UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Departamento de Álxebra

Varias perspectivas sobre las bases de Gröbner: forma normal de Smith, algoritmo de Berlekamp y álgebras de Leibniz

Manuel Avelino Insua Hermo

2005

Varias perspectivas sobre las Bases de Gröbner: Forma Normal de Smith, Algoritmo de Berlekamp y Álgebras de Leibniz

Manuel Avelino Insua Hermo

Universidad de Santiago de Compostela

Departamento de Álxebra

Santiago de Compostela, 7 de Abril de 2005

Varias perspectivas sobre las Bases de Gröbner: Forma Normal de Smith, Algoritmo de Berlekamp y Álgebras de Leibniz

Fdo.: Manuel Avelino Insua Hermo

Memoria para optar al grado de Doctor realizada en el Departamento de Álxebra de la Universidad de Santiago de Compostela bajo la dirección del Profesor D. Manuel Ladra González.

Santiago de Compostela, a 7 de Abril de 2005.

Fdo.: Prof. Dr. Manuel Ladra González

A mi Padre Avelino,
mi Madre Marité,
mi Abuela Juana,
mi hermano Juanjo,
mi mujer Amalia y
mi hija Cristina M^a,
con todo mi cariño.

Agradecimientos

Llegado este momento se impone la necesidad, más que la simple obligación cortés, de expresar mi más sincero agradecimiento a aquellas personas que han contribuido, en una u otra forma a este trabajo. En primer lugar, sin duda, debo agradecer a mi director de tesis D. Manuel Ladra González, el apoyo tanto académico como personal que me ha prestado; así mismo quiero agradecerle desde aquí la confianza que ha mantenido en mí a lo largo de estos años. Sin saberlo Manolo, una simple llamada tuya para interesarte cómo me iba y darme ánimos, me ha servido para mantener la confianza en mí mismo, bien cuando veía que no daba avanzado o bien cuando al estar trabajando las ganas de continuar flaquean; por todo ello te estoy muy agradecido, muchas gracias Manolo. Así mismo quiero hacer extensivo mi agradecimiento a todos los profesores del Departamento de Álxebra, por el apoyo recibido.

Ya en el terreno más personal quiero desde aquí, agradecer a mis padres Avelino y Marité todo el cariño, apoyo incondicional y confianza en mis posibilidades que siempre me han mostrado, siempre habéis estado conmigo en los buenos y malos momentos; siempre habéis sabido darme el consejo adecuado; hora es ya de que retorne algo de lo que he recibido (por pequeña que sea esta satisfacción). Por todo ello, y seguramente mucho más que se me olvida, quiero expresar mi reconocimiento y mi cariño más sincero.

Continuando en orden cronológico, quiero desde aquí agradecer a mi mujer Amalia todo el apoyo y comprensión recibidos. Gracias por entender la importancia que esta tesis tiene para mí.

Resumen de la tesis de doctoramiento: Varias Perspectivas sobre las Bases de Gröbner: Forma Normal de Smith, Algoritmo de Berlekamp y Álgebras de Leibniz

En 1965 Buchberger introdujo el concepto de Base de Gröbner para un ideal del anillo de polinomios conmutativos y proporcionó un algoritmo de cálculo para calcular dichas bases. Desde entonces, la teoría de Bases de Gröbner ha experimentado un notable desarrollo, tanto en el terreno de sus aplicaciones que son abundantes y variadas, como en la extensión del concepto inicial de Base de Gröbner a otras estructuras matemáticas más complejas que el anillo de polinomios conmutativos.

En este trabajo se incide en estas dos vertientes de las Bases de Gröbner. A lo largo de los cuatros capítulos de los que consta se trata, en el primero el problema del cálculo de la Forma Normal de Smith utilizando la teoría de Bases de Gröbner. La idea clave que va a permitir hacer este cálculo vía Bases de Gröbner es el hecho de que estas bases proporcionan una matriz equivalente en filas con la matriz inicial. En el segundo capítulo se aborda el problema de la factorización de polinomios en una sola variable con coeficientes sobre un cuerpo finito a través del algoritmo de Berlekamp utilizando las Bases de Gröbner como herramienta de cálculo; además se programa el algoritmo obtenido de Berlekamp con bases de Gröbner en lenguaje Maple. En el tercer capítulo se aborda el problema de extender el concepto de Base de Gröbner al álgebra envolvente universal de un álgebra de Leibniz. El camino seguido para introducir este concepto en el álgebra envolvente fue a través del teorema de Poincaré-Birkhoff-Witt (que en este trabajo se prueba utilizando técnicas de Bases de Gröbner) y del concepto de FG-base de Gröbner, que es el concepto de Base de Gröbner definido para cocientes del álgebra asociativa no conmutativa libre. Por último en el capítulo cuatro se proporcionan los listados del paquete SmithGroebner en el cual se programaron en lenguaje Mathematica los resultados más destacados obtenidos en el primer capítulo.

Summary of the Ph.D. :
Varias Perspectivas sobre las Bases de Gröbner: Forma Normal de Smith, Algoritmo de Berlekamp y Álgebras de Leibniz

In 1965 Buchberger introduced the notion of Gröbner bases for a polynomial ideal and an algorithm for their computation. Since then, Gröbner bases has experimented a notable development in the area of its applications that they are abundant and assorted , as well as in the extension of the initial Gröbner bases concept to others more complex rings than the commutative polynomial ring.

This work has four chapters. In the first one, we calculate Smith Normal Form using Gröbner Bases. The key idea is that these bases provide a matrix which is equivalent with the initial matrix. In the second chapter we factor polynomials in one variable over a finite field through Berlekamp algorithm using Gröbner Bases, and we program Berlekamp with Gröbner Bases in Maple. In the third chapter we define the concept of Göbner Bases in the universal enveloping algebra of a Leibniz algebra. Poincaré-Birkhoff-Witt theorem, which we demonstrate using Gröbner Bases techniques, and FG-Bases are the essential tools to reach our objective. Finally, the main results of chapter one are programmed in package SmithGroebner which we write in Mathematica. The lists of SmithGroebner are chapter four.

Contenidos

Introducción

Algoritmos y Bases de Gröbner, son las palabras que mejor definen el contenido de esta memoria.

Muchos son los algoritmos matemáticos construidos desde el algoritmo de Euclides; éstos se han ido creando a lo largo de los siglos, en mayor o menor medida, en todas las ramas de las matemáticas para resolver constructivamente los más diversos problemas. El término algoritmo no deja de ser, en cierta forma paradójico, pues aunque en principio significa que se tiene un determinado procedimiento para resolver un cierto problema, en matemáticas por ejemplo, puede suceder en la práctica que éste no sea aplicable en general o en determinados casos, debido al gran número de operaciones que se deben realizar. Cierto es que desde la invención de las computadoras ha mejorado mucho la situación, se podría decir incluso que las computadoras han contribuido a "empequeñecer" en gran medida esta paradoja, aunque siga presente. La solución que se ha adoptado en la segunda mitad del siglo XX, frente a esta paradoja de tener un algoritmo, y por lo tanto la solución de un determinado problema, pero que éste no sea aplicable en un tiempo razonable debido a la gran cantidad de operaciones a realizar (hay algoritmos cuya aplicación llevaría a una computadora cientos de años) ha sido la de reformular el concepto mismo de algoritmo. Frente al concepto tradicional de algoritmo, que es considerado como un procedimiento por el cual en un número finito de pasos, partiendo de unos datos de entrada, se construye una solución a un determinado problema (son los que hoy en día se conocen

como algoritmos deterministas), surge un nuevo tipo de algoritmos, los algoritmos probabilísticos. Este tipo de procedimientos, que son muy utilizados en los test de primalidad y en la factorización de polinomios en una o varias variables, están basados en utilizar métodos probabilísticos para obtener el resultado deseado.

Hoy en día, no cabe la menor duda de que en matemáticas las palabras algoritmo y computadora van íntimamente ligadas, pues son las computadoras los instrumentos con los que poner en práctica los algoritmos desarrollados sobre el papel. El rol que han desempeñado y todavía desempeñan las computadoras no se quedó aquí, pues a su vez, el hecho de poder mediante una computadora, aplicar un algoritmo para resolver un problema determinado, ha servido como estímulo para el desarrollo de nuevos algoritmos; e incluso se puede decir más, el uso de las computadoras en matemáticas impulsó e impulsa la "parte constructivista", por llamarla así, de sus distintas ramas, de hecho, por ejemplo en Álgebra en la actualidad se habla de la disciplina de Álgebra Computacional.

Las aportaciones de las computadoras no se quedaron aquí, pues hubo quien intuyó una nueva funcionalidad: el cálculo simbólico.

La idea de hacer cálculo simbólico con un ordenador es relativamente reciente. Desde 1960 hasta nuestros días se han realizado numerosos lenguajes de computación simbólica, desde el pionero Lisp, pasando por Alpak, Camal, Formac, Saint, en la década de los años sesenta, hasta los más actuales como Axiom, Bergman, Cayley, Cocoa, Gap, Magma, Maple, Mathematica, Reduce y Singular.

Al igual que la idea de hacer cálculo simbólico con un ordenador, la construcción de algoritmos en el campo del álgebra conmutativa y geometría algebraica es también relativamente reciente. Hermann [Her26] fue una de los primeros matemáticos que se interesó en la construcción de algoritmos, en el campo de la teoría de ideales del anillo de polinomios en n variables sobre un cuerpo $\mathbb{K}$ ($n \geq 1$). Descubrió un algoritmo que calculaba la des-

composición primaria de un ideal y otro que servía para comprobar si un polinomio pertenecía a un ideal dado; éste último algoritmo reducía el problema a resolver un sistema de ecuaciones lineales. Más tarde Seidenberg [Sei74] publicaría una revisión de ambos métodos dados por Hermann.

El avance más significativo en la construcción de algoritmos para álgebra conmutativa y geometría algebraica, surge de la teoría y técnicas que desarrolló Bruno Buchberger en su tesis doctoral [Buc65]. La teoría de Buchberger o Bases de Gröbner, permite manipular polinomios en varias variables casi de la misma forma, que uno manipula polinomios en una variable. El algoritmo de Buchberger, piedra angular de la teoría de las Bases de Gröbner, es una generalización del Algoritmo de Euclides al caso multivariable, si se ve desde el punto de vista de la teoría de ideales sobre $\mathbb{K}[x_1, ..., x_n]$ ($\mathbb{K}$ cuerpo), debido a la unicidad del resto de la división de un polinomio por una Base de Gröbner. También se puede considerar como una generalización del algoritmo de eliminación de Gauss al caso no lineal, si se ve desde el punto de vista de la teoría de variedades, pues, en el caso lineal, los polinomios que definen una variedad lineal en el sistema final, tras haber aplicado Gauss, forman una Base de Gröbner respecto al orden monomial lexicográfico. Incluso se puede considerar como una generalización de la resultante de Sylvester para el cálculo del polinomio mínimo de dos polinomios en una variable.

Buscando las primeras "huellas" de lo que posteriormente serían las bases de Gröbner, es necesario retroceder hasta finales del siglo XIX, y situarse, con cierta sorpresa, en el contexto de la Teoría de Invariantes, más concretamente en el "tira y afloja" que mantuvieron dos matemáticos de prestigio de la época, éstos son Hilbert y Gordan.

En 1890 Hilbert dio a conocer su primera prueba del teorema de Finitud [H1890], que establece que el anillo de invariantes asociado a un grupo finito está finitamente generado. Aunque de gran interés teórico, su demostración no tenía por si misma gran interés práctico, pues la prueba dada por Hilbert es no constructiva; fue precisamente esta no constructividad de la demostración

dada por Hilbert el hecho que criticó Gordan, y es justo en este contexto al revisar la demostración dada por Hilbert, cuando Gordan pronunció la ya famosa frase: "esto es teología y no matemáticas". Posteriormente, el propio Gordan en cierta forma se retractó de lo dicho afirmando que: "la teología también tiene sus ventajas", al publicar él mismo [G1899] una nueva prueba del teorema de la Base de Hilbert (responsable de la no constructividad de la prueba de 1890); en el citado artículo utiliza por primera vez ideas similares a las bases de Gröbner, que él llama "le systeme irréductible N", y la generación finita de los ideales monomiales para deducir el teorema de la Base de Hilbert.

Con el transcurso del tiempo se ha podido comprobar que la demostración de 1890 no era en el fondo tan no constructiva, como en un principio se pensó, pues basándose en ella Derksen [Der99] ha construido un algoritmo de computación de los invariantes fundamentales de un grupo lineal reductivo, utilizando las bases de Gröbner como herramienta de cálculo.

Motivado por las críticas recibidas, Hilbert da tres años más tarde una nueva prueba, esta vez constructiva, del teorema de Finitud utilizando sistemas homogéneos de parámetros [H1893].

El uso de un concepto más o menos aproximado de lo que es una base de Gröbner no se quedó aquí; Macaulay [M1916] introdujo el concepto de H-base de un ideal en $\mathbb{K}[x_1, ..., x_n]$. Una H-base es un subconjunto finito F de $\mathbb{K}[x_1, ..., x_n]$ tal que a cada $0 \neq g \in < F >$ tiene una H-representación, esto es, una representación de la forma:

$$g = \sum_{f \in F} h_f \cdot f$$

con $max\{grado(h_f \cdot f)/f \in F\} \leq grado(g)$. Bajo un orden monomial adecuado, se puede probar que toda base de Gröbner es una H-base, aunque el recíproco no es cierto en general. Macaulay probó la existencia de una H-base para un ideal dado de forma no constructiva, como una simple aplicación del teorema de la Base de Hilbert.

También se debe a Macaulay la introducción de órdenes totales en el conjunto de los monomios de un anillo de polinomios [M1927], órdenes que utilizó para caracterizar las posibles funciones de Hilbert de ideales graduados, comparándolas con ideales monomiales.

Más tarde, Gröbner [Grö39] publicó aplicaciones de la idea de Macaulay de ordenar los monomios y de encontrar explícitamente una base del $\mathbb{K}$-espacio vectorial $\frac{\mathbb{K}[x_1,\ldots,x_n]}{I}$, siendo I un ideal de dimensión cero, aunque sólo resolvió parcialmente este problema.

En 1964, Gröbner propuso a su estudiante Bruno Buchberger el cálculo de tales bases como tema para su tesis doctoral. La intención de ambos era resolver el problema del cálculo de una $\mathbb{K}$-base de $\frac{\mathbb{K}[x_1,\ldots,x_n]}{I}$ siendo I un ideal de dimensión cero. Para su sorpresa, consiguieron desarrollar un algoritmo que era válido para cualquier ideal I. Incomprensiblemente los resultados obtenidos por Buchberger recibieron escasa atención hasta principios de los años setenta, fue entonces cuando Buchberger acuñó el término base de Gröbner.

En el terreno científico sucede muchas veces que las nuevas teorías suelen ser obtenidas casi simultánea e independientemente por distintos investigadores, de hecho así fue en este caso pues a la vez que se desarrollaba la teoría de las bases de Gröbner, Hironaka [Hir64] introduce, aunque de modo no constructivo, las bases estándar ("standard bases") para ideales en el anillo de series de potencias; estas bases han resultado ser análogas de las bases de Gröbner. El trabajo de Hironaka fue independiente del de Buchberger, y no fue hasta los años setenta cuando la analogía fue sacada a la luz. Knuth y Bendix [KnB70] desarrollaron la idea de la compleción del par crítico, estructura que recuerda al algoritmo de Buchberger, para la completa reescritura de sistemas de ecuaciones ("term-rewriting systems"). Incluso antes que el correspondiente concepto en álgebras asociativas conmutativas libres, las bases de Gröbner para ideales en álgebras de Lie libres fueron introducidas por Shirshov [Shi62].

Desde su creación, la teoría de las Bases de Gröbner ha experimentado un notable crecimiento. Si se restringe a sus aplicaciones, se ve que estas aparecen por doquier, en áreas tan diversas de las matemáticas como la integración indefinida de funciones racionales, la teoría de códigos, la estadística, las ecuaciones en derivadas parciales, la teoría de grafos, programación entera, funciones hipergeométricas, análisis numérico, diseño geométrico asistido por ordenador, teoría de homotopía combinatoria, etc (véase [BuW98]). En el caso particular del álgebra conmutativa y geometría algebraica, proporciona métodos para el cálculo de inversas de aplicaciones racionales, la resolución de sistemas de ecuaciones polinómicas, la inversión de matrices, el cálculo del máximo común divisor de varios polinomios, la pertenencia de un polinomio a un ideal del anillo de polinomios en n variables, el cálculo de las ecuaciones implícitas de una variedad a partir de sus ecuaciones paramétricas, etc (véase [CLO92, BeW93]).

El trabajo que a continuación se presenta consta de dos partes claramente diferenciadas. En la primera, se utilizan bases de Gröbner para calcular la Forma Normal de Smith (primer capítulo) y para calcular la factorización de un polinomio libre de cuadrados en una variable, con coeficientes en un cuerpo finito (segundo capítulo). En la segunda parte (tercer capítulo) se aborda un tema más arduo, la definición del concepto de base de Gröbner en el álgebra envolvente universal de un álgebra de Leibniz.

Como en todo trabajo de investigación, se plantean al principio del mismo los objetivos que se pretenden alcanzar, en este caso, el director de esta tesis propuso un doble objetivo, por una parte la obtención de la forma canónica de Jordan de una matriz dada, utilizando las Bases de Gröbner como herramienta de cálculo, y por otra, la definición de bases de Gröbner en el álgebra envolvente universal de un álgebra de Leibniz $\mathfrak{g}$ de dimensión finita.

El cálculo, utilizando bases de Gröbner, de la forma canónica de Jordan era un procedimiento que no se conocía, pero se tenía el convencimiento de

que tal procedimiento existía, de que había una conexión entre las bases de Gröbner y el cálculo de la forma canónica de Jordan. Puesto que las formas canónicas de las matrices (la forma canónica de Jordan, la forma canónica de Frobenius, la segunda forma canónica y la forma canónica de Jacobson [Ser02, Jea99]) se pueden obtener a partir de la Forma Normal de Smith, se cambió el objetivo inicial por uno más ambicioso, el cálculo de la Forma Normal de Smith con bases de Gröbner.

El primer paso que se dio para lograr calcular la Forma Normal de Smith utilizando las bases de Gröbner, fue el análisis del algoritmo clásico de obtención de dicha forma canónica [Jac74]. Un análisis con detalle muestra que es posible una reformulación de dicho algoritmo, las ideas básicas son ahora diagonalización y reordenación de los divisores elementales. Se llega de esta forma a un algoritmo nuevo de cálculo de la Forma Normal de Smith, la traducción de este algoritmo al contexto de bases de Gröbner proporcionará el procedimiento buscado.

Sea $A \in Mat_{m\times n}(D)$ una matriz con coeficientes en un DIP y el ideal I_A de $D[x_1, ..., x_n]$ generado por las filas de la matriz $A \cdot (x_1, ..., x_n)^T$.

El cálculo de la base de Gröbner del ideal I_A no es otra cosa que aplicar el algoritmo de eliminación de Gauss a la matriz A. Teniendo en cuenta que al aplicar Gauss a una matriz se están haciendo sólo operaciones en filas, y que para obtener la Forma Normal de Smith es necesario hacer, por norma general, operaciones en filas y columnas, ¿cómo es posible obtener la Forma Normal de Smith si no se está operando en columnas?

Esta dificultad que se presenta es fácilmente salvable, sin más que tener en cuenta que operar en columnas no es más que operar en filas en la traspuesta. De esta forma surge la idea de aplicar Gauss a una matriz (operaciones en filas), trasponer el resultado, volver a aplicar el algoritmo de Gauss (operaciones en columnas en la matriz original), trasponer el resultado y volver a empezar el proceso. Inmediatamente surge la pregunta de si esta sucesión de bases de Gröbner conduce a la Forma Normal de Smith. La respuesta, como a

continuación se verá es que no, este proceso conduce siempre hacia una matriz diagonal equivalente a la matriz inicial (diagonalización). No obstante, una vez aquí, la obtención de la Forma Normal de Smith se consigue sin mayor dificultad sin más que reordenar los divisores elementales (reordenación).

Obtenido el algoritmo se plantean otros dos desafíos no menos interesantes. El primero es la obtención de las matrices de paso y, como consecuencia, de las formas canónicas más conocidas: la forma canónica racional o de Frobenius, la segunda forma canónica, las formas canónicas de Jacobson y Jordan (objetivo inicial) pues todas ellas se pueden obtener a partir de la Forma Normal de Smith.

Un trabajo sobre un tema de álgebra computacional no se puede entender actualmente sin la programación de los algoritmos obtenidos, pues bien, este punto constituye el segundo desafío.

El planteamiento que se ha hecho, desde un principio, del cálculo de la Forma Normal de Smith ha sido tomar como base un DIP cualquiera, por coherencia con esta forma de proceder, se ha desarrollado un paquete en lenguaje Mathematica, el paquete SmithGroebner, que es capaz de operar en cualquier DIP que el usuario desee. Esta capacidad que posee el paquete SmithGroebner de manejar cualquier DIP, que es muy interesante desde un punto de vista matemático, paradójicamente es también su principal desventaja, pues al tener que programar en un DIP cualquiera, se está obligado "a priori" a utilizar rutinas lo más generales posibles, no contempladas en Mathematica y por lo tanto no compiladas, que ya de partida van a hacer más lento su funcionamiento. Aún así, no hay actualmente disponible ningún programa de computación simbólica que ofrezca unas prestaciones y una flexibilidad a la hora de escoger el DIP de trabajo, comparables a las que proporciona el paquete SmithGroebner, téngase en cuenta que se está hablando de un paquete en el que tiene cabida cualquier DIP.

Fermat es un lenguaje de computación simbólica que permite calcular la Forma Normal de Smith de matrices con coeficientes en $\mathbb{Q}, \mathbb{Z}$ o $\mathbb{F}_q$, posibilidad cubierta también por el paquete SmithGroebner.

Magma permite calcular la Forma Normal de Smith de matrices con coeficientes en un dominio euclídeo, posibilidad cubierta y ampliada por el paquete SmithGroebner, pues éste da la ocasión a mayores de trabajar en un DIP que no sea dominio euclideo, por ejemplo el anillo $\mathbb{Z}[\frac{1+\sqrt{-19}}{2}]$ [Cam88, Gre97].

Reduce [Hea95] posee el paquete *normform* para calcular la Forma Normal de Smith de matrices con coeficientes en $\mathbb{Z}$, $\mathbb{Q}[x]$ o en anillos de la forma $Q(y_1, ..., y_n)[x]$. También calcula la forma canónica de Frobenius, la forma canónica de Jacobson y la forma canónica de Jordan. Este paquete que posee Reduce es quizás el que más se acerca a las prestaciones que ofrece SmithGroebner.

Mathematica posee los paquetes *IntegerSmithNormalForm.m* y *PolynomialSmithNormalForm.m* que calculan la Forma Normal de Smith de matrices con coeficientes en $\mathbb{Z}$ y $\mathbb{Q}[x]$ respectivamente. El paquete SmithGroebner que aquí se presenta, se puede considerar como una extensión de estos dos paquetes.

Maple proporciona rutinas para calcular la Forma Normal de Smith de matrices con coeficientes en $\mathbb{Z}$ o $\mathbb{F}[x]$, siendo $\mathbb{F}$ un cuerpo, posibilidad cubierta por el paquete SmithGroebner.

Los DIPs que por defecto maneja el paquete SmithGroebner, aunque como se ha dicho es posible añadir cualquier otro que desee el usuario, son:[1]

$\mathbb{Z}$	$\mathbb{Z}[i]$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{C}$	$\mathbb{Z}_p$
$\mathbb{F}_q$	$\mathbb{Z}[\frac{1+\sqrt{-19}}{2}]$	$\mathbb{Q}[x]$	$\mathbb{R}[x]$	$\mathbb{C}[x]$	$\mathbb{Z}_p[x]$
$\mathbb{F}_q[x]$					

[1] $\mathbb{F}_q$: Cuerpo finito de q elementos

El paquete SmithGroebner es capaz de:

- Calcular la Forma Normal de Smith y las matrices de paso de una matriz con coeficientes en cualquier DIP de los anteriores.
- Calcular la inversa de una matriz con coeficientes en cualquier DIP de los anteriores.
- Calcular la forma canónica de Frobenius, la segunda forma canónica y las formas canónicas de Jacobson y Jordan, así como sus respectivas matrices de paso, de una matriz con coeficientes en uno de los cuerpos anteriormente citados.
- Calcular los generadores z_i de los cíclicos en los que descompone un $\mathbb{K}[x]$-módulo finitamente generado ($\mathbb{K}$ cuerpo) y los generadores de sus ideales aniquiladores $(0 : z_i)$; los cuales se obtienen como tesis en el Teorema de Estructura de Módulos finitamente generados definidos sobre un DIP.

Paralelamente al desarrollo del paquete SmithGroebner, se ha ido escribiendo un notebook en Mathematica en el que se explican los DIPs disponibles y el modo de funcionamiento de los comandos al servicio del usuario, que proporciona el paquete SmithGroebner. Dicha ayuda, una vez instalada, puede ser consultada dentro de la ayuda del propio Mathematica en la sección Add-ons, subsección bases de Gröbner

La última sección de este primer capítulo, está dedicada a la utilización del paquete SmithGroebner para mostrar algunas aplicaciones del cálculo de la Forma Normal de Smith, en tres campos bien distintos:

- La identificación de grupos abelianos finitamente generados.
- La resolución de sistemas de ecuaciones diferenciales lineales.
- La resolución de sistemas de ecuaciones diofánticas.

Durante el desarrollo del paquete SmithGroebner surgió la necesidad de factorizar polinomios en una variable sobre un cuerpo finito, para poder implementar los algoritmos de cálculo de las distintas formas canónicas que manejan los divisores elementales. Como Mathematica no tiene, hasta donde el que suscribe sabe, un comando específico, ni un paquete, de cálculo de dicha factorización, surgió la necesidad de programar el algoritmo de Berlekamp [Ber67] para obtenerla. El algoritmo de Berlekamp proporciona una forma rápida y elegante de factorizar polinomios sobre cuerpos finitos de orden pequeño q. La idea clave en la que se basa este algoritmo es la de sacar partido de las soluciones de la ecuación:

$$g(x)^q - g(x) = 0 \quad mod\ f(x)$$

siendo $f(x)$ el polinomio mónico de grado n a factorizar. Para a continuación resolver un cierto sistema de ecuaciones lineales de orden n, y con cada elemento $h(x)$ de una base del espacio solución, ir desgranando los factores irreducibles del polinomio $f(x)$ al utilizar la fórmula:

$$f(x) = \prod_{c \in \mathbb{F}_q} GCD(f(x), h(x) - c)$$

Este algoritmo también proporciona una forma de factorizar polinomios sobre $\mathbb{Z}$, pues basta con factorizar sobre $\mathbb{Z}_n$ siendo n un entero lo suficientemente grande [Ber72].

La pista que proporcionó la traducción del algoritmo de Berlekamp a un contexto con bases de Gröbner, y con ello el embrión de lo que sería el segundo capítulo, fueron los artículos de Lazard [Laz85] y Czichowski [Czi95]. En el último de estos artículos el autor, basándose en el trabajo de Lazard, calcula la integral de una función racional $\frac{P(x)}{Q(x)}$, siendo $P(x)$, $Q(x) \in \mathbb{K}[x]$ ($\mathbb{K}$ cuerpo de característica cero) y $Q(x)$ un polinomio mónico y libre de cuadrados. Para ello, da precisamente el método de calcular los máximos comunes di-

visores que se necesitan saber para aplicar el algoritmo de Berlekamp, pues el procedimiento que expone se traslada completamente de un contexto de característica cero a uno de característica p.

Es esta traducción del algoritmo de Berlekamp a un contexto con bases de Gröbner más su programación en lenguaje Maple, el contenido del segundo capítulo.

El tercer y último capítulo de esta tesis trata sobre el concepto de base de Gröbner en el álgebra envolvente universal $UL(\mathfrak{g})$ de un álgebra de Leibniz $\mathfrak{g}$ de dimensión finita.

Al comienzo del desarollo de este tercer y último capítulo, como no podía ser de otra forma, se revisaron varios trabajos existentes sobre la construcción de bases de Gröbner en distintas estructuras matemáticas [ApL88, Ape00, KaW90, Wei92, Mor94, BGC98, GRZ02, Rei95, Li02], llegando a la conclusión de que Buchberger en su tesis [Buc65] no sólo estableció qué es una base de Gröbner, sino que también marcó las pautas a seguir para definir el concepto de base de Gröbner en otras estructuras matemáticas, pues casi todas las construcciones que se han hecho, tratan de imitar en la medida de lo posible, el camino que marcó Buchberger.

Las pautas fijadas por Buchberger son:

- Definición de un orden monomial (el cual debe cumplir la condición de cadena descendente)

- Definición de un algoritmo de división (el cual termina en un número finito de pasos cuando se maneja un orden monomial, por cumplir éste la condición de cadena descendente)

Una vez que se han definido los dos puntos anteriores, se está en condiciones de definir, en primer lugar, el concepto de término principal de un polinomio y, en segundo lugar, el concepto mismo de base de Gröbner.

Así pues, los primeros esfuerzos estuvieron encaminados hacia la definición del concepto de orden monomial en $UL(\mathfrak{g})$. Tras varios intentos fallidos

debidos a la existencia en $UL(\mathfrak{g})$ de divisores de cero, que impedían que se cumpliese la condición de cadena descendente, y como no se intuía ninguna posible solución, se optó por probar por una vía diferente. Se empezó por profundizar un poco más en la estructura de $UL(\mathfrak{g})$, con la firme convicción de que un conocimiento más profundo de dicha álgebra, al final, conducirá al objetivo deseado.

Teniendo en cuenta que el álgebra tensorial es isomorfa como $\mathbb{K}$-álgebra al álgebra asociativa no conmutativa libre, es posible visualizar $UL(\mathfrak{g})$ como un cociente de la citada álgebra:

$$\pi : \mathbb{K}<x_1, ..., x_n> \twoheadrightarrow UL(\mathfrak{g})$$

con este enfoque, ya es posible probar la versión correspondiente del teorema de Poincaré-Birkhoff-Witt para $UL(\mathfrak{g})$ y constatar su noetherianidad.

Llegados a este punto, es conveniente recordar que el origen de las bases de Gröbner está relacionado con el cálculo de una $\mathbb{K}$-base del $\mathbb{K}$-espacio vectorial $\frac{\mathbb{K}[x_1,...,x_n]}{I}$ siendo I un ideal de $\mathbb{K}[x_1, ..., x_n]$. Si se traslada este hecho al contexto que nos ocupa, se tiene que para un ideal $\overline{J}$ de $UL(\mathfrak{g})$ el cociente

$$\frac{UL(\mathfrak{g})}{\overline{J}} = \frac{\frac{\mathbb{K}<x_1,...,x_n>}{Ker\pi}}{\frac{J}{Ker\pi}} \cong \frac{\mathbb{K} < x_1, ..., x_n >}{J}$$

esto es, el tercer teorema de isomorfía va a permitir "dar un paso atrás" y volver al álgebra asociativa no conmutativa libre, donde hay perfectamente establecida una teoría de bases de Gröbner [Mor94].

La forma más lógica de proceder será entonces definir el concepto de base de Gröbner en $UL(\mathfrak{g})$, que aquí se denomina FG-base de Gröbner finita, a partir del concepto de FG-base ("Factor Gröbner Basis") introducido en [Nor01] para un cociente del álgebra asociativa no conmutativa libre.

La ventaja de esta forma de proceder es muy importante pues, se evita trabajar directamente en $UL(\mathfrak{g})$, que es intrínsecamente más complicado que hacerlo en el álgebra asociativa no conmutativa libre. El enfoque que

aquí se presenta, además ofrece la posibilidad de calcular explícitamente FG-bases de Gröbner finitas, pues existen un lenguaje de computación simbólica, que es Bergman, y un paquete de Mathematica, que es NCAlgebra, que permiten calcular bases de Gröbner en el álgebra asociativa no conmutativa libre. Aunque se han utilizado ambos en esta memoria, es más recomendable utilizar NCAlgebra y no Bergman, pues éste último sólo es capaz de manipular polinomios homogéneos.

Una vez que se ha establecido el concepto de FG-base de Gröbner finita en $UL(\mathfrak{g})$, surge de modo inmediato la pregunta de si el desarrollo que se ha realizado es exportable a otros cocientes conocidos del álgebra asociativa no conmutativa libre, en los que ya existe una teoría establecida de bases de Gröbner, como por ejemplo el anillo conmutativo de polinomios en n variables o el álgebra envolvente universal de un álgebra de Lie, y comparar los conceptos de "FG-base de Gröbner finita" y de base de Gröbner. La conclusión a la que se llega es que la construcción de toda "FG-base de Gröbner finita" en ambas estructuras nos da una base de Gröbner y recíprocamente, toda base de Gröbner que tenga un número finito de elementos es una FG-base de Gröbner finita.

Capítulo 1

Bases de Gröbner y la Forma Normal de Smith

Una vez más, las bases de Gröbner demuestran su gran adaptabilidad para la resolución de los más diversos problemas; en esta ocasión, en el cálculo de la Forma Normal de Smith de una matriz con coeficientes en un dominio de ideales principales.

Aunque este hecho pueda sorprender en un principio, a lo largo de este capítulo se podrá comprobar que, el origen de tal cálculo se localiza en el algoritmo mismo de obtención de una base de Gröbner, "sólo hay que seguir" los polinomios que interesan.

1.1. Introducción

Casi tan antiguo como el concepto mismo de matriz, es el interés por calcular su forma canónica.

Para poder hablar de forma canónica, hay que, ciertamente, establecer previamente una relación de equivalencia entre matrices; respecto a la cual, una matriz es la forma canónica de otra. En este capítulo, dos matrices A y B

se dicen equivalentes si y sólo si, existen matrices inversibles P y R, llamadas matrices de paso, tales que B=PAR.

Actualmente, es conocido que toda matriz con coeficientes sobre un dominio de ideales principales, de m filas por n columnas, es equivalente a una matriz diagonal, que es la Forma Normal de Smith, cuyos elementos no nulos $d_1, ..., d_p$, que son los Factores Invariantes , verifican que d_1 divide a d_2, d_2 divide a d_3,..., d_{p-1} divide a d_p (véase [Jac74]). Este hecho fue probado por primera vez para matrices de números enteros por H. J. S. Smith [S1861] en 1861. A su vez, el trabajo de Smith se apoyó en un resultado que, diez años antes, Sylvester había publicado [S1851]: el máximo común divisor de los menores de orden i de B es igual al máximo común divisor de los menores de orden i de A.
Por aquel entonces, mediados del siglo XIX, muchos de los resultados más importantes sobre Teoría de Grupos Abelianos eran ya conocidos. Introducido por Gauss, el concepto de grupo abeliano se desarrolló simultáneamente en teoría de números por parte de Gauss, Schering, Kronecker y Dirichlet, y en teoría de funciones elípticas e integrales abelianas por Gauss, Abel y Jacobi. Sin embargo, no fue hasta 1879 cuando Frobenius y Stickelberger [F1879] descubrieron y utilizaron explícitamente la conexión entre grupos abelianos finitamente generados y el Teorema de Smith. En el mismo año, Frobenius probó que la Forma Normal de Smith extendida a matrices sobre anillos de polinomios, se puede utilizar para clasificar las matrices cuadradas con coeficientes en un cuerpo.

Posteriormente, se ha reinterpretado el cálculo de la Forma Normal de Smith en el contexto del álgebra moderna: La Forma Normal de Smith proporciona la descomposición en suma directa de módulos cíclicos, de un módulo de tipo finito definido sobre un dominio de ideales principales (véase [DuF99]).

Es evidente la ventaja, en lo que a simplicidad de cálculo se refiere, de manejar la forma canónica de Smith de una matriz. Es seguramente esta

ventaja, la que ha propiciado, sobre todo desde el uso masivo de los ordenadores, la gran literatura acerca de su cálculo, véase [Vil94, Vil95, Vil97, Mal72, HaH83, Ser02, Jac74].

En este capítulo se presenta una nueva aproximación al problema de la computación de la Forma Normal de Smith, desde la perspectiva de las bases de Gröbner. El objetivo principal de este primer capítulo es mostrar la estrecha relación existente entre el cálculo de la Forma Normal de Smith y la Teoría de Bases de Gröbner; en ningún caso se pretende, sea dicho esto ya de paso, la construcción de un algoritmo que sea más eficiente que los mejores algoritmos actuales [ChC82, KaB79].

El algoritmo clásico de obtención de la Forma Normal de Smith de una matriz de m filas y n columnas $A = (a_{ij})$ con coeficientes en un DIP, "grosso modo", consiste de $min(m, n)$ etapas. En la primera etapa, se obtiene a partir de A una matriz de la forma:

$$\begin{pmatrix} d_1 & 0 & ... & 0 \\ 0 & b_{22} & ... & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{m2} & ... & b_{mn} \end{pmatrix}$$

tal que $d_1|b_{ij}\ \forall i, j$. En la segunda, se vuelve a pasar el mismo proceso que se siguió en la primera etapa pero esta vez a la matriz:

$$\begin{pmatrix} b_{22} & ... & b_{2n} \\ \vdots & \ddots & \vdots \\ b_{m2} & ... & b_{mn} \end{pmatrix}$$

con lo cual se obtiene al final de esta etapa, una matriz de la forma:

$$\begin{pmatrix} d_1 & 0 & 0 & ... & 0 \\ 0 & d_2 & 0 & ... & 0 \\ 0 & 0 & c_{33} & ... & c_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & c_{m3} & ... & c_{mn} \end{pmatrix}$$

con $d_1|d_2|c_{ij}\ \forall i,j$. Tras $min(m,n)$ etapas como mucho, el proceso termina con la obtención de la Forma Normal de Smith:

$$\begin{pmatrix} d_1 & & & & & 0 \\ & \ddots & & & & \\ & & d_p & & & \\ & & & 0 & & \\ & & & & \ddots & \\ 0 & & & & & 0 \end{pmatrix}$$

con $d_1|d_2|...|d_p$.

Computacionalmente hablando, como posteriormente se comprobará, es un cálculo pesado, pedir que en la etapa i-ésima d_i divida a todos los elementos de la matriz que están por debajo de él. Surge de modo natural entonces la pregunta ¿qué sucede si se quita sólo esta condición de divisibilidad? La respuesta es casi inmediata pues, se tendría un nuevo algoritmo que en un número finito de etapas, conduciría a una matriz diagonal que tiene la misma Forma Normal de Smith que la matriz de partida. Una vez aquí, el paso a la Forma Normal de Smith es relativamente fácil, pues basta con reordenar los factores irreducibles en los que se descomponen los elementos de la diagonal principal [Mal72]. Se llega de este modo a un nuevo algoritmo de cálculo de la Forma Normal de Smith, las ideas básicas son ahora:

- Diagonalización

- Reordenación

La traducción a un contexto con bases de Gröbner del proceso de diagonalización, es la clave para la obtención del algoritmo buscado. Para dicha traducción se procede del siguiente modo. Es conocido, que para obtener la Forma Normal de Smith de una matriz, es necesario por norma general, hacer operaciones elementales en filas y columnas. Al aplicar el algoritmo de eliminación de Gauss a una matriz, se están haciendo sólo operaciones en

filas; se puede pensar entonces que no existe tal conexión entre las bases de Gröbner y el cálculo de la Forma Normal de Smith, pues no se actúa sobre las columnas de la matriz.

Esta dificultad que se presenta es fácilmente salvable, sin más que tener en cuenta que operar en columnas no es más que operar en filas en la traspuesta. Surge entonces de modo natural, la idea de aplicar el algoritmo de eliminación de Gauss a una matriz dada (=operaciones elementales en filas), trasponer el resultado y volver a aplicar el algoritmo de eliminación de gaussiana a esta nueva matriz (=operaciones elementales en columnas en la matriz original), trasponer el resultado y volver a empezar el proceso. Inmediatamente surge la pregunta de si esta "sucesión de bases de Gröbner" conduce o no a la Forma Normal de Smith, la respuesta a esta cuestión es que no, pues en general este proceso conduce a la obtención de una matriz diagonal que tiene la misma Forma Normal de Smith que la de partida.

Como posteriormente se comprobará, es precisamente esta forma de actuar, la traducción a un contexto con bases de Gröbner del proceso de diagonalización citado anteriormente, pues aunque en principio no lo parezca, un análisis más en profundidad mostrará que se está haciendo lo mismo en ambos casos.

Obtenido el algoritmo de cálculo de la Forma Normal de Smith, se plantean otros dos desafíos no menos interesantes. El primero de ellos es la obtención de las matrices de paso y, como consecuencia, de las formas canónicas más conocidas: la forma canónica Racional o de Frobenius, la segunda forma canónica, las formas canónicas de Jacobson y Jordan.

El punto clave para lograr calcular las matrices de paso, es darse cuenta que la Reordenación se puede lograr aplicando el algoritmo de Diagonalización a una matriz adecuada.

Si a una matriz $A = (a_{ij})$ de m filas y n columnas le aplicamos el algoritmo de diagonalización esbozado anteriormente, obtendremos una matriz diagonal $D = diag(d_1, ..., d_p, 0, ..., 0)$ que posee la misma Forma Normal de Smith que

la matriz inicial. Para conseguir la Forma Normal de Smith se procederá del siguiente modo, en primer lugar se construye la matriz:

$$D_1 = \begin{pmatrix} d_1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ \vdots & \ddots & \vdots & & & & & \vdots \\ d_r & \dots & d_r & & & & & \vdots \\ 0 & & & 0 & & & & \vdots \\ \vdots & & & & \ddots & & & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & \dots & 0 \end{pmatrix}$$

D_1 posee la misma Forma Normal de Smith que las matrices A y D. Si se aplica a D_1 el algoritmo de diagonalización se obtiene una matriz de la forma:

$$D_2 = \begin{pmatrix} mcd(d_1, ..., d_r) & & & & & & 0 \\ & d_2' & & & & & \\ & & \ddots & & & & \\ & & & d_r' & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ 0 & & & & & & 0 \end{pmatrix}$$

por la propia mecánica heredada del algoritmo clásico de cálculo de la Forma Normal de Smith, para matrices que sean de la misma forma que D_1. Aplicando de nuevo el proceso a la submatriz de D_2 obtenida al eliminar la primera fila y columna y así sucesivamente, se conseguirá en un número finito de etapas, obtener una matriz de la forma:

$$\begin{pmatrix} e_1 & & & & & 0 \\ & \ddots & & & & \\ & & e_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ 0 & & & & & 0 \end{pmatrix}$$

con $e_1|e_2|...|e_r$, esto es, habremos calculado la Forma Normal de Smith de la matriz A. Llegados a este punto, es posible concluir que el proceso de diagonalización es la clave para obtener la Forma Normal de Smith, a su vez, el algoritmo de eliminación de Gauss aplicado de forma reiterada, es el núcleo del algoritmo de Diagonalización.

Dada una matriz A de m filas y n columnas, las matrices de paso se van a ir construyendo de la siguiente forma, en primer lugar se aplica el algoritmo de eliminación de Gauss a la matriz $(A|I_m)$, siendo I_m la matriz identidad de orden m, de esta forma se obtiene una matriz $(B|P)$, y se consigue no sólo hacer operaciones en las filas de la matriz A, sino que además, en las posiciones de I_m se "guarda memoria" de lo hecho, esto es, $B = PA$. A continuación se repite el mismo proceso anterior pero para la traspuesta de B y la matriz I_n, con lo cual se obtendrá una matriz $(C|Q)$ verificando $C = QB^T$, se puede concluir entonces que la matriz traspuesta de C $(= C^T)$ es la matriz A después de haber realizado operaciones elementales en las filas y en las columnas, así pues $C^T = PAQ^T$. Aplicando el algoritmo de diagonalización de este modo, dentro del algoritmo de cálculo de la Forma Normal de Smith, es posible ir construyendo las matrices de paso.

La aproximación que en este capítulo se presenta, al cálculo de la Forma canónica Racional o de Frobenius es mediante un algoritmo determinista. Se han hecho también aproximaciones a este cálculo mediante algoritmos probabilísticos [GiS02].

En lo que sigue, D denota a un dominio de ideales principales (esto es, D es un anillo unitario y conmutativo tal que todo ideal está generado por uno de los elementos del propio ideal), m, n y r son números naturales.

1.2. Planteamiento Inicial

En esta segunda sección, se va a exponer el algoritmo de cálculo de la Forma Normal de Smith, véase [Jac74], que servirá como base a partir de

la cual, se construirá un nuevo algoritmo de obtención de la Forma Normal de Smith, que utiliza las bases de Gröbner como herramienta de cálculo. Para ello es necesario establecer, previamente, las siguientes definiciones y resultados.

Definición 1.2.1 *Dado $d \in D$, se denomina longitud de d, brevemente $l(d)$, al número de elementos primos de su factorización.*

Ejemplo 1.2.2 *Sea $D = \mathbb{Z}$*
$l(8) = l(2 \cdot 2 \cdot 2) = 3$
$l(6) = l(2 \cdot 3) = 2$
$l(-1) = 0$

Proposición 1.2.3 *Si $u \in D$ es unidad, entonces $l(u) = 0$.*

Definición 1.2.4 *Sobre el conjunto $Mat_{m\times n}(D)$ se define la siguiente relación binaria:*
$A, B \in Mat_{m\times n}(D)$ se dicen equivalentes , brevemente $A \simeq B$, si existen $P \in Mat_{m\times m}(D)$ y $R \in Mat_{n\times n}(D)$ ambas inversibles, tales que $B = PAR$.

Proposición 1.2.5 *$\simeq$ es una relación de equivalencia.*

Definición 1.2.6 *Las matrices P y Q se denominan matices de paso.*

Definición 1.2.7 *Se denomina matriz elemental , a una matriz que es de alguna de estas tres formas:*

- *Dado $b \in D$, $i, j, n \in \mathbb{N}$ tales que $i \neq j, i \leq n, j \leq n$.*
  $T_{ij}(b)$ es una matriz cuadrada de orden n, con un uno en cada posición de su diagonal principal, b en la posición (i,j) y cero en el resto.
- *Dado $u \in D$ unidad, $i, n \in \mathbb{N}$ tales que $i \leq n$.*
  $D_i(u)$ es una matriz cuadrada de orden n y diagonal, con u en el lugar (i,i) y un uno en las demás posiciones de su diagonal principal.

- *Dados $i, j, n \in \mathbb{N}$, tales que $i, j \leq n$.*
  P_{ij} es la matriz que se obtiene a partir de la matriz identidad de orden n, intercambiando las filas i y j.

Definición 1.2.8 *Dada $A \in Mat_{m\times n}(D)$, son operaciones elementales en A, la multiplicación por la izquierda o por la derecha, por matrices elementales.*

Las operaciones elementales son pues:

1. $T_{ij}(b) \cdot A \longrightarrow$ Matriz obtenida a partir de A, sumándole a la fila i la fila j multiplicada por b.
2. $A \cdot T_{ij}(b) \longrightarrow$ Matriz obtenida a partir de A, sumándole a la columna j la columna i multiplicada por b
3. $D_i(u) \cdot A \longrightarrow$ Matriz obtenida a partir de A, multiplicando la fila i por u.
4. $A \cdot D_i(u) \longrightarrow$ Matriz obtenida a partir de A, multiplicando la columna i por u.
5. $P_{ij} \cdot A \longrightarrow$ Matriz obtenida a partir de A, intercambiando las filas i y j.
6. $A \cdot P_{ij} \longrightarrow$ Matriz obtenida a partir de A, intercambiando las columnas i y j.

Observaciones:

- Multiplicar por la izquierda por una matriz elemental, conlleva una transformación en filas
- Multiplicar por la derecha por una matriz elemental, conlleva una transformación en columnas.

- Al realizar una operación elemental sobre una matriz, se obtiene una matriz equivalente a la inicial.

Teorema 1.2.9 *[Jac74] Cualquier matriz de m filas por n columnas con coeficientes en D, es equivalente a una matriz diagonal, cuyos elementos no nulos,* $d_1, ..., d_r$*, verifican que* d_1 *divide a* d_2*,* d_2 *divide a* d_3*,...,* d_{r-1} *divide a* d_r*. Además, esta matriz diagonal es única, salvo multiplicación de los elementos diagonales no nulos por unidades de D, y se denomina Forma Normal de Smith.*

Definición 1.2.10 *Sea* $A \in Mat_{m\times n}(D)$ *y* $S = diag(d_1, ..., d_r, 0, ..., 0)$ *su Forma Normal de Smith (*$d_1, ..., d_r$ *no nulos). Entonces:*

- $d_1, ..., d_r$ *se denominan factores invariantes de A.*
- *Considérense las factorizaciones en irreducibles de los factores invariantes:*

  $d_i = p_{i1}^{e_{i1}} \cdot ... \cdot p_{ir_j}^{e_{ir_j}}, p_{ij} \neq p_{is}$ *si* $j \neq s, i = 1, ..., r$

  Entonces cada $p_{ij}^{e_{ij}}$ *se denomina divisor elemental sobre D de A.*

La propia demostración del teorema anterior constituye un algoritmo de cálculo de la Forma Normal de Smith. Es precisamente este algoritmo el punto de partida; una reformulación del mismo más la traducción de esta reformulación al contexto de bases de Gröbner son los pasos que se seguirán para lograr el objetivo de este capítulo. A continuación se presenta la citada demostración en formato de algoritmo de cálculo de la Forma Normal de Smith; debido a su extensión y para facilitar su comprensión, se presenta troceado en dos rutinas, el algoritmo Smith (la rutina principal) y el algoritmo Divisible (una subrutina de la rutina principal).

