$ el coeficiente principal de F. Considerando $F'=\alpha F$. Si $F\equiv G_kH_k \bmod p^k$ con $\mathrm{mcd}(G_k,H_k)=1 \bmod p^k$ entonces*

$$F'\equiv\left(\alpha\frac{G_k}{\mathrm{lc}(G_k}\right)\left(\alpha\frac{H_k}{\mathrm{lc}(H_k)}\right) \bmod p^k \quad \text{y} \quad \mathrm{mcd}\left(G'_k=\alpha\frac{G_k}{\mathrm{lc}(G_k},H'_k=\alpha\frac{H_k}{\mathrm{lc}(H_k)}\right)=1$$

entonces para evitar el problema del coeficiente principal basta añadir al algoritmo de Hensel una adaptación de los coeficientes haciendo

$$G'_k=\alpha\frac{G_k}{\mathrm{lc}(G_k} \quad \text{y lo mismo con} \quad H'_k=\alpha\frac{H_k}{\mathrm{lc}(H_k)}.$$

Ejemplo 2.3.8. *Sean $F(X)=24X^4+22X^3+29X^2+16X+5\in\mathbb{Z}[X]$ y $p=5$, entonces $\overline{F}(X)=4X^4+2X^3+4X^2+X$ y un divisor de $\overline{F}$ es $G_1(X)=X^2+2X+4$.*

Se calcula $H_1(X)=\overline{F}(X)/G(X)=4X^2+4X$ en $\mathbb{F}_5$ y los valores de las cotas para los criterios de parada

$$b=2\begin{pmatrix}\lfloor\mathrm{gr}(F)/2\rfloor\\ \lfloor\mathrm{gr}(F)/4\rfloor\end{pmatrix}\|F\|=186{,}85 \quad \text{y} \quad n=\lfloor\log_p(b)\rfloor+2=5$$

.

Como F no es mónico, se considera $F'(X)=24F(X)=576X^4+528X^3+696X^2+384X+120$, $G_1(X)=4X^2+3X+1$ y $H_1(X)=4X^2+4X$. Y ahora se hace un cambio para que el coeficiente principal de G_1 y H_1 sea el mismo que el de F y no hallan problemas al dar la factorización con este coeficiente, ya que no se cambia a lo largo del algoritmo. Entonces se tiene

$$F(X)=576X^4+528X^3+696X^2+384X+120,$$

$$G_1(X)=24X^2+3X+1 \quad \text{y} \quad H_1(X)=24X^2+4X.$$

Ahora se hallan A y B de manera que cumplan que $AG_1+BH_1=1$ y se obtienen $A(X)=3X+1$ y $B(X)=2X+1$.

El algoritmo requiere que se tomen como negativos los coeficientes que sean mayores que $5^1/2$, entonces $G_1(X)=24X^2-2X+1$ y $H_1(X)=24X^2-X$.

Se hallan $C=(F-G_1H_1)/5=120X^3+134X^2+77X+24$, como $C\neq 0$ se tienen que calcular u y v tales que $G_2=G_1+5u$ y $H_2=H_1+5v$ sean factorización de F en $\mathbb{F}_{5^2}$. Para ello se hace la división

$$\frac{CA}{H_1}=\frac{3X^3+X+1}{2X+1} \quad \text{sobre} \quad \mathbb{F}_5$$

y se obtiene como cociente $q(X) = 3X + 2$ *y como resto* $v(X) = X + 4$*, entonces se halla* $u(X) = CB + qG_1 = 4X + 1$ *en* $\mathbb{F}_5$*.*

Los nuevos polinomios que factorizan a F *en* $\mathbb{F}_{5^2}$ *son*

$$G_2(X) = G_1(X)+5(4X+1) = 24X^2+18X+6 \quad \text{y} \quad H_2(X) = H_1(X)+5(X+4) = 24X^2+4X+20.$$

Se comprueba que no se han superado los criterios de parada calculando las alturas de G_2 *y* H_2*,* $H(G_2) = H(H_2) = 24 < b = 128{,}85$ *y se continua con el algoritmo.*

k	G_k	H_k	C
1	$24X^2 - 2X + 1$	$24X^2 - X + 1$	$120X^3 + 134X^2 + 77X + 24$
2	$24X^2 - 7X + 6$	$24X^2 + 4X - 5$	$24X^3 + 28X^2 + 13X + 6$
3	$24X^2 + 18X + 6$	$24X^2 + 4X + 20$	0

Se debe deshacer el cambio que se hizo al principio. Se halla el contenido de G *y se hace* $m = \operatorname{mcd}(24, \operatorname{cont}(G_3))$ *y*

$$G(X) = \frac{G_3(X)}{m} = 4X^2 + 3X + 1 \quad \text{y} \quad H(X) = \frac{H_3(X)}{24/m} = 6X^2 + X + 5.$$

Entonces se ha obtenido la factorización:

$$F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 = (4X^2 + 3X + 1)(6X^2 + X + 5).$$

Ejemplo 2.3.9. *Sea* $F(X) = 12X^3 + 10X^2 - 36X + 35 \in \mathbb{Z}[X]$ *y* $p = 5$*, entonces* $G_1(X) = 2X$ *y* $H_1(X) = X^2 + 2$ *son dos divisores de* $\overline{F}$*. Se definen los nuevos polinomios* $\tilde{F}(X) = 12F(X) = 144X^3 + 120X^2 - 432X + 420$*,* $\tilde{G}(X) = 12G(X)/2 = 2X$ *y* $\tilde{H}(X) = 12H(X) = 2X^2 - 1$*. Y se aplica el algoritmo de Hensel.*

k	$\tilde{G}_k$	$\tilde{H}_k$	C
1	$2X$	$2X^2 - 1$	$24X^2 - 84X + 84$
2	$12X + 5$	$12X^2 + 5X - 11$	$13X + 19$
3	$12X + 30$	$12X^2 - 20X + 14$	0

Termina la factorización y se deshace el cambio inicial

$$G(X) = \frac{\tilde{G}_3(X)}{\operatorname{cont}(G_3)} = 2X + 5 \quad \text{y} \quad H(X) = \frac{\tilde{H}_3(X)}{\operatorname{cont}(H_3)} = 6X^2 - 10X + 7.$$

Nota 2.3.10. *Se debe tener en cuenta que el algoritmo de elevación lineal no aumenta el grado de los divisores de* F*, por lo que se debe introducir el polinomio de mayor grado en la factorización sobre* $\mathbb{F}_p$*.*

Nota 2.3.11. *El método de Hensel no siempre obtiene una factorización del polinomio. Por ejemplo,* $F(X) = X^3 + 2X \in \mathbb{Z}[X]$*,* $p = 3$ *y los polinomios* $G(X) = X + 1$ *y* $H(X) = X^2 - X$ *divisores de* F *sobre* $\mathbb{F}_3[X]$

Se hallan los criterios de parada $b = 4{,}472$ *y* $n = 3$. *Empleando el algoritmo de la división euclídea se calcula* $A(X) = X + 1$ *y* $B(X) = 2$ *tales que* $AG + BH = 1$.

Se calcula $C = (F - GH)/3 = X$, *dado que* $C \neq 0$ *se halla la división*

$$\frac{CA}{H} = \frac{X^2 + X}{X^2 - X}$$

donde el cociente es $q = 1$, *el resto es* $V(X) = 2X$ *y se tiene* $U(X) = 1$.

Y la nueva descomposición es $G(X) = X + 4$ *y* $H(X) = X^2 - 4X$. *Y se empieza de nuevo el algoritmo.*

Se halla $C = (F - GH)/3^2 = 2X$ *entonces hallamos la división*

$$\frac{CA}{H} = \frac{2X^2 + 2X}{X^2 - 4X},$$

el cociente es $q = 2$, *el resto es* $V(X) = X$ *y se halla* $U(X) = 2$.