Algoritmo 1.2.11 *Smith*
Entrada: $A \in Mat_{m\times n}(D)$ *tal que A es no nula.*
Salida: La Forma Normal de Smith de A.

1. *Se coloca, mediante transformaciones elementales (cambios de filas y columnas), un elemento no nulo y con longitud mínima, en la posición (1,1). Esta matriz así obtenida, es equivalente a la inicial y pasa a ser A.*

2. $A = Divisible(A)$

3. *Todos los elementos no nulos de la primera fila y columna son divisibles por a_{11}, a diferencia de lo que ocurre con el resto de los elementos de la matriz que no tienen por qué serlo. Mediante operaciones elementales sobre filas y columnas de la matriz A, se llega a una matriz de la forma:*

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & b_{22} & ... & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{m2} & ... & b_{mn} \end{pmatrix}$$

4. *Mientras a_{11} no divida a todos los elementos de la matriz.*

   a) *Se escoge un b_{ij} que no sea divisible por a_{11}.*

   b) $A = T_{1i}(1) \cdot A$ *(se le suma a la primera fila de A, la i-ésima)*

   c) $A = Divisible(A)$

   d) *Todos los elementos no nulos de la primera fila y columna, son divisibles por a_{11}, a diferencia de lo que ocurre con el resto de los elementos de la matriz que no tienen porque serlo. Mediante operaciones elementales sobre las filas y columnas de la matriz A, se llega a una matriz de la forma:*

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & b_{22} & ... & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{m2} & ... & b_{mn} \end{pmatrix}$$

5. *A es de la forma:*

$$\begin{pmatrix} d_{11} & 0 & ... & 0 \\ 0 & c_{22} & ... & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & c_{m2} & ... & c_{mn} \end{pmatrix}$$

verificando que d_{11} *divide a* c_{ij}, $i \in \{2, ..., m\}, j \in \{2, ..., n\}$

6. *Se aplica este algoritmo de nuevo, a la submatriz:*

$$\begin{pmatrix} c_{22} & ... & c_{2n} \\ \vdots & \ddots & \vdots \\ c_{m2} & ... & c_{mn} \end{pmatrix}$$

7. *Fin.*

Algoritmo 1.2.12 *Divisible*
Enrada: $A \in Mat_{m\times n}(D)$ *tal que A es no nula*
Salida: Una matriz equivalente a la matriz de entrada, que verifica que todo elemento no nulo de la primera fila y primera columna, es divisible por el elemento que ocupa la posición (1,1).

1. *Mientras todo elemento no nulo de la primera fila y columna no sea divisible por* a_{11}.

   a) *¿Existe en la primera fila un elemento no nulo que no sea divisible por* a_{11}*?*

      1) *Sí*

         i) *Se lleva este elemento hasta la posición (1,2) mediante un intercambio de columnas. A es ahora de la forma:*

$$\begin{pmatrix} a & b & a_{13} & & a_{1n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & & a_{mn} \end{pmatrix}$$

ii) Se calculan $d,\alpha,\beta,s,t \in D$ *tales que:*

- $d = mcd(a,b)$
- $\alpha \cdot a + \beta \cdot b = d$
- $a = s \cdot d$
- $b = t \cdot d$

iii) $A = \begin{pmatrix} a & b & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} \alpha & -t & 0 & \dots & 0 \\ \beta & s & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} =$

$$\begin{pmatrix} d & 0 & a_{13} & \dots & a_{1n} \\ \alpha \cdot a_{21} + \beta \cdot a_{22} & -t \cdot a_{21} + s \cdot a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha \cdot a_{m1} + \beta \cdot a_{m2} & -t \cdot a_{m1} + s \cdot a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix}$$

[Nota: $l(a) > l(d)$*. La longitud del elemento que ocupa la posición (1,1) disminuye.]*

b) *¿Existe en la primera columna un elemento no nulo que no sea divisible por* a_{11}*?*

1) Sí

i) Se lleva ese elemento hasta la posición (2,1), mediante un intercambio de filas. A es de la forma:

$$\begin{pmatrix} a & a_{12} & \dots & a_{1n} \\ b & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{3m} & \dots & a_{mn} \end{pmatrix}$$

ii) Se calculan $e,\alpha,\beta,s,t \in D$ *tales que:*

- $e = mcd(a,b)$
- $\alpha \cdot a + \beta \cdot b = e$
- $a = s \cdot e$
- $b = t \cdot e$

iii) $A = \begin{pmatrix} \alpha & \beta & 0 & \dots & 0 \\ -t & s & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} a & a_{12} & \dots & a_{1n} \\ b & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} =$

$$\begin{pmatrix} e & \alpha \cdot a_{12} + \beta \cdot a_{22} & \dots & \alpha \cdot a_{1n} + \beta \cdot a_{2n} \\ 0 & -t \cdot a_{12} + s \cdot a_{22} & \dots & -t \cdot a_{1n} + s \cdot a_{2n} \\ a_{31} & \dots & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & \dots & \dots & a_{mn} \end{pmatrix}$$

[Nota: $l(a) > l(e)$*. La longitud del elemento que ocupa la posición (1,1) disminuye.]*

2. Devuelve A

3. Fin.

Dos hechos fundamentales son los que sustentan este algoritmo. El primero, es pedir que el anillo base sea un dominio de ideales principales, pues esto garantiza que siempre existe el máximo común divisor de dos elementos y se verifica la identidad de Bezout. El segundo, es la ingeniosa introducción de la función longitud, de un elemento de un dominio de ideales principales. Con ella se relacionan los números naturales y el cálculo de la Forma Normal de Smith.

Gracias a la función longitud, se tiene asegurado que el algoritmo Smith termina tras un número finito de pasos, ya que si esto no fuese así, se podría construir una sucesión infinita de números naturales acotada superiormente y estrictamente decreciente, lo cual es imposible.

Que el conjunto de los números naturales sea bien ordenado, con respecto a la relación "$\leq$ ", es en última instancia, el garante de que el algoritmo Smith, termina tras un número finito de pasos. Esta propiedad del conjunto de los números naturales, más la función longitud, que actúa como un puente entre el algoritmo de Cálculo de la Forma Normal de Smith y el conjunto de los números naturales, garantizan que todo el proceso termina en un número finito de pasos.

1.3. Reformulación del algoritmo de cálculo

Ciertamente, el problema del cálculo teórico de la Forma Normal de Smith, queda resuelto con el algoritmo Smith. No obstante, desde un punto de vista práctico, no es el más adecuado por la cantidad de operaciones innecesarias que se hacen. No es preciso colocar un término de longitud mínima en la posición (1,1), aunque esta elección pueda en algunos casos acelerar, en otros retrasar, la obtención de la Forma Normal de Smith. Considérese, por ejemplo, el caso concreto de una matriz que contenga una unidad y ésta no esté en la posición (1,1); aplicar el paso 1 del algoritmo Smith, supone colocar la unidad en la posición (1,1). Gracias a esto, se pasa directamente al

paso 3. El paso 1 ha contribuido a acelerar la obtención de la Forma Normal de Smith, ya que de no haberse realizado sí se entraría probablemente en el paso 2. Por otra parte, considérese el caso hipotético de que el elemento que ocupa la posición (1,1), sea el elemento de longitud mínima. En esta ocasión, aplicar el paso 1 no modifica la matriz original y supone un retraso en la obtención de la Forma Normal de Smith.

A todo esto, se une el hecho de que obtener un elemento de longitud mínima, obliga a calcular por norma general, las longitudes de todos los elementos de la matriz y luego escoger uno de longitud mínima. Por otra parte, el cálculo de la longitud de un elemento, no es en general, un cálculo trivial.

Tampoco es necesario, que cada vez que la matriz sea de la forma:

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & b_{22} & ... & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{m2} & ... & b_{mn} \end{pmatrix}$$

exigir que a_{11} divida a todos los elementos de la matriz; lo que se pretende con esta condición, claro está, es obtener al final la Forma Normal de Smith. Como a continuación quedará patente, es posible sustituir este paso por otro que se aplica sólo al final de un proceso de cálculo y no en cada paso intermedio.

Un examen detallado del algoritmo Smith, revela que si se eliminan los pasos 1,4 y 5, el algoritmo resultante proporcionaría una matriz diagonal, equivalente a la inicial (hecho que se muestra en el algoritmo Diagonaliza, que se expone a continuación). Desde luego, no es posible esperar que esta matriz diagonal sea la Forma Normal de Smith, en algunos casos si lo será y en otros no. No obstante, como el conjunto de los divisores elementales de una matriz diagonal, es el mismo que el de su Forma Normal de Smith [Mal72], sólo queda reordenar este conjunto convenientemente para obtener la matriz que interesa. Esta reordenación se consigue haciendo un barrido sobre los elementos de la diagonal principal.

Estos dos hechos, permiten realizar una reformulación del algoritmo Smith; las ideas básicas son ahora:

- Diagonalización
- Reordenación de los divisores elementales.

Algoritmo 1.3.1 *SmithR (Reformulación)*
Entrada: $A \in Mat_{m\times n}(D)$ *tal que* A *es no nula.*
Salida: La Forma Normal de Smith de A

1. *A=Diagonaliza(A) /*Diagonalización*/*
2. *¿A es la Forma Normal de Smith? /*Reordenación*/*
   a) *No*
      - *Bucle desde i=1 hasta mínimo(m,n)-1*
        - $P=I_n + B$ *siendo*
          - I_n*: matriz identidad de orden n*
          - *B: matriz nula salvo en las posiciones* $\{(j,i)\}_{j=i+1}^{\mín(m,n)}$ *que contienen un uno.*
        - $A=A\cdot P$
        - *A=Diagonaliza(A)*
        - *¿A es la Forma Normal de Smith?*
          - *Sí*
            - *Salir del bucle.*
3. *Devuelve A*
4. *Fin*

Algoritmo 1.3.2 *Diagonaliza*
Entrada: $A \in Mat_{m\times n}(D)$ tal que A es no nula.
Salida: Una matriz diagonal equivalente a la inicial (son los pasos 2,3 y 6 del algoritmo Smith)

1. *A=Divisible(A)*

2. *Todos los elementos no nulos de la primera fila y columna son divisibles por a_{11}, a diferencia de lo que ocurre con el resto de los elementos de la matriz que no tienen porque serlo. Mediante operaciones elementales en filas y columnas se llega a una matriz de la forma:*

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & b_{22} & ... & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{m2} & ... & b_{mn} \end{pmatrix}$$

3. *Se aplica este algoritmo de nuevo, a la submatriz:*

$$B = \begin{pmatrix} b_{22} & ... & b_{2n} \\ \vdots & \vdots & \vdots \\ b_{m2} & ... & b_{mn} \end{pmatrix}$$

   esto es:

   - *B=Diagonaliza(B)*
   - $A = \begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & & & \\ \vdots & & B & \\ 0 & & & \end{pmatrix}$

4. *Fin*

Hay varias formas de reordenar los divisores elementales para conseguir los factores invariantes. Una de ellas viene reflejada en [Mal72], y está basada en la factorización en irreducibles de los elementos diagonales, y en su posterior reordenación según su exponente. Este no es el camino que se va a seguir; las ideas del algoritmo Smith, siguen siendo el punto de partida. Esta forma de proceder, viene justificada por el hecho de que se pretenden conseguir las matrices de paso, a medida que se va avanzando en el cálculo de la Forma Normal de Smith.

La justificación del paso 2 del algoritmo SmithR, es la siguiente:

Sea $A = diag(d_1, ..., d_r, 0, ..., 0) \in Mat_{m\times n}(D)$ una matriz diagonal

Primera iteración (i=1).

Se cambia A por la matriz equivalente que se obtiene al sumar a la primera columna, todas las columnas que están a su derecha (en el algoritmo esto se obtiene al hacer A·P). La matriz obtenida es de la forma:

$$\begin{pmatrix} d_1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ \vdots & \ddots & \vdots & & & & & \vdots \\ d_r & \dots & d_r & & & & & \vdots \\ 0 & & & 0 & & & & \vdots \\ \vdots & & & & \ddots & & & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & \dots & 0 \end{pmatrix}$$

Si se diagonaliza esta matriz con el algoritmo Diagonaliza, y se sigue el proceso, se observa que éste devuelve una matriz diagonal que tiene en su posición (1,1) el máximo común divisor de $d_1, ..., d_r$; esto es, tiene en su posición (1,1) el primer factor invariante de A.

Segunda Iteración (i=2).

Para calcular el segundo factor invariante, se procede de forma similar a la anterior. Primero se le suma a la segunda columna de la matriz obtenida en el paso previo, todas las columnas que están a su derecha. El resultado es una matriz de la forma:

$$\mathrm{E}=\begin{pmatrix} f_1 & 0 & & \dots & & 0 & \dots & 0 \\ 0 & e_1 & 0 & \dots & & & & \\ \vdots & \vdots & \ddots & & & & & \\ \vdots & e_r & \dots & e_r & & \vdots & & \vdots \\ & & & & \ddots & & & \\ 0 & & & & 0 & 0 & \dots & 0 \end{pmatrix}$$

Si se vuelve a aplicar el algoritmo Diagonaliza a esta nueva matriz se obtendrá otra diagonal que tiene en su posición (2,2), el primer factor invariante de la submatriz $diag(e_1, ..., e_r) \in Mat_{(m-1)\times(n-1)}(D)$ que es, el segundo factor invariante de la matriz E. Es importante destacar que la primera fila y la primera columna no influyen a la hora de aplicar el algoritmo Diagonaliza a la matriz E, es más, no sufren ninguna modificación.

Si se continua este proceso (el bucle del algoritmo SmithR), se obtendrá al final la Forma Normal de Smith.

A la vista del algoritmo SmithR, se puede observar que, la base sobre la que se apoya la reformulación del algoritmo de cálculo de la Forma Normal de Smith, es el proceso de diagonalización. De hecho, una traducción de este algoritmo de diagonalización al contexto de bases de Gröbner, proporcionaría de modo inmediato un algoritmo de cálculo de la Forma Normal de Smith. Este es, precisamente el objetivo de la próxima sección.

1.4. Una solución con Bases de Gröbner

Antes de exponer el teorema central, pilar básico como se verá, del algoritmo de computación de la Forma Normal de Smith, es necesario establecer previamente unos conceptos.

Definición 1.4.1 *Dada $A = (a_{ij}) \in Mat_{m\times n}(D)$, se llamará ideal asociado a A, brevemente I_A, al ideal de $D[x_1, ..., x_n]$ generado por:*

$$\begin{aligned} f_1 &= a_{11}x_1 + ... + a_{1n}x_n \\ &\vdots \\ f_m &= a_{m1}x_1 + ... + a_{mn}x_n \end{aligned}$$

Definición 1.4.2 *Dadas $A = (a_{ij}) \in Mat_{m\times n}(D)$ y $B \in Mat_{(m+r)\times n}(D)$, se dirá que B es igual a A salvo un número finito de filas nulas, brevemente $B =_0 A$, si:*

$$B = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix}$$

Proposición 1.4.3 *Dadas $A \in Mat_{m\times n}(D)$ y $B \in Mat_{(m+r)\times n}(D)$, se verifica:*
Si $B =_0 A$, entonces A y B tienen el mismo rango y los mismos factores invariantes.

Definición 1.4.4 $Mat_{\mathbb{N}\times n}(D) := \bigsqcup_{m\in\mathbb{N}} Mat_{m\times n}(D)$

Definición 1.4.5 *Sean $A, B \in Mat_{\mathbb{N}\times n}(D)$.*
B es equivalente a A salvo un número finito de filas nulas, brevemente $B \simeq_0 A$, si la Forma Normal de Smith de B es igual, salvo un número finito de filas nulas, a la Forma Normal de Smith de A.

Proposición 1.4.6 *$\simeq_0$es transitiva.*

Proposición 1.4.7 *Dadas $A, B \in Mat_{\mathbb{N}\times n}(D)$, se verifica:*
Si $B \simeq_0 A$, entonces A y B tienen el mismo rango y los mismos factores invariantes.

Fijado un orden monomial cualquiera en $D[x_1, x_2, ..., x_n]$ con $x_1 > x_2 > ... > x_n$, se establece:

Definición 1.4.8 *Dados* $f, g \in D[x_1, x_2, ..., x_n]$

1. *lp(f) denota la potencia principal de f.*
2. *lc(f) denota el coeficiente principal de f.*
3. *lt(f) denota al producto de lc(f) por lp(f).*
4. *La sicigia de f y g es:*
   $S(f,g) = \frac{mcm(lc(f),lc(g))}{lc(f)} \cdot \frac{mcm(lp(f),lp(g))}{lp(f)} f - \frac{mcm(lc(f),lc(g))}{lc(g)} \cdot \frac{mcm(lp(f),lp(g))}{lp(g)} g$

El teorema que sigue, es la base del algoritmo de cálculo que en este capítulo se presenta. Dada $A \in Mat_{\mathbb{N}\times n}(D)$, se demuestra que cualquier base de Gröbner Fuerte de I_A [AdL94], proporciona una matriz equivalente a A.

Teorema 1.4.9 *Sea* $A = (a_{ij}) \in Mat_{m\times n}(D)$ *una matriz no nula. Se verifica:*

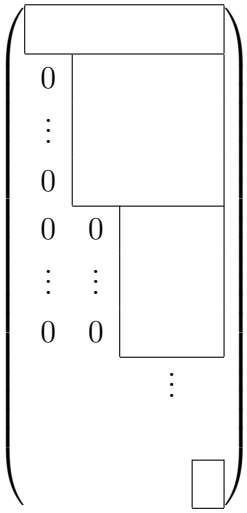
1. *Cualquier base de Gröbner Fuerte de* I_A *es de la forma:*
   $\{g_1 = a_1^1 x_1 + a_2^1 x_2 + ... + a_n^1 x_n, g_2 = a_2^2 x_2 + ... + a_n^2 x_n, ..., g_n = a_n^n x_n\} \cup B$
   siendo B, o el conjunto vacío, o un conjunto formado por polinomios homogéneos en $x_1, ..., x_n$*, de grado mayor o igual que 2.*
2. *La matriz*

$$Lad(A) := \begin{pmatrix} a_1^1 & \dots & a_n^1 \\ & \ddots & \vdots \\ & & a_n^n \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{pmatrix} \in Mat_{m\times n}(D)$$

que se obtiene a partir de la base de Gröbner Fuerte de apartado anterior, es equivalente a A.

3. *Existe* $P \in Mat_{m\times m}(D)$ *inversible, tal que* $Lad(A) = P \cdot A$

Demostración:
Sea $A = (a_{ij}) \in Mat_{m\times n}(D)$, tal que A es no nula. Se supondrá que las filas de A están ordenadas de la siguiente forma:



Esto es, en primer lugar las filas cuya primera columna es no nula, luego, a continuación, las filas cuya primera columna es nula y la segunda columna es no nula, y así sucesivamente. Esta ordenación no es restrictiva, esto es, no hay que reordenar la matriz antes de calcular la base de Gröbner. Es simplemente una cuestión de resaltar que se va a considerar cada bloque de filas por separado, y para simplificar notación. Se tiene entonces una partición de $\{1, ..., m\}$.

Se define:

- $I_1 = \{1, ..., p_1\}$, índices correspondientes a las filas de A cuya primera columna es no nula.
- $I_2 = \{p_1 + 1, ..., p_2\}$, índices correspondientes a las filas de A cuya primera columna es nula y la segunda es no nula.

- ...
- $I_n = \{p_{n-1} + 1, ..., m\}$, índices correspondientes a las filas de A cuya primera columna no nula es la columna n.

$$I_1 \sqcup I_2 \sqcup ... \sqcup I_n = \{1, ..., m\}.$$

Cabe la posibilidad de que uno o varios I_i sean el conjunto vacío.

Sea $\{f_1, ..., f_m\}$ el conjunto de generadores de I_A.

De aquí en adelante, se utilizará la palabra bloque, para designar el conjunto de todos los polinomios cuyos subíndices están en un mismo I_i.

$\{f_1, ..., f_m\}$ es un conjunto de polinomios homogéneos de grado 1. El carácter homogéneo de los polinomios iniciales es heredado por las sicigias y sus reducidos módulo un determinado conjunto de polinomios también homogéneos. Esto es, todos los polinomios no nulos que aparecen al aplicar el algoritmo de cálculo de una base de Gröbner, son homogéneos de grado mayor o igual que 1. Además, sólo al hacer la sicigia de dos polinomios cuyos subíndices están en el mismo I_k, se puede obtener un polinomio homogéneo de grado uno. Esto es así, por el carácter homogéneo de grado 1, de los polinomios iniciales.

A continuación se procede a aplicar el algoritmo de cálculo de una base de Gröbner a I_A. Se empieza por calcular las sicigias y sus reducidos, al llegar al cálculo de una sicigia de dos polinomios cuyos subíndices están en el mismo I_k, se razona de la siguiente forma:

Se supone que f_1 y f_2 están en el primer bloque, esto es $\{1, 2\} \subseteq I_1$, y se quiere calcular $S(f_1, f_2)$.

$$f_1 = a_{11}x_1 + ... + a_{1n}x_n;\ a_{11} \neq 0$$
$$f_2 = a_{21}x_1 + ... + a_{2n}x_n;\ a_{21} \neq 0$$

$$S(f_1, f_2) = \frac{mcm(a_{11},a_{21})}{a_{11}} f_1 - \frac{mcm(a_{11},a_{21})}{a_{21}} f_2 = \frac{a_{21}}{mcd(a_{11},a_{21})} f_1 - \frac{a_{11}}{mcd(a_{11},a_{21})} f_2 =$$

$$= \frac{a_{21}}{mcd(a_{11},a_{21})}(a_{11}x_1 + ... + a_{1n}x_n) - \frac{a_{11}}{mcd(a_{11},a_{21})}(a_{21}x_1 + ... + a_{2n}x_n) =$$

$$= \frac{a_{21}a_{12}-a_{11}a_{22}}{mcd(a_{11},a_{21})}x_2 + ... + \frac{a_{21}a_{1n}-a_{11}a_{2n}}{mcd(a_{11},a_{21})}x_n$$

Se comprueba que este polinomio se puede obtener, multiplicando por una matriz inversible.

Se define:

$$B = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \\ 0 & \dots & 0 \end{pmatrix} ; B =_0 A$$

$$\begin{pmatrix} 1 & 0 & \dots & & 0 \\ 0 & 1 & & & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ & & & \ddots & \vdots \\ \frac{a_{21}}{mcd(a_{11},a_{21})} & -\frac{a_{11}}{mcd(a_{11},a_{21})} & 0 \quad ... & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \\ 0 & \dots & 0 \end{pmatrix} =$$

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \\ 0 & \frac{a_{21}a_{12}-a_{11}a_{22}}{mcd(a_{11},a_{21})} & \dots & \frac{a_{21}a_{1n}-a_{11}a_{2n}}{mcd(a_{11},a_{21})} \end{pmatrix} = C$$

La última fila de C contiene los coeficientes de $S(f_1, f_2)$. Queda por calcular $\overline{S(f_1, f_2)}^{\{f_1,\dots,f_m\}}$.

$$\begin{array}{c|c|c|c} S(f_1, f_2) & f_1 & \dots & f_m \\ \hline f_s & q_1 & ... & q_m \end{array}$$

s es el índice que le corresponde al nuevo polinomio, hay que tener en cuenta que es posible que se hayan añadido otros polinomios antes de calcular esta sicigia.

$$S(f_1, f_2) = \sum_{i=1}^{m} q_i f_i + f_s \Longrightarrow f_s = S(f_1, f_2) - \sum_{i=1}^{m} q_i f_i$$

Esto es lo mismo que hacer:

$$\underbrace{\begin{pmatrix} 1 & 0 & & & 0 \\ 0 & 1 & & & \\ \vdots & \vdots & \ddots & & \\ 0 & 0 & & \ddots & 0 \\ -q_1 & -q_2 & \dots & -q_m & 1 \end{pmatrix}}_{Inversible} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \\ 0 & \frac{a_{21}a_{12}-a_{11}a_{22}}{mcd(a_{11},a_{21})} & \dots & \frac{a_{21}a_{1n}-a_{11}a_{2n}}{mcd(a_{11},a_{21})} \end{pmatrix} = D$$

La última fila de D contiene los coeficientes de f_s. Además se verifica:

$$D \simeq C \simeq B =_0 A$$

entonces,

$$D \simeq_0 A$$

y por lo tanto A y D tienen los mismos factores invariantes.

Si $f_s \neq 0$ entonces $lp(f_s) = x_k$ para algún $k \in \{1, ..., n\}$ y se añade el índice de este nuevo polinomio, s, a I_k. Es muy importante ir añadiendo cada índice nuevo al bloque al que pertenezca a medida que se van obteniendo los polinomios lineales, pues esta demostración se basa en los números que aparecen en cada I_k.

Así mismo, se lleva cuenta de las matrices por las que se va multiplicando (para al final obtener P); para ello se define:

$$P_1 = \begin{pmatrix} 1 & 0 & & & 0 \\ 0 & 1 & & & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & & \ddots & 0 \\ -q_1 & -q_2 & \dots & -q_m & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & & & & 0 \\ 0 & 1 & & & & 0 \\ \vdots & \vdots & & \ddots & & \vdots \\ & & & & \ddots & \vdots \\ \frac{a_{21}}{mcd(a_{11},a_{21})} & -\frac{a_{11}}{mcd(a_{11},a_{21})} & 0 & \dots & 0 & 1 \end{pmatrix}$$

$P_1 \cdot A' = D$ siendo A' la matriz A más una fila de ceros.

Se prosigue con los cálculos de las sicigias y sus reducidos, hasta que se llega a otra sicigia de dos polinomios cuyos índices vuelven a estar los dos en el mismo I_k. Razonando igual que antes, se define E como la matriz D más una fila de ceros. Calcular el polinomio reducido de la sicigia, no es más que multiplicar E por una determinada matriz inversible H; sea F el resultado.

Por ser $F \simeq E =_0 D$ se deduce que $F \simeq_0 D$, como además $D \simeq_0 A$, entonces se sigue que $F \simeq_0 A$

Para seguir llevando cuenta de las matrices por las que se multiplica, se define:

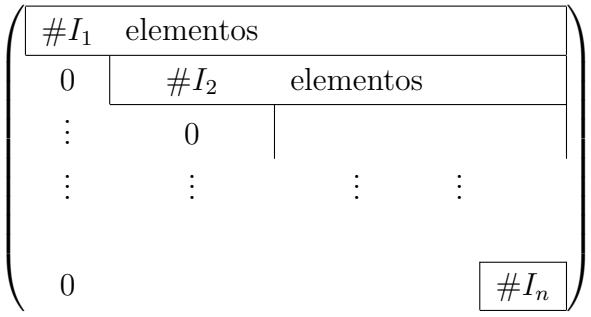
$$P_2 = H \cdot \begin{pmatrix} p_{11} & \dots & p_{1m+1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ p_{m+11} & \dots & p_{m+1m+1} & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

siendo $P_1 = \begin{pmatrix} p_{11} & \dots & p_{1m+1} \\ \vdots & \ddots & \vdots \\ p_{m+11} & \dots & p_{m+1m+1} \end{pmatrix}$

$P_2 \cdot A' = F$ siendo A' la matriz A más dos filas de ceros.

Siguiendo esta estrategia de cálculo, se llegaría, al finalizar la computación de la base de Gröbner de I_A, a una matriz Δ, de r filas y n columnas, que contendría a todos los polinomios de grado 1 de la base de Gröbner, y a una matriz Ω cuadrada e inversible de orden r, tales que $\Omega \cdot A' = \Delta$ (siendo A' la matriz A más r-m filas nulas). Además, por construcción, $\Delta \simeq_0 A$.

Δ se puede volver a pensar, para simplificar notación, que es de la forma:



Una vez obtenida una base de Gröbner de I_A, hay que calcular una base de Gröbner Fuerte para I_A.

$I_1, ..., I_n$ son conjuntos saturados [AdL94]. De cualquier otro conjunto saturado J, al calcular f_J se va a obtener un polinomio homogéneo de grado mayor o igual que dos. Sólo al calcular f_{I_k}, con I_k no vacío, se obtendrá un polinomio homogéneo de grado 1. Así, cada bloque I_k no vacío, contribuye con un polinomio f_{I_k}. Se ilustra este cálculo para I_1, para los demás casos es similar.

El primer paso es calcular $\{a_j\}_{j \in I_1}$ tales que:

$$\sum_{j \in I_1} a_j \cdot lc(f_j) = mcd(\{lc(f_j)/j \in I_1\}) \tag{1.1}$$

Por definición:

$$f_{I_1} = \sum_{j \in I_1} a_j \cdot f_j$$

De nuevo, este polinomio también se puede obtener multiplicando por la izquierda por una matriz inversible.

Se define γ como la matriz Δ más una fila de ceros ($\gamma =_0 \Delta$).

$$\underbrace{\begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ a_1 & \dots & a_{\#I_1} & 0 & \dots & 1 \end{pmatrix}}_{Inversible} \cdot \begin{pmatrix} \#I_1 & \text{elementos} & & \\ 0 & \#I_2 & \text{elementos} & \\ \vdots & 0 & & \\ \vdots & \vdots & \vdots & \vdots \\ 0 & & & \#I_n \\ 0 & & \dots & 0 \end{pmatrix} = \Sigma$$

La última fila de Σ está formada por los coeficientes de f_{I_1} en $x_1, ..., x_n$.

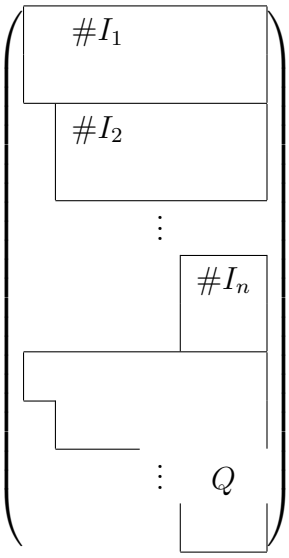
De nuevo $\Sigma \simeq \gamma =_0 \Delta$, y por lo tanto $\Sigma \simeq_0 \Delta$, como además $\Delta \simeq_0 A$, se verifica entonces que $\Sigma \simeq_0 A$.

No hay que olvidar que también se está llevando cuenta de las matrices por las que se multiplica. Se define:

$$\Psi = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & \ddots & \\ a_1 & \dots & a_{\#I_1} & 0 & \dots & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \Omega_{11} & & \Omega_{1r} & 0 \\ & & & \\ & & & \\ & & & \\ \Omega_{r1} & & \Omega_{rr} & 0 \\ 0 & & 0 & 1 \end{pmatrix},$$

$$\text{con } \Omega = \begin{pmatrix} \Omega_{11} & \dots & \Omega_{1r} \\ \vdots & \ddots & \vdots \\ \Omega_{r1} & \dots & \Omega_{rr} \end{pmatrix}$$

Razonando de igual modo con el resto de los bloques I_k no vacíos, se obtendrían $\sum_{Final} \in Mat_{u \times n}(D)$ y $\Psi_{Final} \in Mat_{u \times u}(D)$ tales que:

- $\sum_{Final} =$ 

- $\sum_{Final} = \Psi_{Final} \cdot A'$

Una base de Gröbner Fuerte de I_A será pues de la forma:

$$\underbrace{(\bigcup_{k/I_k \neq \emptyset} \{f_{I_k}\})}_{Q} \cup B$$

Con B un conjunto o bien vacío, o bien formado por polinomios homogéneos de grado mayor o igual que 2 (apartado 1 probado). Cualquier polinomio de grado 1, que esté en el ideal, va a reducirse a cero al dividir entre Q. Es por esto que, realizando operaciones elementales en filas sobre $\sum_{Final}$, y llevando cuenta de lo que se hace; al final se obtienen dos matrices $P' \in Mat_{u \times u}(D)$ y $X \in Mat_{u \times n}(D)$, tales que:

- $X = \begin{pmatrix} & & & \\ & & & Q \\ & & \vdots & \\ & & & \\ 0 & \dots & \dots & 0 \\ \vdots & & & \vdots \\ 0 & \dots & \dots & 0 \end{pmatrix}$, donde Q está formado por los coeficientes de los polinomios f_{I_k} con $I_k \neq \emptyset$.

- P' es inversible y $P' \cdot A' = X$, siendo A' la matriz A más $u - m$ filas de ceros.

La submatriz de X formada por los coeficientes de los elementos de Q, más las filas de ceros que hagan falta hasta completar una matriz de tamaño $m \times n$, es equivalente a A; pues tienen los mismos factores invariantes ($X \simeq_0 A$) y las mismas dimensiones. Esta matriz es $Lad(A)$ (apartado 2 probado). Del mismo modo, la submatriz P de P' formada por las m primeras columnas y las m primeras filas es inversible y verifica que $Lad(A) = P \cdot A$, esto es así por ser $P' \cdot A' = X$, y A' la matriz A más un número finito de filas nulas (apartado 3 probado). □

Con la demostración anterior en mente, es necesario realizar, antes de continuar, las siguientes puntualizaciones:

- *No es necesario un cálculo completo de la base de Gröbner Fuerte* para obtener los polinomios homogéneos de grado 1 de la base de Gröbner Fuerte de I_A, con un cálculo parcial basta. Es más, la forma de proceder en la demostración sugiere el método a seguir. Éste, consiste en calcular las sicigias y sus reducidos de los polinomios del primer bloque, añadir los polinomios resultantes a los bloques que pertenezcan, procesar el bloque 2 al igual que se hizo con el bloque 1, y así sucesivamente.

El último bloque no es necesario procesarlo, pues está formado por monomios y la sicigia de dos monomios es siempre cero.

- *Sobre la elección de los coeficientes* a_i en (1.1): aunque en principio, para obtener una matriz equivalente, cualquier familia de coeficientes que cumpla la condición vale, con el objeto de evitar bucles infinitos en el proceso que a continuación se describirá, se restringe su elección con la siguiente condición:
  Si alguno de los polinomios, tiene por coeficiente principal el máximo común divisor de los coeficientes principales, entonces se toma directamente, como integrante de la base de Gröbner Fuerte de I_A, al primer polinomio del bloque cuyo coeficiente principal sea el máximo común divisor de todos.
  A esta condición se hará referencia a lo largo del texto como $\{C1\}$. A partir de ahora, siempre que se hable de base de Gröbner Fuerte, se supondrá que se utilizó esta restricción a la hora de calcularla.

- *El primer bloque nunca aumenta de tamaño*, pues todas las sicigias tienen, como mucho, a x_2 por monomio principal. El resto de los bloques, si pueden aumentar de tamaño. Este hecho es relevante, desde el punto de vista de que ya se puede saber desde un principio, el polinomio que aporta el primer bloque a la base de Gröbner Fuerte de I_A.

A continuación, se ilustra la idea de la demostración del Teorema anterior, con un ejemplo

Ejemplo 1.4.10 *Se calculará, utilizando la forma de proceder en la demostración anterior, una matriz equivalente a*

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \end{pmatrix} \in Mat_{4\times 3}(\mathbb{Z})$$

$I_A :=< \{f_1 = x - y, f_2 = -5y + 6z, f_3 = 2x + 3y + 7z, f_4 = 3y + 2z\} >$
$I_1 = \{1, 3\}, I_2 = \{2, 4\}, I_3 = \emptyset$
$I_1 \cup I_2 \cup I_3 = \{1, 2, 3, 4\}$

En lo que sigue, para todo número natural i, se define A_i como la matriz A más i filas de ceros.

$S(f_1, f_3) = 2(x - y) - (2x + 3y + 7z) = -5y - 7z$

Matricialmente se corresponde con:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & -1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ \underbrace{\boxed{0 \quad -5 \quad -7}}_{S(f_1,f_3)} \end{pmatrix}$$

Se reduce $S(f_1, f_3)$

$$\begin{array}{l|l|l|l|l} -5y - 7z & x - y & -5y + 6z & 2x + 3y + 7z & 3y + 2z \\ \hline \dfrac{5y - 6z}{-13z = f_5} & & 1 & & \end{array}$$

Matricialmente:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & -5 & -7 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ \underbrace{\boxed{0 \quad 0 \quad -13}}_{f_5} \end{pmatrix} = C$$

Se lleva cuenta de las matrices por las que se multiplica.

$$P_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & -1 & 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 2 & -1 & -1 & 0 & 1 \end{pmatrix}$$

Se verifica:

$$C = P_1 \cdot A_1$$

Se añade 5 *a* I_3*, quedando entonces* $I_1 = \{1, 3\}, I_2 = \{2, 4\}, I_3 = \{5\}$

$S(f_2, f_4) = 3(-5y + 6z) + 5(3y + 2z) = 28z$

Matricialmente:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ \underbrace{\boxed{0 \quad 0 \quad 28}}_{S(f_2,f_4)} \end{pmatrix} = D$$

El resultado de reducir $S(f_2, f_4)$ por $\{f_1, f_2, f_3, f_4, f_5\}$ es $S(f_2, f_4)$. Se define por lo tanto $f_6 = S(f_2, f_4)$. Se añade $\{6\}$ a I_3, quedando entonces $I_1 = \{1, 3\}, I_2 = \{2, 4\}, I_3 = \{5, 6\}$.

Hay que seguir llevando cuenta de las matrices por las que se multiplica. Se define:

$$P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 \end{pmatrix} \cdot \left(\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 \end{pmatrix}$$

y se verifica que $P_2 \cdot A_2 = D$. Además una base de Gröbner para I_A es entonces de la forma $\{f_1, f_2, f_3, f_4, f_5, f_6\} \cup B$, siendo B, o el conjunto vacío, o un conjunto de polinomios homogéneos de grado mayor o igual que 2.

Falta por calcular una base de Gröbner Fuerte. Cada bloque I_k no vacío, contribuye con un polinomio.

Primer Bloque.

$I_1 = \{1, 3\}, f_{I_1} = f_1$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \left(\begin{array}{ccc} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & 28 \\ \hline 0 & 0 & 0 \end{array}\right) = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & 28 \\ \underbrace{\boxed{1 \quad -1 \quad 0}}_{f_{I_1}} \end{pmatrix} = E$$

Se define:

$$P_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \left(\begin{array}{cccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Se verifica que $E = P_3 \cdot A_3$

El segundo bloque está formado por dos polinomios ($I_2 = \{2, 4\}$)*; el máximo común divisor de sus coeficientes principales* ($mcd(-5, 3)$) *es 1. Teniendo en cuenta que* $1 = 1 \cdot (-5) + 2 \cdot 3$ *se define* $f_{I_2} = 1 \cdot (-5y + 6z) + 2 \cdot (3y + 2z) = y + 10z$.

$$F = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \left(\begin{array}{ccc} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & -28 \\ 1 & -1 & 0 \\ \hline 0 & 0 & 0 \end{array}\right) =$$

$$= \left(\begin{array}{ccc} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & -28 \\ 1 & -1 & 0 \\ \underbrace{\boxed{0 \quad 1 \quad 10}}_{f_{I_2}} \end{array}\right) = F$$

Se define:

$$P_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \left(\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Se verifica que $F = P_4 \cdot A_4$

El tercer bloque está formado por dos polinomios ($I_3 = \{5, 6\}$)*; el máximo común divisor de sus coeficientes principales* ($mcd(-13, 28)$) *es 1. Teniendo en cuenta que* $1 = (-13) \cdot (-13) - 6 \cdot 28$ *se define* $f_{I_3} = z$.

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -13 & -6 & 0 & 0 & 1 \end{pmatrix} \cdot \left(\begin{array}{ccc} 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & 28 \\ 1 & -1 & 0 \\ 0 & 1 & 10 \\ \hline 0 & 0 & 0 \end{array}\right) =$$

$$= \begin{pmatrix} \\ 1 & -1 & 0 \\ 0 & -5 & 6 \\ 2 & 3 & 7 \\ 0 & 3 & 2 \\ 0 & 0 & -13 \\ 0 & 0 & 28 \\ \underbrace{\begin{array}{|ccc|}\hline 1 & -1 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \\ \hline\end{array}}_{Q} \end{pmatrix} = G$$

$$P_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -13 & -6 & 0 & 0 & 1 \end{pmatrix} \cdot$$

$$\left(\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ -26 & -5 & 13 & -30 & -13 & -6 & 0 & 0 & 1 \end{pmatrix}$$

Se verifica que $G = P_5 \cdot A_5$. *Además sólo queda hacer operaciones elementales en* G, *para obtener la matriz buscada.*

$$\begin{aligned} f_1 &= x - y = 1 \cdot (x - y) + 0 \cdot (y + 10z) + 0 \cdot z \\ f_2 &= -5y + 6z = 0 \cdot (x - y) + (-5) \cdot (y + 10z) + 56 \cdot z \\ f_3 &= 2x + 3y + 7z = 2 \cdot (x - y) + 5 \cdot (y + 10z) + (-43) \cdot z \\ f_4 &= 3y + 2z = 0 \cdot (x - y) + 3 \cdot (y + 10z) + (-28) \cdot z \\ f_5 &= -13z = 0 \cdot (x - y) + 0 \cdot (y + 10z) + (-13) \cdot z \\ f_6 &= 28z = 0 \cdot (x - y) + 0 \cdot (y + 10z) + 28 \cdot z \end{aligned}$$

Por lo tanto:

$$\begin{aligned} T_{17}(-1) &\longrightarrow \textit{hace cero la primera fila de } G \\ T_{29}(-56) \cdot T_{28}(5) &\longrightarrow \textit{hace cero la segunda fila de } G \\ T_{39}(43) \cdot T_{38}(-5) \cdot T_{37}(-2) &\longrightarrow \textit{hace cero la tercera fila de } G \\ T_{49}(28) \cdot T_{48}(-3) &\longrightarrow \textit{hace cero la cuarta fila de } G \\ T_{59}(13) &\longrightarrow \textit{hace cero la quinta fila de } G \\ T_{69}(-28) &\longrightarrow \textit{hace cero la sexta fila de } G \end{aligned}$$

De este modo, si se define Ψ *de modo que* $\Psi := T_{17}(-1) \cdot T_{29}(-56) \cdot T_{28}(5) \cdot T_{39}(43) \cdot T_{38}(-5) \cdot T_{37}(-2) \cdot T_{49}(28) \cdot T_{48}(-3) \cdot T_{59}(13) \cdot T_{69}(-28)$ *entonces se verifica que*

$$\Psi \cdot G = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{pmatrix}$$

si además se intercambian las filas convenientemente, entonces

$$\underbrace{P_{39} \cdot P_{28} \cdot P_{17} \cdot \Psi}_{\Phi} \cdot G = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\Phi \cdot G = \Phi \cdot P_5 \cdot A_5 \begin{pmatrix} -1 & 1 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\Psi \cdot P_5 = \left(\begin{array}{cccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ -26 & -5 & 13 & -30 & -13 & -6 & 0 & 0 & 1 \\ -728 & -143 & 364 & -845 & -364 & -168 & 0 & -3 & 28 \\ \hline -336 & -66 & 168 & -390 & -168 & -78 & 0 & 0 & 13 \\ 728 & 143 & -364 & 845 & 364 & 169 & 0 & 0 & -28 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1456 & 286 & -728 & 1690 & 728 & 336 & 0 & 5 & -56 \\ -1120 & -220 & 560 & -1300 & -559 & -258 & -2 & -5 & 43 \end{array}\right)$$

$$Lad(A) = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ -26 & -5 & 13 & -30 \\ -728 & -143 & 364 & -845 \end{pmatrix}$$

La matriz P verifica que es invertible y que $Lad(A) = P \cdot A$

Para calcular, en general la Forma Normal de Smith de una matriz, es necesario hacer transformaciones en filas y columnas. Como se acaba de comprobar; mediante el cálculo de una base de Gröbner Fuerte del ideal asociado a una matriz dada, se puede obtener una matriz equivalente en filas a la primera. Al calcular una base de Gröbner Fuerte, no se realizan operaciones en las columnas de la matriz, sólo en las filas. Este hecho podría inducir a pensar que no hay un nexo claro entre la Teoría de las bases de Gröbner y el cálculo de la Forma Normal de Smith (falta operar en columnas). No

obstante, esta dificultad es fácilmente salvable teniendo en cuenta que las operaciones en columnas no son más, que operaciones en filas en la traspuesta. Estos dos hechos, por una parte que una base de Gröbner Fuerte proporcione una matriz equivalente en filas, y por otra, saber que las operaciones en columnas no son más que operaciones en filas en la traspuesta, son el origen de un concepto que va a desempeñar un papel central, en el algoritmo de diagonalización, el concepto de par equivalente.

Definición 1.4.11 *Se llamará par equivalente de una matriz no nula* $A = (a_{ij})$*, brevemente* $\widetilde{A}$*, a un par de matrices* $({}^1A = ({}^1a_{ij}), {}^2A = ({}^2a_{ij}))$*, tales que:*

- ${}^1A = Lad(A)$
- ${}^2A = Lad({}^1A^T)$*, siendo* ${}^1A^T$ *la matriz traspuesta de* ${}^1A$*.*

Lema 1.4.12 $A \simeq {}^1A \simeq {}^2A^T$

Lema 1.4.13 *Sea* i *un número natural tal que* a_{i1} *es el primer elemento no nulo de la primera columna de una matriz no nula* $A = (a_{ij})$*. Se verifica:* $l(a_{i1}) \geq l({}^1a_{11}) \geq l({}^2a_{11})$*.*

Demostración:
Trivial si se tiene en cuenta que ${}^1a_{11}$ $({}^2a_{11})$ es el máximo común divisor de un conjunto de elementos de D, entre los que está a_{i1} $({}^1a_{11})$.

□

Lema 1.4.14 *Sea* i *un número natural tal que* a_{i1} *es el primer elemento no nulo de la primera columna de una matriz no nula* $A = (a_{ij})$*.*
Si $l(a_{i1}) \geq l({}^1a_{11}) = l({}^2a_{11})$*, entonces* ${}^2A$ *es de la forma:*

$$\begin{pmatrix} {}^1a_{11} & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & R & \\ 0 & & & \end{pmatrix}$$

siendo R una matriz triangular superior.

Demostración:

- $\widetilde{A} = ({}^1A,\, {}^2A)$ con ${}^1A$ y ${}^2A$ triangulares superiores.
- ${}^2a_{11}$ es el máximo común divisor de los elementos no nulos de la primera columna de ${}^1A^T$, entre ellos ${}^1a_{11}$.
- $l({}^1a_{11}) = l({}^2a_{11})$ garantiza que ${}^1a_{11} = u \cdot {}^2a_{11}$, con u unidad en D. Luego ${}^1a_{11}$ es el máximo común divisor de los elementos no nulos de la primera columna de ${}^1A^T$. Este hecho, unido a la restricción {C1}, garantiza que el polinomio que aporta el primer bloque es ${}^1a_{11}x_1$. Con lo cual la matriz ${}^2A$ es de la forma:

$$\begin{pmatrix} {}^1a_{11} & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathrm{R} & \\ 0 & & & \end{pmatrix}$$

□

Definición 1.4.15 *Se entenderá por "cálculo reiterado del par equivalente de una matriz A", al proceso de construcción de la sucesión* $\{\widetilde{A}_n\}_{n\in\mathbb{N}\cup\{0\}}$*, donde*

$$\begin{array}{lllll} \widetilde{A}_0 = \widetilde{A} & & & & \\ \widetilde{A} & \longrightarrow & ({}^1A, {}^2A) & A_1 = {}^2A^T \\ \widetilde{A}_1 & \longrightarrow & ({}^1A_1, {}^2A_1) & A_2 = {}^2A_1^T \\ \widetilde{A}_2 & \longrightarrow & ({}^1A_2, {}^2A_2) & A_3 = {}^2A_2^T \\ \vdots & & & \end{array}$$

Lema 1.4.16 *La sucesión* $\{\widetilde{A}_n\}_{n\in\mathbb{N}\cup\{0\}}$*, está formada por matrices equivalentes todas entre sí.*

Demostración:
Trivial si se tiene en cuenta el Lema 1.4.12. □

Teorema 1.4.17 *El cálculo reiterado del par equivalente de una matriz no nula A, conduce a una matriz diagonal equivalente a A.*

Demostración:
Sea $A = (a_{ij}) \in Mat_{m\times n}(D)$ tal que A es no nula, $\widetilde{A} = ({}^1A, {}^2A)$ con ${}^1A = ({}^1a_{ij})$ y ${}^2A = ({}^2a_{ij})$. Sea además a_{i1} el primer elemento no nulo de la primera columna. Por el Lema 1.4.13 se verifica que $l(a_{i1}) \geq l({}^1a_{11}) \geq l({}^2a_{11})$. Sólo se pueden dar uno de estos cuatro casos:

$$\begin{array}{ll} i) & l(a_{i1}) > l({}^1a_{11}) > l({}^2a_{11}) \\ ii) & l(a_{i1}) > l({}^1a_{11}) = l({}^2a_{11}) \\ iii) & l(a_{i1}) = l({}^1a_{11}) > l({}^2a_{11}) \\ iv) & l(a_{i1}) = l({}^1a_{11}) = l({}^2a_{11}) \end{array}$$

Si se verifica $ii)$ o $iv)$, entonces ${}^2A$; por el Lema 1.4.14, va a ser de la forma:

$$\begin{pmatrix} {}^1a_{11} & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & R & \\ 0 & & & \end{pmatrix}$$

con R una matriz triangular superior.
Cada vez que se calcula el par equivalente de una matriz, se presenta la casuística anterior. En el proceso de cálculo reiterado del par equivalente, los casos i) y iii) no se pueden dar un número infinito de veces, ya que este hecho implicaría la existencia de una sucesión de números naturales estrictamente decreciente, acotada superiormente y con infinitos términos,

lo cual es imposible. Se puede asegurar entonces que, tras el cálculo de un número finito de términos de la sucesión $\{\widetilde{A}_n\}_{n\in\mathbb{N}\cup\{0\}}$, se llega a un término $\widetilde{A}_{p-1}$ tal que:

$$l({}_{p-1}a_{i1}) \geq l({}^1_{p-1}a_{11}) = l({}^2_{p-1}a_{11})$$

siendo $A_{p-1} = ({}_{p-1}a_{ij})$, ${}^1A_{p-1} = ({}^1_{p-1}a_{ij})$, ${}^2A_{p-1} = ({}^2_{p-1}a_{ij})$ y ${}_{p-1}a_{i1}$ el primer elemento no nulo de la primera columna de A_{p-1}. Por lo tanto:

$${}^2A_{p-1} = \begin{pmatrix} {}^1_{p-1}a_{11} & 0 & 0 \\ 0 & & \\ \vdots & & \\ 0 & & \end{pmatrix}$$

esto es:

$$A_p = \begin{pmatrix} {}^1_{p-1}a_{11} & 0 & 0 \\ 0 & & \\ \vdots & & \\ 0 & & \end{pmatrix}$$

Una vez que se llega a A_p, el comportamiento del término que ocupa la posición $(1,1)$, a lo largo del cálculo reiterado del par equivalente, se traslada al término que ocupa la posición $(2,2)$. Razonando de modo análogo, tras un número finito de pasos, se llega a un término:

$$\begin{pmatrix} a_1 & 0 & \dots & \dots & 0 \\ 0 & a_2 & & & \\ \vdots & & & & \\ \vdots & & & & \\ 0 & & & & \end{pmatrix}$$

Continuando este proceso, al final se llegaría a un término:

$$\begin{pmatrix} a_1 & 0 & \dots & 0 \\ 0 & \ddots & & \\ & & a_t & \\ \vdots & & 0 & \\ & & & \ddots \\ 0 & & & 0 \end{pmatrix}$$

$(t = rango(A))$ equivalente a A. □

A la vista de este resultado, la estrategia a seguir para diagonalizar una matriz es clara; basta con realizar un cálculo reiterado del par equivalente.

Algoritmo 1.4.18 *DiagonalizaG*
Entrada: $A \in Mat_{m\times n}(D)$ tal que A es no nula.
Salida: Una matriz diagonal equivalente a la inicial.

1. *Mientras A no diagonal.*
   a) *Calcular $\widetilde{A}$*
   b) $A = {}^2A^T$
2. *Return A*
3. *Fin*

Los algoritmos Diagonaliza y DiagonalizaG, comparten una misma forma de proceder, a la hora de diagonalizar una matriz; la estrategia que siguen consiste en:

$$\begin{pmatrix} a_{11} & \dots & \dots & a_{1n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{m1} & \dots & \dots & a_{mn} \end{pmatrix} \rightarrow \begin{pmatrix} d_1 & 0 & \dots & \dots & 0 \\ 0 & b_{22} & & \dots & b_{2n} \\ \vdots & & & & \\ \vdots & \vdots & & & \vdots \\ 0 & b_{m2} & & & b_{mn} \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} d_1 & 0 & \dots & \dots & 0 \\ 0 & d_2 & 0 & \dots & 0 \\ \vdots & 0 & c_{13} & \dots & c_{3n} \\ \vdots & \vdots & & & \\ & & & & \\ 0 & 0 & c_{m3} & \dots & c_{mn} \end{pmatrix} \rightarrow \begin{pmatrix} d_1 & & & & 0 \\ & \ddots & & & \\ & & d_r & & \\ & & & 0 & \\ & & & & \ddots \\ 0 & & & & 0 \end{pmatrix}$$

Meter ceros a la derecha y debajo de cada elemento de la diagonal principal. Desde este punto de vista, se puede afirmar que DiagonalizaG es la traducción al contexto de las bases de Gröbner del algoritmo Diagonaliza. Cabe esperar entonces que al reemplazar en el algoritmo SmithR, el algoritmo Diagonaliza por DiagonalizaG, se obtenga el algoritmo de cálculo de la Forma Normal de Smith que se busca.

Algoritmo 1.4.19 *SmithG (Reformulación)*
Entrada: $A \in Mat_{m\times n}(D)$ tal que A es no nula.
Salida: La Forma Normal de Smith de A

1. *A=DiagonalizaG(A) /*Diagonalización*/*
2. *¿A es la Forma Normal de Smith? /*Reordenación*/*
   a) *No*
      - *Bucle desde i=1 hasta mínimo(m,n)-1*
        - *P=I_n + B siendo*
          - *I_n: matriz identidad de orden n*
          - *B: matriz nula salvo en las posiciones $\{(j,i)\}_{j=i+1}^{\text{mín}(m,n)}$ que contienen un uno.*
        - *A=A·P*
        - *A=DiagonalizaG(A)*

- *¿A es la Forma Normal de Smith?*
  - *Sí*
    - *Salir del bucle.*

3. *Devuelve A*

4. *Fin*

Demostración:
Sólo queda por justificar el paso 2 de este algoritmo; para ello sea $A = diag(a_1, ..., a_r, 0, ..., 0)$.
Primera Iteración (i=1).
Hacer $A = A \cdot P$ cambia A por la matriz que se obtiene al sumarle a la primera columna de A, todas las columnas a su derecha. Esto es, A es ahora de la forma:

$$\begin{pmatrix} a_1 & & & & & \\ \vdots & \ddots & & & & \\ a_r & \dots & a_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

A continuación se aplica el algoritmo DiagonalizaG.
$\{a_1x_1, a_2x_1 + a_2x_2, \dots, a_rx_1 + a_rx_r\}$ es el conjunto de los polinomios que constituyen el primer bloque. ${}^1a_{11}$ es el máximo común divisor de $a_1, ..., a_r$, esto es, es el primer factor invariante de A. Con este hecho en mente, más la restricción $\{C1\}$, se puede asegurar que ${}^2A$ va a ser de la forma:

$$\begin{pmatrix} {}^1a_{11} & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \boxed{} & \\ 0 & & & \end{pmatrix}$$

Como resultado del algoritmo DiagonalizaG, se obtiene una matriz diagonal A, equivalente a la inicial y con el primer factor invariante situado en la posición $(1,1)$.
Segunda Iteración (i=2).
Hacer $A = A \cdot P$ cambia A por la matriz que se obtiene al sumarle a la segunda columna de A, todas las columnas a su derecha. Esto es, A es ahora de la forma:

$$\begin{pmatrix} d_1 & & & & & & \\ & b_2 & & & & & \\ & \vdots & \ddots & & & & \\ & b_r & \dots & b_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}$$

A continuación se aplica el algoritmo DiagonalizaG.
$\{b_2x_2, b_3x_2 + b_3x_3, \dots, b_rx_2 + b_rx_r\}$ es el conjunto de polinomios que constituyen el segundo bloque. Como sólo hay un polinomio en el primer bloque (a_1x_1), no es posible esperar que al segundo bloque se le añadan nuevos polinomios (resultado de la reducción de sicigias), es por esto, que es posible conocer de antemano el polinomio con el que contribuye a la base de Gröbner Fuerte. Este polinomio tiene por coeficiente principal al máximo común divisor de $b_2, \dots, b_r$, esto es, ${}^1a_{22}$ es el segundo factor invariante. Con este hecho en mente más la restricción $\{C1\}$, se puede asegurar que ${}^2A$ va a ser de la forma:

$$\begin{pmatrix} d_1 & & 0 \\ & {}^1a_{22} & \\ & & \boxed{} \\ 0 & & \end{pmatrix}$$

Como resultado del algoritmo DiagonalizaG, se obtiene una matriz diagonal A, equivalente a la inicial y con el primer factor invariante situado en la posición $(1,1)$, y el segundo factor invariante en la posición $(2,2)$. Continuando este bucle hasta su final, se llegaría a la Forma Normal de Smith.

□

No se puede terminar esta sección, sin hacer mención especial a la condición $\{C1\}$. Esta restricción es muy importante, pues garantiza que el algoritmo de diagonalización utilizando bases de Gröbner, no entra en un bucle infinito, esto es, que el proceso de cálculo reiterado del par equivalente, conduce a una matriz diagonal. Para comprender este hecho hay que remontarse al Teorema 1.4.9, siguiendo el hilo de su demostración, se ve que cada bloque no vacío, una vez que se han calculado todas las sicigias y sus reducidos, aporta a la base de Gröbner Fuerte un polinomio, que es combinación lineal con coeficientes en el anillo base D, de los polinomios que forman el bloque, y cuyo coeficiente principal es el máximo común divisor de los coeficientes principales de todos los polinomios que forman el bloque. El problema surge en la arbitrariedad a la hora de escoger esa combinación lineal; desde luego si lo que se quiere es calcular una matriz equivalente a la inicial, cualquier combinación lineal de los polinomios del bloque, que verifique la condición anterior, es válida.

No obstante, la situación cambia radicalmente a la hora de realizar un cálculo reiterado del par equivalente.

La condición $\{C1\}$ surge realmente del Lema 1.4.14, que se verifique $l({}^1a_{11}) = l({}^2a_{11})$, implica que ${}^1a_{11}$ es el máximo común divisor de los coeficientes principales de los polinomios del primer bloque. Pues bien, a la hora de calcular la primera fila de ${}^2A$, es este monomio, ${}^1a_{11}x_1$, el que se debe escoger para poder ir diagonalizando la matriz. Para asegurar que en el caso de que ${}^1a_{11}$ sea el máximo común divisor de los elementos no nulos

de la primera columna, se escogerá ${}^1a_{11}x_1$ como el polinomio que aporta el primer bloque, se impone la condición $\{C1\}$.

A continuación se expone un ejemplo en el que la no aplicación de $\{C1\}$, conduce a un bucle infinito al aplicar el algoritmo DiagonalizaG.

Ejemplo 1.4.20 *Se Diagonalíza siguiendo los pasos del algoritmo DiagonalizaG, la siguiente matriz.*

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

- *Cálculo de* $\widetilde{A}$.
  - *Cálculo de* ${}^1A$.
    - $\begin{array}{l} f_1 = x \\ f_2 = -x + y \end{array} \Longrightarrow \begin{array}{l} I_1 = \{1, 2\} \\ I_2 = \emptyset \end{array}$
    - $S(f_1, f_2) = y = f_3 \Longrightarrow I_2 = \{3\}$
    - $f_{I_1} = -x + y$ *(No se aplica la condición* $\{C1\}$ *y se escoge* f_2*).*
    - $f_{I_2} = y$
    - ${}^1A = Lad(A) = \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}$
  - *Cálculo de* ${}^2A$.
  - ${}^1A^T = \begin{pmatrix} -1 & 0 \\ 1 & 1 \end{pmatrix}$
  - $\begin{array}{l} f_1 = -x \\ f_2 = x + y \end{array} \Longrightarrow \begin{array}{l} I_1 = \{1, 2\} \\ I_2 = \emptyset \end{array}$
  - $S(f_1, f_2) = y = f_3 \Longrightarrow I_2 = \{3\}$
  - $f_{I_1} = x + y$ *(No se aplica la condición* $\{C1\}$ *y se escoge* f_2*).*
  - $f_{I_2} = y$

- ${}^2A = Lad({}^1A^T) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$

- $A = {}^2A^T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$
- *Cálculo de* $\widetilde{A}$.
  - *Cálculo de* ${}^1A$.
    - $\begin{matrix} f_1 = x \\ f_2 = x + y \end{matrix} \Longrightarrow \begin{matrix} I_1 = \{1, 2\} \\ I_2 = \emptyset \end{matrix}$
    - $S(f_1, f_2) = -y = f_3 \Longrightarrow I_2 = \{3\}$
    - $f_{I_1} = x + y$ *(No se aplica la condición* $\{C1\}$ *y se escoge* f_2*).*
    - $f_{I_2} = -y$
    - ${}^1A = Lad(A) = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$
  - *Cálculo de* ${}^2A$.
  - ${}^1A^T = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$
  - $\begin{matrix} f_1 = x \\ f_2 = x - y \end{matrix} \Longrightarrow \begin{matrix} I_1 = \{1, 2\} \\ I_2 = \emptyset \end{matrix}$
  - $S(f_1, f_2) = y = f_3 \Longrightarrow I_2 = \{3\}$
  - $f_{I_1} = x - y$
  - $f_{I_2} = y$
  - ${}^2A = Lad({}^1A^T) = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$

- $A = {}^2A^T = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$

Llegando de este modo a la matriz inicial, con lo que se podrían repetir los cálculos desde el principio una y otra vez.

1.5. Matrices de Paso

Si bien es importante conocer la Forma Normal de Smith de una matriz, también es importante conocer sus matrices de paso; sobre todo de cara a la utilización de la Forma Normal de Smith para la resolución de efectiva de problemas.

1.5.1. Construcción

El punto de partida para la construcción de las matrices de paso, es también el punto de partida del algoritmo DiagonalizaG; éste es el Teorema 1.4.9. Cada vez que se aplica este resultado, esto es, cada vez que se calcula para una matriz dada A, $Lad(A)$, existe una matriz P inversible tal que $Lad(A) = P \cdot A$. Pues bien, simplemente llevando cuenta de las matrices P, por las que se va multiplicando, se obtendrá al final tanto la Forma Normal de Smith, como sus matrices de paso.

Teorema 1.5.1 *Llevando cuenta de las matrices por las que se va multiplicando en el algoritmo SmithG, se obtienen las matrices de paso.*

Demostración:
Dada $B \in Mat_{m\times n}(D)$ no nula, se le aplica el algoritmo SmithG.

- $A = B$
- Paso 1: Aplicar DiagonalizaG a A.
  - Si A no es diagonal, se calcula $\widetilde{A}$.
    - ${}^1A = Lad(A)$
      Por el Teorema 1.4.9, existe $F_1 \in Mat_{m\times m}(D)$ inversible, tal que ${}^1A = Lad(A) = F_1A$.

- ${}^2A = Lad({}^1A^T)$
  Por el Teorema 1.4.9, existe $C_1 \in Mat_{n\times n}(D)$ inversible, tal que ${}^2A = Lad({}^1A^T) = C_1 \cdot {}^1A^T$
- $\begin{matrix} {}^1A = F_1A \\ {}^2A = C_1 \cdot {}^1A^T \end{matrix} \rightarrow {}^2A = C_1 \cdot {}^1A^T = C_1 \cdot (F_1B)^T = C_1 \cdot B^T \cdot F_1^T$
- Sea $A = {}^2A^T = F_1 \cdot B \cdot C_1^T$

- Si A no es diagonal, se calcula $\widetilde{A}$.

  - ${}^1A = Lad(A)$
    Por el Teorema 1.4.9, existe $F_2 \in Mat_{m\times m}(D)$ inversible, tal que ${}^1A = Lad(A) = F_2A$.
  - ${}^2A = Lad({}^1A^T)$
    Por el Teorema 1.4.9, existe $C_2 \in Mat_{n\times n}(D)$ inversible, tal que ${}^2A = Lad({}^1A^T) = C_2 \cdot {}^1A^T$
  - $\begin{matrix} {}^1A = F_2A \\ {}^2A = C_2 \cdot {}^1A^T \end{matrix} \rightarrow {}^2A = C_2 \cdot {}^1A^T = C_2 \cdot (F_2A)^T = C_2 \cdot A^T \cdot F_2^T$
  - $A = {}^2A^T (= F_2 \cdot F_1 \cdot B \cdot C_1^T \cdot C_2^T = F_2 \cdot F_1 \cdot B \cdot (C_2 \cdot C_1)^T)$

- Si A no es diagonal, se calcula $\widetilde{A}$.

- ...

Este proceso termina tras un número finito de pasos, con una matriz diagonal equivalente a la inicial y de la forma:

$$A = {}^2A^T = \underbrace{(F_r \cdot ... \cdot F_1)}_{F} \cdot B \cdot \underbrace{(C_r \cdot ... \cdot C_1)^T}_{C}$$

- Paso 2: Reordenación de los divisores elementales.

  - Bucle $i = 1, ..., \text{mín}(m, n) - 1$

    - $i = 1$

      - Se construye la matriz P

- $A = A \cdot P$
- $C = C \cdot P$
- Al aplicar DiagonalizaG a A se recogen las matrices D_1, G_1 por las que al multiplicar A se obtiene una matriz Diagonal (véase paso 1)
- $A = D_1 \cdot A \cdot G_1$
- $C = C \cdot G_1$
- $F = D_1 \cdot F$

- $i = 2$

...

Como resultado de este bucle se llegará a dos matrices F y C inversibles tales que la Forma Normal de Smith de B es: $F \cdot B \cdot C$. □

Para que el cálculo de las matrices de paso sea completo, sólo queda por explicar cómo, dada una matriz A, se puede obtener una matriz inversible P, tal que $Lad(A) = P \cdot A$.

El método natural es el que se expone en la demostración del Teorema 1.4.9; ir construyendo P, a medida que se avanza en el cálculo de la base de Gröbner Fuerte de I_A. No obstante, no es este el camino que aquí se seguirá, ya que supondría manejar matrices de tamaño arbitrariamente grande (recuérdese que el número de filas, depende del número de sicigias que se calculan para obtener una base de Gröbner Fuerte de I_A).

El proceso que se seguirá para obtener P, es el siguiente:
Dada $A \in Mat_{m\times n}(D)$, se construye la matriz $(A|I_m)$, esto es, a la matriz A se le añade la matriz identidad de orden m. Se calcula una base de Gröbner Fuerte para $I_{(A|I_m)} =< \{f_1(x_1, ..., x_n) + y_1, ..., f_m(x_1, ..., x_n) + y_m\} >$ con respecto a un orden monomial cualquiera con $x_1 > ... > x_n > y_1 > ... > y_m$. El conjunto de los polinomios de grado 1 de esta base de Gröbner Fuerte es de la forma:

$\{g_1(x_1,...,x_n)+h_1(y_1,...,y_m),...,g_t(x_1,...,x_n)+h_t(y_1,...,y_m),u_1(y_1,...,y_m),$
$...,u_s(y_1,...,y_m)\}$.

A la hora de calcular una base de Gröbner, es importante la elección del par de polinomios del que se va a calcular su sicigia. Escoger uno u otro par, supondrá al final, llegar seguramente a bases de Gröbner distintas. Si para calcular una base de Gröbner Fuerte de $I_A =< \{f_1,...,f_m\} >$, se siguen los mismos pasos que se siguieron en la construcción de la base de Gröbner Fuerte de $I_{(A|I_m)}$ para obtener $\{g_1(x_1,...,x_n)+h_1(y_1,...,y_m),...,g_t(x_1,...,x_n)+h_t(y_1,...,y_m)\}$, al final, cuando se complete el cálculo de la base de Gröbner Fuerte de I_A, se tendrá que el conjunto de polinomios de grado 1 para esa base de Gröbner Fuerte de I_A es $\{g_1(x_1,...,x_n),...,g_t(x_1,...,x_n)\}$; esto es así, pues se han hecho las "mismas cuentas" para I_A que para $I_{(A|I_m)}$, en lo que a cálculo de polinomios de grado 1 en $\{x_1,...,x_n\}$ se refiere. La única diferencia sustancial entre un cálculo y el otro es que en el segundo se arrastran términos adicionales (la parte que depende de $\{y_1,...,y_m\}$) al hacer las cuentas, y en el primero no. Este hecho se expresa y amplía en el siguiente resultado.

Teorema 1.5.2 *Dada $A \in Mat_{m\times n}(D)$, G una base de Gröbner Fuerte de $I_{(A|I_m)}$ y $Lad(A|I_m)$ la matriz que se obtiene a partir de G, entonces existe $P \in Mat_{m\times m}(D)$ inversible y existe una base de Gröbner Fuerte de I_A tal que su correspondiente matriz $Lad(A)$ verifica:*

- $Lad(A|I_m) = (Lad(A)|P)$
- $Lad(A) = P \cdot A$

Demostración:
Como se acaba de comprobar, es posible obtener los mismos polinomios de grado 1, salvo los sumandos que dependen de $\{y_1,...,y_m\}$, tanto al calcular una Base de Gröbner Fuerte de I_A como de $I_{A|I_m}$. Este hecho, garantiza que la submatriz de $Lad(A|I_m)$, formada por las primeras m filas y n columnas, es $Lad(A)$.

Por el Teorema 1.4.9 existe $P \in Mat_{m\times m}(D)$ inversible tal que:

$$Lad(A|I_m) = P \cdot (A|I_m) = (\underbrace{P \cdot A}_{Lad(A)}|P)$$

□

El papel que juega la matriz I_m al añadirse a A, es el de "guardar memoria" de la matriz de paso P. Con este hecho en mente, es fácil modificar los algoritmos DiagonalizaG y SmithG para obtener las matrices de paso.

Algoritmo 1.5.3 *DiagonalizaG_Paso*
Entrada: $A \in Mat_{m\times n}(D)$ *tal que* A *es no nula.*
Salida: Este algoritmo devuelve tres matrices B, F, C *tales que:*

- $B \in Mat_{m\times n}(D)$ *es diagonal y equivalente a* A.
- $F \in Mat_{m\times m}(D)$ *inversible.*
- $C \in Mat_{n\times n}(D)$ *inversible.*
- $B = F \cdot A \cdot C$

1. $F = I_m$; $C = I_n$ (I_j: *Matriz identidad de orden j)*
2. *Mientras* A *no diagonal.*
   - *Se calcula* $Lad(A|I_m)(= {}^1(A|I_m))$
     /*$Lad(A|I_m) = (Lad(A)|P); Lad(A) = P \cdot A$*/
   - ${}^1A = Lad(A)$
   - $F = P \cdot F$
   - *Se calcula* $Lad({}^1A^T|I_n)$
     /*$Lad({}^1A^T|I_n) = (Lad({}^1A^T)|Q); Lad({}^1A^T) = Q \cdot {}^1A^T$*/
   - ${}^2A = Lad({}^1A^T)$
   - $C = Q \cdot C$ /**Se traspone al final**/
   - $A = {}^2A^T$

3. $C = C^T; B = A$

4. *Devuelve* B, F, C

5. *Fin*

Algoritmo 1.5.4 *SmithG_Paso*
Entrada: $A \in Mat_{m\times n}(D)$ *tal que* A *es no nula.*
Salida: Este algoritmo devuelve tres matrices G,H,K tales que:

- $G \in Mat_{m\times n}(D)$ *es la Forma Normal de Smith de* A.
- $H \in Mat_{m\times m}(D)$ *inversible.*
- $K \in Mat_{n\times n}(D)$ *inversible.*
- $G = H \cdot A \cdot K$

1. $(A, H, K) = DiagonalizaG\_Paso(A)$

2. *¿*A *es la Forma Normal de Smith?*

   - *No*
     - $P = I_n + B$
       I_n *es la matriz identidad de orden n,*
       B *es la matriz nula, salvo en las posiciones* $(j, i)_{j=j+i}^{min(m,n)}$ *que contienen un uno.*
     - $A = A \cdot P$
     - $K = K \cdot P$
     - $(A, F, C) = DiagonalizaG\_Paso(A)$
     - $H = F \cdot H$
     - $K = K \cdot C$
     - *¿*A *es la Forma Normal de Smith?*

- Sí, entonces salir del bucle.

3. $G = A$

4. Devuelve G,H,K

5. Fin

La forma de proceder en el algoritmo SmithG_Paso, es el modo más completo de calcular la Forma Normal de Smith, puesto que, a medida que se va avanzando en su cálculo, se van construyendo las matrices de paso. Además, todo este proceso se realiza utilizando las bases de Gröbner como herramienta de cálculo.

1.5.2. Cálculo de la inversa de una matriz

Seguramente llama la atención, tratar a estas alturas de capítulo, el tema de decidir si una matriz es inversible o no, y en caso afirmativo calcular su inversa. No obstante, esto está plenamente justificado, pues en la subsección siguiente de cálculo de Formas Canónicas, se necesitará calcular inversas de matrices.

Dos son los algoritmos que se presentan, y aunque sólo el primero de ellos es válido para su utilización en la siguiente subsección, ambos ponen de manifiesto la relación entre el cálculo de la inversa de una matriz y el cálculo de la Forma Normal de Smith. El primero se fundamenta en el cálculo de las matrices de paso, y el segundo en la obtención del polinomio mínimo.

Cálculo de la Inversa y Matrices de Paso.

Sea D, como hasta ahora, un dominio de ideales principales y $A \in Mat_n(D)$. Entonces, existen matrices S, P, Q de orden n, tales que:

- S es la Forma Normal de Smith de A.

- P y Q son inversibles.
- $S = P \cdot A \cdot Q$

De la última igualdad se concluye que para que A sea inversible, todos los elementos que forman la diagonal principal de S, deben ser unidades de D. De ser cierto $A^{-1} = Q \cdot S^{-1} \cdot P$.

Este razonamiento da pie al siguiente algoritmo:

Algoritmo 1.5.5 *Inversa_Paso*
Entrada: $A \in Mat_n(D)$ *tal que* A *es no nula.*
Salida: Si A *es inversible devuelve su inversa, de no serlo devuelve* 0.

1. $(S, P, Q) = SmithG\_Paso(A)$ /* $S = P \cdot A \cdot Q$ */
2. *¿Todos los elementos de la diagonal principal de* S *son unidades en* D*?*
   - *Sí.*
     - $Inversa\_A = Q \cdot S^{-1} \cdot P$
   - *No.*
     - $Inversa\_A = 0$
3. *Devuelve* $Inversa\_A$
4. *Fin*

Cálculo de la Inversa y el Polinomio Mínimo.

En esta subsección se parte de un cuerpo conmutativo $\mathbb{K}$ y de una matriz $A \in Mat_n(\mathbb{K})$. Es conocido que el polinomio mínimo de A, $m(x) = x^m + a_{m-1}x^{m-1} + ... + a_0$, es el factor invariante de mayor grado de la matriz $xI_n - A$, donde I_n es la matriz identidad de orden n [DuF99].

Que la matriz A sea inversible depende del término independiente de $m(x)$, pues $a_0 = 0$ es condición necesaria y suficiente para que la matriz

no sea inversible. Por otra parte, si a_0 no fuese cero, se podría razonar del siguiente modo para obtener A^{-1}.

$$m(A) = 0 \Longleftrightarrow A^m + a_{m-1}A^{m-1} + ... + a_0 I_n = 0 \overset{a_0 \neq 0}{\Longleftrightarrow}$$

$$A^{m-1} + a_{m-1}A^{m-2} + ... + a_0 A^{-1} = 0 \Longleftrightarrow$$

$$A^{-1} = \tfrac{-1}{a_0} \cdot (A^{m-1} + a_{m-1}A^{m-2} + ... + a_1 I_n)$$

Estos hechos dan pie al siguiente algoritmo.

Algoritmo 1.5.6 *Inversa_Minimo*
Entrada: $A \in Mat_n(\mathbb{K})$ *tal que* A *es no nula.*
Salida: Si A *es inversible devuelve su inversa, de no serlo devuelve* 0.

1. $B = SmithG(x \cdot I_n - A)$
2. *Sea* f *el factor invariante de mayor grado de* B
   - f *es el polinomio mínimo de* A.
   - $f = x^m + a_{m-1} \cdot x^{m-1} + ... + a_0$
3. *¿*$a_0 = 0$*?*
   - *Sí.* /* A *no es inversible* */
     - $Inversa\_A = 0$
   - *No.*
     - $Inversa\_A = \tfrac{-1}{a_0} \cdot (A^{m-1} + a_{m-1}A^{m-2} + ... + a_1 I_n)$
4. *Devuelve* $Inversa\_A$
5. *Fin*

1.5.3. Cálculo de Formas Canónicas

Mediados del siglo XIX; ésta es la época en la que se descubrió la Forma Normal de Smith [S1861]. Aunque en su origen estaba definida sobre matrices de números enteros, con el tiempo, este concepto se ha interpretado en el contexto de la Teoría de Módulos: La Forma Normal de Smith, proporciona la descomposición en suma directa de módulos cíclicos, de un módulo de tipo finito sobre un dominio de ideales principales. Esta reinterpretación del concepto, se recoge en el siguiente Teorema [Jac74]:

Teorema 1.5.7 *Teorema de Estructura de Módulos de Tipo Finito .- Sea D un dominio de ideales principales y M un D-módulo de tipo finito. Entonces, M es suma directa de D-módulos cíclicos:*

$$M = Dz_1 \oplus ... \oplus Dz_s$$

tal que:

$$(0 : z_1) \supseteq (0 : z_2) \supseteq ... \supseteq (0 : z_s);\ (0 : z_k) \neq D,\ k = 1, ..., s$$

Este teorema de estructura, es el pilar central sobre el que se apoya, el desarrollo para la obtención de las Formas Canónicas de una matriz, que en esta subsección se presenta. De hecho, su demostración [Jac74] es el punto de partida.
Cuatro son las Formas Canónicas que se obtendrán [Ser02, Bar96]:

1. La Forma Canónica Racional o de Frobenius.
2. La Segunda Forma Canónica.
3. La Forma Canónica de Jacobson.
4. La Forma Canónica de Jordan.

La mecánica para obtener una de las Formas Canónicas anteriores, es siempre la misma. Primero, se descompone el espacio vectorial sobre el que está definido el endomorfismo, en suma directa de subespacios invariantes (es en este punto donde interviene el Teorema de Estructura). Segundo, se elige un base adecuada en cada subespacio. La unión de todas estas bases, proporcionará la base en la cual el endomorfismo se expresa matricialmente en la forma canónica buscada.

Sea $\mathbb{K}$ un cuerpo conmutativo, V un $\mathbb{K}$-espacio vectorial de dimensión finita n, $B = \{v_1, ..., v_n\}$ una $\mathbb{K}$-base de V y $T : V \longrightarrow V$ una aplicación $\mathbb{K}$-lineal.

$$\begin{array}{rccl} T: & V & \longrightarrow & V \\ & v_1 & \longrightarrow & T(v_1) = a_{11}v_1 + ... + a_{n1}v_n \\ & & \vdots & \\ & v_n & \longrightarrow & T(v_n) = a_{1n}v_1 + ... + a_{nn}v_n \end{array}$$

La matriz de T en la base B es:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

Tomando como ley de composición externa Θ:

$$\begin{array}{ccc} \mathbb{K} \times V & & \\ \downarrow & \overset{*}{\searrow} & \\ \mathbb{K}[x] \times V & \xrightarrow{\Theta} & V \end{array}$$

siendo:

- $\Theta(a_m x^m + ... + a_0, v) := a_m T^m(v) + ... + a_0 v$
- $*$ la operación externa de V como $\mathbb{K}$-espacio vectorial.

Lema 1.5.8 *[Jac74] T dota a V de una estructura de $\mathbb{K}[x]$-módulo.*

Lema 1.5.9 *[Jac74] V es un $\mathbb{K}[x]$-módulo de torsión.*

Demostración:
Trivial por ser V un $\mathbb{K}$-espacio vectorial de dimensión finita n y no haber, por lo tanto, $n+1$ vectores linealmente independientes. □

Lema 1.5.10 *[Jac74] V es un $\mathbb{K}[x]$-módulo finitamente generado.*

Demostración:
Trivial por ser V un $\mathbb{K}$-espacio vectorial de dimensión finita n. □

Una aplicación apresurada del Teorema de Estructura, conduciría a afirmar que, como V es un $\mathbb{K}[x]$-módulo finitamente generado y $\mathbb{K}[x]$ es un dominio de ideales principales (por ser $\mathbb{K}$ cuerpo), entonces

$$V = \mathbb{K}[x]z_1 \oplus ... \oplus \mathbb{K}[x]z_s$$

con

$$(0 : z_1) \supseteq (0 : z_2) \supseteq ... \supseteq (0 : z_s);\ (0 : z_k) \neq D, k = 1, ..., s$$

Esta aplicación directa del Teorema de Estructura, aunque importante pues muestra la composición de V como $\mathbb{K}[x]$-módulo, desde un punto de vista práctico, no aporta lo que realmente interesa, ¿cómo calcular $z_1, ..., z_s$? La obtención de $z_1, ..., z_s$ es importante, pues como se verá, es a partir de estos elementos, de donde se obtendrán todas las Formas Canónicas. Afortunadamente, la prueba del Teorema de Estructura [Jac74] es totalmente constructiva, por lo que se seguirán sus pasos para obtener $z_1, ..., z_s$.

Para empezar, todo $\mathbb{K}[x]$-módulo finitamente generado, es cociente de un $\mathbb{K}[x]$-módulo libre, esto es equivalente a decir que existe una aplicación

$\mathbb{K}[x]$-lineal sobreyectiva de $(\mathbb{K}[x])^n$ en V. Esta aplicación es la siguiente:

$$\begin{array}{lcl} \mathbb{K}[x]^n & \overset{\Psi}{\twoheadrightarrow} & V \\ e_1 & \longrightarrow & \Psi(e_1) = v_1 \\ \vdots & & \\ e_n & \longrightarrow & \Psi(e_n) = v_n \end{array}$$

con $C = \{e_1, ..., e_n\}$ la base canónica de $\mathbb{K}[x]^n$.

$$Ker\Psi \overset{i}{\rightarrowtail} \mathbb{K}[x]^n \overset{\Psi}{\twoheadrightarrow} V$$

Teorema 1.5.11 *[Jac74]* $F = \overset{n}{\underset{j=1}{\cup}} \{f_j = \sum_{i=1}^{n} xe_j - a_{ij}e_i\}$ *es una* $\mathbb{K}[x]$*-base de* $Ker\Psi$*.*

La matriz de la inclusión i en las bases F y C es:

$$\begin{pmatrix} x - a_{11} & -a_{12} & \dots & -a_{1n} \\ -a_{21} & x - a_{22} & \dots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \dots & x - a_{nn} \end{pmatrix} = xI_n - A$$

Definición 1.5.12 *Dada* $A \in Mat_{n\times n}(\mathbb{K})$*, se llama matriz característica de* A*, a la matriz* $xI_n - A$*.*

Para $xI_n - A$, existen $S, P, Q \in Mat_{n\times n}(\mathbb{K}[x])$ tales que:

- S es la Forma Normal de Smith de $xI_n - A$.
- P y Q son matrices inversibles
- $S = P \cdot (xI_n - A) \cdot Q$

Esto es:

$$\begin{array}{ccccc} {}_{F}Ker\Psi & \overset{xI_n-A}{\rightarrowtail} & (\mathbb{K}[x]^n)_C & \overset{\Psi}{\twoheadrightarrow} & V \\ \uparrow_Q & & \downarrow_P & & \\ {}_{F'}Ker\Psi & \overset{S}{\rightarrowtail} & (\mathbb{K}[x]^n)_{C'} & & \end{array}$$

Lo que se está haciendo al obtener la Forma Normal de Smith de la matriz característica de A, es calcular una base F' en $Ker\Psi$ y otra C' en $\mathbb{K}[x]^n$, respecto a las cuales, la matriz asociada al morfismo inclusión tiene una expresión diagonal, que es la matriz S.

Q y P son las matrices de cambio de base, las columnas de Q son las coordenadas de F' con respecto a F, las columnas de P son las coordenadas de C con respecto a C' y las columnas de P^{-1} son las coordenadas de C' con respecto a la base C.

S es de la forma:

$$\begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & d_1 & & \\ & & & & \ddots & \\ & & & & & d_s \end{pmatrix} (d_1|d_2|...|d_s)$$

siendo $d_1|d_2|...|d_s$ no nulos.

Un examen, a posteriori, de la prueba del Teorema de Estructura, muestra que el número de posiciones nulas en la diagonal principal de S, se corresponde con el rango libre o número de Betti [DuF99] del $\mathbb{K}[x]$-módulo V, esto es, es el mayor entero p tal que $\mathbb{K}[x]^p \subseteq V$. Como V es un $\mathbb{K}[x]$-módulo de torsión, carece de componente libre [Jac74], y por lo tanto p es cero. Esta es la razón por la cual se puede asegurar que $d_1, ..., d_s$ son polinomios no nulos.

En la prueba del Teorema de Estructura se demuestra finalmente, que las imágenes por Ψ de los últimos s vectores de la base

$$C' = \{e_1' = \sum_{i=1}^{n} {}^1p_{i1}e_i, e_2' = \sum_{i=1}^{n} {}^1p_{i2}e_i, ..., e_n' = \sum_{i=1}^{n} {}^1p_{in}e_i\}$$

$(P^{-1} = ({}^1p_{ij}))$, son las que determinan los elementos de V cuyos cíclicos generan V. Así pues se verifica:

$$z_1 = \Psi(e'_{n-s+1}), ..., z_s = \Psi(e'_n)$$

$$V = \mathbb{K}[x]z_1 \oplus ... \oplus \mathbb{K}[x]z_s \tag{1.2}$$

Además

- $(0 : z_i) = (d_i)$ con $d_i = x^{m_i} + a_{m_i-1}x^{m_i-1} + ... + a_{0_i}, i = 1, ..., s$
- $(0 : z_1) \supseteq (0 : z_2) \supseteq ... \supseteq (0 : z_s)$

Este hecho da pie al siguiente algoritmo.

Algoritmo 1.5.13 *Generadores_Primera_Descomposición.*
Entrada:

- $\mathbb{K}$ *cuerpo y* V *un* $\mathbb{K}$*-espacio vectorial.*
- $B = \{v_1, ..., v_n\}$ *una* $\mathbb{K}$*-base de* V*.*
- $A \in Mat_{n\times n}(\mathbb{K})$ *no nula, tal que* A *es la matriz del endomorfismo de* V *en la base* B*.*

Salida: $z_1, ..., z_s \in V$ *y* $d_1, ..., d_s \in \mathbb{K}[x]$ *tales que*

- $V = \mathbb{K}[x]z_1 \oplus ... \oplus \mathbb{K}[x]z_s$
- $(0 : z_i) = (d_i), i = 1, ..., s$

- $(0 : z_1) \supseteq (0 : z_2) \supseteq ... \supseteq (0 : z_s)$.

- $d_1, ..., d_s$ *son polinomios mónicos.*

1. $(S, P, Q) = SmithG\_Paso(xI_n - A)$ /* $S = diag(1, ..., 1, d_1, ..., d_s)$ */

2. $s =$ *número de elementos distintos de 1 en la diagonal principal de* S.

3. $Inversa\_P = Inversa\_Paso(P)$ /* $\Psi(e_i) = v_i$, *siendo* $\{e_1, ..., e_n\}$ *la base canónica de* $\mathbb{K}[x]^n$ */

4. $z_i = \Psi($*columna* $n - s + i$*-ésima de* $Inversa\_P)$, $i = 1, ..., s$

5. $d_i =$*elemento de* S *que ocupa la posición* $(n-s+i, n-s+i), i = 1, ..., s$

6. *Se hacen mónicos* $d_1, ..., d_s$

7. *Devuelve* $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\}$

8. *Fin*

Ejemplo 1.5.14 *Se aplica el algoritmo Generadores_Primera_Descomposición a:*

- $A = \begin{pmatrix} 2 & -1 \\ 0 & 1 \end{pmatrix} \in Mat_{2\times 2}(\mathbb{Q})$

- $B = \{v_1 = (-1, 0, 2), v_2 = (0, 1, 1)\}$

$V =< v_1, v_2 > \subseteq \mathbb{Q}^3$

1. $(S, P, Q) = SmithG\_Paso(xI_2 - A)$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & (x-1)(x-2) \end{pmatrix}$$

$$P = \begin{pmatrix} -1 & 0 \\ 1-x & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & 1 \\ 1 & 2-x \end{pmatrix}$$

2. $s = 1$

3. $Inversa\_P = Inversa\_Paso(P)$.

$$Inversa\_P = \begin{pmatrix} -1 & 0 \\ 1-x & 1 \end{pmatrix}$$

4. $z_1 = \Psi(0 \cdot e_1 + 1 \cdot e_2) = 0 \cdot v_1 + 1 \cdot v_2 = v_2 = (0, 1, 1)$

5. $d_1 = (x-1) \cdot (x-2) = x^2 - 3x + 2$

6. *Devuelve* $\{\{(0, 1, 1)\}, \{x^2 - 3x + 2\}\}$

7. *Fin*

Se puede afirmar entonces que:

- $V =< v_1, v_2 >= \mathbb{K}[x] \cdot v_2$ *(*V *es un* $\mathbb{K}[x]$*-módulo cíclico).*

- $(0 : v_2) = (x^2 - 3x + 2)$

Definición 1.5.15 *Dado un polinomio mónico* $f = x^n + a_{n-1}x^{n-1} + ... + a_0 \in \mathbb{K}[x]$*, se denomina matriz compañera asociada a* f*, a la matriz* $C(f) \in Mat_{n\times n}(\mathbb{K})$ *cuyo término* (i, j) *es:*

- *1, si* $j+1=i$.
- $-a_{i-1}$, *si* $j=n$.
- *0, en cualquier otro caso.*

Ejemplo 1.5.16 *La matriz compañera de* x^3-2x^2+4 *es:*

$$C(x^3-2x^2+4)=\begin{pmatrix}0&0&-4\\1&0&0\\0&1&2\end{pmatrix}$$

Teorema 1.5.17 *[Bar96, Ser02] Dada una matriz* A *de orden* n *sobre* $\mathbb{K}$*, existe una matriz* P *de orden* n *e inversible, tal que:*

$$(F=)P^{-1}AP=diag(C(f_1),...,C(f_t)),\ con\ f_1|f_2|...|f_t.$$

F se dice que es la Forma Canónica Racional o de Frobenius de A.

La Forma Canónica Racional de A [Jac74], se obtiene a partir de la primera descomposición de V (1.2), que proporciona el Teorema de Estructura. Tomando como $\mathbb{K}$-base para cada módulo cíclico $\mathbb{K}[x]z_i$ [Jac74]

$$\{z_i, x\cdot z_i, ..., x^{m_i-1}\cdot z_i\}$$

se obtiene una $\mathbb{K}$-base para V de la forma

$$B'=\{z_1, x\cdot z_1, ..., x^{m_1-1}\cdot z_1, z_2, x\cdot z_2, ..., x^{m_2-1}\cdot z_2, ..., z_s, x\cdot z_s, ..., x^{m_s-1}\cdot z_s\}.$$

Si P es la matriz que se obtiene al colocar en columnas las coordenadas de los vectores de B', entonces se verifica que $P^{-1}AP$ es una matriz diagonal por bloques, formada por las matrices compañeras de los factores invariantes de la matriz característica de A, esto es, se obtiene la Forma Canónica Racional de A.
De este razonamiento se sigue el siguiente algoritmo:

Algoritmo 1.5.18 *Frobenius.*
Entrada:

- $\mathbb{K}$ *cuerpo.*
- $A \in Mat_{n\times n}(\mathbb{K})$ *no nula, la matriz del endomorfismo de* $\mathbb{K}^n$ *en las bases canónicas.*

Salida:

- *Una matriz* R *tal que* R *es la Forma Canónica Racional de* A*.*
- *Una matriz inversible* P*, tal que* $P^{-1} \cdot A \cdot P = R$

1. $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\}$=*Generadores_Primera_ Descomposición(A,C)*
   / C es la base canónica de* $\mathbb{K}^n$ **/*
2. $B' = \emptyset$*; R =matriz nula de tamaño* $n \times n$*; indicador_bloque=1*
3. *Bucle* $i = 1$ *hasta* s */*Se construye* B' **/*
   - $m = grado(d_i)$
   - $B' = B' \cup \{z_i, x \cdot z_i, ..., x^{m-1} \cdot z_i\}$
   - *Se sustituye el rectángulo de* R *de coordenadas*
     - $(indicador\_bloque, indicador\_bloque)$
       / Esquina superior izqda. */*
     - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1)$
       / Esquina inferior dcha. */*

     por la matriz compañera de d_i
   - $indicador\_bloque = indicador\_bloque + m$
4. P *es la matriz que se obtiene al poner en columnas los vectores de* B'
5. *Devuelve* R, P

6. Fin

Ejemplo 1.5.19 *Se calculará la Forma Canónica Racional de:*

$$A = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix} \in Mat_{6\times 6}(\mathbb{Q})$$

1. $\{\{z_1, ...d_s\}, \{d_1, ..., d_s\}\}$*=Generadores_Primera_Descomposición*(A, C)

- $z_1 = (-3, 3, -1, 0, -1, 0)$*;* $d_1 = x - 1$
- $z_2 = (1, -1, 0, 0, -1, 0)$*;* $d_2 = x - 1$
- $z_3 = (0, \frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{4})$*;* $d_2 = x^4 - 7x^3 + 17x^2 - 17x + 6$

2. $B' = \emptyset$*; R= matriz nula de orden 6; indicador_bloque=1*

3. Bucle i=1 hasta 3

- $i = 1$
  - $m = 1$
  - $B' = \{z_1\}$
  - $(indicador\_bloque, indicador\_bloque) = (1, 1)$
  - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (1, 1)$
  - *Se sustituye el bloque de R de coordenadas (1,1)-(1,1) por la matriz compañera de* $x - 1$*.*

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- $indicador\_bloque = 2$

- $i = 2$
  - $m = 1$
  - $B' = \{z_1, z_2\}$
  - $(indicador\_bloque, indicador\_bloque) = (2, 2)$
  - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (2, 2)$
  - *Se sustituye el bloque de R de coordenadas (2,2)-(2,2) por la matriz compañera de* $x - 1$.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

  - $indicador\_bloque = 3$

- $i = 3$
  - $m = 4$
  - $B' = \{z_1, z_2\} \cup \{z_3, A \cdot z_3 = (\frac{-1}{2}, 1, \frac{1}{2}, 2, \frac{-1}{4}, \frac{3}{4}),$
    $A^2 \cdot z_3 = (-1, \frac{3}{2}, \frac{1}{2}, 4, -2, \frac{9}{4}),$
    $A^3 \cdot z_3 = (\frac{-3}{2}, 2, \frac{1}{2}, 8, \frac{-13}{2}, \frac{27}{4})\}$
  - $(indicador\_bloque, indicador\_bloque) = (3, 3)$
  - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (6, 6)$

- *Se sustituye el bloque de R de coordenadas (3,3)-(6,6) por la matriz compañera de* $x^4 - 7x^3 + 17x^2 - 17x + 6$.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -6 \\ 0 & 0 & 1 & 0 & 0 & 17 \\ 0 & 0 & 0 & 1 & 0 & -17 \\ 0 & 0 & 0 & 0 & 1 & 7 \end{pmatrix}$$

- *indicador_bloque* $= 7$

4. $P = \begin{pmatrix} -3 & 1 & 0 & \frac{-1}{2} & -1 & \frac{-3}{2} \\ 3 & -1 & \frac{1}{2} & 1 & \frac{3}{2} & 2 \\ -1 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 & 2 & 4 & 8 \\ -1 & -1 & 0 & \frac{-1}{4} & -2 & \frac{-13}{2} \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} & \frac{9}{4} & \frac{27}{4} \end{pmatrix}$

5. *Devuelve* R, P

6. *Fin*

Teorema 1.5.20 *[Jac74, DuF99, Coh74] Sea* M *un* $\mathbb{K}[x]$*-módulo,* $x \in M$*,* $d \in \mathbb{K}[x]$ *tal que* $(0 : x) = (d)$ *y* $d = p_1^{e_1} \cdot ... \cdot p_t^{e_t}$*,* $p_i \neq p_j$ *si* $i \neq j$*, la descomposición de* d *en primos. Entonces:*

- $Dx = Dx_1 \oplus ... \oplus Dx_t$ *con* $(0 : x_i) = (p_i^{e_i})$

- $x_i = p_1^{e_1} \cdot ... \cdot p_{i-1}^{e_{i-1}} \cdot p_{i+1}^{e_{i+1}} \cdot ... \cdot p_t^{e_t}$*,* $i = 1, ..., t$.