Entonces la nueva descomposición es $G(X) = X + 22$ *y* $H(X) = X^2 + 5X$

Se ha llegado al número máximo de elevaciones $n = 3$ *y no se ha encontrado la factorización.*

Capítulo 3

Polinomios enteros

En esta sección se estudiará la factorización de polinomios con coeficientes enteros, que es uno de los principales problemas algorítmicos con polinomios.

El primer algoritmo de factorización lo realizó F. T. Schubert (1793) y lo desarrolló L. Kronecker (1882). Otros métodos de factorización más eficientes que se van a tratar son los introducidos por A. Hausmann y L. Kronecker, E. R. Berlekamp, H. Zassenhaus y D. G. Cantor (1969) y A. K. Lenstra, H. W. Lenstra y L. Lóvasz (1982).

3.1. Método de Factorización de Kronecker

Sea $F \in \mathbb{Z}$, $\mathrm{gr}(F) = d \geq 1$. Según los resultados de la Sección 1.3 si F es reducible en $\mathbb{Z}[X]$, entonces existe un divisor G de F en $\mathbb{Z}[X]$ de grado menor o igual que $d/2$ y $L(G) \leq 2$.

Sea $s = \lfloor d/2 \rfloor$. Se fija una sucesión $F(d_0), F(d_1), \ldots, F(d_s)$ donde $d_0, d_1, \ldots, d_s$ son enteros cualesquiera distintos.

Si $G \mid F$, entonces $G(d_0) \mid F(d_0), \ldots, G(d_s) \mid F(d_s)$. Dado que $F(d_i)$ tiene un número finito de divisores, existe sólo un número finito de posibles valores para cada $G(d_i)$, con $i \in \{0, 1, \ldots, s\}$. A una combinación fija de estos valores $e = (e_0, e_1, \ldots, e_s)$, tal que e_i divide a $F(d_i)$ para todo i, le corresponde un único polinomio $H_e \in \mathbb{Z}[X]$ tal que

$$H_e(d_i) = e_i, \text{ para todo } i \in \{0, 1, \ldots, s\}.$$

El polinomio H_e se obtiene por la interpolación de Lagrange. Después de calcular el polinomio H_e, se establece mediante la división de polinomios si es o no un divisor de F en $\mathbb{Z}[X]$. Si $H_e \mid F$, entonces se ha encontrado un divisor no trivial de F en $\mathbb{Z}[X]$ y se aplica el mismo procedimiento para el polinomio cociente F/H_e. Después de un número finito de pasos se obtiene la factorización en irreducibles de F sobre $\mathbb{Z}$.

Mediante inducción sobre el grado de F se tiene:

Teorema 3.1.1 (L. Kronecker)**.** *La factorización en factores irreducibles de cualquier polinomio en $\mathbb{Z}[X]$ se puede lograr por este método en un número finito de pasos.*

Descripción del algoritmo de Kronecker

- Sea $F \in \mathbb{Z}[X]$, se escogen $s = \lfloor \mathrm{gr}(F)/2 \rfloor + 1$ números, para simplificar se toman $M = \{0, 1, \ldots, s\}$ y se halla $F(i)$, $\forall i \in M$. Se comprueba que ninguno sea raíz de F.
- Si existe $j \in M$ tal que $F(j) = 0$, se modifica el polinomio $F = F/(X - j)$ tantas veces como sea necesario. Y se obtiene otro conjunto M.
- Se hallan los divisores de $F(i)$ para $i \in M$ y se calcula el polinomio de Lagrange F_{Lag} tomando M y $F(i)$, $i \in M$, $(e_0, \ldots, e_s)$.
- Se comprueba si algún polinomio es entero y si divide a F y se repite todo el proceso con F/F_{Lag} y F_{Lag}.
- Si ningún F_{Lag} divide a F o no es entero, entonces F es irreducible.

Ejemplo 3.1.2. *Sea $F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 \in \mathbb{Z}[X]$, con $\mathrm{gr}(F) = 4$. Se escogen $\lfloor \mathrm{gr}(F)/2 \rfloor + 1 = 3$ enteros $i = 0, 1, 2$ en los que se evalúa $F(i)$ y se hallan sus divisores.*

i	$F(i)$	*Divisores*
0	5	± 1, ± 5
1	96	± 1, ± 2, ± 3, ± 4, ± 6, ± 8, ± 12, ± 16, ± 24, ± 32, ± 48, ± 96
2	713	± 1, ± 23, ± 31, ± 713

Ahora se consideran las combinaciones de divisores de la forma $e = (e_0, e_1, e_2)$ tal que e_i divide a $F(i)$ para $i = 0, 1, 2$. Para cada una de las combinaciones se tiene que calcular el polinomio de Lagrange H_e con $H_e(i) = e_i$, $\forall i \in \{0, 1, 2\}$. Algunas combinaciones con las que se obtienen polinomios enteros son, entre otras, $e^1 = (1, 1, 23)$ con $H_{e^1}(X) = 11X^2 - 11X + 1$, $e^2 = (1, 2, 31)$ con $H_{e^2}(X) = 14X^2 - 13X + 1$ o $e^3 = (1, 8, 23)$ con $H_{e^3}(X) = -7X^2 + 14X + 1$.

Pero H_e además de ser entero debe dividir a F. Así con $e = (1, 8, 23)$ obtenemos el polinomio $H_e(X) = 4X^2 + 3X + 1$ que divide a F.

Entonces con el polinomio $H_e(X) = 4X^2 + 3X + 1$ se obtiene $G_e(X) = F/H_e = 6X^2 + X + 5$ y se aplica el algoritmo de Kronecker a los polinomios H_e y G_e.

Sea $H_e(X) = 4X^2 + 3X + 1$, entonces igual que antes, se escogen los enteros $i = 0, 1$ y se evalúan $H_e(i)$.

i	$H_e(i)$	*Divisores*
0	5	± 1, ± 5
1	12	± 1, ± 2, ± 3, ± 4, ± 6, ± 12

Se construyen de nuevo las combinaciones $e = (e_0, e_1)$ y se halla el polinomio de Lagrange $H'_e(i) = e_i$ para $i = 0, 1$. En este caso de las 48 combinaciones que se obtienen ningún polinomio divide a H_e, por lo que H_e es irreducible en $\mathbb{Z}[X]$.

Sea ahora $G_e(X) = 6X^2 + X + 5$, se escogen los enteros $\{0, 1\}$ y se evalúa $G_e(i)$ con $i = 0, 1$.

i	$G_e(i)$	*Divisores*
0	1	± 1
1	8	± 1, ± 2, ± 4, ± 8

De nuevo se construyen las combinaciones y se hallan los polinomios de Lagrange. Tampoco se

encuentra ninguno que divida a G_e con lo cual, G_e es irreducible sobre $\mathbb{Z}[X]$.

Entonces la factorización en factores irreducibles de $F(X)$ es

$$F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 = (4X^2 + 3X + 1) \cdot (6X^2 + X + 5)$$

3.2. El algoritmo de Kronecker-Hausmann

A. Hausmann obtuvo en 1937 [10] una mejora del algoritmo de Kronecker. Este algoritmo se diferencia del anterior en que a partir de la lista $G(d_0), G(d_1), \ldots, G(d_s)$, decide si el correspondiente polinomio $G(X)$ es o no un divisor de $F(X)$ en $\mathbb{Z}[X]$ sin calcular el polinomio $G(X)$.