Teniendo en cuenta este resultado, cada $\mathbb{K}[x]$-módulo cíclico de la descomposición (1.2), se puede volver a descomponer en suma directa de $\mathbb{K}[x]$-módulos cíclicos. Se obtiene así, una segunda descomposición de V:

$$V = \mathbb{K}[x] \cdot u_1 \oplus ... \oplus \mathbb{K}[x] \cdot u_r \tag{1.3}$$

siendo $(0 : u_i) = (p_i(x)^{e_i})$, con $p_i(x)$ un polinomio mónico e irreducible en $\mathbb{K}[x]$, $i = 1, ..., r$.

Teorema 1.5.21 *[Bar96, Ser02] Dada una matriz A de orden n sobre $\mathbb{K}$, existe una matriz P de orden n e inversible, tal que:*

$$(F =)P^{-1}AP = diag(C(q_1^{a_1}), ..., C(q_t^{a_t})),\ q_1, ..., q_t \text{ irreducibles sobre } \mathbb{K}[x].$$

F se dice que es la Segunda Forma Canónica de A.

Tomando de nuevo como $\mathbb{K}$-base de cada módulo cíclico $\mathbb{K}[x]u_i$

$$\{u_i, x \cdot u_i, ..., x^{h_i-1} \cdot u_i\}$$

siendo h_i el grado del polinomio $p_i^{e_i}$, se obtiene una $\mathbb{K}$-base para V de la forma

$$B'' = \{u_1, x \cdot u_1, ..., x^{h_1-1} \cdot u_1, ..., u_r, x \cdot u_r, ..., x^{h_r-1} \cdot u_r\}$$

Si P es la matriz que se obtiene al colocar en columnas las coordenadas de los vectores de B'', entonces se verifica que $P^{-1}AP$ es una matriz diagonal por bloques, formada por las matrices compañeras de los divisores elementales de la matriz característica de A, esto es, se obtiene la Segunda Forma Canónica de A.
Esta forma de proceder da lugar a:

Algoritmo 1.5.22 *Segunda_Forma_Canónica.*
Entrada:

- $\mathbb{K}$ *cuerpo*
- $A \in Mat_{n\times n}(\mathbb{K})$ *no nula, la matriz del endomorfismo de* $\mathbb{K}^n$ *en las bases canónicas.*

Salida:

- *Una matriz R tal que R es la Segunda Forma Canónica de A*
- *Una matriz inversible P, tal que $P^{-1} \cdot A \cdot P = R$*

1. $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\}$ =*Generadores_Primera_Descomposición*(A, C)
   /* *C es la base canónica de* $\mathbb{K}^n$ */
2. $B'' = \emptyset$; *R es la matriz nula de orden n; indicador_bloque=1.*
3. *Bucle* $i = 1$ *hasta* s
   - *Se factoriza d_i en primos (sobre $\mathbb{K}[x]$)*
     - $d_i = p_{i,1}^{e_{i,1}} \cdot ... \cdot p_{i,t}^{e_{i,t}}$, $p_{i,j} \neq p_{i,k}$ *si* $j \neq k$.
     - *t=número de factores primos en la descomposición de d_i.*
   - /* *Se descompone $\mathbb{K}[x]z_i$ en suma directa de cíclicos* */
   - *Bucle* $j = 1$ *hasta* t
     - $u = \frac{d_i}{p_{i,j}^{e_{i,j}}} z_i$
     - $m = grado(p_{i,j}) \cdot e_{i,j}$
     - $B'' = B'' \cup \{u, x \cdot u, ..., x^{m-1}u\}$
     - *Se sustituye el rectángulo de R:*
       - $(indicador\_bloque, indicador\_bloque)$
         /* *Esquina superior izquierda* */
       - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1)$
         /* *Esquina inferior derecha* */

       por la matriz compañera de $p_{i,j}^{e_{i,j}}$.
   - $indicador\_bloque = indicador\_bloque + m$
4. *P es la matriz que se obtiene al poner en columnas los vectores de B''.*

5. *Devuelve* R, P

6. *Fin*

Ejemplo 1.5.23 *Se calculará la Segunda Forma Canónica de la matriz:*

$$A = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix} \in Mat_{6\times 6}(\mathbb{Q})$$

1. $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\} =$*Generadores_Primera_Descomposición*(A, C)

   - $z_1 = (-3, 3, -1, 0, -1, 0)$*;* $d_1 = x - 1$
   - $z_2 = (1, -1, 0, 0, -1, 0)$*;* $d_2 = x - 1$
   - $z_3 = (0, \frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{4})$*;* $d_3 = x^4 - 7x^3 + 17x^2 - 17x + 6$

2. $B'' = \emptyset$*; R es la matriz nula de orden 6; indicador_bloque=1*

3. *Bucle* $i = 1$ *hasta 3*

   - $i = 1$
     - $d_1 = x - 1$
     - *Bucle* $j = 1$ *hasta 1*
       - $u = z_1$
       - $m = 1$
       - $B'' = \{z_1\}$
       - $(indicador\_bloque, indicador\_bloque) = (1, 1)$
       - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (1, 1)$

- *Se sustituye el rectángulo de R de coordenadas (1,1)-(1,1), por la matriz compañera de* $x-1$.

$$R=\begin{pmatrix}1&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{pmatrix}$$

- *indicador_bloque=2*

- $i=2$

  - $d_2=x-1$
  - *Bucle* $j=1$ *hasta 1*
    - $u=z_2$
    - $m=1$
    - $B''=\{z_1,z_2\}$
    - $(indicador\_bloque, indicador\_bloque)=(2,2)$
    - $(indicador\_bloque+m-1, indicador\_bloque+m-1)=(2,2)$
    - *Se sustituye el rectángulo de R de coordenadas (2,2)-(2,2), por la matriz compañera de* $x-1$.

$$R=\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{pmatrix}$$

    - *indicador_bloque=3*

- $i=3$

- $d_3 = (x-1)^2 \cdot (x-2) \cdot (x-3)$
- *Bucle* $j = 1$ *hasta 3*
  - $j = 1$
    - $u = (x-2) \cdot (x-3) \cdot z_3 = (\frac{3}{2}, \frac{-1}{2}, 1, 0, \frac{1}{2}, 0)$
    - $m = 2$
    - $B'' = \{z_1, z_2\} \cup \{u, xu = (\frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{2}, 0)\}$
    - $(indicador\_bloque, indicador\_bloque) = (3, 3)$
    - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (4, 4)$
    - *Se sustituye el rectángulo de* R *de coordenadas (3,3)-(4,4), por la matriz compañera de* $(x-1)^2$*.*

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

    - $indicador\_bloque$*=5*
  - $j = 2$
    - $u = (x-1)^2 \cdot (x-3) \cdot z_3 = (0, 0, 0, -1, 0, 0)$
    - $m = 1$
    - $B'' = \{z_1, z_2, (\frac{3}{2}, \frac{-1}{2}, 1, 0, \frac{1}{2}, 0), (\frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{2}, 0)\} \cup \{(0, 0, 0, -1, 0, 0)\}$
    - $(indicador\_bloque, indicador\_bloque) = (5, 5)$
    - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (5, 5)$
    - *Se sustituye el rectángulo de* R *de coordenadas (5,5)-(5,5), por la matriz compañera de* $x - 2$*.*

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- *indicador_bloque=6*

- $j = 3$
  - $u = (x-1)^2 \cdot (x-2) \cdot z_3 = (0,0,0,0,-1,1)$
  - $m = 1$
  - $B'' = B'' \cup \{(0,0,0,0,-1,1)\}$
  - $(indicador\_bloque, indicador\_bloque) = (6,6)$
  - $(indicador\_bloque + m - 1, indicador\_bloque + m - 1) = (6,6)$
  - *Se sustituye el rectángulo de R de coordenadas (6,6)-(6,6), por la matriz compañera de* $x - 3$.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

  - *indicador_bloque=7*

4. $P = \begin{pmatrix} -3 & 1 & \frac{3}{2} & \frac{1}{2} & 0 & 0 \\ 3 & -1 & \frac{-1}{2} & \frac{1}{2} & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & -1 & \frac{1}{2} & \frac{1}{2} & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

5. *Devuelve* R, P

6. *Fin*

Definición 1.5.24 *Se llama matriz hiperasociada o hipercompañera de un polinomio mónico* $p(x)^q$ *a una matriz que verifica:*

- *Si* $q = 1$ *es la matriz compañera de* $p(x)$.
- *Si* $q > 1$ *es una matriz cuadrada de orden* $grado(p(x)) \cdot q$, *de la forma:*

$$C_q(p) = \begin{pmatrix} C(p) & & & & 0 \\ N & C(p) & & & \\ & N & \ddots & & \\ & & \ddots & \ddots & \\ 0 & & & N & C(p) \end{pmatrix}$$

siendo $C(p)$ *la matriz compañera de* p, *y* N *una matriz con el mismo orden de* $C(p)$, *que tiene un uno en la esquina superior derecha y cero en el resto de las posiciones.*

Ejemplo 1.5.25 *La matriz hiperasociada de* $(x^2 + 2x - 1)^4$ *es:*

$$C_4(x^2 + 2x - 1) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Teorema 1.5.26 *[Jea99, Ser02] Dada una matriz A de orden n sobre $\mathbb{K}$, existe una matriz P de orden n e inversible, tal que:*

$$(F =)P^{-1}AP = diag(C_{a_1}(q_1), ..., C_{a_t}(q_t)) \quad q_1, ..., q_t \text{ irreducibles.}$$

F se dice que es la Forma Canónica de Jacobson de A.

La Forma Canónica de Jacobson, se obtiene al tomar como $\mathbb{K}$-base de cada cíclico $\mathbb{K}[x]u_i$ en (1.3) [Jac74]

$$\{u_i, x \cdot u_i, ..., x^{m_i-1} \cdot u_i, p_i \cdot u_i, x \cdot p_i \cdot u_i, ..., x^{m_i-1} \cdot p_i \cdot u_i, ..., p_i^{e_i-1} \cdot u_i, \\ x \cdot p_i^{e_i-1} \cdot u_i, ..., x^{m_i-1} \cdot p_i^{e_i-1} \cdot u_i\}$$

siendo $(m_i = grado(p_i(x)))$. La unión de todas estas $\mathbb{K}$-bases proporciona una $\mathbb{K}$-base para V, en la cual, el endomorfismo que representa la matriz A tiene asociada una matriz diagonal por bloques, formada por las matrices hipercompañeras de los divisores elementales de la matriz característica de A, esto es, se obtiene la Forma Canónica de Jacobson de A. Este razonamiento da lugar al siguiente algoritmo.

Algoritmo 1.5.27 *Jacobson*
Entrada:

- $\mathbb{K}$ *cuerpo*
- $A \in Mat_{n\times n}(\mathbb{K})$ *no nula, la matriz del endomorfismo de* $\mathbb{K}^n$ *en las bases canónicas.*

Salida:

- *Una matriz R tal que R es la Forma Canónica de Jacobson de A.*
- *Una matriz inversible P, tal que $P^{-1} \cdot A \cdot P = R$*

1. $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\}$ =*Generadores_Primera_Descomposición*(A, C)
   /* C *es la base canónica de* $\mathbb{K}^n$ */

2. $B''' = \emptyset$; R *es la matriz nula de orden n; indicador_bloque=1.*

3. *Bucle* $i = 1$ *hasta* s

   - *Se factoriza* d_i *en primos (sobre* $\mathbb{K}[x]$*)*
     - $d_i = p_{i,1}^{e_{i,1}} \cdot ... \cdot p_{i,t}^{e_{i,t}}$, $p_{i,j} \neq p_{i,k}$ *si* $j \neq k$.
     - t=*número de factores primos en la descomposición de* d_i.
   - /* *Se descompone* $\mathbb{K}[x]z_i$ *en suma directa de cíclicos* */
   - *Bucle* $j = 1$ *hasta* t
     - $u = \frac{d_i}{p_{i,j}^{e_{i,j}}} z_i$
     - $m = grado(p_{i,j}) \cdot e_{i,j}$; $gradop = m \cdot e_{ij}$
     - $B''' = B''' \cup \{u, x \cdot u, ..., x^{m-1} \cdot u, p_{i,j} \cdot u, x \cdot p_{i,j} \cdot u, ..., x^{m-1} \cdot p_{i,j} \cdot u, ..., p_{i,j}^{e_{ij}-1} \cdot u, x \cdot p_{i,j}^{e_{ij}-1} \cdot u, ..., x^{m-1} \cdot p_{i,j}^{e_{ij}-1} \cdot u\}$
     - *Se sustituye el rectángulo de* R*:*
       - $(indicador\_bloque, indicador\_bloque)$
         /* *Esquina superior izquierda* */
       - $(indicador\_bloque+gradop-1, indicador\_bloque+gradop-1)$
         /* *Esquina inferior derecha* */

       por la matriz hipercompañera de $p_{i,j}^{e_{i,j}}$.
   - $indicador\_bloque = indicador\_bloque + gradop$

4. P *es la matriz que se obtiene al poner en columnas los vectores de* B'''.

5. *Devuelve* R, P

6. *Fin*

Ejemplo 1.5.28 *Se calculará la Forma Canónica de Jacobson de la matriz:*

$$A = \begin{pmatrix} 0 & 0 & 0 & -4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 0 \end{pmatrix} \in Mat_{4\times 4}(\mathbb{Q})$$

1. $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\}$ =*Generadores_Primera_Descomposición*(A, C)

$$z_1 = (0, 0, \tfrac{1}{4}, 0);\ d_1 = x^4 + 4x^2 + 4$$

2. $B''' = \emptyset$; R *la matriz nula de orden 4; indicador_bloque=1*

3. *Bucle* $i = 1$ *hasta 1*

- $i = 1$
  - $d_1 = (x^2 + 2)^2$
  - $t = 1$
  - *Bucle* $j = 1$ *hasta 1*
    - $u = z_1$
    - $m = 2; gradop = 4$
    - $B''' = \{z_1, x \cdot z_1 = (0, 0, 0, \frac{1}{4}),$
      $(x^2+2) \cdot z_1 = (-1, 0, \frac{-1}{2}, 0), x(x^2+2) \cdot z_1 = (0, -1, 0, \frac{-1}{2})\}$
      - $(indicador\_bloque, indicador\_bloque) = (1, 1)$
      - $(indicador\_bloque+gradop-1, indicador\_bloque+gradop-1) = (4, 4)$

      por la matriz hipercompañera de $(x^2 + 2)^2$

$$R = \left(\begin{array}{cc|cc} 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \end{array}\right)$$

- *indicador_bloque* $= 5$

4. $P = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ \frac{1}{4} & 0 & \frac{-1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{-1}{2} \end{pmatrix}$

5. *Devuelve* R, P

6. *Fin*

Queda, por último, tratar el cálculo de la Forma Canónica de Jordan de una matriz. La Forma Canónica de Jordan es la apariencia que toma la Forma Canónica de Jacobson, cuando los divisores elementales de la matriz característica, son potencias de polinomios de grado 1 [Ser02].

El método natural para calcular entonces la Forma Canónica de Jordan de una matriz, se reduce a comprobar que las raíces del polinomio característico están todas en $\mathbb{K}$, y, en caso afirmativo, aplicar el algoritmo de cálculo de la Forma Canónica de Jacobson.

Ejemplo 1.5.29 *Se calculará la Forma Canónica de Jordan de la matriz*

$$A = \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix} \in Mat_{6\times 6}(\mathbb{Q})$$

Los factores invariantes no triviales son $x-1, x-1, x^4-7x^3+17x^2-17x+6$. *Todos tienen sus raíces en* $\mathbb{Q}$, *por lo que si se aplica el algoritmo Jacobson se obtendrá entonces la Forma Canónica de Jordan de* A.

1. $\{\{z_1, ..., z_s\}, \{d_1, ..., d_s\}\} =$*Generadores_Primera_Descomposición*(A, C)

- $z_1 = (-3, 3, -1, 0, -1, 0)$; $d_1 = x - 1$
- $z_2 = (1, -1, 0, 0, -1, 0)$; $d_2 = x - 1$
- $z_3 = (0, \frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{4})$; $d_3 = x^4 - 7x^3 + 17x^2 - 17x + 6$

2. *$B'' = \emptyset$; R es la matriz nula de orden 6; indicador_bloque=1*

3. *Bucle $i = 1$ hasta 3*

- $i = 1$
  - $d_1 = x - 1$
  - $t = 1$
  - *Bucle $j = 1$ hasta 1*
    - $u = z_1$
    - $m = 1$; $gradop = 1$
    - $B''' = \{z_1\}$
    - $(indicador\_bloque, indicador\_bloque) = (1, 1)$
    - $(indicador\_bloque + gradop - 1, indicador\_bloque + gradop - 1) = (1, 1)$
    - *Se sustituye el rectángulo de R de coordenadas (1,1)-(1,1), por la matriz hipercompañera de $x - 1$.*

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

    - *indicador_bloque=2*

- $i = 2$
  - $d_2 = x - 1$

- $t = 1$
- *Bucle* $j = 1$ *hasta 1*
  - $u = z_2$
  - $m = 1$*;* $gradop = 1$
  - $B''' = \{z_1, z_2\}$
  - $(indicador\_bloque, indicador\_bloque) = (2, 2)$
  - $(indicador\_bloque + gradop - 1, indicador\_bloque + gradop - 1) = (2, 2)$
  - *Se sustituye el rectángulo de* R *de coordenadas (2,2)-(2,2), por la matriz hipercompañera de* $x - 1$*.*

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

  - *indicador_bloque=3*

- $i = 3$
  - $d_3 = (x-1)^2 \cdot (x-2) \cdot (x-3)$
  - $t = 3$
  - *Bucle* $j = 1$ *hasta 3*
    - $j = 1$
      - $u = (x-2) \cdot (x-3) \cdot z_3 = (\frac{3}{2}, \frac{-1}{2}, 1, 0, \frac{1}{2}, 0)$
      - $m = 1$*;* $gradop = 2$
      - $B''' = \{z_1, z_2\} \cup \{u, (x-1)u = (-1, 1, 0, 0, 0, 0)\}$
      - $(indicador\_bloque, indicador\_bloque) = (3, 3)$

- $(indicador\_bloque+gradop-1, indicador\_bloque+gradop-1) = (4,4)$
- *Se sustituye el rectángulo de R de coordenadas (3,3)-(4,4), por la matriz hipercompañera de* $(x-1)^2$.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- *indicador_bloque=5*

- $j = 2$
  - $u = (x-1)^2 \cdot (x-3) \cdot z_3 = (0,0,0,-1,0,0)$
  - $m = 1$; $gradop = 1$
  - $B''' = \{z_1, z_2, (\frac{3}{2}, \frac{-1}{2}, 1, 0, \frac{1}{2}, 0), (-1,1,0,0,0,0)\} \cup \{(0,0,0,-1,0,0)\}$
  - $(indicador\_bloque, indicador\_bloque) = (5,5)$
  - $(indicador\_bloque+gradop-1, indicador\_bloque+gradop-1) = (5,5)$
  - *Se sustituye el rectángulo de R de coordenadas (5,5)-(5,5), por la matriz hipercompañera de* $x-2$.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

  - *indicador_bloque=6*

- $j = 3$

- $u = (x-1)^2 \cdot (x-2) \cdot z_3 = (0,0,0,0,-1,1)$
- $m = 1$; $gradop = 1$
- $B''' = B''' \cup \{(0,0,0,0,-1,1)\}$
- $(indicador\_bloque, indicador\_bloque) = (6,6)$
- $(indicador\_bloque+gradop-1, indicador\_bloque+gradop-1) = (6,6)$
- *Se sustituye el rectángulo de R de coordenadas (6,6)-(6,6), por la matriz compañera de* $x-3$.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

- $indicador\_bloque=7$

4. $P = \begin{pmatrix} -3 & 1 & \frac{3}{2} & -1 & 0 & 0 \\ 3 & -1 & \frac{-1}{2} & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & -1 & \frac{1}{2} & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

5. *Devuelve* R, P

6. *Fin*

Observaciones:

- *Los algoritmos de cálculo de la Segunda Forma Canónica y de la Forma Canónica de Jacobson, son casi idénticos*, las únicas diferencias radican

a la hora de construir la $\mathbb{K}$-base, y en el hecho de que en uno se maneja la matriz compañera y en el otro la matriz hipercompañera. Este comportamiento simplifica enormemente la programación del cálculo de ambas Formas Canónicas.

- *No siempre es posible, en la práctica, obtener la Segunda Forma Canónica y la Forma Canónica de Jacobson*, pues para su cálculo se debe conocer previamente la factorización en irreducibles de los factores invariantes (=divisores elementales), y esto no siempre es factible. En $\mathbb{C}$, por ejemplo, aunque teóricamente todo polinomio de grado mayor que uno con coeficientes complejos, se puede expresar como producto de polinomios de grado uno, en la práctica, no siempre se puede llegar a esta factorización.

- *La Forma Canónica de Jordan, no es más que la apariencia que adopta la Forma Canónica de Jacobson, en el caso de que los divisores elementales sean potencias de polinomios lineales*, o lo que es equivalente, cuando los factores invariantes tengan todas sus raíces en el cuerpo $\mathbb{K}$ sobre el que se trabaja.

1.5.4. Más Bases de Gröbner

Una vez que se ha conseguido, utilizando la teoría de bases de Gröbner, reformular los tres algoritmos clásicos de cálculo de formas canónicas, es difícil no caer en la tentación de preguntarse, ¿ qué más pueden aportar las bases de Gröbner ?, esto es, ¿ sería posible utilizar bases de Gröbner para realizar algún cálculo más ? Un examen detallado de los tres algoritmos de cálculo de formas canónicas (Frobenius, Segunda_Forma_Canónica, Jacobson) muestra que tres son los cálculos más importantes que se hacen, el primero es el de calcular los elementos $\{z_1, ..., z_s\}$ y los factores invariantes, que ya se hace utilizando bases de Gröbner; el segundo es la factorización de los factores invariantes (en Segunda_Forma_Canónica y Jacobson), y el tercero es el cálculo

de la acción de un polinomio $f(x) \in \mathbb{K}[x]$ sobre un vector $v \in V$. Este cálculo es muy importante, pues se necesita para construir la $\mathbb{K}$-base de V en la cual el endomorfismo que define la matriz A, se expresa matricialmente mediante una de las cuatro formas canónicas descritas anteriormente.

La cuestión que subyace tras el cálculo de la acción de un polinomio sobre un vector, es la evaluación de un polinomio en una matriz ($f(x) \cdot v = f(A) \cdot v$) [Gie93]. Este cálculo en principio puede parecer trivial, sin embargo, es un problema muy complicado de resolver, pues aunque la mecánica es muy sencilla, en la práctica, el número de operaciones a realizar se dispara a medida que crece el orden de la matriz y el grado del polinomio sobre el que actúa.

En esta subsección se va a explicar cómo calcular la acción de un polinomio $f(x) \in \mathbb{K}[x]$ sobre un vector $v \in V$, utilizando bases de Gröbner. El único objetivo de esta subsección es el de mostrar que este cálculo se puede hacer utilizando la teoría de bases de Gröbner, sin entrar a discutir en ningún momento, la rapidez o lentitud del procedimiento que aquí se expone.

Se define, igual que en la subsección anterior, el epimorfismo:

$$\Psi : \mathbb{K}[x]^n \twoheadrightarrow V$$

Ψ es una presentación libre del $\mathbb{K}[x]$-módulo V siendo $Ker\Psi$ su módulo de relaciones. Se puede afirmar entonces que se tiene un isomorfismo de $\mathbb{K}[x]$-módulos entre $\frac{\mathbb{K}[x]^n}{Ker\Psi}$ y V.

En la mayoría de los libros de álgebra, se manejan siempre los elementos de R-módulo R^n (con R un anillo cualquiera), de la forma $\sum_{i=1}^{n} a_i e_i$, siendo $\{e_1, ..., e_n\}$ la base canónica de R^n y $a_1, ..., a_n \in R$. Esto es, se piensan, por comodidad para su manipulación, como polinomios en las variables $e_1, ..., e_n$ con coeficientes en R.

Este modo de actuar define, de modo natural, un monomorfismo $\mathbb{K}[x]$-lineal:

$$\begin{array}{ccc} \theta : \mathbb{K}[x]^n & \rightarrowtail & \mathbb{K}[x, E_1, ..., E_n] \\ g = (g_1(x), ..., g_n(x)) & \rightsquigarrow & \theta(g) = g_1(x) \cdot E_1 + ... + g_n(x) \cdot E_n \end{array}$$

A partir de los morfismos de $\mathbb{K}[x]$-módulos θ y Ψ, y definiendo I como el $\mathbb{K}[x]$-módulo $\theta(Ker\Psi)$, es posible construir de modo natural el siguiente diagrama conmutativo de $\mathbb{K}[x]$-módulos:

$$\begin{array}{ccc} Ker\ \Psi & \xrightarrow[\cong]{\theta} & I \\ \downarrow & & \downarrow \\ K[x]^n & \xrightarrow{\theta} & \mathbb{K}[x, E_1, ..., E_n] \\ \downarrow & & \downarrow \\ V \cong \frac{\mathbb{K}[x]^n}{Ker\ \Psi} & \longrightarrow & \frac{\mathbb{K}[x, E_1, ..., E_n]}{I} \end{array}$$

La propiedad universal del conúcleo garantiza la existencia de un único morfismo inyectivo

$$\overline{\theta} : V \cong \frac{\mathbb{K}[x]^n}{Ker\Psi} \longrightarrow \frac{\mathbb{K}[x, E_1, ..., E_n]}{I}$$

que hace el diagrama conmutativo.

En lo tocante al $\mathbb{K}[x]$-módulo I, es posible concretar un poco más y afirmar:

Proposición 1.5.30 *I está generado como $\mathbb{K}[x]$-módulo por*

$$\{xE_1 - \sum_{i=1}^{n} a_{i1}E_i, ..., xE_n - \sum_{i=1}^{n} a_{in}E_i\}$$

Demostración:
Trivial teniendo en cuenta que $\bigcup_{j=1}^{n}\{f_j = \sum_{i=1}^{n} xe_j - a_{ij}e_i\}$ es una $\mathbb{K}[x]$-base de $Ker\Psi$ □

Corolario 1.5.31 *V es un $\mathbb{K}[x]$-submódulo del $\mathbb{K}[x]$-módulo $\frac{\mathbb{K}[x,E_1,...,E_n]}{I}$*

Lema 1.5.32 *Una base de Gröbner para el ideal*

$$J =< \{f_1 = xE_1 - \sum_{i=1}^{n} a_{i1}E_i, ..., f_n = xE_n - \sum_{i=1}^{n} a_{in}E_i\} >$$

con respecto al orden monomial lexicográfico con $x > E_1 > ... > E_n$ es de la forma $G = \{xE_1 - \sum_{i=1}^{n} a_{i1}E_i, ..., xE_n - \sum_{i=1}^{n} a_{in}E_i\} \cup H$, siendo H un conjunto de polinomios homogéneos en $E_1, ..., E_n$, de grado mayor o igual que 2.

Demostración:
Se comienza esta demostración aplicando el algoritmo de Buchberger [CLO92] de cálculo de bases de Gröbner al conjunto $G = \{f_1, ..., f_n\}$.

$S(f_1, f_2) = E_2 \cdot (xE_1 - \sum_{i=1}^{n} a_{i1}E_i) - E_1 \cdot (xE_2 - \sum_{i=1}^{n} a_{i2}E_i) = \sum_{i=1}^{n}(-a_{i1} \cdot E_2 + a_{i2} \cdot E_1)E_i$

El reducido de $S(f_1, f_2)$ por G es $S(f_1, f_2)$, pues ninguno de sus monomios es divisible por ningún monomio principal de ningún polinomio de G. Se añade $f_{n+1} = S(f_1, f_2)$ a G.

Procediendo de igual modo que antes, se tiene que $S(f_1, f_3) = \sum_{i=1}^{n}(-a_{i1} \cdot E_3 + a_{i3} \cdot E_1)E_i$. Como mucho, $S(f_1, f_3)$ se puede dividir entre f_{n+1}, y como este polinomio es homogéneo de grado 2, entonces, el reducido con respecto a G de $S(f_1, f_3)$, sea éste f_{n+2}, es homogéneo de grado 2.

Si se continúa el algoritmo de Buchberger hasta el final, se obtendrá una base de Gröbner con la forma descrita. □

Corolario 1.5.33 *$\{\overline{E_1}, ..., \overline{E_n}, \overline{1}, \overline{x}, ..., \overline{x^n}, ...\}$ es una $\mathbb{K}$-base de $\frac{\mathbb{K}[x,E_1,...,E_n]}{I}$*

Lema 1.5.34 *$\{e_i + Ker\Psi\}_{i=1}^{n}$ es un conjunto $\mathbb{K}$-linealmente independiente.*

Demostración:
$\alpha_1(e_1 + Ker\Psi) + ... + \alpha_n(e_n + Ker\Psi) = 0 + Ker\Psi \iff \alpha_1 e_1 + ... + \alpha_n e_n \in Ker\Psi \iff \Psi(\alpha_1 e_1 + ... + \alpha_n e_n) = \alpha_1 v_1 + ... + \alpha_n v_n = 0 \iff \alpha_1 = ... = \alpha_n = 0$
□

Lema 1.5.35 *$\{\overline{E_1}, ..., \overline{E_n}\}$ es una $\mathbb{K}$-base para V.*

Demostración:
V es un $\mathbb{K}$-espacio vectorial de dimensión n y $\{e_i + Ker\Psi\}_{i=1}^n$ es un conjunto $\mathbb{K}$-linealmente independiente de V. Es por esto por lo que se puede concluir que $\{e_i + Ker\Psi\}_{i=1}^n$ es una $\mathbb{K}$-base de V; y, por lo tanto $\{\overline{E_1} = \overline{\theta}(e_1 + Ker\Psi), ..., \overline{E_n} = \overline{\theta}(e_n + Ker\Psi)\}$ es una $\mathbb{K}$-base para V. □

Corolario 1.5.36 $\frac{\mathbb{K}[x, E_1, ..., E_n]}{I} = V \oplus \mathbb{K}[x]$

Conclusión 1.5.37 *Todo elemento $h(x, E_1, ..., E_n) \in V$ (que en realidad es una clase por ser un elemento de $\frac{\mathbb{K}[x, E_1, ..., E_n]}{I}$) tiene por representante canónico a un elemento que es combinación $\mathbb{K}$-lineal de $\{\overline{E_1}, ..., \overline{E_n}\}$.*

Corolario 1.5.38 *Dado $f(x) \in \mathbb{K}[x], v = v_1 e_1 + ... + v_n e_n \in V$ la acción de $f(x)$ sobre v ($f(x) \cdot v$) es el resto de la división del polinomio $f(x) \cdot (v_1 E_1 + ... + v_n E_n)$ entre $\{xE_1 - \sum_{i=1}^{n} a_{i1} E_i, ..., xE_n - \sum_{i=1}^{n} a_{in} E_i\}$, respecto al orden monomial lexicográfico con $x > E_1 > ... > E_n$.*

Ejemplo 1.5.39 *Dado $V = \mathbb{Q}^2$ y $A = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}$, se calculará la acción de $x^2 + 2x - 2$ sobre $v = (1, -1)$.*

La acción de $x^2 + 2x - 2$ sobre $v = (1, -1)$ se corresponde con el representante canónico del elemento $(x^2 + 2x - 2) \cdot (E_1 - E_2)$, dicho representante no es más que el resto de su división entre $\{(1 - x)E_2, (1 - x)E_1 - 2E_2\}$, respecto al orden monomial lexicográfico con $x > E_1 > E_2$.

$(x^2+2x-2)(E_1-E_2) = (5+x)((1-x)E_2)+(-3-x)((1-x)E_1-2E_2)+E_1-9E_2$

Así pues $(x^2 + 2x - 2) \cdot (1, -1) = (1, -9)$

1.6. Programación de los resultados obtenidos

Puesto que es el Álgebra Computacional la disciplina a la que pertenece la teoría que se expone en esta memoria, es lógico esperar que parte de dicha memoria, consista en el desarrollo de un paquete de funciones, las cuáles implementan los resultados obtenidos hasta el momento. Éste es el motivo por el que se desarrolló, en lenguaje Mathematica, el paquete *SmithGroebner*.

La primera cuestión que hubo que decidir, a la hora de sentarse a programar el paquete *SmithGroebner*, fue la de escoger los DIPs sobre los que iba a actuar el paquete; en principio había dos opciones, la primera era la de escoger unos pocos DIPs y programar cada caso en particular, y la segunda era la de hacer un paquete en el que tuviese cabida cualquier DIP. Ambas opciones tienen sus ventajas e inconvenientes; la primera opción posee la ventaja de que al centrar la atención sólo en varios DIPs, es posible hacer rutinas más rápidas pues se puede acceder a funciones del propio sistema, esto es, menos DIPs más rapidez; mientras la segunda opción, posee la ventaja notable de ser un paquete que puede manejar cualquier DIP; no obstante, aunque parezca contradictorio, ésta es también su principal desventaja, pues al necesitar las rutinas lo más generales posibles, obliga a no poder utilizar las rutinas del propio sistema, y por lo tanto ya compiladas, que proporciona Mathematica. Sustituir estas rutinas por otras del paquete va a ralentizar, ya de partida, su funcionamiento.

Piénsese por ejemplo en los DIPs $\mathbb{Z}$ y $\mathbb{Q}$; si se desarrolla un paquete en el que prime la rapidez, se debería, por ejemplo, programar el algoritmo de Buchberger para $\mathbb{Z}$ [NaG94] y para $\mathbb{Q}$ por separado (para $\mathbb{Q}$ se puede utilizar la función GroebnerBasis). Si se desarrolla pensando en la segunda opción, se

programaría un único algoritmo de cálculo de bases de Gröbner para ideales de polinomios sobre DIPs [Mol88, Pan88].

Debido a su mayor interés matemático, es la segunda opción el camino escogido como planteamiento inicial.

El paquete *SmithGroebner* se diseñó pensando en un dominio de ideales principales cualquiera. De hecho, el nivel de abstracción que se alcanza es tal, que se puede añadir cualquier otro DIP que se desee, modificando solamente la función *CargaFunciones*.

Únicamente añadiendo un trocito de código en esta función, se puede ampliar, aún más, el rango de los DIPs que puede manejar el paquete; eso sí, sin perder ninguna de las funciones que ofrece. Para ello sólo se hubo que fijar en las operaciones que se hacían de distinta forma, según se estuviese en un DIP u otro (no es lo mismo, por ejemplo, calcular el m.c.d. en $\mathbb{Z}$ que en $\mathbb{Q}[x]$ o $\mathbb{Z}[\frac{1+\sqrt{-19}}{2}]$); definir unas funciones genéricas y, dependiendo del DIP escogido por el usuario, construir esas funciones genéricas de modo adecuado en tiempo de ejecución.

Las funciones genéricas que se utilizan son:

1. $MaximoComunDivisor[a, b]$
   Proporciona el *m.c.d.* de a y b.

2. $MaximoComunDivisorEx[a, b]$
   Proporciona la lista $\{d, \{\alpha, \beta\}\}$, siendo

   - d es el m.c.d. de a y b
   - $\alpha \cdot a + \beta \cdot b = d$

3. $DividirQ[a, b]$
   Función lógica que retorna True si a divide a b y False en otro caso.

4. $Cociente[a, b]$
   Proporciona el cociente de la división de a entre b (esta función sólo se utiliza cuando b divide a a)

5. $Normaliza[A, B : 0]$[1]
   Se utiliza para formatear la Forma Normal de Smith, antes de mostrarla al usuario (por ej., si se trabaja sobre $\mathbb{Z}$, Normaliza devuelve una matriz de enteros positivos).

   - $Normaliza[A]$ devuelve la matriz A formateada.

   - $Normaliza[A, B]$ devuelve $\{C, D\}$; siendo C la matriz A formateada y D la matriz de paso en columnas B, convenientemente modificada, pues hay que llevar cuenta de los cambios realizados.

6. $UnidadQ[Valor]$
   Retorna True si Valor es una unidad en el DIP en el que se trabaja, False en otro caso.

El funcionamiento del paquete es muy sencillo. Una vez que el usuario pulse "Return", antes de hacer ningún cálculo, se identifica el anillo escogido y se definen estas seis funciones de modo correcto (esta es la labor principal de la función CargaFunciones). Una vez hecho esto, se empieza a calcular. La función CargaFunciones actúa de intermediaria entre el usuario y el paquete.

Si se revisa el listado de las funciones, se puede observar que hay alguna función genérica más, pero las esenciales son estas que se acaban de describir.

Las líneas generales de funcionamiento, como se acaba de ver, son muy sencillas; sólo queda entonces por describir las funciones que proporciona el paquete *SmithGroebner* y los anillos sobre los que opera.

Ocho son las funciones que están a disposición del usuario:

1. *GroebnerEquivalente*
   Calcula una matriz Lad(A). Si se le añade la opción " Paso ->Sí " proporcionará además la matriz de paso.

[1] B:0 quiere decir que B es un argumento optativo, en caso de no aparecer, Mathematica toma B=0

2. *Smith*
   Calcula la Forma Normal de Smith de una matriz. Si se le añade la opción " Paso ->Sí " proporcionará además las matrices de paso.

3. *InversaPaso*
   Calcula la inversa de una matriz

4. *InversaMinimo*
   Calcula la inversa de una matriz de complejos

5. *Frobenius*
   Calcula la Forma Canónica Racional de una matriz y la matriz de cambio de base.

6. *SegundaFormaC*
   Calcula la Segunda Forma Canónica de una matriz y la matriz de cambio de base.

7. *Jacobson*
   Calcula la Forma Canónica de Jacobson de una matriz y la matriz de cambio de base.

8. *Generadores1D*
   Programación del algoritmo Generadores_Primera_Descomposicion, el cual se utiliza para calcular los elementos z_i y d_i, que proporcionan la descomposición del $K[x]$-módulo V en suma directa de cíclicos (véase 1.2).

Por último, los anillos de trabajo que maneja el paquete son:[2]

$\mathbb{Z}$	$\mathbb{Z}[i]$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{C}$	$\mathbb{Z}_p$
$\mathbb{F}_q$	$\mathbb{Z}[\frac{1+\sqrt{-19}}{2}]$	$\mathbb{Q}[x]$	$\mathbb{R}[x]$	$\mathbb{C}[x]$	$\mathbb{Z}_p[x]$
$\mathbb{F}_q[x]$					

[2] $\mathbb{F}_q$: Cuerpo finito de q elementos

De entre todos los DIPs que maneja el paquete escogidos inicialmente, llama seguramente la atención el anillo $\mathbb{Z}[\frac{1+\sqrt{-19}}{2}]$, este anillo es el único para el que no hay definido en Mathematica ningún comando específico; para todos los demás existen funciones definidas en mayor o menor medida. $\mathbb{Z}[\frac{1+\sqrt{-19}}{2}]$ constituye uno de los pocos ejemplos conocidos, de anillos que son DIPs y no son dominios euclídeos [Cam88, Gre97]. Es precisamente este motivo por el que se incluyó este anillo en el paquete. Para más información, se recomienda instalar la ayuda del paquete y consultar en el HelpBrowser de Mathematica, la sección Add-ons, subsección bases de Gröbner.

1.7. Aplicaciones

Con el correr del tiempo, la Forma Normal de Smith se ha ido aplicando para resolver los más diversos problemas; ahora, una vez que ha quedado de manifiesto que se puede calcular la Forma Normal de Smith mediante un cálculo sucesivo de bases de Gröbner; todas estas aplicaciones de la Forma Normal de Smith, pasan a ser problemas resolubles con técnicas de la Teoría de las bases de Gröbner.

1.7.1. Grupos Abelianos Finitamente Generados.

Si bien era necesaria una sección de aplicaciones del cálculo de la Forma Normal de Smith, donde trabajar con el paquete construido; no es menos cierto que, hablar del cálculo de la Forma Normal de Smith, y no hablar en primera instancia de la identificación de R-módulos finitamente generados sobre un dominio de ideales principales, en este caso $R = \mathbb{Z}$, sería incomprensible. Pues las dos, el cálculo de la Forma Normal de Smith y la identificación de R-módulos finitamente generados sobre un dominio de ideales principales, van íntimamente ligadas [Jac74, DuF99, F1879].

Todo grupo abeliano finitamente generado queda definido por un conjunto de generadores y uno de relaciones, y recíprocamente. La Forma Normal de Smith de la matriz del conjunto de relaciones, es la que proporciona la información que se necesita, para identificar el grupo.

Ejemplo 1.7.1 *Se identificará el grupo abeliano M generado por $\{a, b, c, d\}$, sujeto a las relaciones $\{a + b + d = 0, 2a - b = c\}$*

La matriz de relaciones es

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 2 & -1 & -1 & 0 \end{pmatrix}$$

Se calcula la Forma Normal de Smith

$$Smith[\{\{1, 1, 0, 1\}, \{2, -1, -1, 0\}\}, Z]$$

y se obtiene

$$S = \begin{pmatrix} d_1 = 1 & 0 & 0 & 0 \\ 0 & d_2 = 1 & 0 & 0 \end{pmatrix}$$

Entonces [DuF99]

$$M \cong \frac{\mathbb{Z}}{d_1\mathbb{Z}} \oplus \frac{\mathbb{Z}}{d_2\mathbb{Z}} \oplus \mathbb{Z} \oplus \mathbb{Z} \cong \mathbb{Z} \oplus \mathbb{Z}$$

Ejemplo 1.7.2 *Se identificará el grupo abeliano M generado por $\{a, b, c, d\}$, sujeto a las relaciones $\{2a - 3b = c, a + d = 0, 5d = 0\}$*

La matriz de relaciones es

$$\begin{pmatrix} 2 & -3 & -1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

Se calcula la Forma Normal de Smith

$$Smith[\{\{2,-3,-1,0\},\{1,0,0,1\},\{0,0,0,5\}\},Z]$$

y se obtiene

$$S=\begin{pmatrix} d_1=1 & 0 & 0 & 0 \\ 0 & d_2=1 & 0 & 0 \\ 0 & 0 & d_3=5 & 0 \end{pmatrix}$$

Entonces [DuF99]

$$M \cong \frac{\mathbb{Z}}{d_1\mathbb{Z}} \oplus \frac{\mathbb{Z}}{d_2\mathbb{Z}} \oplus \frac{\mathbb{Z}}{d_2\mathbb{Z}} \oplus \mathbb{Z} \cong \mathbb{Z}_5 \oplus \mathbb{Z}$$

1.7.2. Ecuaciones Diferenciales Lineales

Uno de los métodos de resolución de un sistema de m ecuaciones diferenciales lineales de orden n con coeficientes constantes, es el método de eliminación [NaS92]. Este método se basa en manejar la matriz de operadores diferenciales lineales asociada al sistema de ecuaciones. Esta matriz de operadores, se puede considerar como una matriz de polinomios en una variable; surge entonces de modo natural, la idea de aplicar el cálculo de la Forma Normal de Smith para su resolución.

Ejemplo 1.7.3 *Se calculará la solución general de:*

$$x''(t)+y'(t)-x(t)+y(t)=-1$$
$$x'(t)+y'(t)-x(t)=t^2$$

El sistema a resolver en notación de operadores diferenciales lineales es:[3]

[3] $L=d/dt$

$$\underbrace{\begin{pmatrix} L^2-1 & L+1 \\ L-1 & L \end{pmatrix}}_{A}\underbrace{\begin{pmatrix} x(t) \\ y(t) \end{pmatrix}}_{X(t)} = \underbrace{\begin{pmatrix} -1 \\ t^2 \end{pmatrix}}_{B(t)}$$

El primer paso a dar es el cálculo de la Forma Normal de Smith:

$$Smith[\{\{L^2-1, L+1\}, \{L-1, L\}\}, Q[L], Paso- > Sí]$$

De esta forma se obtiene

- $S = \begin{pmatrix} 1 & 0 \\ 0 & L^3-L^2-L+1 \end{pmatrix}$

- $P = \begin{pmatrix} 0 & 1 \\ -1 & -L^2+L+2 \end{pmatrix}$

- $Q = \begin{pmatrix} -1 & -L \\ 1 & L-1 \end{pmatrix}$

siendo S la Forma Normal de Smith de A, P la matriz de paso en filas y Q la matriz de paso en columnas ($S = P \cdot A \cdot Q$).

$$A \cdot X(t) = B(t) \Longleftrightarrow P \cdot A \cdot X(t) = P \cdot B(t)$$

Haciendo el cambio de variable $X(t) = Q \cdot Y(t)$

$$P \cdot A \cdot Q \cdot Y(t) = P \cdot B(t) \Longleftrightarrow S \cdot Y(t) = P \cdot B(t) \Longleftrightarrow$$

$$\Longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & L^3-L^2-L+1 \end{pmatrix}\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} t^2 \\ 2t^2+2t-1 \end{pmatrix}$$

El sistema a resolver es pues:

$$\begin{array}{l} y_1(t) = t^2 \\ y_2^{3)}(t) - y_2^{2)}(t) - y_2^{)}(t) + y_2(t) = 2t^2 + 2t - 1 \end{array}$$

Su solución es:

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} t^2 \\ C_1e^t + C_2te^t + C_3e^{-t} + 2t^2 + 6t + 9 \end{pmatrix}$$

Para obtener la solución del sistema planteado inicialmente, sólo queda por deshacer el cambio de variable:

$$X(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = Q \cdot Y(t) =$$

$$= \begin{pmatrix} -C_1e^t - C_2e^t - C_2te^t + C_3e^{-t} - t^2 - 4t - 6 \\ -2C_3e^{-t} + C_2e^t - t^2 - 2t - 3 \end{pmatrix}$$

Ejemplo 1.7.4 *Se calculará la solución general de:*

$$\begin{array}{l} x_1'(t) + x_2'(t) + x_3'(t) + 2x_1(t) + x_2(t) + 3x_3(t) = 0 \\ x_1'''(t) + x_2'''(t) + 2x_3'''(t) + 2x_1''(t) + x_2''(t) + 3x_3''(t) + x_1'(t) + x_2'(t) + x_3'(t) = 0 \\ x_1''(t) + x_2''(t) + 3x_3''(t) + 3x_1'(t) + 2x_2'(t) + 6x_3'(t) + 2x_1(t) + x_2(t) + 3x_3(t) = 0 \end{array}$$

El sistema a resolver en notación de operadores diferenciales lineales es:[4]

$$\underbrace{\begin{pmatrix} L+2 & L+1 & L+3 \\ L^3+2L^2+L & L^3+L^2+L & 2L^3+3L^2+L \\ L^2+3L+2 & L^2+2L+1 & 3L^2+6L+3 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix}}_{X(t)} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}}_{B(t)}$$

[4] $L=d/dt$

El primer paso a dar es el cálculo de la Forma Normal de Smith de A:

$$Smith[\{\{L+2, L+1, L+3\}, \{L^3+2L^2+L, L^3+L^2+L, 2L^3+3L^2+L\},$$
$$\{L^2+3L+2, L^2+2L+1, 3L^2+6L+3\}\}, Q[L], Paso- > Sí]$$

De esta forma se obtiene

- $S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & L^2+L \end{pmatrix}$

- $P = \begin{pmatrix} \frac{1}{2}+\frac{L^2}{2} & \frac{-1}{2} & 0 \\ L^3+2L^2+L & -L-2 & 0 \\ L+1 & 0 & -1 \end{pmatrix}$

- $Q = \begin{pmatrix} 1 & \frac{1}{2} & \frac{-1}{2}L^3-\frac{1}{2}L^2+1 \\ 0 & -1 & \frac{1}{2}L^3+L^2-\frac{1}{2} \\ 0 & 0 & \frac{-1}{2} \end{pmatrix}$

Haciendo el cambio de variable $X(t) = Q \cdot Y(t)$, el sistema a resolver es ahora:

$$\left\{ \begin{array}{l} y_1(t) = 0 \\ y_2'(t) = 0 \\ y_3''(t) + y_3'(t) = 0 \end{array} \right\}$$

Su solución es:

$$\begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} = \begin{pmatrix} 0 \\ C_1 \\ C_2 + C_3 e^{-t} \end{pmatrix}$$

Deshaciendo el cambio de variable:

$$X(t) = \begin{pmatrix} \frac{1}{2}C_1 + C_2 + C_3 e^{-t} \\ -C_1 - \frac{1}{2}C_2 \\ \frac{-1}{2}C_2 - \frac{1}{2}C_3 e^{-t} \end{pmatrix}$$

1.7.3. Ecuaciones Diofánticas

En esencia, el procedimiento que se utiliza para resolver los sistemas de ecuaciones diferenciales de la subsección anterior, es válido para resolver un sistema de ecuaciones lineales diofánticas $Ax = b$. Los pasos a dar son los mismos; primero se calcula la Forma Normal de Smith de A, esta vez eso sí, sobre $\mathbb{Z}$; se obtienen de este modo tres matrices S, P, Q, tales que S es la Forma Normal de Smith de A y $S = PAQ$; segundo, se multiplica el sistema por la matriz P y se hace el cambio de variable $X = Q \cdot Y$. En tercer lugar se resuelve, si es que se puede, el sistema $S \cdot Y = PB$, que es trivial pues S es diagonal, y, por último, se deshace el cambio de variable para obtener X, para una consulta más detallada sobre este particular se recomienda [Lak96].