Teorema 3.2.1. *Sea $F \in \mathbb{Z}[X]$, $d = \mathrm{gr}(F) \geq 1$ y se considera $a_i = F(m_i)$ para $i = 1, 2, \ldots, d+1$, con $m_i = m_0 + ia$, $a \in \mathbb{Z}$, $a_i = b_i c_i$ para todo i, b_i, $c_i \in \mathbb{Z}$.*

Sean $G_1, G_2 \in \mathbb{Z}[X]$ tales que $G_1(m_i) = b_i$, $G_2(m_i) = c_i$ para todo i. Si el orden diferencial de la sucesión $(b_1, \ldots, b_{d+1})$ es u y el orden diferencial de $(c_1, \ldots, c_{d+1})$ es v, entonces $u + v \geq d$.

Por otra parte, si $u + v = d$, entonces $F = G_1 G_2$.

Demostración. Se tiene $(G_1 \cdot G_2)(m_i) = b_i c_i = a_i$ para $i = 1, 2, \ldots, d+1$. Por los Lemas 1.5.2 y 1.5.4se tiene $u + v \geq d$.

Si $u + v \leq d$, sea $G = F - G_1 G_2$. Puesto que G se anula en $d+1$ puntos y $\mathrm{gr}(G) \leq d$, se deduce que $G = 0$. Entonces $F = G_1 G_2$. □

Se supone ahora que F es estable, es decir, todas las raíces de F tendrán una parte real negativa. En principio este método trabaja con polinomios estables aunque como se vio en la Nota 1.6.4 ésta no es una limitación real. Entonces se tiene el siguiente resultado.

Teorema 3.2.2 (A. Hausmann [10])**.** *Sea $F \in \mathbb{Z}[X]$ estable de grado $d \geq 1$, $a_i = F(m_i)$, donde $m_1, m_2, \ldots, m_{d+1}$ son $d+1$ enteros de una progresión aritmética creciente con $m_1 > 0$. Sean $b_i, c_i \in \mathbb{Z}$ satisfaciendo las condiciones:*

i) $a_i = b_i c_i$ *para todo* $i = 1, 2, \ldots, d+1$;

ii) $b_0, c_0 \geq 2$;

iii) $b_0 < b_1 < \ldots < b_d$;

iv) $c_0 < c_1 < \ldots < c_d$.

Entonces existen dos polinomios $G_1, G_2 \in \mathbb{Z}[X]$ de grados $u \geq 1$ y $v \geq 1$ respectivamente, tales que $F = G_1 G_2$, $b_i = G_1(m_i)$ y $c_i = G_2(m_i)$ para todo i, si y sólo si, las diferencias de orden u en la tabla de diferencias de los b_i son una constante no nula, las diferencias de los c_i de orden v son también una constante no nula y $u + v = d$.

Demostración. Se supone que existen los polinomios G_1 y G_2 del enunciado. Como F es estable, los polinomios G_1 y G_2 son estables. En particular, tienen coeficientes positivos.

Por Lema 1.5.2 el orden diferencial de la sucesión $(b_1, \ldots, b_{d+1})$ es u y el orden diferencial de la sucesión $(c_1, \ldots, c_{d+1})$ es v. Por otro lado se tiene

$$u + v = \mathrm{gr}(G_1) + \mathrm{gr}(G_2) = \mathrm{gr}(G_1 G_2) = \mathrm{gr}(F) = d.$$

La implicación contraria se deduce del Teorema 3.2.1. □

Descripción del algoritmo de Kronecker-Hausmann

- Sea $F \in \mathbb{Z}[X]$ con $s := \mathrm{gr}(F)$, se comprueba si es estable. En el caso de que no lo sea, se modifica como se indica en la Nota 1.6.4.
- Se evalúa F en el conjunto $M = \{1, 2, \ldots, s+1\}$. Si alguno es raíz de F, se modifica el polinomio y el conjunto y se repite el proceso. Si $\mathrm{gr}(F) = 1$ ya se tiene la factorización.
- Si $\mathrm{gr}(F) \neq 1$ entonces se halla el conjunto de los divisores de $\{F(i),\ i \in M\} = \{(a_1, \ldots, a_{s+1})\}$, $\mathcal{B} = \{(b_1, \ldots, b_{s+1})\}$ y $\mathcal{C} = \{(c_1, \ldots, c_{s+1})$ tal que $c_i = a_i/b_i\}$.
- Se escogen las sucesiones de $\mathcal{B}$ y $\mathcal{C}$ de manera que cumplan las condiciones del Teorema 3.2.2. Si no se encuentran divisores que cumplan las condiciones, ya se tiene la factorización.
- Si se cumplen las condiciones del Teorema 3.2.2, se halla la tabla de diferencias de cada $b \in \mathcal{B}$ y el correspondiente $c \in \mathcal{C}$. Si la suma de los órdenes diferenciales de b y c no suman $s = \mathrm{gr}(F)$, se toman otros valores.
- Si los órdenes suman s, se calculan los polinomios de Newton N_b y N_c y se repite el algoritmo con cada uno.

Como se tienen las condiciones adicionales

$$b_0 < b_1 < \ldots < b_s, \quad c_0 < c_1 < \ldots < c_d$$

en muchos casos se pueden considerar un número mucho menor de tablas diferenciales.

Ejemplo 3.2.3. *Sea $F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 \in \mathbb{Z}[X]$. Véase si es estable, para ello se emplea el Teorema de Hurwitz 1.6.3 y se calculan los siguientes determinantes:*

$$\Lambda_1 = |22| > 0, \qquad \Lambda_2 \begin{vmatrix} 16 & 5 \\ 0 & 0 \end{vmatrix} = 0$$

Entonces, como Λ_2 no es positivo, se tiene que F no es estable. Aplicando las fórmulas de la Nota 1.6.4 se obtiene que $Q(X) = F(X+2) = 24X^4 + 214X^3 + 737X^2 + 1164X + 713$ es un polinomio estable.

Se escogen $\mathrm{gr}(Q)+1$ enteros $\{1, 2, 3, 4, 5\}$ y se evalúa $Q(i)$ con $i = 1, 2, 3, 4, 5$. Se hallan los divisores y se quitan los que no cumplan $b_1 < b_2 < \ldots < b_5$, entonces se tiene

i	$H_2(i)$	*Divisores*
1	2852	2, 4, 23, 31, 46, 62, 92, 124, 713, 1426
2	8085	3, 5, 7, 11, 15, 21, 33, 35, 49, 55, 77, 105, 147, 165, 231, 245, 385, 539, 735, 1155, 1617, 2695
3	18560	4, 5, 8, 10, 16, 20, 29, 32, 40, 58, 64, 80, 116, 128, 145, 160, 232, 290, 320, 464, 580, 640, 928, 1160, 1856, 2320, 3712, 4640
4	37001	163, 227
5	66708	204, 218, 306, 327

Ahora, como en Kronecker, se deben hallar todas las combinaciones de divisores $b = (b_1, \ldots, b_5)$ *y también de* $c = (c_1, \ldots, c_5)$*, pero se hace de manera que cumplan que* $b_i < b_{i+1}$ *y* $c_i < c_{i+1}$*, con* $c_i = Q(i)/b_i$ *para* $i = 1, 2, 3, 4, 5$.