Ejemplo 1.7.5 *Se resolverá la ecuación diofántica lineal*

$$8x + 10y = 1000$$

El primer paso a dar es expresar el sistema en forma matricial.

$$\begin{pmatrix} 8 & 10 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = (1000)$$

A continuación, se calcula la Forma Normal de Smith de A:

$$Smith[\{\{8, 10\}\}, Z, Paso- > Sí]$$

Se obtiene de este modo

$$S = \begin{pmatrix} 2 & 0 \end{pmatrix} \quad P = (1) \quad Q = \begin{pmatrix} -1 & 5 \\ 1 & -4 \end{pmatrix}$$

siendo S la Forma Normal de Smith ($S = PAQ$)

$$AX = b \Longleftrightarrow PAX = Pb$$

Haciendo el cambio de variable $X = QY$, queda:

$$PAQY = Pb \Longleftrightarrow SY = Pb \Longleftrightarrow \begin{pmatrix} 2 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = 1000$$

Que tiene por solución

$$\begin{aligned} y_1 &= 500 \\ y_2 &= \lambda \end{aligned}$$

Deshaciendo el cambio de variable

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 & 5 \\ 1 & -4 \end{pmatrix} \begin{pmatrix} 500 \\ \lambda \end{pmatrix} = \begin{pmatrix} -500 + 5\lambda \\ 500 - 4\lambda \end{pmatrix}$$

Por lo tanto la solución del sistema inicial es:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -500 + 5\lambda \\ 500 - 4\lambda \end{pmatrix} \lambda \in \mathbb{Z}$$

Ejemplo 1.7.6 *Se resolverá el sistema lineal de ecuaciones diofánticas lineales*

$$\begin{aligned} 3x + 6y + 5z + 2t &= -2 \\ x + y + z - 5t &= 0 \\ -7x + 2y - 2z + t &= 1 \end{aligned}$$

El primer paso a dar es expresar el sistema en forma matricial.

$$\underbrace{\begin{pmatrix} 3 & 6 & 5 & 2 \\ 1 & 1 & 1 & -5 \\ -7 & 2 & -2 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} -2 \\ 0 \\ 1 \end{pmatrix}}_{b}$$

A continuación, se calcula la Forma Normal de Smith de A:

$$Smith[\{\{3,6,5,2\},\{1,1,1,-5\},\{-7,2,-2,1\}\},Z,Paso->Sí]$$

Se obtiene de este modo

$$S=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&3&0\end{pmatrix}$$

$$P=\begin{pmatrix}0&1&0\\1&-3&0\\-4&19&1\end{pmatrix}$$

$$Q=\begin{pmatrix}1&0&40&-39\\0&1&53&-51\\0&-1&-88&85\\0&0&1&-1\end{pmatrix}$$

siendo S la Forma Normal de Smith ($S=PAQ$)

$$AX=b\Longleftrightarrow PAX=Pb$$

El sistema a resolver es

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&3&0\end{pmatrix}\begin{pmatrix}y_1\\y_2\\y_3\\y_4\end{pmatrix}=\begin{pmatrix}0\\-2\\9\end{pmatrix}$$

Cuya solución es

$$\begin{array}{l} y_1 = 0 \\ y_2 = -2 \\ y_3 = 3 \\ y_4 = \lambda \end{array} \quad \lambda \in \mathbb{Z}$$

Deshaciendo el cambio de variable se obtiene la solución del sistema de partida

$$\begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} 120 - 39\lambda \\ 157 - 51\lambda \\ -262 + 85\lambda \\ 3 - \lambda \end{pmatrix} \lambda \in \mathbb{Z}$$

Capítulo 2

Bases de Gröbner y el Algoritmo de Berlekamp

Se puede afirmar, sin temor a equivocarse, que es un hecho frecuente cuando se está trabajando en algún proyecto de investigación, que se obtengan resultados colaterales, en principio inesperados; al menos esto es lo que ha ocurrido en este caso, y fruto del desarrollo del extenso paquete *Smith-Groebner*, se ha obtenido, sin pretenderlo inicialmente, una versión con bases de Gröbner del algoritmo de Berlekamp [Ber67]. Para evitar confusiones es necesario decir que no se busca, en ningún momento, dar un algoritmo más rápido que el algoritmo clásico de Berlekamp; el verdadero objetivo es el de establecer la relación téorica existente entre dicho algoritmo y las bases de Gröbner.

El uso de las bases de Gröbner para factorizar polinomios no es novedoso, pues para polinomios en dos variables, sorprendentemente, dichas técnicas son útiles en la práctica. Noro y Yokoyama [NoY02], usando cambios de ordenes en bases de Gröbner de ideales de dimensión 0, propusieron un algoritmo en $\mathbb{F}_q[x, y]$ en tiempo polinómico que puede ser implementado eficientemente.

Este segundo capítulo consta de cuatro secciones; en la primera sección se presenta una breve introducción sobre la factorización de polinomios, y el lugar que ocupa el algoritmo de Berlekamp en este contexto; a continuación se expone brevemente el algoritmo inicial de Berlekamp para, en la siguiente sección, enunciar el teorema que justifica una versión con bases de Gröbner de dicho algoritmo. Por último, en la cuarta sección se proporciona el listado, en Maple, de las cinco funciones que implementan la versión con bases de Gröbner del algoritmo de Berlekamp.

2.1. Introducción

El problema de factorizar un polinomio en una o varias variables sobre un cuerpo finito, los números racionales o los números complejos es uno de los mayores éxitos en el área de la computación simbólica en los últimos cuarenta años. Los ejemplos más importantes son los anillos $\mathbb{F}_q[x], \mathbb{Z}[x]$ y, como ejemplo más reciente de interés computacional, $\mathbb{Q}(X)[d/dX]$ (este anillo no es conmutativo y la factorización en irreducibles no es única, pero su interés radica en que permite encontrar relaciones algebraicas entre soluciones de ecuaciones diferenciales lineales).

Aquí, consideramos el problema de factorizar polinomios sobre cuerpos finitos. La factorización de polinomios sobre cuerpos finitos, a parte de su interés matemático intrínseco, juega un papel muy importante en muchas áreas de las Matemáticas, especialmente en Criptografía, ampliamente desarrollada desde la popularización de Internet, Teoría de Números y Teoría de Códigos.

Por otro lado, conocer cómo se factorizan polinomios sobre cuerpos finitos es muy importante, pues es la base sobre la que se construyen algoritmos de factorización de polinomios sobre otros anillos, tales como

$$\mathbb{Z}[x], \mathbb{Q}[x], \mathbb{Z}[x_1, ..., x_n], \mathbb{Q}[x_1, ..., x_n], \mathbb{F}_q[x_1, ..., x_n], \mathbb{Q}(\alpha)[x], \mathbb{Q}(\alpha)[x_1, ..., x_n]$$

La factorización de polinomios en cuerpos finitos se usa como un subproblema en algoritmos para factorizar polinomios sobre los enteros; la construcción de Hensel permite reducir problemas de factorización de polinomios en $\mathbb{Z}[x_1, ..., x_n]$ a factorizar polinomios en $\mathbb{F}_p[x]$, p primo.

Polinomios con coeficientes en un cuerpo finito y sus propiedades de factorización fueron estudiadas desde mediados del siglo XIX. Sin embargo, no fue hasta 1967 cuando Berlekamp [Ber67] presentó un algoritmo eficiente que factoriza polinomios en una variable sobre un cuerpo finito de q elementos.

Hablando en términos generales, los métodos de factorización de polinomios que existen actualmente se pueden clasificar en dos partes, perfectamente diferenciadas: técnicas deterministas y técnicas probabilísticas.

Mientras las técnicas deterministas se basan en los algoritmos clásicos, las técnicas probabilísticas están basadas en algoritmos que utilizan métodos probabilísticos para obtener el resultado deseado.

Utilizar un método determinista garantiza que la solución (al menos en teoría) siempre se alcanzará; aunque en la práctica, el número de operaciones sea tan alto que haga imposible la aplicación de dicho algoritmo en un tiempo razonable. En frente a esta paradoja de tener un algoritmo, y por lo tanto la solución de un determinado problema, pero que este algoritmo no sea aplicable debido al gran número de operaciones que se deben llevar a cabo, se encuentran los métodos probabilísticos, que son los más utilizados. En esta clase de métodos, al contrario de lo que pasaba con las técnicas deterministas, existe la posibilidad de que el algoritmo conduzca a un error; evidentemente, la probabilidad de que esto suceda, siempre se busca que sea lo más pequeña posible. Una de las formas de conseguirlo, que es la que más se utiliza, por ejemplo cuando se aplica el algoritmo de *Cantor-Zassenhaus* [CaZ81], es la de pasar el algoritmo varias veces. Este pequeño e ingenioso truco reduce la probabilidad de que el algoritmo falle. En compensación, este tipo de algoritmos suelen ser más rápidos que los algoritmos deterministas.

El algoritmo de Berlekamp [Ber67] y el algoritmo de Niederreiter [Nie94] son dos ejemplos de técnicas deterministas de factorización de polinomios. El algoritmo probabilístico de Berlekamp tipo Las Vegas [Ber70] y los algoritmos de Cantor-Zassenhaus [CaZ81], de von zur Gathen-Shoup [GaS92] y de Kaltofen-Shoup [KaS98] son ejemplos de ténicas probabilísticas de factorización. Estos algoritmos probabilísticos son en tiempo polinómico mientras que sigue siendo un problema abierto si existe un algoritmo determinista en tiempo polinómico.

El interés de este capítulo está centrado en el algoritmo de Berlekamp; este algoritmo proporciona la factorización de un polinomio libre de cuadrados con coeficientes sobre un cuerpo finito; nótese que la verdadera dificultad a la hora de factorizar un polinomio es dicha factorización, pues existen algoritmos, relativamente sencillos, que determinan la factorización libre de cuadrados de un polinomio [GCL92, pág. 337].

La clave para obtener la versión con bases de Gröbner del algoritmo de Berlekamp, está en utilizar dichas bases para calcular los máximos comunes divisores que se necesitan cuando se aplica el algoritmo de Berlekamp.

2.2. El Algoritmo de Berlekamp

Sea $\mathbb{F}_q$ un cuerpo finito de q elementos, característica p, y $a(x) \in \mathbb{F}_q[x] - \{0\}$ un polinomio mónico de grado n mayor que cero y libre de cuadrados.

Definición 2.2.1 *Sea* $W := \{[h(x) \in \frac{\mathbb{F}_q[x]}{<a(x)>}/[h(x)]^q = [h(x)]\}$.

Proposición 2.2.2 *[GCL92] W es un* $\mathbb{F}_q$*-subespacio vectorial de* $\frac{\mathbb{F}_q[x]}{<a(x)>}$.

Por ser $a(x)$ libre de cuadrados es posible encontrar k polinomios irreducibles $a_1(x), ..., a_k(x)$ tales que $a(x) = a_1(x) \cdot ... \cdot a_k(x)$. Gracias a esta

factorización en irreducibles, es posible definir la siguiente aplicación

$$\begin{array}{lcl} \phi : \frac{\mathbb{F}_q[x]}{<a(x)>} & \longrightarrow & \frac{\mathbb{F}_q[x]}{<a_1(x)>} \times ... \times \frac{\mathbb{F}_q[x]}{<a_k(x)>} \\ [h(x)] & \rightsquigarrow & \phi([h(x)]) = ([h(x)], ..., [h(x)]) \end{array}$$

ϕ es un homomorfismo de anillos. Más aún, por ser $a_1(x), ..., a_k(x)$ irreducibles, se verifica que los ideales $< a_1(x) >, ..., < a_n(x) >$ son primos entre sí, y además, que $\bigcap_{i=1}^{k} < a_i(x) > = < a(x) >$. Es por esto que se puede concluir, gracias al Teorema Chino de los Restos, que ϕ es un isomorfismo de anillos.

Sea $h(x) \in \mathbb{F}_q(x)$, se verifica:

$$\text{Si } h(x)^q \equiv h(x) \, mod \, a(x) \Rightarrow h(x)^q \equiv h(x) \, mod \, a_i(x)$$

Así la restricción de ϕ a W da lugar al siguiente homomorfismo de anillos (que también lo es de $\mathbb{F}_q$-espacios vectoriales):

$$\psi = \phi_{|W} : W \longrightarrow W_1 \times ... \times W_k$$

siendo $W_i := \{[h(x)] \in \frac{\mathbb{F}_q[x]}{<a_i(x)>} / [h(x)]^q = [h(x)]\}$.

Más aún, considerando a W_i como el conjunto de raíces del polinomio $z^q - z \in \frac{\mathbb{F}_q[x]}{<a_i(x)>}[z]$, se deduce que W_i es isomorfo como $\mathbb{F}_q$-espacio vectorial a $\mathbb{F}_q$, esto es, es un subespacio vectorial de $\frac{\mathbb{F}_q[x]}{<a(x)>}$ de dimensión 1.

Teorema 2.2.3 *[GCL92] La aplicación ψ es un isomorfismo de anillos y de $\mathbb{F}_q$-espacios vectoriales. En particular, la dimensión de W es el número de factores irreducibles de la factorización de $a(x)$.*

Así, $a(x)$ es irreducible si y sólo si $\dim_{\mathbb{F}_q} W = 1$.

Definición 2.2.4 *[GaG99] Sea Q la matriz que tiene por filas (y en este orden), los restos de $x^0, x^q, ..., x^{(n-1)q}$ módulo $a(x)$. La matriz Q se llama Matriz de Berlekamp-Petr.*

Teorema 2.2.5 *[GCL92]*

$$W = \{v = (v_0, ..., v_{n-1}) / v(Q - I) = 0\}^1$$

Teorema 2.2.6 *[GCL92, LiN97] Sea $h(x) \in W$ un polinomio no constante. Entonces se verifica que:*

$$a(x) = \prod_{c \in \mathbb{F}_q} GCD(a(x), h(x) - c)$$

El algoritmo de Berlekamp [Ber67] consiste en:

1. Calcular una $\mathbb{F}_q$-base para W $(\{1, h_2(x), ..., h_k(x)\})$
2. Aplicar el Teorema 2.2.6, tomando $h(x) = h_2(x)$
3. Si $a(x)$ factorizase en un producto de k factores, entonces el proceso se para. De no ser así, se aplica el Teorema 2.2.6 a cada factor de $a(x)$ obtenido previamente, tomando $h(x) = h_3(x)$; y así sucesivamente.

2.3. Berlekamp con Bases de Gröbner

Calcular una $\mathbb{F}_q$-base para W, es el primer punto a tratar a la hora de aplicar el algoritmo de Berlekamp. El Teorema 2.2.5 proporciona el modo de hacerlo; para ello sólo hay que resolver el siguiente sistema lineal de ecuaciones:

$$(x_1, ..., x_n)(Q - I) = (0, ..., 0)$$

para obtener una $\mathbb{F}_q$-base para W. Una de las aplicaciones más conocidas de la teoría de las bases de Gröbner es la resolución de sistemas de ecuaciones polinómicas [GCL92, CLO92, AdL94], razón por la cual este punto no representa un problema y puede ser resuelto calculando una base de Gröbner.

[1] *I es la matriz identidad de orden n*

Desde cierto punto de vista, se podría decir que el algoritmo de Berlekamp es la aplicación reiterada del Teorema 2.2.6. La mayor dificultad en este resultado es calcular todos los máximos comunes divisores.

Un problema similar a este, fue tratado y resuelto con bases de Gröbner por Czichowski [Czi95]. El contexto era bien distinto, pues el artículo de Czichowski trata sobre el cálculo de integrales de funciones racionales sobre un cuerpo de característica cero; y aquí se trata el problema de la factorización de un polinomio en una variable sobre un cuerpo finito. De todas formas, la idea de cómo calcular estos máximos comunes divisores con bases de Gröbner es la misma en ambos casos.

El punto de partida del artículo de Czichowski es la fórmula de integración dada por Rothstein [Rot76], Trager [Tra76] y Lazard y Rioboo [LaR90]

$$\int \frac{A(x)}{D(x)}dx = \sum_{c/R(c)=0} c \cdot \log(GCD(A(x) - c \cdot \frac{dD}{dx}(x), D(x))) \qquad (2.1)$$

siendo $R(z) = Resultante_x(A(x) - z \cdot \frac{dD}{dx}(x), D(x))$, $D(x)$ un polinomio mónico y libre de cuadrados, de grado mayor que cero, $\deg(A(x)) < \deg(D(x))$ y $GCD(A(x), D(x)) = 1$.

Czichowski probó que la base de Gröbner reducida del ideal $< A(x) - z \cdot \frac{dD}{dx}(x), D(x) >$ respecto al orden monomial lexicográfico con $x > z$, proporciona los máximos comunes divisores buscados.

Las ideas de Czichowski, como posteriormente se verá, trasladan completamente al contexto de la factorización de polinomios sobre un cuerpo finito. Curiosamente, las ideas de Czichowski se adaptan mucho mejor a un contexto de cuerpos finitos que a uno de característica cero; pues en el contexto de cuerpos finitos, como se verá a continuación, las raíces de la resultante siempre se pueden calcular, mientras en característica cero no siempre se puede.

Teorema 2.3.1 *Sean $a(x), h(x) \in \mathbb{F}_q[x]$, con $a(x)$ mónico, libre de cuadrados y $deg(a(x)) = n \geq 1$, $h(x) \in W$, $I =< a(x), h(x) - z > \subset \mathbb{F}_q[x, z]$, y sea $G = \{P_1(x,z) = R_1(z) \cdot x^{n_1} + ..., P_2(x,z) = R_2(z) \cdot x^{n_2} + ..., ..., P_m(x,z) = R_m(z) \cdot x^{n_m} + ...\}$ la base de Göbner Reducida de I, respecto al orden monomial lexicográfico con $x > z$. Se supone además, que G está ordenado con respecto a sus términos principales en orden ascendente. Se verifica:*

a) $R_{k+1}(z)|R_k(z)$, $k = 1, ..., m-1$.

b) $R_m(z) \in F_q$.

c) $P_1(x,z) = R_1(z)$

d) $R_1(z)$ *tiene las mismas raíces que* $Resultante_x(a(x), h(x) - z)$ *pero con multiplicidad uno.*

e) $Resultante_x(a(x), h(x) - z)$ *posee todas sus raíces en* $\mathbb{F}_q$.

f) $P_k(x,z) = R_k(z) \cdot S_k(x,z), k = 1, ..., m.$
siendo $S_k(x,z)$ *un polinomio mónico con respecto a su mayor potencia en* x *(*$S_k(x,z) = x^{n_k} + ...$*)*

g) *Si* $Q_k(z), k = 1, ...m-1$, *denota al polinomio definido por* $R_k(z) = Q_k(z) \cdot R_{k+1}(z)$, *entonces:*

$$a(x) = \prod_{k=1,...,m-1} \left(\prod_{c/Q_k(c)=0} S_{k+1}(x,c) \right)$$

Demostración:

a)

Sea $k \in \{1, 2, ..., m-1\}$, con $n_k < n_{k+1}$ y G la base de Gröbner Reducida del Ideal I, entonces $deg(R_k(z)) > deg(R_{k+1}(z))$. Además existen $A(z), B(z) \in \mathbb{F}_q[z]$ tales que $A(z) \cdot R_k(z) + B(z) \cdot R_{k+1}(z) = GCD(R_k(z), R_{k+1}(z))$.

Sea $P(x,z) = A(z) \cdot x^{n_{k+1}-n_k} \cdot P_k(x,z) + B(z) \cdot P_{k+1}(x,z) = GCD(R_k(z), R_{k+1}(z)) \cdot x^{n_{k+1}} + ...$ Puesto que $P(x,z) \in I$, entonces $P(x,z) \Longrightarrow_G 0$. El término principal de $P(x,z)$ es $GCD(R_k(z), R_{k+1}(z)) \cdot x^{n_{k+1}}$ y sólo es divisible por el término principal de $P_{k+1}(x,z)$. Es por esto, que se verifica que $GCD(R_k(z), R_{k+1}(z)) = R_{k+1}(z)$, y por lo tanto $R_{k+1}(z)|R_k(z)$ para todo k, $k \in \{1, ..., m-1\}$

b)

Puesto que $a(x) \Longrightarrow_G 0$ y el término principal de $a(x)$ es x^n, entonces $R_m(z)$ debe ser constante.

c)

Es bien conocida de Geometría Algebraica la correspondencia entre ideales de $\mathbb{K}[x_1, ..., x_n]$ y variedades de $\mathbb{K}^n$ ($\mathbb{K}$ cuerpo), que viene dada por:

Para $V \subseteq \mathbb{K}^n$

$$i(V) = \{f \in \mathbb{K}[x_1, ..., x_n] : \ f(x) = 0 \ \forall x \in V\}$$

Para un ideal $I \subseteq \mathbb{K}[x_1, ..., x_n]$

$$v(I) = \{x \in \mathbb{K}^n : \ f(x) = 0 \ \forall f \in I\}$$

Para obtener el resultado, se comprobará en primer lugar que $iv(I) = I$.

$\{a(x), h(x) - z\}$ es una base de Gröbner Reducida de I, respecto al orden monomial lexicográfico con $z > x$ (este hecho es de comprobación rutinaria). Sea $F(x, z) \in iv(I)$, por el algoritmo de Euclides $F(x, z) = Q_1(x, z) \cdot a(x) + Q_2(x, z) \cdot (h(x) - z) + R(x)$, con $deg(R(x)) < n$. Ya que $R(x) \in iv(I)$ y $a(x)$ es libre de cuadrados entonces $R(x) = 0$ y $iv(I) = I$.

Por hipótesis, $a(x)$ es libre de cuadrados, es por esto que la ecuación $a(x) = 0$ tiene n soluciones diferentes (posiblemente en la clausura algebraica de $\mathbb{F}_q$); sean estas $x_1, ..., x_n$.

Se define $z_i = h(x_i)$, $i \in \{1, ...n\}$. Puesto que $h(x) \in W$ si y sólo si $h^q(x) \equiv h(x) \ mod \ a(x)$ entonces $h^q(x_i) = h(x_i)$, $i \in \{1, ..., n\}$ y por lo tanto $z_i = h(x_i) \in \mathbb{F}_q$, $i \in \{1, ..., n\}$.

Los puntos de la variedad $v(I)$, siempre tienen su segunda componente en $\mathbb{F}_q$. Sea $g(z) = \prod_{i=1,...,n} (z - z_i) \in iv(I) = I$. Ya que g tiene todas sus raíces en $\mathbb{F}_q$ entonces $I_1 = I \cap \mathbb{F}_q[z] \neq \emptyset$. Teniendo en cuenta el orden de G y el Teorema de Eliminación [CLO92], es posible concluir que $P_1(x, z) = R_1(z)$ y $I_1 =< R_1(z) >$.

d)

Como $g(z) \in I_1$, existe $b(z) \in \mathbb{F}_q[z]$ tal que $g(z) = R_1(z) \cdot b(z)$; se sigue entonces que $R_1(z)$ posee todas sus raíces en $\mathbb{F}_q$.

Ya que $\text{Resultante}_x(a(x), h(x) - z) \in \mathbb{F}_q[z] \cap$ I entonces se verifica que $R_1(z) | Resultante_x(\, a(x), h(x) - z)$ y $Resultante_x(a(x), h(x) - z) = \prod_{i=1,...,n} (h(x_i) - z) = g(z)$ [GCL92] luego $g, R_1 \in iv(I) = I$, $g = R_1 \cdot b$ y las raíces de g son $z_1, ..., z_n$.

Por lo tanto se puede afirmar que g y R_1 tienen las mismas raíces, pero con distintas multiplicidades; además $R_1(z)$ tiene las mismas raíces que $Resultante_x(a(x), h(x) - z)$, pero con multiplicidad uno.

e)

$Resultante_x(a(x), h(x) - z) = g(z)$, con lo cual tiene todas sus raíces en $\mathbb{F}_q$.

f)

Se comprueba ahora que $P_k(x, z) = R_k(z) \cdot S_k(x, z)$ para $k = 1, ..., m$.

Se procede por inducción.

Para $k = 1$

$P_1(x, z) = R_1(z) \cdot S_1(x, z)$, siendo $S_1(x, z) = 1$ (véase c))

Se supone que el resultado es cierto para k. Sea $Q_k(z) \in \mathbb{F}_q[z]$ tal que $R_k(z) = Q_k(z) \cdot R_{k+1}(z)$.

Se define:

$$P(x, z) = Q_k(z) \cdot P_{k+1}(x, z) - x^{n_{k+1} - n_k} \cdot P_k(x, z) \in I$$
$$P(x, z) \Longrightarrow_G 0, \text{ e incluso más, } P(x, z) \Longrightarrow_{\{P_1, ..., P_k\}} 0$$

Entonces:

$$Q_k(z) \cdot P_{k+1}(x, z) = \sum_{i=1,...,k} c_i(x, z) \cdot P_i(x, z)$$
$$P_{k+1}(x, z) = \underbrace{\boxed{\frac{R_k(x,z)}{Q_k(z)}}}_{R_{k+1}(z)} \cdot \sum_{i=1,...,k} c_i(x, z) \cdot \underbrace{\boxed{\frac{P_i(x,z)}{R_k(x,z)}}}_{R_k(z) | R_i(z) \;\; i = 1, ..., k-1}$$

y por lo tanto $R_{k+1}(z) | P_{k+1}(x, z)$

g)

Sea $k \in \{1, ..., m-1\}, c \in \mathbb{F}_q$ tal que $Q_k(c) = 0$. Se procederá a comprobar en primer lugar que $S_{k+1}(x, c) = GCD(a(x), h(x) - c)$

Sea $j > k$, se define

$$P(x, z) = Q_j(z) \cdot P_{j+1}(x, z) - x^{n_{j+1}-n_j} \cdot P_j(x, z) \Longrightarrow_G 0$$

Entonces

$$Q_j(z) \cdot P_{j+1}(x, z) = \sum_{i=1,...,j} c_i(x, z) \cdot P_i(x, z),$$

evaluado en c es

$$P_{j+1}(x, c) = \sum_{i=k+1,...,j} \frac{c_i(x,c)}{Q_j(c)} \cdot P_i(x, c)$$

Se puede concluir entonces que

$$P_{k+1}(x, c) | P_j(x, c) \quad j \in \{k + 1, ..., m\}$$

Por lo tanto

$$P_{k+1}(x, c) = GCD(\{P_j(x, c)\}_{j=k+1}^{m}),$$

o si se escoge un polinomio mónico

$$S_{k+1}(x, c) = GCD(\{P_j(x, c)\}_{j=k+1}^{m}).$$

Como $\{P_1(x, z), ...P_m(x, z)\}$ es una base de Gröbner $I =< a(x), h(x) - z >$, se verifica que:

$$< P_1(x, c), ..., P_m(x, c) >=< a(x), h(x) - c >.$$

Este hecho, junto al Teorema 2.2.6, proporciona el resultado deseado:

$$a(x) = \prod_{k=1,...,m-1} \Big(\prod_{c/Q_k(c)=0} S_{k+1}(x, c) \Big)$$

□

Después de este desarrollo teórico, se hace necesario explicar la potencia de este resultado con un ejemplo.

Ejemplo 2.3.2 *Se factoriza el polinomio libre de cuadrados* $a(x) = x^4 + 2 \in \mathbb{Z}_3[x]$.

Nota: Téngase en cuenta que se está trabajando en el anillo cociente $\frac{\mathbb{Z}_3[x]}{<a(x)>}$ *, el cual tiene subyacente una estructura de* $\mathbb{F}_3$*-espacio vectorial con*

base $\{1, x, x^2, x^3\}$*; y que este es el motivo por el que se hablará indistintamente del vector* (a, b, c, d) *o del polinomio* $a + bx + cx^2 + dx^3$.
El primer paso a dar es construir la matriz de Berlekamp-Petr:

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Una Z_3*-base para* W *es* $\{(1,0,0,0), (0,0,1,0), (0,1,0,1)\}$*, y por lo tanto el número de factores irreducibles en los cuales* $a(x)$ *se descompone es 3.*

Se aplica el Teorema 2.3.1 al vector $(0,0,1,0) = x^2$.

Puesto que el ideal I *es:*

$$I =< x^4 + 2, x^2 - z >.$$

y la base de Gröbner Reducida de I *con respecto al orden monomial lexicográfico, con* $x > z$*, es*

$$G = \{2 + z^2, x^2 - z\}.$$

Entonces

$$\begin{aligned} P_1(z) &= 2 + z^2 = R_1(z) \\ P_2(x, z) &= x^2 - z \\ R_2(z) &= 1 \\ S_2(x, z) &= x^2 - z \end{aligned}$$

y, como consecuencia del Teorema 2.3.1:

$$a(x) = \prod_{k=1,\ldots,2-1} \Big(\prod_{c/Q_k(c)=0} S_{k+1}(x, c) \Big) = \prod_{c/c^2+2=0} S_2(x, c) =$$

$$= S_2(x, 1) \cdot S_2(x, 2) = (x^2 - 1) \cdot (x^2 - 2)$$

Como $a(x)$ descompone en 2 factores (y deben ser 3), hay que continuar aplicando el Teorema 2.3.1. Para hacerlo, se toma el siguiente elemento de la base y uno de los factores; se aplica entonces el Teorema 2.3.1 al vector $(0,1,0,1) = x + x^3$ y al primer factor, el ideal I es ahora:

$$I =< x^2 - 1, x^3 + x - z >$$

La base de Gröbner Reducida de I con respecto al orden monomial lexicográfico, con $x > z$ es

$$G = \{z^2 + 2, x + z\}$$

Así pues

$$P_1(x,z) = z^2 + 2 = (z-1)\cdot(z-2) = R_1(z)$$
$$P_2(x,z) = x + z$$
$$R_2(z) = 1$$
$$S_2(x,z) = x + z$$

Entonces

$$x^2 - 1 = S_2(x,2)\cdot S_2(x,1) = (x+2)\cdot(x+1)$$

$$\boxed{a(x) = (x+2)\cdot(x+1)\cdot(x^2-2)}$$

2.4. Programación en Maple

De entre el gran abanico de posibilidades de cálculo que ofrece Maple se encuentra la factorización de un polinomio en una variable con coeficientes sobre un cuerpo finito. Para ello, se proporcionan dos comandos **sqrfree(f) mod p** y **Berlekamp(g,x,$\mathbb{K}$) mod p**; con el primero se obtiene la factorización del polinomio **f** en producto de polinomios libres de cuadrados, mientras que aplicar el segundo comando a cada uno de estos polinomios por separado proporciona su factorización. El objetivo final de este capítulo no es otro que hacer el segundo paso utilizando las bases de Gröbner como

herramienta de cálculo, sin entrar en consideraciones de que método es el más rápido.

Con este objetivo en mente se han desarrollado cinco funciones en Maple 9.5, que implementan el algoritmo de factorización que se desprende del Teorema 2.3.1, éstas funciones son:

1. *GBerlekamp*
   Única función que está a disposición del usuario y que calcula la factorización de un polinomio en una variable, libre de cuadrados y con coeficientes sobre un cuerpo finito.

2. *BerlekampPetrMatrix*
   Subrutina de la función GBerlekamp que obtiene la Matriz de *Berlekamp-Petr*.

3. *WBasis*
   Subrutina de la función GBerlekamp que calcula una base para el subespacio W.

4. *GroebnerKernel*
   Subrutina de la función GBerlekamp que calcula la factorización de un polinomio mónico y libre de cuadrados.

5. *CanonicalForm*
   Subrutina de la función GBerlekamp que calcula representantes canónicos de elementos de un cuerpo finito.

En las líneas que siguen se muestra, en primer lugar, un par de ejemplos de utilización de este paquete y, en segundo lugar el listado de dichas funciones.

Ejemplo 2.4.1 *Se calculará la factorzación de* $x^4 + 2 \in \mathbb{Z}_3[x]$

```
> GBerlekamp(x^4+2,x,3,alpha,alpha);
```

$$\{2 + x,\ x^2 + 1,\ 1 + x\}$$

Ejemplo 2.4.2 *Se calculará la factorización de* $(1+\alpha+\alpha^3)+\alpha x+(1+\alpha^2+\alpha^3)x^3+(1+\alpha+\alpha^3)x^4+x^5 \in \frac{\mathbb{Z}_2[\alpha,x]}{<\alpha^4+\alpha+1>} = \mathbb{F}_{16}$.

```
> GBerlekamp(1+alpha+alpha^3+alpha*x+(1+alpha^2+alpha^3)*x^3+
> (1+alpha+alpha^3)*x^4+x^5,x,2,alpha^4+alpha+1,alpha);
```

$$\{\alpha^3+1+(\alpha^3+\alpha)\,x+x^3+x^2\,(\alpha^2+\alpha+1),\ \alpha^2+1+(\alpha^3+\alpha^2)\,x+x^2\}$$

```
# Para ejecutar correctamente este código, es necesario cargar
# previamente el paquete GB de Andreas Pirklbauer's. Este paquete, que
# está localizado en la Share Library, sirve para calcular bases de Gröbner
# sobre cuerpos finitos.
```

```
GBerlekamp:=proc(apoly,var,p::prime,irredpoly,alpha)

# Esta función devuelve la factorización de un polinomio mónico
# y libre de cuadrados.
    local GB_apolydeg,GB_card::integer,GB_W,proceslist,aux,icont,
          hcont,dimW::integer,blnContinue,cardproceslist::integer,
          domloop, apolycan,irredpolycan;

    apolycan:=CanonicalForm(apoly,irredpoly,alpha,p);
    irredpolycan:=Normal(irredpoly) mod p;
    proceslist:={apolycan};
    blnContinue:=true;
    GB_apolydeg:=degree(apoly,var);
    GB_card:=p^degree(irredpoly,alpha);

# Se calcula una base para W.
    GB_W:= WBasis(apolycan,var,GB_apolydeg,p,GB_card,
                  irredpolycan,alpha);

    GB_W:=evalm([seq(var^icont,icont=0..GB_apolydeg-1)]
```

```
                    &* GB_W);
    GB_W:=convert(GB_W,set);
    dimW:=nops(GB_W);
    if dimW > 1 then
        for hcont in GB_W[2..-1] while blnContinue
        do
            domloop:=proceslist;
            for icont in domloop while blnContinue
            do
                aux:=GroebnerKernel(icont,hcont,var,p,
                                irredpolycan,alpha);
                if nops(aux) > 1 then
                    proceslist:=proceslist union {op(aux)};
                    proceslist:=proceslist minus {icont};
                fi;
                if dimW=nops(proceslist) then
                    blnContinue:=false;
                fi;
            end do;
        end do;
    fi;
    return(proceslist);
end proc;
```

```
BerlekampPetrMatrix:=proc(apoly,var,apolydeg,p,cardfinitefield,
                                irredpoly,alpha)
# Esta función devuelve la matriz de Berlekamp-Petr
```

```
    local mat::list,apolydegm1::integer,aux,icont::integer,jcont::integer;
    mat:=array(1..apolydeg,1..apolydeg);
    apolydegm1:=apolydeg-1;
    for icont from 0 to apolydegm1
     do
        aux:=Rem(
                    var^(icont*cardfinitefield),
                    apoly,
                    var) mod p;
        aux:=CanonicalForm(aux,irredpoly,alpha,p);
        for jcont from 0 to apolydegm1
         do
            mat[icont+1,jcont+1]:=coeff(aux,var,jcont);
        end do;
    end do;
    return evalm(mat);
end proc;
```

```
WBasis:=proc(apoly,var,apolydeg,p,cardfinitefield,irredpoly,alpha)
# Esta rutina obtiene una base para W
    local sysmat,tagvars,syspoly,indepvars,aux::integer,
          hbasis,icont::integer,jcont::integer,scont::integer,
          tcont::integer,equats,remvars,dimKer::integer;
    tagvars:=[seq(tagvar[icont],icont=1..apolydeg)];
    indepvars:=[];
    # Matriz del Sistema a resolver
    sysmat:=evalm(BerlekampPetrMatrix(apoly,var,apolydeg,p,
```

```
                                                cardfinitefield,irredpoly,
                                                alpha)-
                                  array(identity,1..apolydeg,1..apolydeg));

    syspoly:=evalm(tagvars&*sysmat);

    syspoly:=GB(convert(syspoly,set) union {irredpoly},
                    [op(tagvars),alpha],plex) mod p;

    syspoly:=convert(syspoly,set) minus {irredpoly};

# Cálculo de las variables independientes
    for icont in tagvars
     do
        aux:=nops(select(has,syspoly,icont));

        if aux<>1 then
            # Variable Independiente encontrada
            indepvars:=[op(indepvars),icont];
        fi;
    end do;

    dimKer:=apolydeg-nops(syspoly);
    if nops(indepvars) < dimKer then
        remvars:=convert(tagvars,set) minus convert(indepvars,set);
        for icont in remvars while nops(indepvars) < dimKer
         do
            aux:=select(has,syspoly,icont);
            if nops(indets(aux[1]) minus {alpha})-\
             nops(indets(aux[1]) intersect convert(indepvars,set)) >1
               then
               # Variable Independiente encontrada
                 indepvars:=[op(indepvars),icont];
            fi;
```

```
    end do;
fi;
hbasis:=array(1..apolydeg,1..nops(indepvars));
scont:=0;
for icont in indepvars
 do
    scont:=scont+1;
    equats:=map(x->if x=icont then x=1 else x=0 fi,indepvars);
    aux:=subs(equats,syspoly);
    tcont:=0;
    for jcont in tagvars
     do
        tcont:=tcont+1;
        if member(jcont,indepvars) then
            if jcont=icont then
                hbasis[tcont,scont]:=1;
            else
                hbasis[tcont,scont]:=0;
            fi;
        else
            hbasis[tcont,scont]:=CanonicalForm(
                                    op(
                                        2,
                                        solve({select(
                                                    has,
                                                    aux,
                                                    jcont)[1]=0},
                                                jcont)[1]),
                                    irredpoly,
                                    alpha,
```

```
                                                    p);
            fi;
        end do;
    end do;

    return(hbasis);
end proc;
```

```
GroebnerKernel:=proc(apoly,hbasispoly,var,p,irredpoly,alpha)
# Esta rutina calcula la factorización de un polinomio mónico y libre de
# cuadrados.
    local factorslist,GB_varz,Rk,Rkp1,GB_G,indset,icont::integer,
          Qroots,Pkp1,aux;

    factorslist:=[];
    GB_G:=GB({apoly,hbasispoly-GB_varz,irredpoly},
                [var,GB_varz,alpha],plex) mod p;

    GB_G:=convert(GB_G,set);
    GB_G:=GB_G minus {irredpoly};
    indset:=map(t->degree(t,var),GB_G);

# Resultante de apoly y hbasispoly-GB_varz
Rk:=select(t->if degree(t,var)=0 then true else false fi,
           GB_G)[1];
# Se borra zero en indset
indset:=indset[2..-1];

# GB no devuelve los polinomios ordenados por su grado en var
    for icont in indset
     do
        Pkp1:=select(t->if degree(t,var)=icont then true else false fi,
                     GB_G)[1];
```

```
        Rkp1:=coeff(Pkp1,var,icont);
        Pkp1:=Quo(Pkp1,Rkp1,var) mod p;
        aux:=Quo(Rk,Rkp1,GB_varz) mod p;
        aux:=subs(alpha=RootOf(irredpoly),aux);

        if degree(irredpoly,alpha)=1 then
            Qroots:=Roots(aux) mod p;
        else
            Qroots:=Roots(aux,RootOf(irredpoly)) mod p;
            Qroots:=subs(RootOf(irredpoly)=alpha,Qroots);
        fi;
        factorslist:=[op(map(t->CanonicalForm(subs(GB_varz=t[1]
                                                   ,Pkp1),
                                              irredpoly,alpha,p),
                             Qroots)),
                      op(factorslist)];
        Rk:=Rkp1;
    end do;

    return(factorslist);
end proc;
```

```
CanonicalForm:=proc(expr,irredpoly,alpha,p)
#Este procedimiento calcula la forma canónica de expr en un cuerpo
# finito.
    local aux;

    if degree(irredpoly,alpha)>1 then
        aux:=subs(alpha=RootOf(irredpoly),expr);
        aux:=Normal(aux) mod p;
        aux:=subs(RootOf(irredpoly)=alpha,aux);
```

```
        else
            aux:=Normal(expr) mod p;
        end if;
        return(aux);
    end proc;
```

Capítulo 3

Bases de Gröbner en $UL(\mathfrak{g})$

Las álgebras de Leibniz son una generalización no conmutativa y no antisimétrica de las álgebras de Lie. El concepto de álgebra envolvente universal para un álgebra de Leibniz, juega un papel fundamental en la teoría de representaciones de álgebras de Leibniz, de manera similar al papel jugado por el álgebra envolvente universal de un álgebra de Lie.

Bases de Gröbner para ideales por la izquierda en álgebras envolventes universales de un álgebra de Lie, fueron introducidas en [ApL88] y para ideales biláteros en [KaW90]. Esto nos conduce a plantear de manera natural, en este último capítulo, las bases de Gröbner para ideales biláteros en el álgebra envolvente universal de un álgebra de Leibniz.

3.1. Introducción

Con el transcurso de los años, se ha ido demostrando que el concepto de base de Gröbner no es exclusivo del anillo de polinomios $\mathbb{K}[x_1, ..., x_n]$ con coeficientes en un cuerpo $\mathbb{K}$, donde originalmente fueron introducidas; sino que es un concepto más universal, más amplio, más general, y exportable a otras estructuras matemáticas.

Bergman [Ber78] extendió el concepto de base de Gröbner a álgebras asociativas libres no conmutativas. En [KaW90, Kre92] se describen bases de Gröbner para álgebras de tipo soluble, que incluyen como caso particular las álgebras envolventes universales de un álgebra de Lie.

Se han definido bases de Gröbner en álgebras libres sobre cuerpos [Mor94]; en anillos de polinomios torcidos ("skew") y extensiones de Ore [Wei92, Pes94]; en grupos cuánticos [BGC98] que contienen como ejemplo las extensiones de Ore iteradas; en estructuras graduadas sobre un anillo [Ape00], estas clases de anillos incluyen las álgebras de polinomios, álgebras envolventes de álgebras de Lie, álgebras de Weyl, álgebras de Hecke y álgebras de Clifford; en ideales de $\mathbb{D}[x_1, ..., x_n]$ donde $\mathbb{D}$ es un DIP [Pan88]; en submódulos de $\mathbb{K}[x_1, ..., x_n]^m$ [Sch80] donde aparece por primera vez la computación de las sicigias por el método del algoritmo de la división (véase también [AdL94]); en ideales de $R[x_1, ..., x_n]$ siendo R un anillo conmutativo y noetheriano cualquiera (ver [AdL94]); en ideales en extensiones del Teorema de Poincaré-Birkhoff-Witt [GRZ02], y en muchas otras estructuras algebraicas.

Existen anillos donde se puede llevar a cabo el proceso de reducción pero no es posible introducir órdenes admisibles noetherianos sobre el conjunto de todos los monomios; para estos casos se han desarrollado otros conceptos para caracterizar las bases de Gröbner. Por ejemplo, en el supuesto de que el anillo contenga divisores de cero un orden admisible noetheriano no sería compatible con la multiplicación del anillo. Esta situación ha sido estudiada, en el caso de que existan divisores de cero en el anillo de coeficientes, por Kapur y Madlener, Green [Gre94], Li [Li02] y por Weispfenning para el caso especial de anillos regulares [Wei89].

En esta ocasión se va a definir lo que se entenderá por base de Gröbner para ideales biláteros del álgebra envolvente universal de un álgebra de Leibniz de dimensión finita. Estas álgebras tienen divisores de cero y por lo tanto no son álgebras de tipo soluble, lo que nos obliga a utilizar un método

diferente del usado en las álgebras envolventes universales de un álgebra de Lie.

3.2. Preliminares

3.2.1. Conceptos Básicos

La gran mayoría de definiciones y resultados que se exponen en esta sección, están sacados fundamentalmente del libro de [Gra00], verdadero hilo conductor de este tercer capítulo.

De aquí en adelante, se supondrá fijado un cuerpo $\mathbb{K}$ y un conjunto de variables $X = \{x_1, ..., x_n\}$, sobre los que se establecen las siguientes definiciones y resultados.

Se denotará por X^* al conjunto de todas las palabras $w = x_{i_1} \cdot ... \cdot x_{i_k}$ que se pueden formar con los elementos de X (esto incluye a la palabra vacía, que se denotará por 1).

X^* con la operación de concatenación de palabras es el monoide libre sobre X.

Definición 3.2.1 *Se llamará monomios a los elementos de X^*. Si $w = x_{i_1} \cdot ... \cdot x_{i_k} \in X^*$, se dice que w es un monomio de grado k, o brevemente, $grad(w) = k$.*

Definición 3.2.2 *Dados $u, v \in X^*$, se dirá que u es un factor de v si u es una subpalabra de v. Si además $v = u \cdot w_1$ (respectivamente $v = w_2 \cdot u$) entonces se dirá que u es un factor por la izquierda (respectivamente por la derecha) de v.*

Se denotará por $\mathbb{K}<x_1, ..., x_n>$ al $\mathbb{K}$-espacio vectorial generado por X^*.

Definición 3.2.3 *A los elementos de $\mathbb{K}<x_1, ..., x_n>$ se les denomina polinomios.*

Proposición 3.2.4 *$\mathbb{K}<x_1, ..., x_n>$ con la operación de concatenación de monomios tiene estructura de $\mathbb{K}$-álgebra asociativa unitaria y no conmutativa de polinomios. $\mathbb{K}<x_1, ..., x_n>$ es el álgebra asociativa no conmutativa libre.*

Una vez que se ha establecido la notación que se va a utilizar, se está en posición de introducir el concepto de orden monomial. Dicho concepto es básico a la hora de definir bases de Gröbner de ideales de $\mathbb{K}<x_1, ..., x_n>$.

Definición 3.2.5 *Dada "$\leq$" una relación de orden en X^*, se dirá que "$\leq$" es un orden monomial, si verifica:*

- *"$\leq$" es una relación de orden total.*
- *"$\leq$" es multiplicativa, esto es, si $u \leq v$ entonces $w \cdot u \leq w \cdot v$ y $u \cdot w \leq v \cdot w$ $\forall u, v, w \in X^*$.*
- *"$\leq$" verifica la condición de cadena descendente, esto es, si $w_1 \geq w_2 \geq ... \geq w_n \geq ...$ es una cadena descendente de palabras, entonces existe $k > 0$ tal que $w_k = w_{k+j}$ $\forall j \geq 0$.*

El concepto de orden monomial juega un papel muy importante dentro de la teoría de bases de Göbner, pues es la garantía de que el algoritmo de la división termina tras un número finito de pasos.

Ejemplo 3.2.6 *El orden lex no es un orden monomial en $\mathbb{K}<x, y>$ [Mor94].*
En $\mathbb{K}<x, y>$ se fija el orden lex con $x > y$.
Si fuese un orden monomial entonces sería multiplicativo:

$$y^2 > y \Longrightarrow y^2 x > yx$$

Sin embargo no es cierto que $y^2 x > yx$. Con lo cual, el orden lex no es un orden monomial en $\mathbb{K}<x, y>$; a diferencia de lo que ocurre en $\mathbb{K}[x_1, ..., x_n]$ que sí lo es.