Se tienen las 5-uplas $b = (23, 55, 116, 163, 204)$ *y* $c = (124, 147, 160, 227, 327)$ *que cumplen las condiciones anteriores. Se halla la tabla de diferencias finitas y se obtiene:*

$$\begin{array}{ccccc} 23 & 55 & 116 & 163 & 204 \\ & 32 & 61 & 47 & 41 \\ & & 29 & -14 & -6 \\ & & & -43 & 8 \\ & & & & 51 \end{array} \quad y \quad \begin{array}{ccccc} 124 & 147 & 160 & 227 & 327 \\ & 23 & 13 & 67 & 100 \\ & & -10 & 54 & 33 \\ & & & -21 & 64 \\ & & & & -85 \end{array}$$

con órdenes 4 los dos, entonces, como $4 + 4 \neq 4$*, esta combinación no sirve, así que se necesitan otras combinaciones.*

Se hallan las tablas de diferencias finitas de $b = (46, 77, 116, 163, 218)$ *y* $c = (62, 105, 160, 227, 306)$

$$\begin{array}{ccccc} 46 & 77 & 116 & 163 & 218 \\ & 31 & 39 & 47 & 55 \\ & & 8 & 8 & 8 \end{array} \quad y \quad \begin{array}{ccccc} 62 & 105 & 160 & 227 & 306 \\ & 43 & 55 & 67 & 79 \\ & & 12 & 12 & 12 \end{array}$$

y los órdenes en este caso sí cumplen $2 + 2 = 4$*, entonces se calculan los polinomios de Newton empleando las diagonales de las tablas anteriores y se obtienen* $N_b(X) = 4X^2 + 19X + 23$ *y* $N_c(X) = 6X^2 + 25X + 31$.

Con estos nuevos polinomios se aplica el algoritmo de Kronecker-Hausmann.

Sea $N_b(X) = 4X^2 + 19X + 23$*, repitiendo el mismo proceso se escogen* $\mathrm{gr}(N_b) + 1$ *enteros* $\{1, 2, 3\}$ *y se evalúan* $N_b(i)$ *con* $i = 1, 2, 3$*. Se hallan los divisores y se mantienen sólo los que cumplan* $b_1 < b_2 < \ldots < b_3$*, entonces se tiene*

i	$H_2(i)$	*Divisores*
1	46	$2, 23$
2	77	$7, 11$
3	116	$\emptyset$

como no se tienen divisores que cumplan las condiciones, $N_b(X) = 4X^2 + 19X + 23$ *es irreducible en* $\mathbb{Z}[X]$.

Ahora sea $N_c(X) = 6X^2 + 25X + 31$*, se escogen* $\mathrm{gr}(N_b) + 1$ *enteros* $\{1, 2, 3\}$ *y se evalúan* $N_b(i)$ *con* $i = 1, 2, 3$*. Se hallan los divisores y se dejan sólo los que cumplan* $b_1 < b_2 < \ldots < b_3$*, entonces se tiene*

i	$H_2(i)$	*Divisores*
1	62	$2, 31$
2	105	$3, 5, 7, 15, 21, 35$
3	160	$4, 5, 8, 10, 16, 20, 32, 40$

Se eligen las combinaciones de b y c que cumplan las condiciones del Teorema 1.6.3 y se hallan las tablas de diferencias finitas. Como ninguna combinación cumple que los órdenes de las tablas sumen el grado de N_c, se tiene que $N_c(X) = 6X^2 + 25X + 31$ es irreducible.

Por tanto la factorización de F en factores irreducibles sobre $\mathbb{Z}[X]$ sería

$$F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 = (4X^2 + 3X + 1) \cdot (6X^2 + X + 5)$$

Nota 3.2.4. *Si se aplicara el algoritmo de Kronecker, se obtendrían* 49280 *combinaciones que se tendrían que probar una a una calculando el polinomio de Lagrange, sin embargo, con esta modificación de Hausmann se tendrán que comprobar exactamente la mitad haciendo tablas diferenciales.*

3.3. El algoritmo de Berlekamp-Zassenhaus-Cantor

Este algoritmo fue desarrollado por E. R. Berlekamp, H. Zassenhaus y D. G. Cantor en 1969 [4]. Está basado en la factorización de polinomios sobre un cuerpo finito y emplea técnicas de elevación de Hensel. Concretamente dado un polinomio $F \in \mathbb{Z}[X]$ se factoriza módulo un primo p y a continuación se eleva a una factorización módulo una potencia lo bastante grande de p, p^k.

Algoritmo de Berlekamp-Zassenhaus-Cantor

Si $F \in \mathbb{Z}[X]$, $d = \mathrm{gr}(F) \geq 1$, una factorización de F en polinomios irreducibles sobre $\mathbb{Z}$ se obtiene a través de los siguientes pasos.

- Si $F(X) = a_d X^d + a_{d-1} X^{d-1} + \cdots + a_0$ no es mónico, se asocia por sustitución un polinomio mónico $F_{mon} \in \mathbb{Z}[X]$, entonces la factorización de F sobre $\mathbb{Z}$ es equivalente a la factorización del polinomio mónico

  $$F_{mon}(X) = a_d^{d-1} F\left(\frac{X}{a_d}\right) = X^d + a_{d-1} X^{d-1} + \cdots + a_d^{d-2} a_1 X + a_d^{d-1} a_0.$$

  Para recuperar el polinomio original se deberá utilizar la transformación inversa. Entonces

  $$F(X) = \frac{1}{a_d^{d-1}} F_{mon}(a_d X),$$

  da una factorización de F a partir de una factorización de F_{mon}. Así se puede suponer que F es mónico.

- Se factoriza F_{mon} en un producto de polinomios mónicos libres de cuadrados en $\mathbb{Z}[X]$. Se puede suponer sin pérdida de generalidad que F es libre de cuadrados. Para ello se calcula

la derivada F', el máximo común divisor $\mathrm{mcd}(F, F')$ y se escribe

$$F = \mathrm{mcd}(F, F') \cdot \frac{F}{\mathrm{mcd}(F, F')}.$$

El polinomio $F/\mathrm{mcd}(F, F')$ es libre de cuadrados.

- Se escoge un primo $p \geq 2$ tal que $\overline{F} = F \bmod p$ es libre de cuadrados sobre $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$. Esta condición se verifica si p no divide al discriminante de F. En efecto, $\mathrm{mcd}(F, F') = 1$ si y sólo si $Res(F, F') \neq 0$.
- Se factoriza $\overline{F}$ en $\mathbb{F}_p[X]$ con el algoritmo de factorización de polinomios Berlekamp sobre cuerpos finitos, $\overline{F} \equiv \prod_{i=1}^{r} G_i \bmod p$. Si $r = 1$ entonces es irreducible.
- Si $r > 1$, se agrupan los resultados de dos en dos de manera que $\overline{F} = \overline{G_0 H_0}$. Aplicando el lema de Hensel, se elevan a $\mathbb{F}_{p^k}$ cumpliendo $\overline{F} = \overline{GH} \bmod p^k$.
- Se comprueba si G/F o H/F. Si $F = G\dfrac{F}{G}$ se aplica el mismo método a G y F/G.
- Si G no divide a F y H no divide a F, se prueba con otra partición de los G_i dados por Berlekamp.

Ejemplo 3.3.1. *Sea* $F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 \in \mathbb{Z}[X]$ *con* $d := \mathrm{gr}(F) = 4$.

Se halla el contenido de F *y* $\mathrm{cont}(F) = 1$*, entonces no se modifica nada. Ahora se convierte* F *en un polinomio mónico*

$$F_{mon}(X) = 24^{4-1}F(X/24) = X^4 + 22X^3 + 696X^2 + 9216X + 69120$$

y se comprueba que es libre de cuadrados $\mathrm{mcd}(F_{mon}, F'_{mon}) = 1$.

Ahora se calcula la resultante $\mathrm{Res}(F_{mon}, F'_{mon}) = 24461584537485312$ *y se busca el menor primo que no la divida* $p = 5$.

Con el algoritmo de Berlekamp se halla la factorización de F_{mon} *en* $\mathbb{F}_5$ *y devuelve los siguientes polinomios* X, $X + 4$ *y* $X^2 + 3X + 4$. *Con estos se construye un conjunto con todas las posibles combinaciones de divisores de* F,

$$\big\{\{X\}, \{X+4\}, \{X^2+3X+4\}, \{X, X^2+3X+4\},$$
$$\{X, X+4\}, \{X^2+3X+4, X+4\}, \{X, X^2+3X+4, X+4\}\big\}$$

y con ayuda del algoritmo de Hensel se elevan a $\bmod 5^k$.