Ejemplo 3.2.7 *El orden graduado lexicográfico por la izquierda (deglex) también llamado simplemente orden graduado lexicográfico, es un orden monomial en X^* [AdL94, Gra00].*

$$u = x_1^{\alpha_1}\cdot\ldots\cdot x_n^{\alpha_n} <_{\deg lex} v = x_1^{\alpha_1}\cdot\ldots\cdot x_n^{\alpha_n} \Longleftrightarrow \begin{cases} grad(u) < grad(v) \\ o \\ grad(u) = grad(v) \Rightarrow \exists i \in \{1,\ldots,n\} \\ \ tal\ que\ \alpha_j = \beta_j\ \ \forall j < i \\ \ y\ \alpha_i < \beta_i \end{cases}$$

Ejemplo 3.2.8 *El orden graduado lexicográfico por la derecha es un orden monomial en X^*.*

$$u = x_1^{\alpha_1}\cdot\ldots\cdot x_n^{\alpha_n} <_{\deg lex} v = x_1^{\beta_1}\cdot\ldots\cdot x_n^{\beta_n} \Longleftrightarrow \begin{cases} grad(u) < grad(v) \\ o \\ grad(u) = grad(v) \Rightarrow \exists i \in \{1,\ldots,n\} \\ \ tal\ que\ \alpha_j = \beta_j\ \ \forall j > i \\ \ y\ \alpha_i < \beta_i \end{cases}$$

Ejemplo 3.2.9 *El orden monomial lexicográfico con pesos a la izquierda:*

Sea $w : \{x_1, ..., x_n\} \to \mathbb{N}$ una aplicación y $W : X^* \to \mathbb{N}$ la extensión natural de w a X^*, esto es, si $u = x_{\alpha_1}x_{\alpha_2}...x_{\alpha_r}$, entonces $W(u) = \sum_{i=1,...,r} w(x_{\alpha_i})$.

$$u < v \Longleftrightarrow \begin{cases} W(u) < W(v) \\ o \\ W(u) = W(v) \Rightarrow \exists i \in \{1,\ldots,n\} \\ \ tal\ que\ \alpha_j = \beta_j\ \ \forall j < i \\ \ y\ \alpha_i < \beta_i \end{cases}$$

De modo análogo se puede definir el orden monomial lexicográfico con pesos a la derecha.

El orden monomial graduado lexicográfico por la izquierda, respectivamente derecha, es un caso particular del orden monomial lexicográfico con pesos a la izquierda, respectivamente derecha, tomando como w la aplicación constante igual a uno.

Ejemplo 3.2.10 *(Órdenes conmutativos) Sea $x_1 < x_2 < ... < x_n$ un orden arbitrario y sea $<_c$ un orden monomial cualquiera definido sobre el conjunto de las palabras conmutativas $\{x_1, ..., x_n\}$. Para cualquier monomio no conmutativo u en $\{x_1, ..., x_n\}$, sea $\overline{u}$ la palabra que se obtiene tratando u como una palabra conmutativa. Se dice que*

$$u < v \Longleftrightarrow \begin{cases} \overline{u} <_c \overline{v} \\ o \\ \overline{u} =_c \overline{v} \Rightarrow u <_{lex} v \end{cases}$$

Ejemplo 3.2.11 *[BuW98] Una variante de los órdenes conmutativos es el orden de Eliminación que se define tomando como $<_c$ el orden monomial lexicográfico y, en lugar del orden lexicográfico $<_{lex}$ el orden monomial graduado lexicográfico. Esto es*

$$u < v \Longleftrightarrow \begin{cases} \overline{u} <_{lex} \overline{v} \\ o \\ \overline{u} =_{lex} \overline{v} \Rightarrow u <_{deglex} v \end{cases}$$

Una vez que se fija un orden monomial, tiene sentido pleno hablar del mayor monomio de un polinomio $f \in \mathbb{K}<x_1, ..., x_n>$.

Definición 3.2.12 *Fijado un orden monomial y dado $f \in \mathbb{K}<x_1, ..., x_n>$, se denota por $lm(f)$ al mayor de los monomios que forman f.*

Si en lugar de un sólo polinomio, se tuviese un subconjunto de $\mathbb{K}<x_1, ..., x_n>$, entonces:

Definición 3.2.13 *Sea* $T \subseteq \mathbb{K}<x_1, ..., x_n>$.

$$lm(T) := \{lm(f)/f \in T\}$$

Al igual que ocurre en el caso conmutativo, se verifica:

Proposición 3.2.14 *[Gra00] Sea* I *un ideal de* $\mathbb{K}<x_1, ..., x_n>$. *Entonces:*

$$\mathbb{K}<x_1, ..., x_n>= C(I) \oplus I$$

siendo

$$C(I) =<\{u \in X^*/u \notin <lm(I)>\}>_{\mathbb{K}}$$

Todo elemento $f \in \mathbb{K}<x_1, ..., x_n>$ se puede expresar entonces de modo único, como suma de un elemento v de I y otro u de $C(I)$. Este elemento u recibe el nombre de forma normal de f módulo I. De forma análoga se puede tomar un subconjunto $G \subset I$ más pequeño, y definir el concepto de forma normal módulo G.

Definición 3.2.15 *Sea* $G \subset \mathbb{K}<x_1, ..., x_n>$ *e* $I =<G>$. *Un elemento* $f \in \mathbb{K}<x_1, ..., x_n>$ *se dice que está en forma normal módulo* G *si para todo* $g \in G$, $lm(g)$ *no es factor de ningún monomio de* f.

Definición 3.2.16 *Sea* $G \subset \mathbb{K}<x_1, ..., x_n>$, $I =<G>$ *y* $u, f \in \mathbb{K}<x_1, ..., x_n>$; u *se dice que es una forma normal módulo* G *de* f *si* u *está en forma normal módulo* G *y* $f - u \in I$.

Ciertamente, a la vista de su definición, el método de cálculo de una forma normal de un elemento $f \in \mathbb{K}<x_1, ..., x_n>$ es bien intuitiva y sencilla de obtener; simplemente bastará con ir reduciendo cada monomio de f con respecto a G. Cuando se obtenga un polinomio tal que ningún monomio sea

divisible por ningún monomio principal de los polinomios que forman G, ese polinomio será una forma normal de f módulo G. Este razonamiento queda recogido más detalladamente en el siguiente algoritmo de cálculo.

Algoritmo 3.2.17 *Forma Normal*
Entrada: $f \in \mathbb{K}<x_1, ..., x_n>, G \subseteq \mathbb{K}<x_1, ..., x_n>$
Salida: $\phi \in \mathbb{K}<x_1, ..., x_n>$ *tal que* ϕ *es una Forma Normal de* f *módulo* G.

1. *Sea* $\phi = 0$; $h = f$
2. *¿*$h = 0$*?*
   - *Si*
     - *Devuelve* ϕ
   - *No.*
     - *Sea* $u = lm(h)$ *y* λ *el coeficiente principal de* h.
3. *¿Existe* $g \in G$ *tal que* $lm(g)$ *sea un factor de* u*?*
   - *No.*
     - $h = h - \lambda u$
     - $\phi = \phi + \lambda u$
     - *Volver al paso 2.*
   - *Si*
     - *Sea* μ *el coeficiente de* $lm(g)$ *en* g *y sean* $v, w \in X^*$ *tales que* $v \cdot lm(g) \cdot w = u$
     - $h = h - \frac{\lambda}{\mu} \cdot v \cdot g \cdot w$
     - *Volver al paso 2.*

El algoritmo Forma Normal no es más que la traducción a $\mathbb{K}<x_1, ..., x_n>$ del algoritmo de la división.

Ejemplo 3.2.18 *Fijado el orden monomial deglex con $x < y$, y^2x e yx son formas normales módulo $G = \{g_1 = xy - x^2, g_2 = x^3 - yx, g_3 = y^3\}$ de $f = x^3y^2$.*

$f = y \cdot g_1 \cdot y + yx \cdot g_1 + g_2 \cdot y^2 + y \cdot g_2 + y^2x$

$f = x^2 \cdot g_1 \cdot y + x^3 \cdot g_1 + x \cdot g_1 \cdot x + g_1 \cdot x + x^2 \cdot g_2 + x \cdot g_2 + g_2 + yx$

A la vista de este ejemplo, se puede concluir que la forma normal módulo G de un polinomio f no es en general única, esto es, el resto de la división de un polinomio f entre $G \subseteq \mathbb{K}<x_1, ..., x_n>$ no es en general único. Para conseguir la unicidad del resto hay que exigir que G sea una base de Gröbner de I, al igual que ocurre en el caso conmutativo.

Definición 3.2.19 *Sea I un ideal de $\mathbb{K}<x_1, ..., x_n>$. Entonces $G \subseteq I$ se dice que es una base de Gröbner de I si para todo f en I existe g en G tal que $lm(g)$ es un factor de $lm(f)$.*

Proposición 3.2.20 *[Gra00] Si en el algoritmo Forma Normal, se pasa como entrada un conjunto G que sea base de Gröbner del ideal $I=<G>$, entonces la forma normal módulo G de f es única y es igual a la forma normal de f módulo I. De aquí en adelante se denotará la forma normal de f módulo I como $\overline{f}^G$ o $\overline{f}^I$.*

Definición 3.2.21 *Un conjunto $G \subseteq \mathbb{K}<x_1, ..., x_n>$ se denomina autorreducido (minimal) si verifica:*

- *$\forall g, h \in G / g \neq h \Rightarrow lm(g)$ no es un factor de $lm(h)$.*
- *G está formado por polinomios mónicos (es decir, coeficiente principal 1).*

Se puede ser un poco más estricto y definir:

Definición 3.2.22 *Un conjunto $G \subseteq \mathbb{K}<x_1, ..., x_n>$ se denomina reducido si verifica:*

- $\forall g, h \in G / g \neq h \Rightarrow lm(g)$ *no es un factor de ningún monomio de* h.
- *G está formado por polinomios mónicos (es decir, coeficiente principal 1).*

Es obvio que todo conjunto reducido es minimal.

Definición 3.2.23 *Sean $g_1, g_2 \in \mathbb{K}<x_1, ..., x_n>$ y $w_i = lm(g_i)$ i=1,2. Supongamos que los coeficientes de w_1, w_2 en g_1, g_2 sean 1. Supóngase además que w_1 no es un factor de w_2 y, recíprocamente, que w_2 no es un factor de w_1. Sean $u, v \in X^*$ tales que $w_1 \cdot u = v \cdot w_2$ y u es un factor propio ($\neq 1$) por la derecha de w_2 y v es un factor propio por la izquierda de w_1. Entonces el elemento $g_1 \cdot u - v \cdot g_2$ se denomina sicigia* [1] *de g_1 y g_2, brevemente, $S(g_1, g_2)$.*

Es en esta definición donde radica la mayor diferencia entre la teoría de bases de Gröbner en el caso conmutativo y en el no conmutativo; pues mientras en el caso conmutativo un par de polinomios sólo proporciona una sicigia, usando el mínimo común de los monomios principales, en el caso no conmutativo un par de polinomios pueden proporcionar más de una sicigia. Incluso se pueden dar en el caso no conmutativo auto-sicigias.

Proposición 3.2.24 *(Lema del diamante)[Ber78] Sea I un ideal del anillo $\mathbb{K}<x_1, ..., x_n>$ generado por un conjunto minimal G. Entonces G es base de Gröbner de I si, y sólo si, el resto de la división de f entre G es cero para todas las posibles sicigias f de los elementos de G.*

Esta proposición proporciona de modo inmediato un algoritmo de cálculo de una base de Gröbner de un ideal I generado por un conjunto finito de polinomios G. Los pasos a seguir son pues, en primer lugar, obtener un

[1] *Aunque no es usual, para mantener la coherencia con los capítulos anteriores se utiliza el nombre de sicigia. Otros autores utilizan el nombre de "overlap relation" o "composition".*

conjunto minimal H a partir de G; en segundo lugar, calcular la sicigia de dos elementos de H y, en caso de no ser cero el resto de la división de la sicigia entre H, añadir este resto a H. La aplicación reiterada de esta forma de proceder, conduce a un conjunto T tal que el resto de la división de la sicigia de dos elementos cualesquiera de T entre T, sea siempre cero.

En el contexto que nos ocupa, el álgebra asociativa no conmutativa libre, no hay ninguna condición que garantice la finitud del proceso de cálculo descrito anteriormente; de hecho podría no terminar en un número finito de pasos.

Ejemplo 3.2.25 *Sea I el ideal de* $\mathbb{K}<x,y>$ *generado por* $f_1 = xyx - yx$. *Basta con dar unos pocos pasos en el algoritmo de cálculo de bases de Gröbner, para darse cuenta de que* $\{f_n = xy^n x - y^n x\}_{n\in\mathbb{N}}$ *es una base de Gröbner de I con respecto al orden monomial graduado lexicográfico con* $x < y$.

Aunque, como se acaba de ver, no hay ninguna condición que en general garantice la finitud del proceso de cálculo de una base de Gröbner en el álgebra asociativa no conmutativa y libre, sí existen condiciones particulares bajo las cuales se puede asegurar la finitud de dicho proceso de obtención de una base de Gröbner.

Proposición 3.2.26 *[Gre94] Todo ideal* $I \subseteq \mathbb{K}<x_1, ..., x_n>$ *tal que el ideal* $<lm(I)>$ *esté finitamente generado posee una base de Gröbner finita.*

Si se centra la atención en el ideal monomial $< lm(I) >$, se tiene que existe un único conjunto de generadores minimal T de $< lm(I) >$. Entonces la única base reducida de Gröbner de I es $G = \{f - \overline{f}^I : f \in T\}$ [Gre94].

3.2.2. El álgebra envolvente universal $UL(\mathfrak{g})$

Un álgebra de Leibniz es una generalización natural del concepto de álgebra de Lie, dicho concepto fue introducido por Loday a lo largo de una serie de trabajos sobre periodicidad en K-teoría algebraica (ver [Lod93, Lod97]).

Definición 3.2.27 *[Lod93] Sea $\mathbb{K}$ un cuerpo y $\mathfrak{g}$ un $\mathbb{K}$-espacio vectorial. Se dice que $\mathfrak{g}$ es un álgebra de Leibniz sobre $\mathbb{K}$ si $\mathfrak{g}$ se puede dotar con una aplicación bilineal*

$$[\_,\_] : \mathfrak{g} \times \mathfrak{g} \longrightarrow \mathfrak{g}$$

verificando la identidad de Leibniz

$$[x,[y,z]] = [[x,y],z] - [[x,z],y]$$

El punto de partida de la teoría de las álgebras de Leibniz es, como se acaba de ver, la teoría de las álgebras de Lie. Surge entonces casi siempre que se presenta el desarrollo de una nueva teoría, que generaliza otra ya conocida, la idea de ver hasta qué punto los conceptos de las álgebras de Lie se pueden generalizar/exportar al nuevo contexto de las álgebras de Leibniz. Entre estos conceptos que se trata de migrar está el concepto de álgebra envolvente universal de un álgebra de Lie [Gra00, HiSt97].

Existen al menos dos definiciones equivalentes posibles de esta álgebra:

Sea $\mathfrak{g}$ un álgebra de Lie y $(-)_L : \mathbf{As} \longrightarrow \mathbf{Lie}$ el funtor entre la categoría de álgebras asociativas y la categoría de álgebras de Lie que a cada álgebra asociativa A le asocia el álgebra de Lie A_L con espacio subyacente A y con el corchete $[a,b] = a \cdot b - b \cdot a$, $\forall a, b \in A$.

(A) Un álgebra envolvente de $\mathfrak{g}$ consiste de un par (U,σ), donde U es un álgebra asociativa unitaria y $\sigma : \mathfrak{g} \to U_L$ es un morfismo de álgebras de Lie, que verifica la propiedad universal siguiente: para toda álgebra asociativa unitaria A y para todo morfismo de álgebras de Lie $f : \mathfrak{g} \to A_L$, existe un único morfismo de álgebras asociativas unitarias $\overline{f} : U \to A$ tal que $\overline{f} \circ \sigma = f$.

(B) Un álgebra envolvente de $\mathfrak{g}$ es un álgebra asociativa unitaria U tal que la categoría de $\mathfrak{g}$-módulos a la derecha es equivalente a la categoría de U-módulos a la derecha.

Tal álgebra U existe y es única salvo isomorfismos. Se denota por $U(\mathfrak{g})$ y se llama el álgebra envolvente universal del álgebra de Lie $\mathfrak{g}$.

Se define así un funtor U : **Lie** $\longrightarrow$ **As** entre la categoría de álgebras de Lie y la categoría de álgebras asociativas que es adjunto por la izquierda del funtor $(-)_L$ [HiSt97].

Surge entonces de modo natural, la cuestión de saber qué pasa si en lugar de la categoría de las álgebras de Lie se pone la categoría de las álgebras de Leibniz.

Toda álgebra de Lie es un álgebra de Leibniz. Así **Lie** es una subcategoría plena de la categoría de álgebras de Leibniz, **Leib**. Inversamente, a toda álgebra de Leibniz $\mathfrak{g}$ se le puede asociar un álgebra de Lie universal $\mathfrak{g}_{Lie} = \frac{\mathfrak{g}}{\mathfrak{g}^{ann}}$, donde $\mathfrak{g}^{ann}$ es el ideal bilátero generado por $\{[x,x]/x \in \mathfrak{g}\}$).

Proposición 3.2.28 *El funtor*

$$(-)_{Lie} : \boldsymbol{Leib} \longrightarrow \boldsymbol{Lie}$$

es adjunto por la izquierda del funtor de olvido

$$\boldsymbol{Lie} \longrightarrow \boldsymbol{Leib}$$

Demostración:
Trivial. □

En cuanto a la búsqueda de un funtor similar entre la categoría de álgebras de Leibniz y la categoría de las álgebras asociativas, verificando las condiciones equivalentes (A) y (B), decir que no existe.

La definición del concepto de álgebra envolvente universal de un álgebra de Leibniz puede verse de dos maneras diferentes. El objeto que jugaría un papel análogo al caso (B) sería el siguiente:

(B') A toda álgebra de Leibniz $\mathfrak{g}$ se le puede asociar un álgebra asociativa unitaria $UL(\mathfrak{g})$ de tal manera que la categoría de $\mathfrak{g}$-representaciones sea equivalente a la categoría de $UL(\mathfrak{g})$-módulos a la derecha.

Una $\mathfrak{g}$-representación consiste de un $\mathbb{K}$-espacio vectorial M y dos aplicaciones bilineales $M \otimes \mathfrak{g} \to M$, $(m,x) \rightsquigarrow m \cdot x$ y $\mathfrak{g} \otimes M \to M$, $(x,m) \rightsquigarrow x \cdot m$ verificando los siguientes axiomas:

$$m \cdot [x,y] = (m \cdot x) \cdot y - (m \cdot y) \cdot x$$
$$x \cdot (m \cdot y) = (x \cdot m) \cdot y - [x,y] \cdot m$$
$$x \cdot (y \cdot m) = [x,y] \cdot m - (x \cdot m) \cdot y$$

donde $m \in M$ y $x, y \in \mathfrak{g}$.

Hay una laboriosa y preciosa construcción hecha por Loday y Pirashvili [LoP93]. La justificación de dicha construcción [LoP98] está basada en los funtores U y $(-)_L$ definidos para las álgebras de Lie. El punto crucial de este desarrollo es la utilización de la categoría de aplicaciones $\mathbb{K}$-lineales $\mathcal{LM}$ equipada con un producto tensor. Partiendo de esta base se prueba que a partir de un *álgebra de Leibniz* cualquiera, es posible construir, y con un comportamiento funtorial, un álgebra de Lie dentro de la categoría $\mathcal{LM}$. A su vez, a partir de esta álgebra de Lie, haciendo actuar el funtor U, se pasa a un álgebra asociativa dentro de la categoría $\mathcal{LM}$. Una vez aquí la construcción de un funtor hasta la categoría de *álgebras asociativas* es un paso trivial, pues basta con hacer un coproducto. Como consecuencia de este desarrollo se define:

Definición 3.2.29 *[LoP93] Sea $\mathfrak{g}$ un álgebra de Leibniz, $\mathfrak{g}^l$ (cuyos elementos se denotarán por l_x con $x \in \mathfrak{g}$) y $\mathfrak{g}^r$ (cuyos elementos se denotarán por r_x con $x \in \mathfrak{g}$) dos copias de $\mathfrak{g}$. Entonces se define el álgebra envolvente universal de $\mathfrak{g}$, brevemente $UL(\mathfrak{g})$, como el cociente del álgebra tensorial $T(\mathfrak{g}^l \oplus \mathfrak{g}^r)$ por el ideal bilátero I generado por los elementos:*

i) $r_{[x,y]} - (r_x r_y - r_y r_x)$

ii) $l_{[x,y]} - (l_x r_y - r_y l_x)$

iii) $(r_y + l_y) l_x$

Proposición 3.2.30 *[LoP93] La categoría de representaciones del álgebra de Leibniz $\mathfrak{g}$ es equivalente a la categoría de módulos a la derecha sobre $UL(\mathfrak{g})$.*

Hay otra manera de ver las cosas.

(A') En [Lod97, Lod01], Loday introduce una versión asociativa de las álgebras de Leibniz, llamadas diálgebras, éstas son $\mathbb{K}$-espacios vectoriales equipados con dos operaciones binarias, $\vdash$ y $\dashv$, que verifican todas las variaciones de la ley asociativa; así álgebras asociativas son diálgebras para las que los dos productos, $\vdash$ y $\dashv$, coinciden. De manera similar al caso de álgebras de Lie, a toda álgebra de Leibniz $\mathfrak{g}$ se le puede asociar una diálgebra $Ud(\mathfrak{g})$, la diálgebra envolvente universal del álgebra de Leibniz $\mathfrak{g}$ [Lod97], que verifica:

Proposición 3.2.31 *[Lod97] El funtor*

$$Ud : \boldsymbol{Leib} \longrightarrow \boldsymbol{Dias}$$

es adjunto por la izquierda del funtor

$$(-)_L : \boldsymbol{Dias} \longrightarrow \boldsymbol{Leib}$$

donde D_L es el álgebra de Leibniz con espacio vectorial subyacente D y con el corchete $[a, b] = a \dashv b - b \vdash a$.

3.3. El Teorema de Poincaré-Birkhoff-Witt

Una vez que se ha establecido el concepto de álgebra envolvente universal de un álgebra de Leibniz, se hace necesario profundizar un poco más para entender mejor su estructura algebraica. Es en este objetivo precisamente donde encaja el teorema de Poincaré-Birkhoff-Witt, pues dicho teorema muestra la estructura del álgebra envolvente universal.

Aquí, haremos hincapié en la versión (B') del teorema de Poincaré-Birkhoff-Witt. Existe también la versión (A') de dicho teorema en la categoría de diálgebras [AyG03] que describe la estructura de la diálgebra envolvente universal: Para un álgebra de Leibniz existe un isomorfismo de diálgebras graduadas $S(\mathfrak{g}_{Lie}) \otimes \mathfrak{g} \approx grUd(\mathfrak{g})$, donde $S(\mathfrak{g}_{Lie})$ es el álgebra simétrica del módulo $\mathfrak{g}_{Lie}$.

En esta sección se presenta una prueba novedosa del teorema de Poincaré-Birkhoff-Witt, utilizando la teoría de bases de Gröbner en el álgebra asociativa no conmutativa y libre, $\mathbb{K}<x_1,...,x_n>$. Para ello es necesario enfocar el concepto de álgebra envolvente universal, desde otro punto de vista más provechoso para poder utilizar las bases de Gröbner como herramienta.

Dada $(\mathfrak{g} =< a_1,...,a_n >_{\mathbb{K}}, [-,-])$ un álgebra de Leibniz de dimensión finita sobre un cuerpo $\mathbb{K}$, el primer paso que hay que dar para la construcción de su álgebra envolvente universal, es la construcción del álgebra tensorial

$$T(\mathfrak{g}^l \oplus \mathfrak{g}^r) = \mathbb{K} \oplus (\mathfrak{g}^l \oplus \mathfrak{g}^r) \oplus (\mathfrak{g}^l \oplus \mathfrak{g}^r) \otimes (\mathfrak{g}^l \oplus \mathfrak{g}^r) \oplus ...$$

Si se profundiza un poco en su estructura, y se pregunta por sus tensores fundamentales, no es difícil darse cuenta de que, en la componente p-ésima se verifica, simplemente por linealidad:

$$\bigotimes_{i=1}^{p} (l_{\alpha_i} + r_{\beta_i}) = l_{\alpha_1} \otimes l_{\alpha_2} \otimes ... \otimes l_{\alpha_p} + l_{\alpha_1} \otimes r_{\beta_2} \otimes ... \otimes l_{\alpha_p} + ... + r_{\beta_1} \otimes r_{\beta_2} \otimes ... \otimes r_{\beta_p}$$

Lema 3.3.1 $TF = \bigcup_{p \in \mathbb{N}} \{\delta_{1,i_1} \otimes ... \otimes \delta_{p,i_p} \ / \ \delta_{k,i_j} = l_{a_{i_j}} \text{ o } \delta_{k,i_j} = r_{a_{i_j}}, i_1,...,i_p \in \{1,...,n\}, k \in \{1,...,p\}\} \cup \{1\}$ es el conjunto de los tensores fundamentales de $T(\mathfrak{g}^l \oplus \mathfrak{g}^r)$.

Todo elemento de $T(\mathfrak{g}^l \oplus \mathfrak{g}^r)$ es pues, combinación $\mathbb{K}$-lineal de los elementos de TF. Esto es, todo elemento del álgebra tensorial, se puede ver como un polinomio no conmutativo con coeficientes en $\mathbb{K}$ y en 2n variables como mucho. Por lo tanto, esencialmente, no hay ninguna diferencia entre $T(\mathfrak{g}^l \oplus \mathfrak{g}^r)$ y $\mathbb{K}<y_1,...,y_n,x_1,...,x_n>$; sólo cambia la forma en la que se escriben los elementos en una u otra estructura matemática.

Se puede establecer entonces la siguiente aplicación $\mathbb{K}$-lineal:

$$\Phi : T(\mathfrak{g}^l \oplus \mathfrak{g}^r) \longrightarrow \mathbb{K}<y_1,...,y_n,x_1,...,x_n>$$

tal que $\Phi(l_{a_i}) = y_i$ y $\Phi(r_{a_i}) = x_i$.

Proposición 3.3.2 Φ *es un isomorfismo de* $\mathbb{K}$*-álgebras.*

La demostración de esta proposición es trivial, pues basta darse cuenta cómo son los tensores fundamentales, y que éstos se corresponden biyectivamente con los monomios en $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$.

El isomorfismo Φ puede aportar mucha más información de la que en un principio puede parecer. En primer lugar, y teniendo en cuenta que el ideal I de las relaciones por las que se divide el álgebra tensorial $T(\mathfrak{g}^l \oplus \mathfrak{g}^r)$ está generado por:

$$r_{[x,y]} - (r_x r_y - r_y r_x)$$

$$l_{[x,y]} - (l_x r_y - r_y l_x)$$

$$(r_y + l_y) l_x$$

cabe preguntarse por $\Phi(I)$, pues el interés último es $UL(\mathfrak{g})$. $\Phi(I)$ está generado por:

$$\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i)$$

$$\Phi(l_{[a_i,a_j]}) - (y_i \cdot x_j - x_j \cdot y_i)$$

$$(x_i + y_i) \cdot y_j$$

Con lo cual:

$$\begin{array}{ccc} I & \longrightarrow & \Phi(I) \\ \downarrow & & \downarrow \\ T(\mathfrak{g}^l \oplus \mathfrak{g}^r) & \longrightarrow & \mathbb{K} < y_1, ..., y_n, x_1, ..., x_n > \\ \downarrow & & \downarrow \\ UL(\mathfrak{g}) & \longrightarrow & \frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)} \end{array}$$

Teorema 3.3.3 $UL(\mathfrak{g}) \approx \frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)}$ *(isomorfismo de $\mathbb{K}$-álgebras).*

El teorema de estructura anterior supone un salto cualitativo importante, pues con este resultado, se sientan las bases para poder utilizar la teoría

de bases de Gröbner sobre $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$; teoría perfectamente establecida y conocida [Mor94], para obtener resultados en el álgebra envolvente universal $UL(\mathfrak{g})$. Con este primer teorema de estructura en mente, se está en condiciones de demostrar el teorema de Poincaré-Birkhoff-Witt para el álgebra envolvente universal de un álgebra de Leibniz de dimensión finita.

Teorema 3.3.4 *Teorema de Poincaré-Birkhoff-Witt [LoP93] Una $\mathbb{K}$-base para el álgebra envolvente universal de un álgebra de Leibniz $\mathfrak{g}$ de dimensión finita, está formada por los monomios de la forma:*

$$\begin{gathered} x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m} \\ x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m} \cdot y_1 \\ ... \\ x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m} \cdot y_n \end{gathered}$$

siendo $n = dim_{\mathbb{K}}\mathfrak{g}$ y $m = dim_{\mathbb{K}}\mathfrak{g}_{Lie}$.

Demostración:
La demostración de este teorema se divide en tres partes. En la primera, se procede a identificar el ideal $\mathfrak{g}^{ann}$, pues dicho ideal juega un papel destacado en esta demostración. En segundo lugar, se obtiene un conjunto minimal G, a partir de los generadores de $\Phi(I)$; y, en tercer y último lugar, se verifica que G es una base de Gröbner para $\Phi(I)$. Una vez probado que G es una base de Gröbner para $\Phi(I)$, sólo queda por aplicar la Proposición 3.2.14 para deducir este resultado.

Hipótesis de partida:

- $(\mathfrak{g} =< a_1, ..., a_n >_{\mathbb{K}}, [-,-])$ un álgebra de Leibniz de dimensión finita.
- $n, m, p \in \mathbb{N}$ tales que $n = dim_{\mathbb{K}}\mathfrak{g}$, $m = dim_{\mathbb{K}}\mathfrak{g}_{Lie}$, $p = dim_{\mathbb{K}}\mathfrak{g}^{ann}$, $(n = m + p)$.
- $\{y_1, ..., y_n, x_1, ..., x_n\}$ un conjunto de variables

- Se fija el orden monomial graduado lexicográfico con: $y_n > ... > y_1 > x_n > ... > x_1$

Por el Teorema 3.3.3:

$$UL(\mathfrak{g}) \approx \frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)}$$

siendo $\Phi(I)$ el ideal generado por

$$\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i)$$

$$\Phi(l_{[a_i,a_j]}) - (y_i \cdot x_j - x_j \cdot y_i)$$

$$(x_i + y_i) \cdot y_j$$

Paso 1: Identificación del ideal $\mathfrak{g}^{ann}$

Sea J el ideal bilátero generado por

$$\{[a_i, a_i]\}_{i=1}^{n} \cup \{[a_i, a_j] + [a_j, a_i]\}_{i,j=1,...,n;i<j},$$

entonces $\mathfrak{g}^{ann} \subseteq J$; además, fijados $i, j \in \{1, .., n\}$ se verifica que $[a_i + a_j, a_i + a_j] = [a_i, a_i] + [a_i, a_j] + [a_j, a_i] + [a_j, a_j]$, y teniendo en cuenta que $[a_i + a_j, a_i + a_j], [a_i, a_i], [a_j, a_j] \in \mathfrak{g}^{ann}$ se puede concluir entonces que $[a_i, a_j] + [a_j, a_i] \in \mathfrak{g}^{ann}$; esto es, $\mathfrak{g}^{ann} = J$.

Puesto que $\mathfrak{g}^{ann}$ es un $\mathbb{K}$-subespacio vectorial de dimensión p, es posible obtener una $\mathbb{K}$-base de $\mathfrak{g}^{ann}$; sea ésta $\{g_1, ..., g_p\}$. Se puede suponer además (para simplificar notación) que tiene la forma:

$$g_1 = a_{m+1} + u_1^1 a_1 + ... + u_m^1 a_m$$

...

$$g_p = a_n + u_1^p a_1 + ... + u_m^p a_m$$

Para conseguir una $\mathbb{K}$-base con esta apariencia, basta con partir de una $\mathbb{K}$-base H de $\mathfrak{g}^{ann}$, tomar el elemento a_i de mayor índice en H, y un elemento cualquiera de H que lo contenga. A ese vector se le llama g_p; a continuación se

elimina, utilizando g_p, todas las expresiones $\alpha \cdot a_i$ de los restantes elementos de H. Se vuelve a coger, de $H - \{g_p\}$, el elemento a_j de mayor índice y un vector que lo contenga. A ese vector se le denota por g_{p-1}; a continuación se elimina $\beta \cdot a_j$ de todos los restantes (incluido g_p). Este proceso se continúa hasta conseguir la base anteriormente descrita (eliminación gaussiana).

El ideal $\mathfrak{g}^{ann}$ juega un papel relevante en esta demostración, pues es muy importante "su visión" dentro de $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$, esto es, los dos $\mathbb{K}$-subespacios vectoriales de $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$ que, de modo natural, son isomorfos a él; y que, a su vez, heredan todas las propiedades de $\mathfrak{g}^{ann}$. Éstos son $\Psi_l(\mathfrak{g}^{ann})$ y $\Psi_r(\mathfrak{g}^{ann})$, siendo Ψ_l y Ψ_r los monomorfismos siguientes:

$$\Psi_l : \mathfrak{g}^{ann} \rightarrowtail \mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$$
$$\Psi_r : \mathfrak{g}^{ann} \rightarrowtail \mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$$

con $\Psi_l(g_i(a_1, ..., a_n)) = g_i(y_1, ..., y_n)$ y $\Psi_r(g_i(a_1, ..., a_n)) = g_i(x_1, ..., x_n)$.

Paso 2: Obtención de un conjunto minimal.

Si se sustituye el conjunto

$$\{\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i)\}_{i,j\in\{1,...,n\}}$$

por

$$\{\Phi(r_{[a_i,a_i]})\}_{i=1}^{n} \cup \{\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i), \Phi(r_{[a_i,a_j]}) + \Phi(r_{[a_j,a_i]})\}_{i,j\in\{1,...,n\};i<j}$$

en el conjunto de los generadores de $\Phi(I)$, entonces este ideal no cambia. A su vez, por lo establecido en el punto anterior, se puede reemplazar, sin que el ideal $\Phi(I)$ cambie, el conjunto

$$\{\Phi(r_{[a_i,a_i]})\}_{i=1}^{n} \cup \{\Phi(r_{[a_i,a_j]}) + \Phi(r_{[a_j,a_i]})\}_{i,j\in\{1,...,n\};i<j}$$

por

$$\{g_1(x_1, ..., x_n), ..., g_p(x_1, ..., x_n)\}.$$

Se puede afirmar entonces que el conjunto

$$\{\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i)\}_{i,j\in\{1,...,n\}}$$

se puede sustituir por

$$C = \{\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i)\}_{i,j\in\{1,...,n\};i<j} \cup \{g_1(x_1,...,x_n),...,g_p(x_1,...,x_n)\}$$

y el ideal $\Phi(I)$ no cambia.

Sea

$$G = \{\Phi(r_{[a_i,a_j]}) - (x_i \cdot x_j - x_j \cdot x_i)\}_{i,j\in\{1,...,m\};i<j} \cup \{g_1(x_1,...,x_n),...,g_p(x_1,\ ...,x_n)\}.$$

Afirmación: G y C generan el mismo ideal ($G \subseteq C$).

Sea $i < j$ con $j \in \{m+1,...,n\}$, $t > 0$ tal que $j = m + t$, se verifica:[2]
$\Phi(r_{[a_i,a_j]}) - x_i x_j + x_j x_i \Rightarrow_{-g_t x_i} \Phi(r_{[a_i,a_j]}) - x_i x_j - u_1^t x_1 x_i - ... - u_m^t x_m x_i \Rightarrow_{x_i g_t}$
$\Phi(r_{[a_i,a_j]}) - u_1^t x_1 x_i - ... - u_m^t x_m x_i + u_1^t x_i x_1 + ... + u_m^t x_i x_m = \Phi(r_{[a_i,a_j]}) +$
$u_1^t(x_i x_1 - x_1 x_i) + ... + u_m^t(x_i x_m - x_m x_i)$

Caso 1: $i > m$, $w > 0$ tal que $i = m + w$ y $k \in \{1,...,m\}$
$x_i x_k - x_k x_i \Rightarrow_{-g_w x_k} -x_k x_i - u_1^w x_1 x_k - ... - u_m^w x_m x_k \Rightarrow_{x_k g_w} -u_1^w x_1 x_k - ... -$
$u_m^w x_m x_k + u_1^w x_k x_1 + ... + u_m^w x_k x_m = u_1^w(x_k x_1 - x_1 x_k) + ... + u_m^w(x_k x_m - x_m x_k) \Rightarrow_G$
$-u_1^w \Phi(r_{[a_1,a_k]}) - ... - u_m^w \Phi(r_{[a_m,a_k]}) \Rightarrow_G u_1^w \Phi(r_{[a_k,a_1]}) + ... + u_m^w \Phi(r_{[a_k,a_m]}) =$
$\Phi(r_{[a_k,g_i-a_i]}) = \Phi(r_{[a_k,g_i]}) - \Phi(r_{[a_k,a_i]}) \Rightarrow \Phi(r_{[a_k,a_i]})$.

pues $[a,x] = 0\ \forall a \in \mathfrak{g}\ \forall x \in \mathfrak{g}^{ann}$.

Por lo tanto la expresión inicial se reduciría a:

[2]Se adopta el convenio en esta demostración de indicar debajo de la flecha de reducción el polinomio/s con los que se va a reducir la expresión.

$$\Phi(r_{[a_i,a_j]}) + u_1^t\Phi(r_{[a_i,a_1]}) + ... + u_m^t\Phi(r_{[a_i,a_m]}) = \Phi(r_{[a_i,g_t]}) = 0$$

Caso 2: $i \leq m$

$\Phi(r_{[a_i,a_j]}) + u_1^t(x_ix_1 - x_1x_i) + ... + u_{i-1}^t(x_ix_{i-1} - x_{i-1}x_i) + u_i^t(x_ix_i - x_ix_i)_{=0} +$ $u_{i+1}^t(x_ix_{i+1} - x_{i+1}x_i) + ... + u_m^t(x_ix_m - x_mx_i) \Rightarrow_G \Phi(r_{[a_i,a_j]}) - u_1^t\Phi(r_{[a_1,a_i]}) -$ $... - u_{i-1}^t\Phi(r_{[a_{i-1},a_i]}) + u_{i+1}^t\Phi(r_{[a_i,a_{i+1}]}) + ... + u_m^t\Phi(r_{[a_i,a_m]}) \Rightarrow_G \Phi(r_{[a_i,a_j]}) +$ $u_1^t\Phi(r_{[a_i,a_1]}) + ... + u_{i-1}^t\Phi(r_{[a_i,a_{i-1}]}) + u_{i+1}^t\Phi(r_{[a_i,a_{i+1}]}) + ... + u_m^t\Phi(r_{[a_i,a_m]}) =$ $\Phi(r_{[a_i,g_t-u_i^ta_i]}) = \Phi(r_{[a_i,g_t]}) - u_i^t\Phi(r_{[a_i,a_i]}) \Rightarrow_G 0.$

Se redefine $G = \{\Phi(r_{[a_i,a_j]}) - (x_ix_j - x_jx_i)\}_{i,j\in\{1,...,m\};i<j} \cup \{g_1, ..., g_p\} \cup \{\Phi(l_{[a_i,a_j]}) -$ $y_ix_j + x_jy_i\}_{i\in\{1,...,n\};j\in\{1,...,m\}}$.

Afirmación: $\{\Phi(l_{[a_i,a_j]}) - y_ix_j + x_jy_i\}_{i\in\{1,...,n\};j\in\{1,...,n\}}$ reduce a cero por G.

Sea $i \in \{1, ..., n\}$, $j > m$, $t > 0$ tal que $j = m + t$.

$\Phi(l_{[a_i,a_j]}) - y_ix_j + x_jy_i \Rightarrow_{y_ig_t} \Phi(l_{[a_i,a_j]}) + x_jy_i + u_1^ty_ix_1 + ... + u_m^ty_ix_m \Rightarrow_{-g_ty_i}$ $\Phi(l_{[a_i,a_j]}) + u_1^ty_ix_1 + ... + u_m^ty_ix_m - u_1^tx_1y_i - ... - u_m^tx_my_i = \Phi(l_{[a_i,a_j]}) + u_1^t(y_ix_1 -$ $x_1y_i) + ... + u_m^t(y_ix_m - x_my_i) \Rightarrow_G \Phi(l_{[a_i,a_j]}) + u_1^t\Phi(l_{[a_i,a_1]}) + ... + u_m^t\Phi(l_{[a_i,a_m]}) =$ $\Phi(l_{[a_i,g_t]}) = 0.$

Por lo tanto, el conjunto:

$G = \{g_{i,j} = \Phi(r_{[a_i,a_j]}) - (x_ix_j - x_jx_i)\}_{i,j\in\{1,...,m\};i<j} \cup \{g_1(x_1, ..., x_n), ..., g_p(x_1, ..., x_n)\} \cup$ $\{h_{i,j} = -\Phi(l_{[a_i,a_j]}) + y_ix_j - x_jy_i\}_{i\in\{1,...,n\};j\in\{1,...,m\}} \cup \{t_{i,j} = x_iy_j + y_iy_j\}_{i,j\in\{1,...,n\}}$ genera el ideal $\Phi(I)$ y además es minimal.

Paso 3: G es una base de Gröbner del ideal $\Phi(I)$.

En esta tercera y última parte de la demostración de este teorema se comprueba que todas las posibles sicigias de los elementos de G, reducen a cero por G.

Sea $j > i > k$, se considera $g_{i,j}$, $g_{k,i}$.

$g_{i,j}x_k - x_j g_{k,i} = \Phi(r_{[a_i,a_j]})x_k - x_i x_j x_k - x_j\Phi(r_{[a_k,a_i]}) + x_j x_k x_i \Rightarrow_{-g_{k,j}x_i} \Phi(r_{[a_i,a_j]})x_k -$
$x_i x_j x_k - x_j\Phi(r_{[a_k,a_i]}) - \Phi(r_{[a_k,a_j]})x_i + x_k x_j x_i \Rightarrow_{x_i g_{k,j}} \Phi(r_{[a_i,a_j]})x_k - x_j\Phi(r_{[a_k,a_i]}) -$
$\Phi(r_{[a_k,a_j]})x_i + x_k x_j x_i + x_i\Phi(r_{[a_k,a_j]}) - x_i x_k x_j \Rightarrow_{g_{k,i}x_j} \Phi(r_{[a_i,a_j]})x_k - x_j\Phi(r_{[a_k,a_i]}) -$
$\Phi(r_{[a_k,a_j]})x_i + x_k x_j x_i + x_i\Phi(r_{[a_k,a_j]}) + \Phi(r_{[a_k,a_i]})x_j - x_k x_i x_j = \Phi(r_{[a_i,a_j]})x_k +$
$x_k(x_j x_i - x_i x_j) + \Phi(r_{[a_k,a_i]})x_j - x_j\Phi(r_{[a_k,a_i]}) - \Phi(r_{[a_k,a_j]})x_i + x_i\Phi(r_{[a_k,a_j]}) \Rightarrow_G$
$\Phi(r_{[a_i,a_j]})x_k - x_k\Phi(r_{[a_i,a_j]}) + \Phi(r_{[a_k,a_i]})x_j - x_j\Phi(r_{[a_k,a_i]}) - \Phi(r_{[a_k,a_j]})x_i + x_i\Phi(r_{[a_k,a_j]})$.

Sea $[a_i, a_j] = \alpha_1^{ij}a_1 + ... + \alpha_n^{ij}a_n$.

$\Phi(r_{[a_i,a_j]})x_k - x_k\Phi(r_{[a_i,a_j]}) = (\alpha_1^{ij}x_1 + ... + \alpha_n^{ij}x_n)x_k - x_k(\alpha_1^{ij}x_1 + ... + \alpha_n^{ij}x_n) =$
$\alpha_1^{ij}(x_1x_k - x_kx_1) + ... + \alpha_{k-1}^{ij}(x_{k-1}x_k - x_kx_{k-1}) + \alpha_k^{ij}\cdot 0 + \alpha_{k+1}^{ij}(x_{k+1}x_k -$
$x_kx_{k+1}) + ... + \alpha_n^{ij}(x_nx_k - x_kx_n) \Rightarrow_G \alpha_1^{ij}\Phi(r_{[a_1,a_k]}) + ... + \alpha_{k-1}^{ij}\Phi(r_{[a_{k-1},a_k]}) -$
$\alpha_{k+1}^{ij}\Phi(r_{[a_k,a_{k+1}]}) - ... - \alpha_n^{ij}\Phi(r_{[a_k,a_n]}) \Rightarrow_G \alpha_1^{ij}\Phi(r_{[a_1,a_k]}) + ... + \alpha_{k-1}^{ij}\Phi(r_{[a_{k-1},a_k]}) +$
$\alpha_{k+1}^{ij}\Phi(r_{[a_{k+1},a_k]}) - ... - \alpha_n^{ij}\Phi(r_{[a_n,a_k]}) = \Phi(r_{[[a_i,a_j],a_k]}) - \alpha_k^{ij}\Phi(r_{[a_k,a_k]})$

$\Rightarrow_G \Phi(r_{[[a_i,a_j],a_k]})$

o

$\Rightarrow_G -\Phi(r_{[a_k,[a_i,a_j]]})$

La expresión anterior se reduce entonces a:

$-\Phi(r_{[a_k,[a_i,a_j]]}) + \Phi(r_{[[a_k,a_i],a_j]}) - \Phi(r_{[[a_k,a_j],a_i]}) = 0$ (identidad de Leibniz).

Sea $j \in \{1, ..., m\}$, $i \in \{1, ..., n\}$ y $k < j$, se considera h_{ij}, g_{kj}.

$h_{ij}x_k - y_ig_{kj} = (-\Phi(l_{[a_i,a_j]}) + y_ix_j - x_jy_i)x_k - y_i(\Phi(r_{[a_k,a_j]}) - x_kx_j + x_jx_k) =$
$-\Phi(l_{[a_i,a_j]})x_k - x_jy_ix_k - y_i\Phi(r_{[a_k,a_j]}) + y_ix_kx_j \Rightarrow_{-h_{ik}x_j} -\Phi(l_{[a_i,a_j]})x_k - x_jy_ix_k -$
$y_i\Phi(r_{[a_k,a_j]}) + \Phi(l_{[a_i,a_k]})x_j + x_ky_ix_j \Rightarrow_{x_jh_{ik}} -\Phi(l_{[a_i,a_j]})x_k - y_i\Phi(r_{[a_k,a_j]}) + \Phi(l_{[a_i,a_k]})x_j +$
$x_ky_ix_j - x_j\Phi(l_{[a_i,a_k]}) - x_jx_ky_i \Rightarrow_{g_{kj}y_i} -\Phi(l_{[a_i,a_j]})x_k - y_i\Phi(r_{[a_k,a_j]}) + \Phi(l_{[a_i,a_k]})x_j +$
$x_ky_ix_j - x_j\Phi(l_{[a_i,a_k]}) + \Phi(r_{[a_k,a_j]})y_i - x_kx_jy_i = -\Phi(l_{[a_i,a_j]})x_k - y_i\Phi(r_{[a_k,a_j]}) +$

$\Phi(l_{[a_i,a_k]})x_j - x_j\Phi(l_{[a_i,a_k]}) + \Phi(r_{[a_k,a_j]})y_i + x_k(y_ix_j - x_jy_i) \Rightarrow_G -\Phi(l_{[a_i,a_j]})x_k - y_i\Phi(r_{[a_k,a_j]}) + \Phi(l_{[a_i,a_k]})x_j - x_j\Phi(l_{[a_i,a_k]}) + \Phi(r_{[a_k,a_j]})y_i + x_k\Phi(l_{[a_i,a_j]})$

Sea $[a_k, a_j] = \alpha_1^{kj}a_1 + ... + \alpha_n^{kj}a_n$.

$\Phi(r_{[a_k,a_j]})y_i - y_i\Phi(r_{[a_k,a_j]}) = (\alpha_1^{kj}x_1 + ... + \alpha_n^{kj}x_n)y_i - y_i(\alpha_1^{kj}x_1 + ... + \alpha_n^{kj}x_n) = \alpha_1^{kj}(x_1y_i - y_ix_1) + ... + \alpha_n^{kj}(x_ny_i - y_ix_n) \Rightarrow_G -\alpha_1^{kj}\Phi(l_{[a_i,a_1]}) - ... - \alpha_n^{kj}\Phi(l_{[a_i,a_n]}) = -\Phi(l_{[a_i,[a_k,a_j]]})$.

$\Phi(l_{[a_i,a_j]})x_k - x_k\Phi(l_{[a_i,a_j]}) = (\alpha_1^{ij}y_1 + ... + \alpha_n^{ij}y_n)x_k - x_k(\alpha_1^{ij}y_1 + ... + \alpha_n^{ij}y_n) = \alpha_1^{ij}(y_1x_k - x_ky_1) + ... + \alpha_n^{ij}(y_nx_k - x_ky_n) \Rightarrow_G \alpha_1^{ij}\Phi(l_{[a_1,a_k]}) + ... + \alpha_n^{ij}\Phi(l_{[a_n,a_k]}) = \Phi(l_{[[a_i,a_j],a_k]})$.

La expresión inicial se reduce entonces a:

$-\Phi(l_{[[a_i,a_j],a_k]}) - \Phi(l_{[a_i,[a_k,a_j]]}) + \Phi(l_{[[a_i,a_k],a_j]}) = 0$ (Identidad de Leibniz).