Aplicando Hensel a $X^2 + 3X + 4$ *en* $\mathbb{Z}_5[X]$ *se obtiene* $X^2 + 18X + 144$ *y* $X^2 + 4X + 480$. *Ahora se aplica a estos polinomios el algoritmo de Berlekamp-Zassenhaus-Cantor.*

Sea $F_1(X) = X^2 + 18X + 144$*, tiene contenido* 1 *y es un polinomio libre de cuadrados. El primo* p *que no divide a la resultante* $\mathrm{Res}(F_1, F_1') = 252$ *es* $p = 5$.

Se halla la factorización en $\mathbb{F}_5$ *de* F_1 *con el algoritmo de Berlekamp sobre cuerpos finitos y se obtiene* $X^2 + 3X + 4$. *Como Berlekamp devuelve un solo polinomio, se asume que* $F_1(X) = X^2 + 18X + 144$ *es ya un divisor irreducible de* F.

Se toma ahora $F_2(X) = X^2+4X+480$*, el contenido es* 1 *y también es libre de cuadrados. El primo* p *que no divide a la resultante* $\mathrm{Res}(F_2, F_2') = 1904$ *es* $p = 3$*. Se aplica el algoritmo de Berlekamp a* F_2 *y se obtienen los polinomios* X *y* $X+1$*, se crea igual que antes el conjunto de todas las posibles combinaciones y se aplica Hensel a cada uno.*

En este caso con el algoritmo de elevación lineal no se obtiene ninguna factorización válida, el polinomio F_2 *es irreducible.*

Ahora que se tiene la descomposición de F *en* F_1 *y* F_2*, se deshace el cambio de mónico que se hizo al principio y queda*

$$F_1(24X) = 576X^2 + 432X + 144 \quad \text{y} \quad F_2(24X) = 576X^2 + 96X + 480.$$

Para terminar se dividen F_1 *y* F_2 *entre el contenido de* F_1 *y* F_2 *respectivamente. Entonces se tiene la siguiente factorización*

$$F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 = (4X^2 + 3X + 1)(6X^2 + X + 5)$$

3.4. El algoritmo de factorización LLL

El algoritmo de Berlekamp-Zassenhaus-Cantor puede tener un coste exponencial, pero A. K. Lenstra, H. W. Lenstra y L. Lovász [17] desarrollaron un método de factorización de polinomios enteros que se puede llevar a cabo en tiempo polinomial.

Este algoritmo hace uso también de la factorización de Berlekamp sobre cuerpos finitos y el Lema de Hensel.

Esencialmente calcula para un polinomio F sobre $\mathbb{Z}$, un primo p convenientemente pequeño y un divisor irreducible H de F en $\mathbb{F}_p[X]$. Después se busca un divisor irreducible H_0 de F en $\mathbb{Z}[X]$ que sea divisible por H en $\mathbb{F}_p[X]$.

La condición de que H divida a H_0 es equivalente al hecho de que H_0 pertenece a un cierto retículo y que los coeficientes de H_0 sean relativamente pequeños.

Definición 3.4.1. *Sea* $n \in \mathbb{N}^*$*. A un subconjunto* L *de* $\mathbb{R}^n$ *se le denomina* retículo *si existe una base* $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ *del espacio vectorial real* $\mathbb{R}^n$ *tal que*

$$L = \sum_{i=1}^{n} \mathbb{Z}\mathbf{b}_i = \left\{ \sum_{j=1}^{n} r_j \mathbf{b}_j \ ; \quad r_j \in \mathbb{Z}, \ j = 1, 2, \ldots, n \right\}.$$

Se dice que $\mathbf{b}_1, \ldots, \mathbf{b}_n$ *forman una* base *de* L*, o generan* L*. El entero* n *se denomina el* rango *del retículo* L *y el número* $d(L) = |det(\mathbf{b}_1, \ldots \mathbf{b}_n)|$ *es el* determinante del retículo L*.*

Definición 3.4.2. *Una base* $\mathbf{b}_1, \ldots, \mathbf{b}_n$ *del retículo* L *es una* base reducida *si siendo* $\mathbf{b}_i^*$, $1 \leq i \leq n$ *la base ortonormal obtenida de la base por el método de Gram-Smidth se tiene*

$$|\mu_{ij}| \leq \frac{1}{2} \quad \text{para} \quad 1 \leq j < i \leq n$$

donde $\mu_{ij} = \dfrac{< \mathbf{b}_i, \mathbf{b}_j^* >}{< \mathbf{b}_j^*, \mathbf{b}_j^* >}$, *y*

$$\|\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*\|^2 \geq \frac{3}{4}\|\mathbf{b}_{i-1}^*\|^2 \quad \text{para} \quad 1 < i \leq n.$$

Sea $p \geq 2$ un primo, $k \in \mathbb{N}^*$ y sea $F \in \mathbb{Z}[X]$ de grado $n \geq 1$, mediante el algoritmo de Berlekamp y la elevación de Hensel se obtiene un polinomio $H \in \mathbb{Z}[X]$ de grado $d \leq n$ que satisface las siguientes condiciones:

1. H es mónico,
2. $H \bmod p^k$ divide a $F \bmod p^k$ en $(\mathbb{Z}/p^k\mathbb{Z})[X]$,
3. $H \bmod p$ es irreducible en $\mathbb{F}_p[X]$,
4. $(H \bmod p)^2$ no divide a $F \bmod p$ en $\mathbb{F}_p[X]$.

Obsérvese que por 4 el polinomio $F \bmod p$ debe ser libre de cuadrados.

Se fija ahora un entero $m \geq d = \mathrm{gr}(H)$ y se considera el conjunto

$$L = \left\{Q \in \mathbb{Z}[X]; \mathrm{gr}(Q) \leq m, H \bmod p^k \ \text{ divide a } \ Q \bmod p^k \text{ en } (\mathbb{Z}/p^k\mathbb{Z})[X]\right\}.$$

L es un subconjunto del $\mathbb{R}$-espacio vectorial de los polinomios de grado menor o igual a m con coeficientes reales el cual se identifica con $\mathbb{R}^{m+1}$ por la aplicación lineal

$$\sum_{i=0}^{m} a_i X^i \longmapsto (a_0, a_1, \ldots, a_m).$$

Entonces L es un retículo y una base de L es

$$\mathcal{B} = \left\{p^k X^i, 0 \leq i < d\right\} \cup \left\{HX^j, 0 \leq j \leq m-d\right\}.$$

El trabajo de Lenstra, Lenstra y Lovász da un algoritmo para calcular una base reducida de un retículo. El resultado principal de este trabajo establece lo siguiente.

Teorema 3.4.3. *Sea* $\mathbf{b}_1, \mathbf{b}_2 \ldots, \mathbf{b}_{m+1}$ *una base reducida de* L *y* $j \in \{1, \ldots, m+1\}$ *tal que*

$$\|\mathbf{b}_j\| < \left(p^{kd}\|F\|^{-m}\right)^{1/n}. \tag{3.1}$$

Sea t *el mayor de los* $j \in \{1, \ldots, m+1\}$. *Entonces* $H_0 = \mathrm{mcd}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_t)$ *es un factor irreducible de* F *tal que* $\mathrm{gr}(H_0) = m+1-t$ *y* $H_0 \bmod p$ *es divisible por* $H \bmod p$.

Descripción del Algoritmo LLL

El algoritmo LLL empieza calculando el menor primo p que no divide a la resultante de F y F', a cada factor G se le calculan las cotas u, m y k

$$u = \left\lfloor \log_2 \frac{n-1}{d} \right\rfloor, \quad m = \left\lfloor \frac{n-1}{2^{u-i}} \right\rfloor, \quad k = \left\lfloor \frac{\log_p \left(2^{mn/2} \binom{2m}{m}^{n/2} \|F\|^{m+n+1}\right)}{d} \right\rfloor + 1.$$

Se eleva a $G_k \in \mathbb{Z}_{p^k}[X]$ mediante el algoritmo de Hensel. Se comprueba si G_k genera un factor de F en $\mathbb{Z}[X]$ haciendo uso del algoritmo de reducción de bases para el retículo asociado a G_k.

En concreto, si $\mathcal{B} = (B_0, \ldots, B_m)$ es la base reducida del retículo asociado a G_k y si $\|B_0\| \geq \left(p^{kd}\|F\|^{-m}\right)^{-1/n}$ entonces el polinomio G_k no genera ningún divisor de F sobre $\mathbb{Z}[X]$. En caso contrario se buscan los primeros t elementos de $\mathcal{B}$ de manera que $B_t \leq \left(p^{kd}\|F\|^{-m}\right)^{-1/n}$ y $B_{t+1} \geq \left(p^{kd}\|F\|^{-m}\right)^{-1/n}$. Se calcula el divisor de F en $\mathbb{Z}[X]$ como $H_0 = \mathrm{mcd}(B_0, \ldots, B_t)$,

Se eliminan de la factorización dada por el algoritmo de Berlekamp aquellos factores que dividan a H_0 y se continua igual con los restantes factores.

Ejemplo 3.4.4. *Sea* $F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 \in \mathbb{Z}[X]$ *y su derivada* $F'(X) = 96X^3 + 66X^2 + 58X + 16$, *se halla la resultante* $\mathrm{Res}(F, F') = 3072050688$ *y se busca el menor primo que no la divida,* $p = 5$.

Ahora se calcula $\overline{F}(X) = 4X^4 + 2X^3 + 4X^2 + X \in \mathbb{F}_5[X]$ *y con el algoritmo de Berlekamp se obtiene la factorización de* $\overline{F}$ *sobre* $\mathbb{F}_p$ *que es* $\overline{F}(X) = X \cdot (X^2 + 2X + 4) \cdot (X + 1)$.

Para cada polinomio de la factorización en $\mathbb{F}_5$ *se aplica el siguiente proceso, que se describe para el factor* $(X + 1)$

$H(X) = X + 1$, $p = 5$. *Se calculan* $u = 1$ *y las cotas* $m = 3$ *y* $k = 26$ *y se aplica la elevación lineal de Hensel a* H *de donde se obtiene* $H_{p^k}(X) = 24X + 1151326532802108409$.

Como $m > \mathrm{gr}(H)$ *se halla la base del retículo*

$$\begin{aligned}\mathcal{B} = \{&(1490116119384765625, 0, 0, 0), (1151326532802108409, 24, 0, 0),\\ &(0, 1151326532802108409, 24, 0), (0, 0, 1151326532802108409, 24)\}.\end{aligned}$$

Por el algoritmo de reducción de bases se obtiene la base reducida:

$$\begin{aligned}\mathcal{B}^* = \{&(-35, 113, -18, 144), (160, 32, 192, 0), (460446812, -201696315, -350089676, 226428576),\\ &(-2918389, 564556110, -91660695, -455186664)\}.\end{aligned}$$

Sea $\mathcal{B}^* = \{B_1, \ldots, B_4\}$ *y* t *el mayor entero tal que* $\|B_t\| < \left(p^{kd}\|F\|^{-m}\right)^{-1/n}$, *se calcula* H_0 *como el máximo común divisor de* $B_1, \ldots, B_t$. *Se obtiene así que* $H_0(X) = 6X^2 + X + 5$.

Se modifica $F(X) = F/H_0 = 4X^2 + 3X + 1$ *y se eliminan de la factorización de Berlekamp los polinomios que sean divisores de* $H_0 \in \mathbb{F}_5[X]$ *y queda* $X^2 + 2X + 4$, *se repite el proceso anterior para* $H(X) = X^2 + 2X + 4$ *y como los grados de* H *y* F *son iguales, se devuelve* $F(X) = 4X^2 + 3X + 1$.

Ya se tiene la factorización de F

$$F(X) = 24X^4 + 22X^3 + 29X^2 + 16X + 5 = (6X^2 + X + 5) \cdot (4X^2 + 3X + 1)$$

Conclusiones

En esta memoria se estudian esencialmente diversos algoritmos para la factorización de polinomios en $\mathbb{Z}[X]$.

Se comienza por el algoritmo más simple debido a Kronecker (1882) y la mejora de Hausmann (1937). A continuación se estudia el algoritmo de Berlekamp-Zassenhaus-Cantor (1969) que hace uso de la factorización sobre cuerpos finitos y finalmente se considera el algoritmo LLL (1982) basado en la teoría de retículos.

Si bien el algotirmo LLL teóricamente es el de menor coste computacional, no resulta ser muy efectivo en muchos casos, por lo que las investigaciones recientes tienden a combinarlo con otros de factorización módulo p.

Desde el punto de vista personal, ha sido emocionante ir entendiendo cada paso de los algoritmos e irlos reproduciendo en la plataforma SageMath, corrigiendo los abundantes fallos encontrados en algunas referencias, así como realizando algunas mejoras menores de los mismos. Cada uno de los algoritmos implementados ha sido validado con numerosos elemplos, si bien nos queda pendiente el análisis de la complejidad de cada uno de ellos.

Consideramos que disponer de estos algoritmos en SageMath puede resultar de utilidad tanto en la docencia del tópico tratado en la memoria, como en diversas aplicaciones de los mismos.

Por otra parte la implementación del algoritmo LLL ha despertado en mi curiosidad por el estudio de la teoría de retículos cuyas aplicaciones van más allá de los problemas de factorización.

Apéndice A

Algoritmos

A.1. Algoritmo de Berlekamp sobre cuerpos finitos

```
#######################            Berlekamp.sage            #######################
import numpy as np
def Berlekamp(F):
    Fact = []
    R = F.parent()
    Z_p = R.base_ring()
    p = Z_p.cardinality()
    with localvars(R, ['x']):
        F1 = F
        mcd = gcd(F1,F1.diff())
        if mcd != 1:
            if mcd != F1:
                F1 = R(F1/mcd)
                Fact += Berlekamp(mcd)
            else:
                expo = F.exponents()
                ee = 0
                gc = gcd(expo)
                while p.divides(gc):
                    gc = gc/p
                    ee += 1
                coef = F.coefficients()
                F2 = 0
                for i in expo:
                    F2 += (coef.pop(0)) * x^(i/(p^ee))
                return (p^ee)*Berlekamp(F2)
        d = F1.degree()
        M = matrix(Z_p,d)
        for i in range(0,d):
            m = (( (x^(i*p)).quo_rem(F1) )[1]).coefficients(sparse=False)
            M[i, 0:len(m)] = vector(m)
        Q = M-matrix.identity(Z_p,d)
        L = list( (Q.kernel()).basis() )
        v = vector(Z_p,d)
        v[0] = 1
        L.remove(v)
        l = d-Q.rank()
```

```
        if l != 1:
            Fact2 = []
            v = L.pop(0)
            G = R(list(v))
            for i in range(0,p):
                g = gcd(F1, G+i)
                if g != 1:
                    Fact2 += [g]
                    F1 = R(F1/g)
            if len(Fact2) != l:
                CFact2 = copy(Fact2)
                for i in CFact2:
                    L = Berlekamp(i)
                    if len(L) != 1:
                        Fact2.remove(i)
                        Fact2 += L
                        if len(Fact2) == l:
                            return Fact + Fact2
            return Fact + Fact2
        else:
            return Fact + [F1]
```