La tercera composición posible es entre t_{ij} y h_{jk} $i, j \in \{1, ..., n\}$, $k \in \{1, ..., m\}$.

$t_{ij}y_k - y_ih_{jk} = (x_iy_j + y_iy_j)x_k - y_i(-\Phi(l_{[a_j,a_k]}) + y_jx_k - x_ky_j) = x_iy_jx_k + y_i\Phi(l_{[a_j,a_k]}) + y_ix_ky_j \Rightarrow_{-h_{ik}y_j} x_iy_jx_k + y_i\Phi(l_{[a_j,a_k]}) + \Phi(l_{[a_i,a_k]})y_j + x_ky_iy_j \Rightarrow_{-x_kt_{ij}} x_iy_jx_k + y_i\Phi(l_{[a_j,a_k]}) + \Phi(l_{[a_i,a_k]})y_j - x_kx_iy_j \Rightarrow_{-x_ih_{jk}} y_i\Phi(l_{[a_j,a_k]}) + \Phi(l_{[a_i,a_k]})y_j - x_kx_iy_j + x_i\Phi(l_{[a_j,a_k]}) + x_ix_ky_j = y_i\Phi(l_{[a_j,a_k]}) + \Phi(l_{[a_i,a_k]})y_j + x_i\Phi(l_{[a_j,a_k]}) + (x_ix_k - x_kx_i)y_j \Rightarrow_G y_i\Phi(l_{[a_j,a_k]}) + \Phi(l_{[a_i,a_k]})y_j + x_i\Phi(l_{[a_j,a_k]}) + \Phi(r_{[a_i,a_k]})y_j = (y_i + x_i)\Phi(l_{[a_j,a_k]}) + (\Phi(l_{[a_i,a_k]}) + \Phi(r_{[a_i,a_k]}))y_j$.

$(y_i + x_i)\Phi(l_{[a_j,a_k]}) = (y_i + x_i)(\alpha_1^{jk}y_1 + ... + \alpha_n^{jk}y_n) = \alpha_1^{jk}(y_i + x_i)y_1 + ... + \alpha_n^{jk}(y_i + x_i)y_n \Rightarrow_G 0$

$(\Phi(l_{[a_i,a_k]}) + \Phi(r_{[a_i,a_k]}))y_j = (\alpha_1^{ik}y_1 + ... + \alpha_n^{ik}y_n + \alpha_1^{ik}x_1 + ... + \alpha_n^{ik}x_n)y_j = \alpha_1^{ik}(y_1 + x_1)y_j + ... + \alpha_n^{ik}(y_n + x_n)y_j \Rightarrow_G 0$

Teniendo en cuenta estas dos reducciones la expresión anterior reduce a cero por G.

La cuarta y última sicigia es la de t_{ij} con t_{jk}, $i,j,k \in \{1,...,n\}$.
$t_{ij}y_k - y_i t_{jk} = (y_i + x_i)y_j y_k - y_i(x_j + y_j)y_k = x_i y_j y_k - y_i x_j y_k \Rightarrow_{h_{ij}y_k} x_i y_j y_k - \Phi(l_{[a_i,a_j]})y_k - x_j y_i y_k \Rightarrow_{-x_i t_{jk}} -\Phi(l_{[a_i,a_j]})y_k - x_j y_i y_k - x_i x_j y_k \Rightarrow_{x_j t_{ik}} -\Phi(l_{[a_i,a_j]})y_k - x_i x_j y_k + x_j x_i y_k = -\Phi(l_{[a_i,a_j]})y_k + (x_j x_i - x_i x_j)y_k \Rightarrow_G -\Phi(l_{[a_i,a_j]})y_k - \Phi(r_{[a_i,a_j]})y_k = -(\Phi(l_{[a_i,a_j]}) + \Phi(r_{[a_i,a_j]}))y_k \Rightarrow_G 0$.

Una vez que se ha probado que toda sicigia de dos elementos de G, reduce a cero por G, se está en posición de afirmar que dicho conjunto es una base de Gröbner para el ideal $\Phi(I)$; con lo cual, por la Proposición 3.2.14
$< lm(\Phi(I)) >=< \{x_j x_i\}_{i,j\in\{1,...,m\};i<j} \cup \{x_{m+1},...,x_n\} \cup \{y_i x_j\}_{i\in\{1,...,n\},j\in\{1,...,m\}} \cup \{y_i y_j\}_{i,j\in\{1,...,n\}} >$; y por lo tanto:

$$x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m}$$
$$x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m} \cdot y_1$$
$$...$$
$$x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m} \cdot y_n$$

es una $\mathbb{K}$-base del álgebra envolvente universal de $\mathfrak{g}$. □

Ejemplo 3.3.5 *Dada ($\mathfrak{g} =< e, f >_{\mathbb{K}}, [-,-]$) un álgebra de Leibniz tal que:*

$$[e,e] = f,\ [e,f] = [f,e] = [f,f] = 0$$

Se calculará una $\mathbb{K}$-base de su álgebra envolvente universal.

Teniendo en cuenta que:

$$\mathfrak{g}_{Lie} = \frac{\mathfrak{g}}{<[x,x]>} = \frac{\mathfrak{g}}{<f>} \cong < e > \ (m = 1)$$

y que el álgebra envolvente universal de $\mathfrak{g}$ es:

$$UL(\mathfrak{g}) \approx \frac{\mathbb{K}<y_1,y_2,x_1,x_2>}{\Phi(I)}$$

siendo $\Phi(I)$ *el ideal generado por las relaciones:*

$$\begin{aligned}
\Phi(r_{[e,e]}) &= 0 \\
\Phi(r_{[e,f]}) &= x_1x_2 - x_2x_1 \\
\Phi(l_{[e,e]}) &= y_1x_1 - x_1y_1 \\
\Phi(l_{[e,f]}) &= y_1x_2 - x_2y_1 \\
\Phi(l_{[f,e]}) &= y_2x_1 - x_1y_2 \\
\Phi(l_{[f,f]}) &= y_2x_2 - x_2y_2 \\
(x_1 + y_1)y_1 &= 0 \\
(x_1 + y_1)y_2 &= 0 \\
(x_2 + y_2)y_1 &= 0 \\
(x_2 + y_2)y_2 &= 0
\end{aligned}$$

entonces, aplicando a los generadores de $\Phi(I)$ *el mismo razonamiento que se siguió en la demostración del teorema de Poincaré-Birkhoff-Witt, se puede afirmar que el conjunto de polinomios*

$$\{x_2, -y_2+y_1x_1-x_1y_1, y_2x_1-x_1y_2, (x_1+y_1)y_2, (x_2+y_2)y_1, (x_2+y_2)y_2, (x_1+y_1)y_1\}$$

forma una base de Gröbner del ideal $\Phi(I)$*, con respecto al orden monomial graduado lexicográfico con* $y_2 > y_1 > x_2 > x_1$*. Con lo cual, una* $\mathbb{K}$*-base de* $UL(\mathfrak{g})$ *será entonces*

$$\{x_1^{\alpha_0}, x_1^{\alpha_1}y_1, x_1^{\alpha_2}y_2\}_{\{\alpha_0,\alpha_1,\alpha_2 \in \mathbb{N}\cup\{0\}\}}$$

El teorema de Poincaré-Birkhoff-Witt que se acaba de probar, se puede resumir en el siguiente teorema de estructura del álgebra envolvente universal de un álgebra de Leibniz de dimensión finita.

Teorema 3.3.6 *(Teorema de Estructura) Sea* $(\mathfrak{g}, [-,-])$ *un álgebra de Leibniz de dimensión finita,* $n = dim_{\mathbb{K}}\mathfrak{g}$ *y* $m = dim_{\mathbb{K}}\mathfrak{g}_{Lie}$*. Entonces, la aplicación* $\mathbb{K}$*-lineal:*

$$\Psi : \mathbb{K}[x_1, ..., x_m]^{n+1} \longrightarrow UL(\mathfrak{g})$$

siendo $\Psi(f_0, ..., f_n) = f_0 + f_1y_1 + ... + f_ny_n$*, es un isomorfismo de* $\mathbb{K}$*-espacios vectoriales.*

Se podría afinar incluso un poquito más y, a la vista de la estructura del álgebra $UL(\mathfrak{g})$ es posible concluir que:

Teorema 3.3.7 *(Teorema de Estructura II) Sea* $(\mathfrak{g}, [-,-])$ *un álgebra de Leibniz de dimensión finita,* $n = dim_{\mathbb{K}}\mathfrak{g}$ *y* $m = dim_{\mathbb{K}}\mathfrak{g}_{Lie}$*. Entonces, la aplicación:*

$$\Phi : U(\mathfrak{g}_{Lie})^{n+1} \longrightarrow UL(\mathfrak{g})$$

siendo $\Phi(f_0, ..., f_n) = f_0 + f_1y_1 + ... + f_ny_n$*, es un isomorfismo de* $U(\mathfrak{g}_{Lie})$*-módulos.*

Demostración:
Partiendo del hecho de que existe un isomorfismo de $\mathbb{K}$-espacios vectoriales entre $U(\mathfrak{g}_{Lie})$ y $\mathbb{K}[x_1, ..., x_m]$ [Gra00] la demostración es trivial, pues basta con darse cuenta de que ambas estructuras matemáticas "funcionan" de igual modo vistos como $U(\mathfrak{g}_{Lie})$-módulos, la única diferencia es la forma en la que se escriben sus elementos, en el primer caso son n+1-uplas de polinomios con coeficientes en $\mathbb{K}[x_1, ..., x_m]$, mientras que en el segundo son polinomios de grado uno o cero en las variables $y_1, ..., y_n$ con coeficientes en $\mathbb{K}[x_1, ..., x_m]$.
□

Aunque no esencial, una condición interesante a estudiar cuando se desean construir bases de Gröbner en una estructura matemática, es la noetherianidad. Puesto que la teoría de las álgebras de Leibniz es, en cierta forma una

generalización de la teoría de las álgebras de Lie, cabe preguntarse en primer lugar si dada un álgebra de Lie $\mathfrak{g}$ su álgebra envolvente universal $U(\mathfrak{g})$, es o no es un anillo noetheriano.

Teorema 3.3.8 *[Dix96]* $U(\mathfrak{g})$ *es un anillo noetheriano para cualquier álgebra de Lie* $\mathfrak{g}$ *de dimensión finita.*

Este resultado, más el Teorema 3.3.7 ponen de manifiesto la condición de anillo noetheriano de $UL(\mathfrak{g})$. Así pues:

Teorema 3.3.9 $UL(\mathfrak{g})$ *es un anillo noetheriano para cualquier álgebra de Leibniz* $\mathfrak{g}$ *de dimensión finita.*

Demostración:
Sea $\tau : \mathfrak{g} \to \mathfrak{g}_{Lie}$ el epimorfismo canónico. Entonces $SL(\tau)$ es isomorfo como $\mathbb{K}$-espacio vectorial a $S(\mathfrak{g}_{Lie}) \oplus S(\mathfrak{g}_{Lie}) \otimes \mathfrak{g}$ [LoP93].

Se verifica que $\mathrm{gr}\, UL(\mathfrak{g}) \cong \mathbb{K}[x_1, ..., x_m]^{n+1}$ (Teorema de estructura 3.3.6). El álgebra $SL(\tau)$ contiene a $S(\mathfrak{g}_{Lie})$ como una subálgebra central, además se tiene un isomorfismo $SL(\tau) \cong S(\mathfrak{g}_{Lie}) \oplus S(\mathfrak{g}_{Lie}) \otimes \mathfrak{g}$ de $S(\mathfrak{g}_{Lie})$-módulos y así $SL(\tau)$ es finitamente generado como $S(\mathfrak{g}_{Lie})$-módulo y por lo tanto $SL(\tau)$ es noetheriano. $UL(\mathfrak{g})$ tiene una filtración tal que el objeto graduado asociado es de la forma $SL(\tau)$ y por lo tanto noetheriano y usando resultados de álgebras filtradas se tiene que $UL(\mathfrak{g})$ es también noetheriano.

□

Una vez cumplido el objetivo principal y último de esta sección, que no es otro que la prueba del teorema de Poincaré-Birkhoff-Witt, no sería adecuado terminar sin destacar unas observaciones acerca de la demostración que se acaba de exponer. Observaciones que tienen que ver con el papel destacado que juega el ideal $\mathfrak{g}^{ann}$ y que, en un principio pudiera pasar desapercibido.

- "Unicidad" de la $\mathbb{K}$-base de $\mathfrak{g}^{ann}$.- Una vez fijada una $\mathbb{K}$-base en $\mathfrak{g}$, la $\mathbb{K}$-base del ideal $\mathfrak{g}^{ann}$ que se construye en la demostración del teorema de Poincaré-Birkhoff-Witt existe y es única.

- Papel de la $\mathbb{K}$-base de $\mathfrak{g}^{ann}$ en $\Phi(I)$.- Aunque en principio, y como en la demostración del teorema de Poincaré-Birkhoff-Witt ha quedado patente, son los generadores de $\mathfrak{g}^{ann}$ (o más correctamente, sus imágenes por Ψ_r) los que aparecen explícitamente en el ideal $\Phi(I)$. Se pudiera pensar entonces, que sólo la imagen por la derecha ($=\Psi_r$) de la $\mathbb{K}$-base de $\mathfrak{g}^{ann}$ juega un papel destacado; nada más lejos de la realidad, lo cierto es que su imagen por la izquierda ($=\Psi_l$) también está presente.

  $\{y_iy_j+x_iy_j\}_{i,j\in\{1,...,n\}}=\{y_iy_j+x_iy_j\}_{\{i\leq m,j\leq n\}}\cup\{y_iy_j+x_iy_j\}_{\{i\geq m+1,j\leq n\}}$

  Sean $i\in\{m+1,...,n\}$, $j\in\{1,...,n\}$.

  $y_iy_j+x_iy_j\Rightarrow_{h_{ji}}y_iy_j-\Phi(l_{[a_j,a_i]})+y_jx_i\Rightarrow_{-y_jg_{i-m}}y_iy_j-\Phi(l_{[a_j,a_i]})-u_1^{i-m}y_jx_1-...-u_m^{i-m}y_jx_m\Rightarrow y_iy_j-\Phi(l_{[a_j,a_i]})-u_1^{i-m}y_jx_1-...-u_m^{i-m}y_jx_m+u_1^{i-m}(-\Phi(l_{[a_j,a_1]})+y_jx_1-x_1y_j)+...+u_m^{i-m}(-\Phi(l_{[a_j,a_m]})+y_jx_m-x_my_j)=y_iy_j-\Phi(l_{[a_j,g_{i-m}(a_1,...,a_n)]})-u_1^{i-m}x_1y_j-...-u_m^{i-m}x_my_j\Rightarrow y_iy_j+u_1^{i-m}y_1y_j+...+u_m^{i-m}y_my_j=\Psi_l(g_{i-m}(a_1,...,a_n))y_j=g_{i-m}(y_1,...,y_n)y_j$

  Se pueden cambiar entonces los generadores $\{y_iy_j+x_iy_j\}_{\{i\geq m+1,j\leq n\}}$ por $\{g_{i-m}(y_1,...,y_n)y_j\}_{i,j\in\{1,...,n\};\,i>m}$ y el conjunto resultante seguiría generando $\Phi(I)$.

- Sobre la elección de la $\mathbb{K}$-base de $\mathfrak{g}^{ann}$.- Otro punto muy importante que tiene que ver con $\mathfrak{g}^{ann}$, es la elección misma de la $\mathbb{K}$-base $\{g_1(a_1,...,a_n),...,g_p(a_1,...,a_n)\}$. Ciertamente, cualquier base que sea de la forma descrita en la demostración del teorema de Poincaré-Birkhoff-Witt es válida. No obstante, hay un punto que se escapa en principio a estudio, ¿qué ocurre si existiese $g_i(a_1,...,a_n)=a_{m+i}$? De existir $g_i(a_1,...,a_n)=a_{m+i}$, entonces x_{m+i} estaría entre los generadores de $\Phi(I)$, esto origina que se elimine de la $\mathbb{K}$-base todo monomio que dependa de x_{m+i} y, por otra parte, en el cociente $y_{m+i}y_j=0\ \forall j\in\{1,...,n\}$, esto es, el producto de dos monomios puede ser cero. Para evitar esta situación basta con escoger como $\mathbb{K}$-base de $\mathfrak{g}$ a $\{b_1=a_1,...,b_{m+i-1}=$

$a_{m+i-1}, b_{m+i} = a_{m+i} + a_1, b_{m+i+1} = a_{m+i+1}, ..., b_n = a_n\}$, esto es, basta con sumar a_i con $i \in \{1, ..., m\}$. Con respecto a esta nueva base $\{b_1, ..., b_n\}$, $g_i(b_1, ..., b_n) = b_{m+i} - b_1$. Procediendo de igual modo con todos los elementos g_i tales que tienen sólo una coordenada no nula, al final se conseguirá una $\mathbb{K}$-base B de $\mathfrak{g}$ tal que se puede encontrar una $\mathbb{K}$-base para $\mathfrak{g}^{ann}$ en las condiciones descritas en la demostración del teorema de Poincaré-Birkhoff-Witt y que además, todos los vectores que la forman tienen al menos dos coordenadas no nulas.

Ejemplo 3.3.10 *Sea ($\mathfrak{g} =< e, f >_{\mathbb{K}}, [-,-]$) el álgebra de Leibniz del ejemplo 3.3.5.*

Tal cual está construido $UL(\mathfrak{g})$ se verifica que $\overline{y_2} \cdot \overline{y_2} = \overline{0}$. Si se quisiese evitar este comportamiento en la multiplicación de dos monomios, bastaría con cambiar la $\mathbb{K}$-base $\{e, f\}$, por $\{a_1 = e, a_2 = e + f\}$. Con respecto a esta nueva base, los generadores del ideal por el que hay que dividir para obtener $UL(\mathfrak{g})$ son $\{x_2 - x_1, -y_2 + y_1 + y_1x_1 - x_1y_1, y_2x_1 - x_1y_2 - y_2 + y_1, (x_1 + y_1)y_1, (x_1 + y_1)y_2, (x_2 + y_2)y_1, (x_2 + y_2)y_2\}$.

El proceso anterior, de cambio de la $\mathbb{K}$-base de $\mathfrak{g}$, se puede hacer siempre que $m \geq 1$, pero, ¿qué pasa cuando $m = 0$? Veamos que esto no es posible como consecuencia del siguiente resultado.

Lema 3.3.11 *Si ($\mathfrak{g}, [-,-]$) un álgebra de Leibniz no nula, entonces $\mathfrak{g}_{Lie} \neq 0$*

Demostración:
Para demostrar este lema es importante darse cuenta de que para cualquier álgebra de Leibniz $\mathfrak{g} =< a_1, ..., a_n >_{\mathbb{K}}$ se verifica:

$$[x, [a_i, a_i]] = 0 \;\; y \;\; [x, [a_i, a_j] + [a_j, a_i]] = 0$$

$\forall\, x \in \mathfrak{g}$ (hecho de comprobación rutinaria aplicando la identidad de Leibniz).

Si $\mathfrak{g}_{Lie} = 0$ entonces $\mathfrak{g} = \mathfrak{g}^{ann} =< [a_i, a_i], [a_i, a_j] + [a_j, a_i] >$. Dado $g \in \mathfrak{g} = \mathfrak{g}^{ann}$, $g = \sum_i [[a_i, a_i], x_i] + \sum_{s,t} [[a_s, a_t] + [a_t, a_s], y_{st}]$, a su vez, por ser x_i e y_{st}

elementos de $\mathfrak{g}^{ann}$ se pueden expresar de modo similar a g en función de los generadores de $\mathfrak{g}^{ann}$. Aplicando la linealidad del corchete y las identidades anteriores se llega a la conclusión de que $g = 0$ y como consecuencia $\mathfrak{g} = 0$.

□

3.4. Bases de Gröbner en $UL(\mathfrak{g})$

Llegados a este punto, se hace necesario recordar el problema inicial que en 1965 Buchberger se planteó, que es el cálculo de una $\mathbb{K}$-base de $\frac{\mathbb{K}[x_1,...,x_n]}{I}$ donde I es un ideal; problema de cuya resolución [Buc65] nacerían las bases de Gröbner. En nuestro caso, nos planteamos la obtención de una $\mathbb{K}$-base para el $\mathbb{K}$-espacio vectorial $\frac{UL(\mathfrak{g})}{\overline{J}}$, siendo $\overline{J}$ un ideal bilátero de $UL(\mathfrak{g})$.

Se tiene definida la siguiente aplicación:

$$\pi : \mathbb{K}<y_1, ..., y_n, x_1, ..., x_n> \longrightarrow UL(\mathfrak{g}) \cong \frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)}$$
$$\Phi(I) \subseteq J \rightsquigarrow \overline{J} = \frac{J}{\Phi(I)}$$

Es conocido que existe una aplicación biyectiva entre ideales de $UL(\mathfrak{g})$ e ideales de $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$ que contienen a $\Phi(I)$. De modo inmediato viene a la mente el tercer teorema de isomorfía [DuF99], gracias al cual se puede afirmar que:

$$\frac{UL(\mathfrak{g})}{\overline{J}} = \frac{\frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)}}{\frac{J}{\Phi(I)}} \cong \frac{\mathbb{K} < y_1, ..., y_n, x_1, ..., x_n >}{J}$$

Esto es, el tercer teorema de isomorfía va a permitir "dar un paso atrás", y volver a $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$, donde hay perfectamente definida una teoría de bases de Gröbner [Mor94, Gra00].

El camino más lógico a seguir, a la vista de lo que se acaba de exponer es pues, el de definir el concepto de base de Gröbner en $UL(\mathfrak{g})$ a partir del concepto de base de Gröbner en $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$.

Cuando Bruno Buchberger definió el concepto de base de Gröbner, no sólo estableció dicha idea, sino que también, paralelamente en cierta forma, fijó los pasos a seguir para definir el concepto de base de Gröbner en otras estructuras matemáticas, pues la gran mayoría de las aproximaciones a este concepto en distintos entornos se hacen imitando, en la medida de lo posible, la construcción hecha por Buchberger para $\mathbb{K}[x_1, ..., x_n]$ [Buc65].

Estos pasos que fijó Buchberger son, esencialmente:

- Definición de un orden monomial

- Algoritmo de la División

No obstante, no es éste el camino que aquí se seguirá; gracias al tercer teorema de isomorfía, y con el fin de evitar trabajar en $UL(\mathfrak{g})$ directamente, que es intrínsecamente más complicado pues, por ejemplo, hay divisores de cero, que complicarían la definición del concepto de orden monomial; se opta por trabajar en el álgebra asociativa no conmutativa libre $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$ y pasar por π a $UL(\mathfrak{g})$.

Este modo de proceder tiene la notable ventaja de aprovechar la teoría de bases de Gröbner existente en $\mathbb{K}<y_1, ..., y_n, x_1, ..., x_n>$, teoría por otra parte perfectamente establecida y conocida, para obtener definiciones y resultados en $UL(\mathfrak{g})$.

Hablando en términos generales, la idea de definir el concepto de base de Gröbner en el álgebra $\frac{\mathbb{K}<x_1,...,x_r>}{I}$ a partir del concepto de base de Gröbner en $\mathbb{K}<x_1, ..., x_r>$ no es nueva, aunque sí reciente. Una primera aproximación a este planteamiento fue llevada a cabo en [Mor94], en este artículo Mora muestra, de un modo breve, las directrices que hay que seguir para establecer el concepto de base de Gröbner en $\frac{\mathbb{K}<x_1,...,x_r>}{I}$ a partir del concepto de base de Gröbner en $\mathbb{K}<x_1, ..., x_r>$. En [Nor01] Nordbeck recoge el testigo que había dejado Mora e introduce el concepto de FG-base ("Factor Gröbner basis") en $\frac{\mathbb{K}<x_1,...,x_r>}{I}$. Las FG-bases son las bases de Gröbner definidas de un modo

general en el álgebra $\frac{\mathbb{K}<x_1,...,x_r>}{I}$ y éstas aparecen como el camino a seguir para definir el concepto de base de Gröbner en $UL(\mathfrak{g})$.

Por motivos que quedarán claros a posteriori y que tienen su razón de ser en la búsqueda de una base de Gröbner con un número finito de elementos, se fija un orden monomial graduado L.

Definición 3.4.1 *[Nor01] Sea $\overline{J}$ un ideal de $\frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)}$, esto es J es un ideal de $\mathbb{K}<y_1,...,y_n,x_1,...,x_n>$ verificando que $\Phi(I) \subset J$. Un conjunto de elementos $F \subset J$ reducidos módulo $\Phi(I)$ se dice que es una FG-base para $\overline{J}$ si, para cada $f \in J$ tal que $\overline{f} \neq 0$ ($\Leftrightarrow f \notin \Phi(I)$) existe $g \in F$ tal que $lm(g) \mid lm(\overline{f})$.*

Bajo cierto punto de vista y como a continuación se comprobará, la definición de FG-base no es más que la extensión de una base de Gröbner de $\Phi(I)$.

Proposición 3.4.2 *[Nor01] Sea $F = \{f_1,...,f_s\}$ un conjunto de elementos reducidos módulo $\Phi(I)$ y G una base de Gröbner del ideal $\Phi(I)$. Entonces F es una FG-base para $\overline{J}$ en $\frac{\mathbb{K}<y_1,...,y_n,x_1,...,x_n>}{\Phi(I)}$ si y sólo si $F \cup G$ es una base de Gröbner del ideal J en $\mathbb{K}<y_1,...,y_n,x_1,...,x_n>$.*

Teniendo en cuenta el resultado anterior, un procedimiento válido para calcular una FG-base de un ideal $\overline{J} = <\overline{f_1},...,\overline{f_s}>$ consiste en calcular una base de Gröbner del ideal de $\mathbb{K}<y_1,...,y_n,x_1,...,x_n>$ generado por $\{f_1,...,f_s\} \cup G$, siendo G una base de Gröbner del ideal $\Phi(I)$.

A continuación se exponen algunos ejemplos que ilustran la definición que se acaba de establecer, para ello se utilizan álgebras de Leibniz de dimensiones 2 y 3.

Si $\mathfrak{g}$ es un álgebra de Leibniz de dimensión 2 con generadores e y f, existen, salvo isomorfismo, tres tipos de álgebras de Leibniz según sea la estructura de $\mathfrak{g}^{ann}$.

Caso I.- Si $dim\mathfrak{g}^{ann} = 0$, entonces $\mathfrak{g}$ es un álgebra de Lie. En las líneas que siguen se muestran un par de ejemplos que ilustran el cálculo de las FG-bases de Gröbner en este supuesto.

Ejemplo 3.4.3 *Dada el álgebra de Leibniz* ($\mathfrak{g} =< e, f >_{\mathbb{Q}}, [-,-] = 0$) *y fijado el orden monomial graduado lexicográfico con* $y_2 > y_1 > x_2 > x_1$, *se calculará una FG-base del ideal* $\overline{J} =< \{\overline{x_1y_1 + x_1^2}, \overline{x_1^2y_1^2 + x_1^4}\} >\subseteq UL(\mathfrak{g})$ *(se supone* $\phi(l_e) = y_1,\ \phi(l_f) = y_2,\ \phi(r_e) = x_1,\ \phi(r_f) = x_2$*).*

Una base de Gröbner del ideal $J =< \{x_2x_1 - x_1x_2, y_1x_1 - x_1y_1, y_1x_2 - x_2y_1, y_2x_1 - x_1y_2, y_2x_2 - x_2y_2, (x_1 + y_1)y_1,\ (x_1 + y_1)y_2, (x_2 + y_2)y_1, (x_2 + y_2)y_2, x_1y_1 + x_1^2, x_1^2y_1^2 + x_1^4\} >$ *con respecto al orden monomial graduado lexicográfico, calculada con Bergman v. 0.96, es:*

$G = \{x_1y_1 + x_1^2, x_2x_1 - x_1x_2, y_1x_1 + x_1^2, y_1x_2 - x_2y_1, y_1^2 - x_1^2, y_1y_2 + x_1y_2, y_2x_1 - x_1y_2, y_2x_2 - x_2y_2, y_2^2 + x_2y_2, y_2y_1 + x_2y_1\} \cup \{x_1^4\} \cup \{-x_1^2x_2^{n-1}y_2 - x_1^2x_2^n, x_1x_2^ny_1 + x_1^2x_2^n\}_{n\in\mathbb{N}}$.

Una FG-base del ideal $\overline{J}$ *es entonces:*

$$\overline{G} = \{\overline{x_1y_1 + x_1^2}, \overline{x_1^4}\} \cup \{\overline{-x_1^2x_2^{n-1}y_2 - x_1^2x_2^n}, \overline{x_1x_2^ny_1 + x_1^2x_2^n}\}_{n\in\mathbb{N}}.$$

Ejemplo 3.4.4 *Dada el álgebra de Leibniz* ($\mathfrak{g} =< e, f >_{\mathbb{Q}}, [-,-]$), *siendo:*

$$[e,e] = [f,f] = 0, \quad [e,f] = -e, \quad [f,e] = e$$

y fijado el orden monomial graduado lexicográfico con $y_2 > y_1 > x_2 > x_1$, *se calculará una FG-base del ideal* $\overline{J} =<\overline{x_1^2 + 2x_1 + x_2^2}, \overline{x_1 + 2x_2}>\subseteq UL(\mathfrak{g})$ *(se supone* $\phi(l_e) = y_1,\ \phi(l_f) = y_2,\ \phi(r_e) = x_1,\ \phi(r_f) = x_2$*).*

Una base de Gröbner del ideal $J =< \{-x_1 + x_2x_1 - x_1x_2, y_1x_1 - x_1y_1, y_1 + y_1x_2 - x_2y_1,\ -y_1 + y_2x_1 - x_1y_2, y_2x_2 - x_2y_2, (x_1 + y_1)y_1,\ (x_1 + y_1)y_2, (x_2 + y_2)y_1, (x_2 + y_2)y_2,\ x_1^2 + 2x_1 + x_2^2, x_1 + 2x_2\} >$ *con respecto al orden monomial graduado lexicográfico, calculada con el paquete NCAlgebra es:*

$$\{x_1, y_1, y_2^2, x_2\}$$

Una FG-base del ideal $\overline{J}$ *es entonces:*

$$\overline{G} = \{\overline{x_1}, \overline{y_1}, \overline{x_2y_2}, \overline{x_2}\}$$

Caso II.- Si $dim\mathfrak{g}^{ann} = 1$ y $\mathfrak{g}^{ann}$ es un módulo trivial sobre $\mathfrak{g}_{Lie}$, entonces $\mathfrak{g}$ es isomorfo al álgebra definida por:

$$[e, e] = [e, f] = [f, e] = 0, \quad [f, f] = e$$

Veamos un ejemplo para este caso.

Ejemplo 3.4.5 *En las condiciones de los dos ejemplos anteriores, se calculará una FG-base del ideal:*

$$\overline{J} = \{\overline{x_2^2y_2 - 3x_2^2}, \overline{x_2y_1 - y_2}\} \subseteq UL(\mathfrak{g})$$

Una base de Gröbner del ideal $J =< \{x_1, y_1x_2 - x_2y_1, -y_1 + y_2x_2 - x_2y_2, (x_1 + y_1)y_1, (x_1+y_1)y_2, (x_2+y_2)y_1, (x_2+y_2)y_2, x_2^2y_2-3x_2^2, x_2y_1-y_2\} >$ *con respecto al orden monomial graduado lexicográfico, calculada con el paquete NCAlgebra es:*

$$\{x_1, y_1, y_2, x_2^2\}$$

Una FG-base del ideal $\overline{J}$ *es entonces:*

$$\{\overline{y_1}, \overline{y_2}, \overline{x_2^2}\}$$

Caso III.- Si $dim\mathfrak{g}^{ann} = 1$ y $\mathfrak{g}^{ann}$ no es un módulo trivial sobre $\mathfrak{g}_{Lie}$, entonces $\mathfrak{g}$ es isomorfo al álgebra definida por:

$$[e, e] = [f, e] = 0, \quad [e, f] = e \quad [f, f] = e$$

Veamos un ejemplo para este caso.

Ejemplo 3.4.6 *En las condiciones anteriores, se calculará una FG-base del ideal:*

$$\overline{J} = \{\overline{x_2^2y_2 - 3x_2^2}, \overline{x_2y_1 - y_2}\} \subseteq UL(\mathfrak{g})$$

Una base de Gröbner del ideal $J =< \{x_1, -y_1 + y_1x_2 - x_2y_1, -y_1 + y_2x_2 - x_2y_2, (x_1+y_1)y_1, (x_1+y_1)y_2, (x_2+y_2)y_1, (x_2+y_2)y_2, x_2^2y_2-3x_2^2, x_2y_1-y_2\} >$ *con respecto al orden monomial graduado lexicográfico, calculada con el paquete NCAlgebra es:*

$$\{x_1, y_1, y_2, x_2^2\}$$

Una FG-base del ideal $\overline{J}$ *es entonces:*

$$\{\overline{y_1}, \overline{y_2}, \overline{x_2^2}\}$$

Caso IV.- Si $\mathfrak{g}$ es un álgebra de Leibniz de dimensión 3 existen 10 clases de álgebras de Leibniz que no son álgebras de Lie [AyO98]. Veamos un ejemplo de uno de estos casos.

Ejemplo 3.4.7 *Dada el álgebra de Leibniz* $(\mathfrak{g} =< e, f, h >_{\mathbb{Q}}, [-,-])$*, siendo:*

$$\begin{array}{lll} [e,e] = 0, & [e,f] = 0, & [e,h] = -2e \\ [f,e] = 0, & [f,f] = e, & [f,h] = e - f \\ [h,e] = 0, & [h,f] = -e + f, & [h,h] = e \end{array}$$

y fijado el orden monomial graduado lexicográfico con $y_3 > y_2 > y_1 > x_3 > x_2 > x_1$*, se calculará una FG-base del ideal (se supone* $\phi(l_e) = y_1,\ \phi(l_f) = y_2,\ \phi(l_h) = y_3,\ \phi(r_e) = x_1,\ \phi(r_f) = x_2,\ \phi(r_h) = x_3$*):*

$$\overline{J} = \{\overline{x_2^2y_2 - 3x_2^2},\ \overline{x_2y_1 - y_3^2}\} \subseteq UL(\mathfrak{g})$$

Una base de Gröbner del ideal $J =< \{-x_1 - x_2 - x_2x_3 + x_3x_2, x_1, y_1x_2 - x_2y_1, 2y_1 + y_1x_3 - x_3y_1, -y_1 + y_2x_2 - x_2y_2, y_1 + y_2 + y_2x_3 - x_3y_2, y_1 - y_2 + y_3x_2 - x_2y_3, -y_1 + y_3x_3 - x_3y_3, (x_1 + y_1)y_1, (x_1 + y_1)y_2, (x_1 + y_1)y_3, (x_2 + y_2)y_1, (x_2 + y_2)y_2, (x_2 + y_2)y_3, (x_3 + y_3)y_1, (x_3 + y_3)y_2, (x_3 + y_3)y_3, x_2^2y_2 - 3x_2^2, x_2y_1 - y_3^2\} >$ *con respecto al orden monomial graduado lexicográfico con* $y_3 > y_2 > y_1 >$

$x_3 > x_2 > x_1$, calculada con el paquete NCAlgebra es:

$$\{x_1, -x_2 - x_2x_3 + x_3x_2, x_2^2, y_3^2, x_3y_3, x_2y_3, x_3y_2, y_2y_3, y_3y_2, y_1, y_3x_3, x_2y_2,$$
$$y_2x_2, y_2 + y_2x_3, -y_2 + y_3x_2, y_2y_2\}$$

Una FG-base de Gröbner del ideal $\overline{J}$ es entonces:

$$\{\overline{x_2^2}, \overline{x_3y_3}, \overline{x_2y_3}, \overline{x_3y_2}, \overline{y_1}, \overline{x_3y_3 + y_1}, \overline{x_2y_2}, \overline{x_2y_2 + y_1}, \overline{x_3y_2 - y_1}, \overline{x_2y_3 - y_1}\}$$

A la vista del ejemplo 3.4.3 las FG-bases en $UL(\mathfrak{g})$ pueden tener infinitos elementos; este comportamiento de las FG-bases no es exclusivo del anillo $UL(\mathfrak{g})$ pues incluso en el anillo de polinomios conmutativos, visto como un cociente del álgebra asociativa no conmutativa libre, también existen FG-bases con infinitos elementos. Esto nos sugiere que el concepto de FG-base no es quizás el más adecuado para tomarlo como definición definitiva de base de Gröbner en $UL(\mathfrak{g})$, al mismo tiempo surge de modo inmediato una nueva cuestión, pues no hay que olvidar que $UL(\mathfrak{g})$ es un anillo noetheriano: ¿es posible establecer una definición de base de Gröbner donde el número de elementos que la componen sea siempre finito?

Para empezar a responder a esta pregunta conviene recordar que en el modelo conmutativo el algoritmo de Buchberger [Buc65] para tras un número finito de pasos gracias la relación de divisibilidad entre monomios, la cual permite construir una cadena estrictamente creciente de ideales que debe ser estacionaria debido a la noetherianidad del anillo, condición que garantiza la finitud del proceso. No hay que olvidar pues el papel importante que debe jugar el criterio de divisibilidad entre monomios, con el fin de repetir el esquema del caso conmutativo.

Mientras en el álgebra asociativa no conmutativa libre el criterio de divisibilidad es muy rígido, pues asegurar que un monomio A divide a otro B

es decir que A es una subpalabra de B. No ocurre así en $UL(\mathfrak{g})$, donde decir que un monomio A divide a otro B es asegurar que existen monomios C y D tales que lm(B)=lm(C·A·D); éste concepto, sin duda, da mucho más juego pues es mucho más amplio y a la postre va a ser una de las garantías para conseguir una base de Gröbner con un número finito de elementos.

No resulta extraño entonces pensar en la posibilidad de armonizar estos tres hechos, por una parte las FG-bases, por otra el cambio del concepto de divisibilidad entre monomios en el álgebra asociativa no conmutativa libre por el concepto de divisibilidad entre monomios en $UL(\mathfrak{g})$ y por otra la noetherianidad de $UL(\mathfrak{g})$, y a partir de ellos construir la nueva definición de base de Gröbner en $UL(\mathfrak{g})$ en la cual se tenga garantizado que dichas bases de Gröbner contengan siempre un número finito de elementos.

El objetivo a conseguir de aquí al final de esta sección es pues, obtener una definición de base de Gröbner en $UL(\mathfrak{g})$ donde el número de elementos que la componen sea siempre finito.

Por el Teorema 3.3.6 de Estructura de la sección anterior, se verifica que existe un isomorfismo de $\mathbb{K}$-espacios vectoriales:

$$\psi : UL(\mathfrak{g}) \longrightarrow \mathbb{K}[x_1, ..., x_m] \oplus \mathbb{K}[x_1, ..., x_m]y_1 \oplus ... \oplus \mathbb{K}[x_1, ..., x_m]y_n$$

Definición 3.4.8 *Sea* $V = \psi^{-1}(\mathbb{K}[x_1, ..., x_m])$.

Partiendo de una FG-base $\overline{G}$ de un ideal $\overline{J}$, se puede construir el conjunto $lm(\overline{G})$ formado por los monomios principales de los polinomios de $\overline{G}$.

Lema 3.4.9 *El isomorfismo* ψ *induce una partición sobre* $lm(\overline{G})$*, de modo que:*

$$lm(\overline{G}) = \overline{D_0} \bigsqcup ... \bigsqcup \overline{D_n}$$

siendo $\overline{D_i}$ *el subconjunto de* $lm(\overline{G})$ *formado por los monomios que están en la componente i-ésima de* $\mathbb{K}[x_1, ..., x_m] \oplus \mathbb{K}[x_1, ..., x_m]y_1 \oplus ... \oplus \mathbb{K}[x_1, ..., x_m]y_n, i \in \{0, ..., n\}$.

Demostración:
Trivial. □

Lema 3.4.10 *Para cada $\overline{D_i}$ se puede extraer un subconjunto finito de monomios $\overline{E_i} \subseteq \overline{D_i}$ tales que $\forall \overline{d} \in \overline{D_i}$ $\exists \overline{e} \in \overline{E_i}$ y $\overline{p} \in V$ tales que $\overline{d} = lm(\overline{p} \cdot \overline{e}), i \in \{0, ..., n\}$*

Demostración:
Sea $i \in \{0, ..., n\}$.
Si $\overline{D_i}$ fuese un conjunto con un número finito de elementos, bastaría tomar $\overline{E_i} = \overline{D_i}$. De no ser así, se considera el ideal generado por $\psi(\overline{D_i})$: $< \psi(\overline{D_i}) > \subseteq \mathbb{K}[x_1, ..., x_m]y_i$[3]. Como $\mathbb{K}[x_1, ..., x_m]y_i$ es un anillo noetheriano, es posible encontrar un subconjunto finito de $\psi(\overline{D_i})$, sea éste $\psi(\overline{E_i})$, tal que todo elemento de $\psi(\overline{D_i})$ es dividido por un elemento de $\psi(\overline{E_i})$, esto es, para todo $\psi(\overline{d}) \in \psi(\overline{D_i})$ existe $\psi(\overline{e}) \in \psi(\overline{E_i})$ y $\psi(\overline{p}) \in \mathbb{K}[x_1, ..., x_m]$ tal que $\psi(\overline{d}) = \psi(\overline{p}) \cdot \psi(\overline{e})$. Por lo tanto, y teniendo en cuenta cómo se multiplica a un lado y al otro, se verifica que $\forall \overline{d} \in \overline{D_i}$, $\exists \overline{e} \in \overline{E_i}$ y $\overline{p} \in V$ tal que $\overline{d} = lm(\overline{p} \cdot \overline{e})$ □

Definición 3.4.11 *Sea $\overline{H_i}$ el subconjunto de $\overline{G}$ formado por los polinomios cuyos monomios principales están en $\overline{E_i}$, $i \in \{0, ..., n\}$, y B la $\mathbb{K}$-base de $UL(\mathfrak{g})$ que se obtiene a partir de una base de Gröbner del ideal $\Phi(I)$.*

Proposición 3.4.12 *El conjunto $\overline{H} = \bigcup_{i=0}^{n} \overline{H_i}$ verifica:*

i) $<\overline{H}>=<\overline{G}>=\overline{J}$.

ii) $\forall \overline{f} \in \overline{J}$, $\exists\, \overline{p}, \overline{q} \in B$ *y* $\overline{g} \in \overline{H}$ *tales que* $lm(\overline{f}) = lm(\overline{p} \cdot lm(\overline{g}) \cdot \overline{q})$.

[3]Por comodidad en la notación se supone $y_0 = 1$

Demostración:
i) Trivial por la propia construcción de $\overline{H}$ y por verificar B la condición de cadena descendente.

ii) Para todo $\overline{f}$ en $\overline{J}$ se puede encontrar $\overline{g}$ en $\overline{G}$ y monomios $\overline{a}, \overline{b}$ en B tales que $lm(\overline{f}) = \overline{a} \cdot lm(\overline{g}) \cdot \overline{b}$.

Sólo son posibles dos opciones, o $\overline{g}$ está en $\overline{H}$, y por lo tanto ya estaría probado, o $\overline{g}$ no está en $\overline{H}$, en cuyo caso se pueden encontrar un polinomio $\overline{t}$ en $\overline{H}$ y $\overline{r}$ en V tales que $lm(\overline{g}) = lm(\overline{r} \cdot lm(\overline{t}))$.

$lm(\overline{f}) = \overline{a} \cdot lm(\overline{r} \cdot lm(\overline{t})) \cdot \overline{b} = lm(\overline{a} \cdot \overline{r} \cdot lm(\overline{t}) \cdot \overline{b}) = lm(lm(\overline{a} \cdot \overline{r}) \cdot lm(\overline{t}) \cdot \overline{b})$.

□

Llegados a este punto se está en condiciones de establecer el concepto de Base de Gröbner en $UL(\mathfrak{g})$.

Definición 3.4.13 *Se denomina FG-base de Gröbner finita obtenida a partir de la FG-base $\overline{G}$, brevemente ${}_f\overline{G}$, al conjunto $\overline{H}$.*

Ejemplo 3.4.14 *Se calculará una FG-base de Gröbner finita, ${}_f\overline{G}$, siendo $\overline{G}$ la FG-base del Ejemplo 3.4.3.*

Puesto que $\overline{G} = \{\overline{x_1y_1 + x_1^2}, \overline{x_1^4}\} \cup \{\overline{-x_1^2x_2^{n-1}y_2 - x_1^2x_2^n}, \overline{x_1x_2^ny_1 + x_1^2x_2^n}\}_{n\in\mathbb{N}}$ *se puede afirmar entonces que* $lm(G) = \{\overline{x_1y_1}, \overline{x_1^4}\} \cup \{\overline{-x_1^2x_2^{n-1}y_2}, \overline{x_1x_2^ny_1}\}_{n\in\mathbb{N}}$.

Procediendo al igual que en el desarrollo teórico anterior, se tienen tres conjuntos $D_0 = \{\overline{x_1^4}\}$, $D_1 = \{\overline{x_1}\} \cup \{\overline{x_1x_2^n}\}$, $D_2 = \{\overline{-x_1^2x_2^{n-1}}\}$*; que a su vez dan lugar a* $E_0 = \{\overline{x_1^4}\}$, $E_1 = \{\overline{x_1}\}$, $E_2 = \{\overline{x_1^2}\}$.

Así pues $\overline{H} = \{\overline{x_1^4}, \overline{x_1y_1 + x_1^2}, \overline{-x_1^2y_2 - x_1^2x_2}\}$ *es una FG-base de Gröbner finita del ideal $\overline{J}$.*

No se debería acabar esta sección sin recalcar la importancia que tiene exigir que el orden monomial sea graduado. Imponer esta condición es un punto clave a la hora de definir el concepto de FG-base de Gröbner finita, pues este hecho garantiza que se verifica:

Proposición 3.4.15 *[Gra00] Sean* $u = x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m}, v = x_1^{\beta_1} \cdot ... \cdot x_m^{\beta_m} \in V$ *entonces* $lm(u \cdot v) = x_1^{\alpha_1+\beta_1} \cdot ... \cdot x_m^{\alpha_m+\beta_m}$.

Proposición 3.4.16 *[Gra00] Sean* $u = x_1^{\alpha_1} \cdot ... \cdot x_m^{\alpha_m}, v = x_1^{\beta_1} \cdot ... \cdot x_m^{\beta_m} \in V$ *entonces* u *es un factor de* v *si y sólo si* $\alpha_i \leq \beta_i \ \forall i$.

3.5. Extensión de las FG-bases de Gröbner finitas.

¿Cuál es la idea básica que se maneja a la hora de definir el concepto de FG-base de Gröbner finita en $UL(\mathfrak{g})$? La idea básica que se maneja a la hora de definir el concepto de FG-base de Gröbner finita en $UL(\mathfrak{g})$, es la de calcular una FG-base y luego extraer de ésta una FG-base de Gröbner finita. Se puede pensar entonces en aplicar este procedimiento a otros cocientes del álgebra asociativa no conmutativa libre donde ya hay establecida una teoría de bases de Gröbner, y estudiar la relación entre las FG-bases de Gröbner finitas y las bases de Gröbner ya definidas. Pues bien, éste es el objetivo de esta sección.

3.5.1. FG-bases de Gröbner finitas en $\mathbb{K}[x_1, ..., x_n]$

Es conocido que el álgebra $\mathbb{K}[x_1, ..., x_n]$ es, en realidad, un cociente del álgebra $\mathbb{K}<x_1, ..., x_n>$ por el ideal conmutador $I = <\{x_i \cdot x_j - x_j \cdot x_i\}_{i,j \in \{1,...,n\}}>$:

$$\mathbb{K}[x_1, ..., x_n] \cong \frac{\mathbb{K}<x_1,...,x_n>}{<\{x_i \cdot x_j - x_j \cdot x_i\}_{i,j \in \{1,...,n\}}>}$$

Por ser un cociente del álgebra asociativa no conmutativa libre es posible definir de modo análogo a $UL(\mathfrak{g})$ el concepto de FG-base en $\mathbb{K}[x_1, ..., x_n]$.

Definición 3.5.1 *[Nor01] Sea* $\overline{J}$ *un ideal de* $\frac{\mathbb{K}<x_1,...,x_n>}{I}$*, esto es* J *es un ideal de* $\mathbb{K}<x_1, ..., x_n>$ *verificando que* $I \subset J$*. Un conjunto de elementos* $F \subset J$ *reducidos módulo* I *se dice que es una FG-base para* $\overline{J}$ *si, para cada* $f \in J$ *tal que* $\overline{f} \neq 0$ *(*$\Leftrightarrow f \notin I$*) existe* $g \in F$ *tal que* $lm(g) \mid lm(\overline{f})$.