A.2. Algoritmo para el levantamiento lineal

```
#######################       Levantamiento lineal.sage        #######################
def Hensel(F, G1, p, n = None, H1 = None):
    Z_p = Integers(p)
    R2.<y> = PolynomialRing(Z_p)
    R = F.parent()
    nomonicos = (F.leading_coefficient() != 1)
    with localvars(R, ['x']):
        if H1 == None:
            H1 = R( R2(F).quo_rem(R2(G1))[0] )
        if n != None:
            b = +oo
        else:
            b = 2*binomial(floor(F.degree()/2), floor(F.degree()/4)) * F.norm(2)
            n = floor(log(b,p)) + 2
        lc = F.leading_coefficient()
        if nomonicos:
            F = F*lc
            G1 = R( lc*(R2(G1)).monic() )
            H1 = R( lc*(R2(H1)).monic() )
        G = R( G1 + (lc - G1.leading_coefficient()) * x^(G1.degree()) )
        H = R( H1 + (lc - H1.leading_coefficient()) * x^(H1.degree()) )
        alt_H = 0
        alt_G = 0
        mcd, A, B = xgcd(R2(G1), R2(H1))
        for k in range(2,n+1):
            Gc = G.coefficients(sparse=False)
            for i in range(G.degree()):
                if Gc[i] >= floor((p^(k-1))/2):
                    G = G-(p^(k-1)) * R(x^i)
            Hc = H.coefficients(sparse=False)
            for i in range(H.degree()):
                if Hc[i] >= floor((p^(k-1))/2):
                    H = H-(p^(k-1)) * R(x^i)
            if b != oo:
                alt_H = (vector(H.coefficients())).norm(oo)
```

```
                alt_G = (vector(G.coefficients())).norm(oo)
                if not(alt_H < b and alt_G < b):
                    break
            C = (F-G*H)/p^(k-1)
            if C == 0:
                if nomonicos:
                    cont = gcd(lc, gcd(G.coefficients()))
                    G = G/(cont)
                    H = H/(lc/cont)
                return True, [G,H]
            (q,v) = R2(C)*(A)).quo_rem(R2(H1)
            u = (R2(C)*(B) + q*R2(G1))
            v = R(R2(v))
            u = R(R2(u))
            G = R(G + p^(k-1)*u)
            H = R(H + p^(k-1)*v)
        if nomonicos:
            cont = gcd(lc, gcd(G.coefficients()))
            G = G/(cont)
            H = H/(lc/cont)
        return False, [G,H]
```

A.3. Algoritmo de Kronecker

```
######################            Kronecker.sage            ######################
import numpy as np
def Kronecker(F):
    R2.<y> = QQ[]
    R = F.parent()
    with localvars(R, ['x']):
        list_G = []
        Q = F
        i = 0
        ev_Q = []
        while i < floor( Q.degree()/2 ) + 1:
            ev = Q(i)
            if ev == 0:
                list_G.append(x-i)
                Q = R(Q/(x-i))
                i = 0
                ev_Q = []
            else:
                ev_Q.append(ev)
                i += 1
        s = floor( Q.degree()/2 )+1
        ev_Q = np.array(ev_Q[0:s], dtype = sage.rings.integer.Integer)
        if Q.degree() != 1:
            lista_div = []
            for i in range(0,s):
                div = ( ev_Q[i].abs() ).divisors()
                lista_div.append(div + list( -1*(vector(div)) ))
            z = xmrange_iter(lista_div, tuple)
            for i in z:
                ff = R2.lagrange_polynomial(zip( range(0,s), i) )
                if ff.denominator() == 1:
                    f_lag = R(ff)
                    if f_lag.divides(Q) and f_lag.degree() != 0:
                        list_G = list_G + Kronecker(R(Q/f_lag)) + Kronecker(f_lag)
                        return list_G
        return list_G + [Q]
```

A.4. Algoritmo de Kronecker-Hausmann

```
#######################        Kronecker-Hausmann.sage         #######################
import numpy as np
def diferencias(b):
    s = len(b)
    for j in range(0,s):
        for k in range(s-1, j, -1):
            b[k] = b[k]-b[k-1]
        if ( b[j+1:s] == b[s-1] ).all():
            diff = b[0:j+2]
            return j+1, diff

def newton(diff):
    R2.<x> = QQ[]
    P = diff[0]
    Prod = (x-1)
    for k in range(2, len(diff) + 1):
        P += diff[k-1]*Prod
        Prod *= (x-k)/k
    return P

def is_stable(F):
    for k in range(1, F.degree()+1):
        M = matrix(k)
        cc = (F+x^(2*k)).coefficients(sparse=False)
        _= cc.pop()
        if not((np.array(cc) >= 0).all() or (np.array(cc) <= 0).all()):
            return False
        cc.reverse()
        v = vector(cc)
        for i in range(k):
            M[i,0:2*i+2] = v[2*k-(i)*(2)-2:min(2*k,2*k-(i)*(2)-2+k)]
            if M.determinant() <= 0:
                return False
    return True

def Hausmann(F):
    R = F.parent()
    Q = F
    list_G = []
    with localvars(R, ['x']):
        i = 1
        ev_Q = []
        while i < Q.degree()+2:
            ev = Q(i)
            if ev == 0:
                list_G.append(x-i)
                Q = R(Q/(x-i))
                i = 1
                ev_Q = []
            else:
                ev_Q.append(ev)
                i += 1
        s = Q.degree()
        ev_Q = np.array(ev_Q[0:s+1], dtype = sage.rings.integer.Integer)
        if Q.degree() != 1:
            lista_div = []
            cota1 = 1
            for i in range(0,s+1):
                div = (ev_Q[i].abs()).divisors()
                l = len(div)
```

```
                while l != 0 and div[0] <= cota1:
                    _=div.pop(l-1)
                    if l != 1:
                        _= div.pop(0)
                    l-= 2
                if l != 0:
                    cota1 = div[0]
                else:
                    return list_G + [Q]
                lista_div.append(div)
            z = xmrange_iter(lista_div)
            c = np.zeros(s+1, dtype = sage.rings.integer.Integer)
            for i in z:
                c[0] = ev_Q[0]/i[0]
                for k in range(0,s):
                    c[k+1] = ev_Q[k+1]/i[k+1]
                    if i[0] >= 2 and i[k+1] > i[k] and c[0] >= 2 and c[k+1] > c[k]:
                        if k == s-1:
                            ord_b, diffb = diferencias(np.array(i, dtype = sage.rings.integer.Integer))
                            ord_c, diffc = diferencias(c)
                            if ord_b+ord_c == Q.degree():
                                Gb = newton(diffb)
                                Gc = newton(diffc)
                                if Gb.denominator() == 1 and Gc.denominator() == 1:
                                    return list_G + Hausmann(R(Gb)) + Hausmann(R(Gc))
                    else:
                        break
        return list_G + [Q]

def Haus2(F):
    R = F.parent()
    with localvars(R, ['x']):
        if is_stable(F):
            return Hausmann(F)
        else:
            n = max(floor( (F.norm(oo)) / abs(F.leading_coefficient()) ) + 1, 1)
            Q = R(F(x+n))
            L = Hausmann(Q)
            L2 = []
            for i in L:
                L2.append(R(i(x-n)))
            return L2
```

A.5. Algoritmo de Berlekamp-Zassenhaus-Cantor

```
#######################       Berlekamp-Zassenhaus-Cantor.sage        #######################
def Zassenhaus(F):
    P = F
    R = F.parent()
    d = P.degree()
    with localvars(R, ['x']):
        cont = gcd(P.coefficients())
        if cont != 1:
            P = R(P/cont)
        lc = P.leading_coefficient()
        if lc != 1:
            P = R((lc^(d-1))*P(x/lc))
        Fact = []
        mcd = gcd(P,diff(P,x))
        if mcd != 1:
            P = R(P/mcd)
            Fact += Zassenhaus(mcd)
```

```
    Res = P.resultant(diff(P,x))
    p = 2
    Primos = Primes()

    while Res%p == 0:
        p = Primos.next(p)
    Z_p = Integers(p)
    R2.<y> = PolynomialRing(Z_p)
    Fact_B = Berlekamp(R2(P))
    if len(Fact_B) == 1:
        Fact += [P]
    else:
        bol = False
        S = list( Subsets(Fact_B, submultiset=True) )
        S.remove([])
        for i in S:
            if len(i) != len(Fact_B):
                G = 1
                for j in i:
                    G *= j
                bol, F_h = (Hensel(P,R(G),p))
                if bol:
                    break
        if bol:
            Fact += Zassenhaus(F_h[0]) + Zassenhaus(F_h[1])
        else:
            return [F]
    if lc != 1:
        Fact2 = []
        for i in Fact:
            j = R(i(x*lc))
            Fact2.append(j/(gcd(j.coefficients())))
        Fact2[0] *= cont
        return Fact2
    Fact[0] *= cont
    return Fact
```

A.6. Algoritmo de Lenstra, Lenstra y Lovász

```
#######################                LLL.sage                #######################
import numpy as np
def Reduccion(B):
    n = len(B)
    C = []
    b = []
    Mu = matrix(QQ, n)
    for i in range(n):
        C.append(B[i])
        for j in range(i):
            Mu[i,j] = (vector(B[i])*vector(C[j])) / (b[j])
            C[i] -= vector(Mu[i,j]*C[j])
        b.append((C[i].norm(2))^2)
    k = 1
    while k < n:
        for i in range(k-1,-1,-1):
            if abs(Mu[k,i]) > 1/2:
                r = floor(Mu[k,i]+1/2)
                B[k] -= r*B[i]
                Mu[k,i] -= r
                for j in range(i):
                    Mu[k,j] -= r*Mu[i,j]
        if b[k] < (3/4-(Mu[k,k-1])^2) * b[k-1]:
```

```
            (B[k-1], B[k]) = (B[k], B[k-1])
            mu = Mu[k,k-1]
            bk1 = b[k]+(mu^2)*(b[k-1])
            bk_k1 = b[k]/bk1
            bk1_k1 = b[k-1]/bk1
            c = mu*bk1_k1
            for i in range(k+1,n):
                Mu[i, k-1:k+1] = [[(Mu[i,k]*(bk_k1)+c*Mu[i,k-1]), Mu[i,k-1]-mu*Mu[i,k]]]
            for j in range(k-1):
                [Mu[k-1,j], Mu[k,j]] = [Mu[k,j],Mu[k-1,j]]
            Mu[k,k-1] = c
            b[k] = b[k-1]*b[k] / bk1
            b[k-1] = bk1
            k = max(0,k-2)
        k += 1
    return B

def A1(F, H, p, m, k):
    R = PolynomialRing(ZZ,x)
    n = F.degree()
    d = H.degree()
    B = []
    for i in range(d):
        b = (p^k*x^i+x^(m+1)).coefficients(sparse=False)
        B.append(vector(b[0:m+1]))
    for i in range(m+1-d):
        b = (H*x^i+x^(m+1)).coefficients(sparse=False)
        B.append(vector(b[0:m+1]))
    B = Reduccion(B)
    if (B[0]).norm(2) >= ( p^(k*d)*(F.norm(2))^(-m) )^(1/n):
        return 0
    t = 0
    while t < len(B) and B[t].norm(2) < ( p^(k*d)*(F.norm(2))^(-m) )^(1/n):
        t += 1
    H0 = gcd(F, R(vector(ZZ, B[0]).list()))
    for i in range(1,t):
        H0 = gcd(H0, R(vector(ZZ, B[i]).list()))
    return H0

def A2(F, H, p):
    n = F.degree()
    d = H.degree()
    if d == n:
        return F
    u = floor((n-1)/d).exact_log(2)
    for i in range(u+1):
        m = floor((n-1)/2^(u-i))
        k = floor( log(2^(m*n/2)*binomial(2*m,m)^(n/2)*F.norm(2)^(m+n+1), p)/d ) + 1
        H2 = (Hensel(F, H, p, n = k))[1][0]
        if  m >= d:
            H0 = A1(F, H2, p, m, k)
            if H0 != 0:
                return H0
    return F

def LLL(F):
    R = F.parent()
    F = R(F)
    Fact = []
    n = F.degree()
    with localvars(R, ['x']):
        Res = F.resultant(diff(F,x))
        if Res == 0:
```

```
    G = gcd(F, diff(F,x))
    F = F/G
    Fact += LLL(G)
    Res = F.resultant(diff(F,x))
p = 2
Primos = Primes()
while Res%p == 0:
    p = Primos.next(p)
Z_p = Integers(p)
R2.<y> = PolynomialRing(Z_p)
F2 = R2(F)
L = Berlekamp(F2)
F_2 = F
while F_2 != 1 and F_2 != -1:
    f = L.pop()
    H0 = A2(F_2, R(f), p)
    Fact.append(H0)
    F_2 = R(F_2/H0)
    for i in L:
        if R2(i).divides(R2(H0)):
            L.remove(i)
return Fact
```

Bibliografía

[1] N. H. Abel. *Mémoire sur les équations algébrique: où on démontre l'impossiblité de la résolution de l'equation générale du cinquième dégré.* Librarian, Faculty of Science, University of Oslo, 1824.

[2] E. R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46(8):1853–1859, 1967.

[3] E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970.

[4] D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981.

[5] D. Cox, J. Little, and D. O'shea. *Ideals, varieties, and algorithms*, volume 3. Springer, 1992.

[6] Von Zur Gathen and Panario. Factoring polynomials over finite fields: A survey. *Journal of Symbolic Computation*, 31(1):3–17, 2001.

[7] K. O Geddes, S. R Czapor, and G. Labahn. *Algorithms for computer algebra.* Kluwer Academic Publishers, 1992.

[8] L. J. Goldstein. *Abstract algebra: a first course.* Englewood Cliffs, N.J.: Prentice-Hall, 1973.

[9] S. Hanif and M. Imran. Factorization algorithms for polynomials over finite fields. *Linnæus University*, 2011.

[10] B. A. Hausmann. A new simplification of kronecker's method of factorization of polynomials. *American Mathematical Monthly*, pages 574–576, 1937.

[11] B. A. Hausmann. *A new simplification of Kronecker's method of factorization of polynomials.* Oxford University Press, 1979.

[12] K. Hensel. *Theorie der algebraischen Zahlen*, volume 1. BG Teubner, 1908.

[13] K Hensel. Eine neue theorie der algebraischen zahlen. *Mathematische Zeitschrift*, 2:433–452, 1918.

[14] A. Hurwitz. Über die bedingungen, unter welchen eine gleichung nur wurzeln mit negativen reellen theilen besitzt. *Mathematische Annalen*, 46(2):273–284, 1895.

[15] L Kronecker. Grundzüge einer arithmetischen theorie der algebraischen grössen.(abdruck einer

festschrift zu herrn ee kummers doctor-jubiläum, 10. september 1881.). *Journal für die reine und angewandte Mathematik*, 92:1–122, 1882.

[16] S. Lang. Algebra revised third edition. *GRADUATE TEXTS IN MATHEMATICS-NEW YORK-*, 2002.

[17] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[18] M. Mignotte and D. Ştefănescu. *Polynomials: An algorithmic approach.* Singapore: Springer-Verlag, 1999.

[19] H. Zassenhaus. On hensel factorization, i. *Journal of Number Theory*, 1(3):291–311, 1969.

[20] H. Zassenhaus. A remark on the hensel factorization method. *Mathematics of Computation*, 32(141):287–292, 1978.