Ejemplo 3.5.2 *Se calculará una FG-base del ideal*

$$\overline{J} =< \overline{xyz}, \overline{-x^2y + z^3}, \overline{x^3y^2} >\subseteq \mathbb{K}[x, y, z]$$

con respecto al orden monomial graduado lexicográfico con $x < y < z$.

Una base de Gröbner del ideal $J =< xyz, -x^2y + z^3, x^3y^2, yx - xy, zx - xz, zy - yz >$, con respecto al orden monomial graduado lexicográfico con $x < y < z$, calculada con el paquete NCAlgebra, *es $\{-x^2y+z^3, x^3y^2, yx-xy, zx-xz, zy-yz\}\cup\{xy^nz\}_{n\in\mathbb{N}}$. Así pues la FG-base del ideal $\overline{J}$ es $\{\overline{-x^2y + z^3}, \overline{x^3y^2}\}\cup\{\overline{xy^nz}\}_{n\in\mathbb{N}}$*

Teorema 3.5.3 *Si $\overline{G}$ es FG-base de $\overline{J}$ con respecto a un orden monomial L entonces, $\overline{G}$ es una base de Gröbner de $\overline{J}$ con respecto al orden monomial L.*

Demostración:
Sea $\overline{f} \in \overline{J}$ y $f \in J$ su representante canónico entonces, existe un polinomio reducido módulo I $g \in G$, $p, q \in X^*$ tales que $lm(f) = p \cdot lm(g) \cdot q$, y por lo tanto $lm(\overline{f}) = \overline{p} \cdot lm(\overline{g}) \cdot \overline{q}$

□

Corolario 3.5.4 *Para todo $\overline{f} \in \overline{J}$ existe siempre $\overline{g} \in \overline{G}$ tal que $lm(\overline{g})$ es una subpalabra de $lm(\overline{f})$.*

Aprovechando que $\mathbb{K}[x_1, ..., x_n]$ es un anillo noetheriano, cabe pensar en la posibilidad, en un principio, de poder definir FG-bases de Gröbner finitas.

Lema 3.5.5 *Si $\overline{G} \subseteq \mathbb{K}[x_1, ..., x_n]$ es FG-base de $\overline{J}$ con respecto a un orden monomial L entonces, del conjunto $\overline{G}$ se puede extraer un subconjunto $\overline{H}$ tal que:*

- $\overline{H}$ es un conjunto finito.

- $\forall \overline{g} \in \overline{G}\ \exists \overline{h} \in \overline{H}\ p, q \in B = \{x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot ... \cdot x_n^{\alpha_n} / (\alpha_1, ..., \alpha_n) \in \mathbb{N}^n\}$ *tales que* $lm(\overline{g}) = lm(\overline{p} \cdot lm(\overline{h}) \cdot \overline{q})$.

Demostración:
Trivial por ser $\mathbb{K}[x_1, ..., x_n]$ un anillo noetheriano, □

Definición 3.5.6 *Se denomina FG-base de Gröbner finita obtenida a partir de la FG-base* $\overline{G}$*, brevemente* ${}_f\overline{G}$*, al conjunto* $\overline{H}$.

Ejemplo 3.5.7 $\{\overline{-x^2y + z^3}, \overline{x^3y^2}, \overline{xyz}\}$ *es una FG-base de Gröbner finita obtenida a partir de la FG-base del ejemplo anterior.*

Teorema 3.5.8 *Si* ${}_f\overline{G}$ *es una FG-base de Gröbner finita con respecto a un orden monomial L de un ideal* $\overline{J}$ *entonces,* ${}_f\overline{G}$ *es una base de Gröbner con respecto al orden monomial L.*

Demostración:
Trivial por la propia definición de FG-base de Gröbner finita. □

Se puede concluir entonces que tanto las FG-bases como las FG-bases de Gröbner finitas calculadas con respecto a un orden monomial L, son siempre una base de Gröbner con respecto al orden monomial L. De modo natural, a la vista de lo que se acaba de exponer, surge la pregunta de si será cierta la implicación contraria, esto es, dada una base de Gröbner con respecto al orden monomial L, ¿es una FG-base?, ¿es una FG-base de Gröbner finita?

Hablando en términos generales y puesto que el algoritmo de Buchberger [Buc65] proporciona bases de Gröbner con un número finito de elementos y las FG-bases pueden contener infinitos elementos, es posible concluir que en general una base de Gröbner no tiene porqué ser una FG-base. Sin embargo:

Teorema 3.5.9 *Sea* $\overline{J} = < \overline{f_1}, ..., \overline{f_r} >$ *un ideal de* $\mathbb{K}[x_1, ..., x_n]$*, L un mismo orden monomial en* $\mathbb{K}[x_1, ..., x_n]$ *y* $\mathbb{K} < x_1, ..., x_n >$ *y* $\overline{G}$ *una base de Gröbner de* $\overline{J}$ *con respecto al orden monomial L, tal que* $\overline{G}$ *tiene un número finito de elementos, entonces* $\overline{G}$ *es una FG-base de Gröbner finita.*

Demostración:
Sea T el conjunto formado por todos los polinomios reducidos módulo I que pertenecen a J, más una base de Gröbner de I con respecto al orden monomial L.

Por construcción T es una FG-base del ideal $\overline{J}$ con respecto al orden monomial L; además $\overline{G} =_f \overline{T}$, esto es, $\overline{G}$ es una FG-base de Gröbner finita obtenida a partir de la FG-base $\overline{T}$. □

3.5.2. FG-bases de Gröbner finitas en $U(\mathfrak{g})$

Sea $(\mathfrak{g}, [-,-])$ un álgebra de Lie de dimensión finita sobre el cuerpo $\mathbb{K}$ con base $\{a_1, ..., a_n\}$ y sea la aplicación lineal $\phi : \mathfrak{g} \to \mathbb{K}<x_1, ..., x_n>$ tal que $\phi(a_i) = x_i$. Es conocido que el álgebra envolvente universal, $U(\mathfrak{g})$, es en realidad un cociente del álgebra $\mathbb{K}<x_1, ..., x_n>$ por el ideal $I =<\{x_j \cdot x_i - x_i \cdot x_j - \phi[a_j, a_i]\}_{i,j\in\{1,...,n\},i<j}>$ [Gra00, HiSt97]:

$$U(\mathfrak{g}) \cong \frac{\mathbb{K}<x_1,...,x_n>}{<\{x_j \cdot x_i - x_i \cdot x_j - \phi[a_j,a_i]\}_{i,j\in\{1,...,n\},i<j}>}$$

Por ser un cociente del álgebra asociativa no conmutativa libre es posible definir de modo análogo a $UL(\mathfrak{g})$ el concepto de FG-base en $U(\mathfrak{g})$.

Al igual que ocurría en $UL(\mathfrak{g})$ y por razones similares, se supondrá que se maneja siempre un orden monomial *graduado* L.

Definición 3.5.10 *[Nor01] Sea $\overline{J}$ un ideal de $\frac{\mathbb{K}<x_1,...,x_n>}{I}$, esto es J es un ideal de $\mathbb{K}<x_1, ..., x_n>$ verificando que $I \subset J$. Un conjunto de elementos $F \subset J$ reducidos módulo I se dice que es una FG-base para $\overline{J}$ si, para cada $f \in J$ tal que $\overline{f} \neq 0$ ($\Leftrightarrow f \notin I$) existe $g \in F$ tal que $lm(g) \mid lm(\overline{f})$.*

Ejemplo 3.5.11 *Sea $\mathfrak{g}$ el álgebra de Lie con base $\{e, f, g\}$ y tabla de multiplicación*

$$[e, f] = g,\ [g, e] = 2e,\ [g, f] = -2f$$

El álgebra envolvente universal es $U(\mathfrak{g}) \cong \frac{\mathbb{K}<x,y,h>}{<\{h\cdot x-x\cdot h-2x, h\cdot y-y\cdot h+2y, y\cdot x-x\cdot y+h\}>}$.

Calculemos la FG-base del ideal $\overline{J} =< \{\overline{x^2}, \overline{y^2}, \overline{h^2-1}\} >\subset U(\mathfrak{g})$*, siendo L el orden monomial graduado lexicográfico con* $x < y < h$.

Una base de Gröbner del ideal $J =< \{x^2, y^2, h^2-1, h\cdot x - x\cdot h - 2x, h\cdot y - y\cdot h + 2y, y\cdot x - x\cdot y + h\} >$*, con respecto a L, calculada con el paquete NCAlgebra de Mathematica, es* $\{x^2, y^2, h^2-1, xh+x, yh-y, xy - \frac{h}{2} - \frac{1}{2}, hy+y, hx-x, yx+\frac{h}{2}-\frac{1}{2}\}$*. Así pues la FG-base del ideal* $\overline{J}$ *es* $\{\overline{x^2}, \overline{y^2}, \overline{h^2-1}, \overline{xh+x}, \overline{yh-y}, \overline{xy-\frac{h}{2}-\frac{1}{2}}\}$.

Ejemplo 3.5.12 *Dada el álgebra de Lie del ejemplo 3.4.4 se calculará una FG-base del ideal* $\overline{J} =<\overline{x_1^2+2x_1+x_2^2}, \overline{x_1+2x_2}>\subseteq U(\mathfrak{g})$ *(se supone* $x_2 > x_1$*).*

Una base de Gröbner del ideal:

$$J =< \{-x_1 + x_2x_1 - x_1x_2, x_1^2 + 2x_1 + x_2^2, x_1 + 2x_2\} >$$

con respecto al orden monomial graduado lexicográfico, calculada con el paquete NCAlgebra es:

$$\{x_1, x_2\}$$

Una FG-base del ideal $\overline{J}$ *es entonces:*

$$\overline{G} = \{\overline{x_1}, \overline{x_2}\}$$

Teorema 3.5.13 *Si* $\overline{G}$ *es una FG-base de* $\overline{J}$ *entonces* $\overline{G}$ *es una base de Gröbner de* $\overline{J}$ *con respecto al orden monomial graduado L.*

Demostración:
La construcción de Bases de Gröbner en $U(\mathfrak{g})$ puede verse en [Gra00].
Sea $\overline{f} \in \overline{J}$ y $f \in J$ su representante canónico entonces, existe un polinomio reducido módulo I $g \in G$, $p, q \in X^*$ tales que $lm(f) = p \cdot lm(g) \cdot q$, y por lo tanto $lm(\overline{f}) = \overline{p} \cdot lm(\overline{g}) \cdot \overline{q}$.

□

Corolario 3.5.14 *Para todo $\overline{f} \in \overline{J}$ existe siempre $\overline{g} \in \overline{G}$ tal que $lm(\overline{g})$ es una subpalabra de $lm(\overline{f})$.*

Aprovechando que $U(\mathfrak{g})$ es un anillo noetheriano [Dix96], cabe pensar en la posibilidad, en un principio, de poder definir FG-bases de Gröbner finitas.

Lema 3.5.15 *Si $\overline{G} \subseteq U(\mathfrak{g})$ es FG-base de $\overline{J}$ con respecto a un orden monomial graduado L entonces, del conjunto $\overline{G}$ se puede extraer un subconjunto $\overline{H}$ tal que:*

- *$\overline{H}$ es un conjunto finito.*
- *$\forall \overline{g} \in \overline{G}\ \exists \overline{h} \in \overline{H}\ p, q \in B$ tales que $lm(\overline{g}) = lm(\overline{p} \cdot lm(\overline{h}) \cdot \overline{q})$.*

Demostración:
Trivial por ser $U(\mathfrak{g})$ un anillo noetheriano.

□

Definición 3.5.16 *Se denomina FG-base de Gröbner finita obtenida a partir de la FG-base $\overline{G}$, brevemente ${}_f\overline{G}$, al conjunto $\overline{H}$.*

Teorema 3.5.17 *Si ${}_f\overline{G}$ es una FG-base de Gröbner finita de un ideal $\overline{J}$ entonces, ${}_f\overline{G}$ es una base de Gröbner con respecto al orden monomial graduado L.*

Demostración:
Trivial por la propia definición de FG-base de Gröbner finita.

□

Se puede concluir entonces que tanto las FG-bases como las FG-bases de Gröbner finitas calculadas con respecto a un orden monomial graduado L, son

siempre una base de Gröbner con respecto al orden monomial graduado L. De modo natural, a la vista de lo que se acaba de exponer, surge la pregunta de si será cierta la implicación contraria, esto es, dada una base de Gröbner con respecto al orden monomial graduado L, ¿es una FG-base?, ¿es una FG-base de Gröbner finita?

Hablando en términos generales y puesto que el algoritmo de Buchberger [Buc65] proporciona bases de Gröbner con un número finito de elementos y las FG-bases pueden contener infinitos elementos, da la impresión de que en general una base de Gröbner no tiene porqué ser una FG-base. Sin embargo:

Teorema 3.5.18 *Sea* $\overline{J} =< \overline{f_1}, ..., \overline{f_r} >$ *un ideal de* $U(\mathfrak{g})$ *y* $\overline{G}$ *una base de Gröbner con respecto al orden monomial graduado L tal que* $\overline{G}$ *tiene un número finito de elementos entonces,* $\overline{G}$ *es una FG-base de Gröbner finita.*

Demostración:
Sea T el conjunto formado por todos los polinomios reducidos módulo I que pertenecen a J, más una base de Gröbner de I con respecto al orden monomial graduado L.

Por construcción T es una FG-base del ideal $\overline{J}$ con respecto al orden monomial L; además $\overline{G} =_f \overline{T}$, esto es, $\overline{G}$ es una FG-base de Gröbner finita obtenida a partir de la FG-base $\overline{T}$.

□

3.6. Aplicaciones

Una vez que se ha hecho un desarrollo teórico de esta índole, sería incomprensible no mostrar por lo menos una aplicación en la cual se utilice toda la potencia de la teoría de las FG-bases de Gröbner finitas, para resolver un problema en concreto.

De entre todo el abanico de posibilidades se ha seleccionado el test de pertenencia de un elemento a un ideal.

3.6.1. Test de pertenencia

Sea $\overline{J}$ un ideal de $UL(\mathfrak{g})$, $\overline{f}$ un elemento de $UL(\mathfrak{g})$ dado en forma canónica y $\overline{G}$ una FG-base de Gröbner finita de $\overline{J}$ con respecto a un orden monomial graduado.

La proposición 3.4.12 garantiza que es condición necesaria, pero no suficiente, para que $\overline{f}$ pertenezca $\overline{J}$ que existan monomios $\overline{p}, \overline{q}$ y un elemento $\overline{g} \in \overline{G}$ tales que $lm(\overline{f}) = lm(\overline{p} \cdot lm(\overline{g}) \cdot \overline{q})$. Con este resultado presente parece claro el método a seguir, en primer lugar se comprueba si existe algún elemento de $\overline{G}$ cuyo monomio principal divida al monomio principal de $\overline{f}$, en caso de que esto no sea así ya se puede afirmar que $\overline{f}$ no pertenece a $\overline{J}$, en el supuesto contrario se reduce $\overline{f}$ por $\overline{g}$ y se vuelve a empezar el proceso con este nuevo polinomio.

Es importante destacar el hecho de que el proceso anteriormente descrito, se basa en construir una sucesión de polinomios que comienza en $\overline{f}$ y cuyos términos principales forman una sucesión estrictamente decreciente. Se podría pensar entonces en la posibilidad de que el algoritmo esbozado anteriormente no acabe en un número finito de pasos; nada más lejos de la realidad pues el hecho de trabajar con un orden monomial va a garantizar que esta sucesión estrictamente decreciente se estaciona en un número finito de etapas, esto es, el proceso anteriormente descrito acaba en un número finito de pasos.

Éstas ideas que se acaban de exponer se recogen en el siguiente algoritmo.

Algoritmo 3.6.1 *Test de Pertenencia*
Entrada: $\overline{f}$, $\overline{J}$, $\overline{G}$.
Salida: True si el polinomio pertenece al ideal dado y False en el caso contrario.

1. $\overline{h} = \overline{f}$, $\Phi = 2$

2. *Mientras* $\Phi = 2$

- *¿ Existen monomios* $\overline{p}, \overline{q}$ *y un polinomio* $\overline{g} \in \overline{G}$ *tales que* $lm(\overline{h}) = lm(\overline{p} \cdot lm(\overline{g}) \cdot \overline{q})$ *?*
  - *Sí*
    - $\overline{h} = \overline{h} - lc(\overline{h}) \cdot \overline{p} \cdot \overline{g} \cdot \overline{q}$
    - *¿* $\overline{h} = 0$ *?*

      Sí.

      $\Phi = 0$
  - *No.*

    $\Phi = 1$

3. *¿* $\Phi = 0$ *?*
   - *Sí. Retornar True*
   - *No. Retornar False*

Ejemplo 3.6.2 *Determinar utilizando el algoritmo anterior si el polinomio* $\overline{f} = \overline{y_2 - 6x_2^2 - x_2y_1 + x_2y_2 + 2x_2^2y_2}$ *pertenece o no al ideal* $\overline{J}$ *del ejemplo 3.4.6.*

1. $\overline{h} = \overline{f}$, $\Phi = 2$
2. $\overline{x_2^2y_2} = \overline{1} \cdot \overline{x_2^2} \cdot \overline{y_2}$
3. $\overline{h} = \overline{h} - 2 \cdot \overline{x_2^2} \cdot \overline{y_2} = \overline{y_2 - 6x_2^2 - x_2y_1 + x_2y_2}$
4. $\overline{x_2y_2} = \overline{x_2} \cdot \overline{y_2} \cdot \overline{1}$
5. $\overline{h} = \overline{h} - \overline{x_2} \cdot \overline{y_2} = \overline{y_2 - 6x_2^2 - x_2y_1}$
6. $\overline{x_2y_1} = \overline{x_2} \cdot \overline{y_1} \cdot \overline{1}$
7. $\overline{h} = \overline{h} + \overline{x_2} \cdot \overline{y_1} = \overline{y_2 - 6x_2^2}$
8. $\overline{x_2^2} = \overline{1} \cdot \overline{x_2^2} \cdot \overline{1}$

9. $\overline{h} = \overline{h} + \overline{6x_2^2} = \overline{y_2}$

10. $\overline{y_2} = \overline{1} \cdot \overline{y_2} \cdot \overline{1}$

11. $\overline{h} = \overline{h} - \overline{y_2} = 0$

12. *True*

Se puede concluir entonces que el polinomio $\overline{f}$ *pertenece al ideal* $\overline{J}$.

Capítulo 4

Listados del paquete SmithGroebner

```
BeginPackage["SmithGroebner`",{"Algebra`PolynomialExtendedGCD`","Algebra`FiniteFields`"}]

    (*
    =============== MENSAJES DE ERROR.

    ===== Función: CargaFunciones.
    *)

    CargaFunciones::MatrizErronea="`1` No es una matriz.";

    CargaFunciones::MatrizNoEnteros="`1` No es una matriz de números Enteros.";

    CargaFunciones::MatrizNoRacionales="`1` No es una matriz de polinomios con coeficientes Racionales
    en la variable `2`. En el contexto en el que nos encontramos, racional quiere decir un número
    entero dividido por otro número entero. Esto es, 0.8 no se considera Racional, pero 8/10 sí.";

    CargaFunciones::MatrizNoGauss="`1` No es una matriz de números Enteros Gaussianos.";

    CargaFunciones::MatrizNoQ="`1` No es una matriz de números racionales. En el contexto que nos
    encontramos, racional quiere decir un número entero dividido por otro número entero. Esto es,
    0.8 no se considera racional, pero 8/10 sí.";

    CargaFunciones::MatrizNoR="`1` No es una matriz de números reales.";

    CargaFunciones::MatrizNoC="`1` No es una matriz de números complejos.";

    CargaFunciones::MatrizNoReales="`1` No es una matriz de polinomios con coeficientes Reales
    en la variable `2`.";

    CargaFunciones::MatrizNoComplejos="`1` No es una matriz de polinomios con coeficientes Complejos
    en la variable `2`.";

    CargaFunciones::MatrizNoModp="`1` No es una matriz de números enteros módulo p (p primo).";

    CargaFunciones::pNoPrimo="`1` No es un número primo y debe serlo.";

    CargaFunciones::AnilloIncorrecto="`1` No es un anillo de trabajo válido. Para consultar los
    anillos de trabajo válidos utilice la función DIP.";

    CargaFunciones::CoefsnoEnteros="Se detectaron polinomios con coeficientes no enteros.";
```

```
CargaFunciones::NoenCuerpo="Se detectaron elementos en la matriz que no pertenecen al
cuerpo ‘1‘ .";

CargaFunciones::NoenAnilloPol="Se detectaron elementos en la matriz que no pertenecen
al anillo ‘1‘ .";

CargaFunciones::polReducible="El polinomio que se quiere utilizar para construír el cuerpo
finito, es reducible sobre el anillo de los enteros módulo ‘1‘ .";

CargaFunciones::ErrLista="Los coeficientes del polinomio irreducible que se utilizará para
construír el cuerpo finito, deben ser enteros entre 0 y ‘1‘ .";

CargaFunciones::dNoPositivo="Galois[p,d], d debe ser estrictamente positivo.";

CargaFunciones::MatrizNo19="‘1‘ No es una matriz de elementos de Z[(1+Sqrt[-19])/2].";

(*
=============== Función: GroebnerEquivalente
*)
GroebnerEquivalente::ErrDatos="Error en los Datos de Entrada. Los datos introducidos no son
coherentes con los valores esperados.";

GroebnerEquivalente::NoOpcion="‘1‘ No está formateado como una opción.";

GroebnerEquivalente::noOpcionOk="La opción tecleada: ‘1‘ no es correcta. Las únicas opciones
posibles son: Paso -> Sí   ó  Paso -> No.";

(*
=============== Función: Smith
*)
Smith::ErrDatos="Error en los Datos de Entrada. Los datos introducidos no son coherentes con los
valores esperados.";

Smith::NoOpcion="‘1‘ No está formateado como una opción.";

Smith::noOpcionOk="La opción tecleada: ‘1‘ no es correcta. Las únicas opciones posibles son:
Paso -> Sí  ó Paso -> No.";

(*
```

```
============== Función: InversaPaso
*)
InversaPaso::noInversible="La matriz dada no es inversible sobre el anillo de las matrices
con coeficientes en `1`";
InversaPaso::noCuadrada="La matriz no es cuadrada. El número de filas es `1`
y el de columnas es `2`.";
InversaPaso::matrizNula="La matriz Nula no es inversible.";
InversaPaso::ErrDatos="Error en los Datos de Entrada. Los datos introducidos no son
coherentes con los valores esperados.";

(*
============== Función: InversaMinimo
*)
InversaMinimo::noCuadrada="La matriz no es cuadrada. El numero de filas es `1`
y el de columnas es `2`.";
InversaMinimo::matrizNula="La matriz nula no es inversible.";
InversaMinimo::matrizSingular="La Matriz `1` es singular";
InversaMinimo::ErrDatos="Error en los Datos de Entrada. Los datos introducidos no son
coherentes con los valores esperados.";

(*
============== Función: ValidaCanonica
*)
ValidaCanonica::ErrDatos="Error en los datos de entrada. Para consultar la lista de los
Anillos de Trabajo teclee DIP .";

ValidaCanonica::NoCuadrada="La matriz no corresponde a ningún endomorfismo, pues
no es cuadrada.";

(*
============== Función: CalculoFormaCanonica
*)
CalculoFormaCanonica::NoDivisores="No se pueden obtener los divisores elementales
sobre el cuerpo seleccionado, por lo que no es posible proporcionar la forma canónica
pedida.";
(*
============== BREVE DESCRIPCIÓN DE LAS FUNCIONES QUE ESTARÁN A DISPOSICIÓN DEL USUARIO.
*)

GroebnerEquivalente::usage="GroebnerEquivalente[Matriz,AnilloBase] obtiene, mediante un cálculo
```

```
parcial de una Base de Groebner, una matriz equivalente a la dada.
GroebnerEquivalente[Matriz,AnilloBase,Paso->Sí] devuelve una lista {A,P}, donde A es una matriz
equivalente a Matriz, y P es una matriz inversible tal que A=P.Matriz.
Para consultar la lista de los Anillos de Trabajo, teclee DIP.";

Smith::usage="Smith[Matriz,AnilloBase] obtiene la Forma Normal de Smith de Matriz.
Smith[Matriz,AnilloBase,Paso->Sí] devuelve una lista {S,P,Q}, donde S es la Forma Normal de Smith
de Matriz, y P y Q son matrices inversibles tales que S=P.Matriz.Q.
Para consultar la lista de los Anillos de Trabajo, teclee DIP.";

InversaPaso::usage="InversaPaso[Matriz,AnilloBase] calcula la Inversa de Matriz sobre el anillo
AnilloBase. Para consultar la lista de los Anillos de Trabajo, teclee DIP.";

InversaMinimo::usage="InversaMinimo[Matriz] calcula la inversa de Matriz con coeficientes sobre el
cuerpo de los números complejos.";

Frobenius::usage="Dada una matriz de un endomorfismo A con coeficientes en un cuerpo K,
Frobenius[A,K] proporciona una lista {R,P}, donde R es la Forma Canónica Racional de A y,
P es la matriz de paso (P^(-1).A.P=R).";

SegundaFormaC::usage="Dada una matriz de un endomorfismo A con coeficientes en un cuerpo K,
SegundaFormaC[A,K] proporciona una lista {S,P}, donde S es la Segunda Forma Canónica de A y,
P es la matriz de paso (P^(-1).A.P=S).";

Jacobson::usage="Dada una matriz de un endomorfismo A con coeficientes en un cuerpo K,
Jacobson[A,K] proporciona una lista {J,P}, donde J es la Forma Canónica de Jacobson de A y,
P es la matriz de paso (P^(-1).A.P=J).";

(*
=============== OPCIONES
*)

Options[GroebnerEquivalente]={Paso->No};
Options[Smith]={Paso->No};

(*
=============== FUNCIÓN: DIP
Nota: Esta función NO puede ir dentro del contexto Private. Ya que el sistema la
interpretaría como una variable y pasaría a llamarse NombrePaquete`Private`DIP.
*)
```

```
DIP:=Module[{},

    CellPrint[
    {Cell["Anillos de Trabajo del Paquete SmithGroebner","Title",FontSize->12,
    CellFrame->True,Background->RGBColor[249/255.,244/255.,225/255.]],
    Cell["Z Anillo de los Enteros.","Section",FontSize->11,CellFrame->True],
    Cell["Z[(1+Sqrt[-19])/2] Clausura Entera de Z en Q(Sqrt[-19]).","Section",
         FontSize->11,CellFrame->True],
    Cell["Z[i] Anillo de los Enteros Gaussianos.","Section",FontSize->11,
    CellFrame->True],
    Cell["Q Cuerpo de los Números Racionales.","Section",FontSize->11,
    CellFrame->True],
    Cell["R Cuerpo de los Números Reales.","Section",FontSize->11,
    CellFrame->True],
    Cell["C Cuerpo de los Números Complejos.","Section",FontSize->11,
    CellFrame->True],
    Cell["Q[x] Anillo de los polinomios con coeficientes Racionales."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["R[x] Anillo de los polinomios con coeficientes Reales."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["C[x] Anillo de los polinomios con coeficientes Complejos."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["Z[p] Anillo de los enteros módulo p (p primo)."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["Z[p,x] Anillo de los polinomios con coeficientes en el cuerpo de los enteros módulo p (p primo)."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["Galois[p,L] Cuerpo Finito de p^(longitud(L)-1) elementos. p es un número primo y L es la lista de los coeficientes,
    ordenados de menor a mayor grado,del polinomio irreducible sobre Z[p], que se utiliza para construír el cuerpo finito."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["Galois[p,d] Cuerpo finito de p^d elementos.","Section",FontSize->11,CellFrame->True],
    Cell["Galois[p,L][x] Anillo de polinomios en una variable sobre el cuerpo finito de p^(longitud(L)-1) elementos."
    ,"Section",FontSize->11,CellFrame->True],
    Cell["Galois[p,d][x] Anillo de polinomios en una variable sobre el cuerpo finito de p^d elementos."
    ,"Section",FontSize->11,CellFrame->True]}];

    ];

(*
============== FUNCIÓN: CargaFunciones
Este procedimiento tiene una doble función:
```

```
    - Validar los datos de entrada, para asegurar su coherencia.
    - Cargar las funciones adecuadas para el dip seleccionado.
Al finalizar este procedimiento se devuelven dos valores:
    ·Un booleano que indica si hubo errores ó no.
    ·La matriz de entrada.
Para cuerpos de enteros módulo p ó para anillos de polinomios en una variable
sobre estos cuerpos, el usuario sólo introduce una matriz de enteros y Z[p] ó
Z[p,x] (con p primo). Esta matriz de enteros que facilita el usuario, es
necesario convertirla a enteros módulo p.
Para evitar los errores de redondeo, se sustituyen los números decimales que aparecen en la matriz,
en el caso de R,C,R[x],C[x], por sus respectivas fracciones generatrices.
Es por esto que se devuelve también la matriz de entrada (ahora convertida).

Nota:
    ·Esta función NO puede ir dentro del contexto Private, debido a las comparaciones
que se deben hacer para identificar el anillo base.

    ·La función MatrizIdentidad sólo se utiliza en las rutinas MatrizEquivalentePaso, para
añadir la matriz identidad, en DiagonalPaso, para inicializar las matrices de Paso, en
FormaNormal, FormaNormalPaso y en NormalizaPolinomios.
*)

CargaFunciones[AnilloBase_,Matriz_List]:=Module[{AnilloOk,numelementos,MatrizEntrada,MatrizFinal},

AnilloOk=False;
MatrizFinal=Matriz;

(* Sólo en el caso de que AnilloBase NO sea un cuerpo de característica cero ni un anillo
de polinomios en una variable sobre un cuerpo de característica cero, es necesario
cambiar la definición de MatrizIdentidad *)
MatrizIdentidad[n_]:=IdentityMatrix[n];

(* Se comprueba que la lista de entrada sea una matriz *)
If[MatrixQ[Matriz] && Matriz =!= {{}},
    Which[
            AnilloBase===Z,
            (* Se comprueba que se introdujo una matriz de enteros *)
            MatrizEntrada=Flatten[Matriz];
            numelementos=Length[MatrizEntrada];
            If[
```

```
    Length[Select[MatrizEntrada,Element[#1,Integers]&]]==numelementos,
        MaximoComunDivisor[a_,b_]:=GCD[a,b];
        MaximoComunDivisorEx[a_,b_]:=ExtendedGCD[a,b];
        DividirQ[a_,b_]:=Mod[b,a]==0;
        Cociente[a_,b_]:=Quotient[a,b];
        Normaliza[A_,B_:0]:=NormalizaEnteros[A,B];
        UnidadQ[Valor_]:=UnidadEnteros[Valor];
        AnilloOk=True;
    ,
    (* Else *)
    Message[CargaFunciones::MatrizNoEnteros,Matriz]
],

AnilloBase===Z[(1+Sqrt[-19])/2],
(* Se comprueba que se introdujo una matriz de elementos de Z[(1+Sqrt[-19])/2] *)
If[
    Element[Expand[2/Sqrt[19]*Flatten[Im[Matriz]]],Integers] &&
    Element[Flatten[Expand[Re[Matriz]-1/Sqrt[19]*Im[Matriz]]],Integers],
        MaximoComunDivisor[elalfa_,elbeta_]:=dipnodeMCD[elalfa,elbeta][[1]];
        MaximoComunDivisorEx[elalfa_,elbeta_]:=dipnodeMCD[elalfa,elbeta];
        DividirQ[elalfa_,elbeta_]:=Element[dipnodeCociente[elbeta,elalfa],Integers];
        Cociente[elalfa_,elbeta_]:=Expand[dipnodeCociente[elalfa,elbeta].{1,(1+Sqrt[-19])/2}];
        Normaliza[A_,B_:0]:=If[B===0,Return[A],Return[{A,B}]];
        UnidadQ[Valor_]:=UnidadEnteros[Valor];
        AnilloOk=True;
    ,
    (* Else *)
    Message[CargaFunciones::MatrizNo19,Matriz]
],

{AnilloBase}===Cases[{AnilloBase},Q[_Symbol]], (* Anillo Q[x] *)
(* Se comprueba que los coeficientes están todos en Q[x] *)
MatrizEntrada=Flatten[CoefficientList[Flatten[Matriz],AnilloBase[[1]]]];
If[
    Length[Select[MatrizEntrada,Element[#1,Rationals]&]]==Length[MatrizEntrada],
        MaximoComunDivisor[a_,b_]:=PolynomialGCD[a,b];
        MaximoComunDivisorEx[a_,b_]:=PolynomialExtendedGCD[a,b];
        DividirQ[a_,b_]:=PolynomialRemainder[b,a,AnilloBase[[1]]]===0;
        Cociente[a_,b_]:=PolynomialQuotient[a,b,AnilloBase[[1]]];
        Normaliza[A_,B_:0]:=NormalizaPolinomios[A,AnilloBase[[1]],B];
```

```
        UnidadQ[Valor_]:=UnidadPolinomios[Valor,AnilloBase[[1]]];
        (* Para el cálculo de formas canónicas *)
        Factorizar[pol_]:=FactQ[pol];
        AnilloOk=True;
,
(* Else *)
    Message[CargaFunciones::MatrizNoRacionales,Matriz,AnilloBase[[1]]];
],

{AnilloBase}===Cases[{AnilloBase},R[_Symbol]], (* Anillo R[x] *)
(* Se comprueba que los coeficientes están todos en R[x] *)
MatrizEntrada=Flatten[CoefficientList[Flatten[Matriz],AnilloBase[[1]]]];
If[
    Length[Select[MatrizEntrada,Element[#1,Reals]&]]==Length[MatrizEntrada],
        (*MaximoComunDivisor[a_,b_]:=PolynomialGCD[a,b,Extension->Automatic]; *)
        MaximoComunDivisor[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]][[1]];
        MaximoComunDivisorEx[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]];
        DividirQ[a_,b_]:=PolynomialRemainder[b,a,AnilloBase[[1]]]===0;
        Cociente[a_,b_]:=PolynomialQuotient[a,b,AnilloBase[[1]]];
        Normaliza[A_,B_:0]:=NormalizaPolinomios[A,AnilloBase[[1]],B];
        UnidadQ[Valor_]:=UnidadPolinomios[Valor,AnilloBase[[1]]];
        MatrizFinal=ConvertirPolinomio[MatrizFinal,AnilloBase[[1]]];
        Factorizar[pol_]:=FactReal[pol];
        AnilloOk=True;
,
(* Else *)
    Message[CargaFunciones::MatrizNoReales,Matriz,AnilloBase[[1]]];
],

{AnilloBase}===Cases[{AnilloBase},C[_Symbol]], (* Anillo C[x] *)
(* Se comprueba que los coeficientes están todos en C[x] *)
MatrizEntrada=Flatten[CoefficientList[Flatten[Matriz],AnilloBase[[1]]]];
If[
    Length[Select[MatrizEntrada,Element[#1,Complexes]&]]==Length[MatrizEntrada],
        (* MaximoComunDivisor[a_,b_]:=PolynomialGCD[a,b,Extension->Automatic]; *)
        MaximoComunDivisor[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]][[1]];
        MaximoComunDivisorEx[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]];
        DividirQ[a_,b_]:=PolynomialRemainder[b,a,AnilloBase[[1]]]===0;
        Cociente[a_,b_]:=PolynomialQuotient[a,b,AnilloBase[[1]]];
        Normaliza[A_,B_:0]:=NormalizaPolinomios[A,AnilloBase[[1]],B];
```

```
        UnidadQ[Valor_]:=UnidadPolinomios[Valor,AnilloBase[[1]]];
        MatrizFinal=ConvertirPolinomio[MatrizFinal,AnilloBase[[1]]];
        Factorizar[pol_]:=FactCompleja[pol];
        AnilloOk=True;
,
(* Else *)
    Message[CargaFunciones::MatrizNoComplejos,Matriz,AnilloBase[[1]]];
],

AnilloBase===Z[i],
(* Se comprueba que los coeficientes son enteros gaussianos *)
MatrizEntrada=Flatten[Matriz];
numelementos=Length[MatrizEntrada];
If[
    Length[Select[Re[MatrizEntrada],Element[#1,Integers]&]]==numelementos &&
    Length[Select[Im[MatrizEntrada],Element[#1,Integers]&]]==numelementos,
        MaximoComunDivisor[a_,b_]:=GCD[a,b];
        MaximoComunDivisorEx[a_,b_]:=ExtendedGCD[a,b];
        DividirQ[a_,b_]:=Mod[b,a]==0;
        Cociente[a_,b_]:=Quotient[a,b];
        Normaliza[A_,B_:0]:=NormalizaEnterosGauss[A,B];
        UnidadQ[Valor_]:=UnidadEnterosGauss[Valor];
        AnilloOk=True,
(* Else *)
    Message[CargaFunciones::MatrizNoGauss,Matriz]
],

AnilloBase===Q,
(* Se comprueba que los coeficientes son números racionales *)
MatrizEntrada=Flatten[Matriz];
numelementos=Length[MatrizEntrada];
If[
    Length[Select[MatrizEntrada,Element[#1,Rationals]&]]==numelementos,
        MaximoComunDivisor[a_,b_]:=a;
        MaximoComunDivisorEx[a_,b_]:={a,{1,0}};
        DividirQ[a_,b_]:=True;
        Cociente[a_,b_]:=a/b;
        Normaliza[A_,B_:0]:=NormalizaCuerpo[A,B];
        UnidadQ[Valor_]:=UnidadCuerpo[Valor];
        AnilloOk=True,
```

```
(* Else *)
    Message[CargaFunciones::MatrizNoQ,Matriz]
],

AnilloBase===R,
(* Se comprueba que los coeficientes son números reales *)
MatrizEntrada=Flatten[Matriz];
numelementos=Length[MatrizEntrada];
If[
    Length[Select[MatrizEntrada,Element[#1,Reals]&]]==numelementos,
        MaximoComunDivisor[a_,b_]:=a;
        MaximoComunDivisorEx[a_,b_]:={a,{1,0}};
        DividirQ[a_,b_]:=True;
        Cociente[a_,b_]:=a/b;
        Normaliza[A_,B_:0]:=NormalizaCuerpo[A,B];
        UnidadQ[Valor_]:=UnidadCuerpo[Valor];
        MatrizFinal=PasoDecimalRacional[MatrizFinal];
        AnilloOk=True,
(* Else *)
    Message[CargaFunciones::MatrizNoR,Matriz]
],

AnilloBase===C,
(* Se comprueba que los coeficientes son números complejos *)
MatrizEntrada=Flatten[Matriz];
numelementos=Length[MatrizEntrada];
If[
    Length[Select[MatrizEntrada,Element[#1,Complexes]&]]==numelementos,
        MaximoComunDivisor[a_,b_]:=a;
        MaximoComunDivisorEx[a_,b_]:={a,{1,0}};
        DividirQ[a_,b_]:=True;
        Cociente[a_,b_]:=a/b;
        Normaliza[A_,B_:0]:=NormalizaCuerpo[A,B];
        UnidadQ[Valor_]:=UnidadCuerpo[Valor];
        MatrizFinal=PasoDecimalRacional[MatrizFinal];
        AnilloOk=True,
(* Else *)
    Message[CargaFunciones::MatrizNoC,Matriz]
]
,
```

```
Cases[{AnilloBase},Galois[_Integer,_List]]=!={}, (* Cuerpo finito, Galois[p,{a0,...,ar}] *)
    (* Se comprueba que p es un número primo *)
    If[
        PrimeQ[AnilloBase[[1]]],
        (* Se comprueba que {a0,...,ar} está formada por enteros entre 0 y p-1 *)
        If[
            ListaEnteros[AnilloBase],
            (* Los coeficientes a0,...,ar, deben corresponder a un polinomio
            a0+a1*x+...+ar*x^r irreducible sobre el cuerpo de los números
            enteros módulo p *)
            If[
                Irreducible[AnilloBase],
                (* Los elementos de la matriz son de la forma GF[p,{a0,...,ar}][{...}] ? *)
                MatrizEntrada=Flatten[Matriz];
                If[
                    (Count[MatrizEntrada,GF[AnilloBase[[1]],AnilloBase[[2]]][_List]]+
                     Count[MatrizEntrada,0])===Times@@Dimensions[Matriz],
                     (* Se comprueba que los coeficientes de los polinomios
                     son números enteros *)
                     MatrizEntrada=Flatten[Map[PrimerElemento,Matriz,{2}]];
                     If[
                        Count[MatrizEntrada,_Integer]===Length[MatrizEntrada],
                        MaximoComunDivisor[a_,b_]:=a;
                        MaximoComunDivisorEx[a_,b_]:={a,{1,0}};
                        DividirQ[a_,b_]:=True;
                        Cociente[a_,b_]:=a/b;
                        Normaliza[A_,B_:0]:=NormalizaCuerpo[A,B];

                        MatrizIdentidad[n_]:=GF[AnilloBase[[1]],AnilloBase[[2]]][{1}] IdentityMatrix[n];
                        UnidadQ[Valor_]:=UnidadCuerpo[Valor];
                        (* Con este pequeño truco se consigue remmplazar el cero
                        del cuerpo por el cero entero *)
                        MatrizFinal=MatrizFinal+MatrizFinal-MatrizFinal;
                        AnilloOk=True;
                    ,
                    (* Else *)
                        Message[CargaFunciones::CoefsnoEnteros];
                    ];
                ,
```

```
                (* Else *)
                    Message[CargaFunciones::NoenCuerpo,AnilloBase];
                ];
            ,
            (* Else *)
                Message[CargaFunciones::polReducible,AnilloBase[[1]]];
            ];
        ,
        (* Else *)
            Message[CargaFunciones::ErrLista,AnilloBase[[1]]-1];
        ];
    ,
    (* Else *)
        Message[CargaFunciones::pNoPrimo,AnilloBase[[1]]]
    ]
,
(* Anillo de polinomios en una variable sobre un cuerpo finito
 (Galois[p,{a0,...,ar}][X]) *)
Cases[{AnilloBase},Galois[_Integer,_List][_Symbol]]=!={},
    (* Se comprueba que p es un número primo *)
    If[
        PrimeQ[Head[AnilloBase][[1]]],
        (* Se comprueba que {a0,...,ar} está formada por enteros entre 0 y p-1 *)
        If[
            ListaEnteros[Head[AnilloBase]],
            (* Los coeficientes a0,...,ar, deben corresponder a un polinomio
            a0+a1*x+...+ar*x^r irreducible sobre el cuerpo de los números
            enteros módulo p *)
            If[
                Irreducible[Head[AnilloBase]],
                (* Los coeficientes de los polinomios que forman la matriz deben ser de la
                forma GF[p,{a0,...,ar}][{...}] *)
                MatrizEntrada=Flatten[CoefficientList[Flatten[Matriz],AnilloBase[[1]]]];
                If[
                    (Count[MatrizEntrada,GF[Head[AnilloBase][[1]],Head[AnilloBase][[2]]][_List]]+
                     Count[MatrizEntrada,0])===Length[MatrizEntrada],
                     (* Se comprueba que los coeficientes de los polinomios
                     son números enteros *)
                     MatrizEntrada=Flatten[Map[PrimerElemento,MatrizEntrada,{1}]];
                     If[
```

```
                        Count[MatrizEntrada,_Integer]===Length[MatrizEntrada],
                        MaximoComunDivisor[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]][[1]];
                        MaximoComunDivisorEx[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]];
                        DividirQ[a_,b_]:=PolynomialRemainder[b,a,AnilloBase[[1]]]===0;
                        Cociente[a_,b_]:=PolynomialQuotient[a,b,AnilloBase[[1]]];
                        Normaliza[A_,B_:0]:=NormalizaPolinomios[A,AnilloBase[[1]],B];

                        MatrizIdentidad[n_]:=GF[Head[AnilloBase][[1]],Head[AnilloBase][[2]]][{1}] IdentityMatrix[n];
                        UnidadQ[Valor_]:=UnidadPolinomios[Valor,AnilloBase[[1]]];
                        (* Con este truco se consigue reemplazar el cero del anillo
                        por el cero entero *)
                        MatrizFinal=MatrizFinal+MatrizFinal-MatrizFinal;
                        Factorizar[pol_]:=FactGalois[pol,Head[AnilloBase][[1]],Length[Head[AnilloBase][[2]]],varx];
                        AnilloOk=True;
                    ,
                    (* Else *)
                        Message[CargaFunciones::CoefsnoEnteros];
                    ];
                ,
                (* Else *)
                    Message[CargaFunciones::NoenAnilloPol,AnilloBase];
                ];
            ,
            (* Else *)
                Message[CargaFunciones::polReducible,Head[AnilloBase][[1]]];
            ];
        ,
        (* Else *)
            Message[CargaFunciones::ErrLista,Head[AnilloBase][[1]]-1];
        ];
    ,
    (* Else *)
        Message[CargaFunciones::pNoPrimo,Head[AnilloBase][[1]]]
    ]
,
Cases[{AnilloBase},Galois[_Integer,_Integer]]=!={}, (* Cuerpo finito con p^d elementos, Galois[p,d] *)
    (* Se comprueba que p es un número primo *)
    If[
        PrimeQ[AnilloBase[[1]]],
        (* El segundo entero debe ser positivo *)
```

```
    If[
        AnilloBase[[2]]>0,
        (* Los elementos de la matriz son de la forma GF[p,d][{...}] ? *)
        MatrizEntrada=Flatten[Matriz];
        If[
            (Count[MatrizEntrada,GF[AnilloBase[[1]],AnilloBase[[2]]][_List]]+
             Count[MatrizEntrada,0])===Times@@Dimensions[Matriz],
             (* Se comprueba que los coeficientes de los polinomios
             son números enteros *)
             MatrizEntrada=Flatten[Map[PrimerElemento,Matriz,{2}]];
             If[
                Count[MatrizEntrada,_Integer]===Length[MatrizEntrada],
                MaximoComunDivisor[a_,b_]:=a;
                MaximoComunDivisorEx[a_,b_]:={a,{1,0}};
                DividirQ[a_,b_]:=True;
                Cociente[a_,b_]:=a/b;
                Normaliza[A_,B_:0]:=NormalizaCuerpo[A,B];

                MatrizIdentidad[n_]:=GF[AnilloBase[[1]],AnilloBase[[2]]][{1}] IdentityMatrix[n];
                UnidadQ[Valor_]:=UnidadCuerpo[Valor];
                (* Con este pequeño truco se consigue reemplazar el
                cero del cuerpo por el cero entero *)
                MatrizFinal=MatrizFinal+MatrizFinal-MatrizFinal;
                AnilloOk=True;
            ,
            (* Else *)
                Message[CargaFunciones::CoefsnoEnteros];
            ];
        ,
        (* Else *)
            Message[CargaFunciones::NoenCuerpo,AnilloBase];
        ];
    ,
    (* Else *)
        Message[CargaFunciones::dNoPositivo];
    ];
,
(* Else *)
    Message[CargaFunciones::pNoPrimo,AnilloBase[[1]]]
]
```

```
    ,
(* Anillo de polinomios en una variable sobre un cuerpo finito
(Galois[p,d][X]) *)
Cases[{AnilloBase},Galois[_Integer,_Integer][_Symbol]]=!={},
    (* Se comprueba que p es un número primo *)
    If[
        PrimeQ[Head[AnilloBase][[1]]],
        (* Los coeficientes de los polinomios que forman la matriz deben ser de la
        forma GF[p,d][{...}] *)
        MatrizEntrada=Flatten[CoefficientList[Flatten[Matriz],AnilloBase[[1]]]];
        If[
            (Count[MatrizEntrada,GF[Head[AnilloBase][[1]],Head[AnilloBase][[2]]][_List]]+
             Count[MatrizEntrada,0])===Length[MatrizEntrada],
             (* Se comprueba que los coeficientes de los polinomios
             son números enteros *)
             MatrizEntrada=Flatten[Map[PrimerElemento,MatrizEntrada,{1}]];
             If[
                Count[MatrizEntrada,_Integer]===Length[MatrizEntrada],
                MaximoComunDivisor[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]][[1]];
                MaximoComunDivisorEx[a_,b_]:=EuclidesRC[a,b,AnilloBase[[1]]];
                DividirQ[a_,b_]:=PolynomialRemainder[b,a,AnilloBase[[1]]]===0;
                Cociente[a_,b_]:=PolynomialQuotient[a,b,AnilloBase[[1]]];
                Normaliza[A_,B_:0]:=NormalizaPolinomios[A,AnilloBase[[1]],B];

                MatrizIdentidad[n_]:=GF[Head[AnilloBase][[1]],Head[AnilloBase][[2]]][{1}] IdentityMatrix[n];
                UnidadQ[Valor_]:=UnidadPolinomios[Valor,AnilloBase[[1]]];
                (* Con este pequeño truco se consigue reemplazar el cero del
                anillo por el cero entero *)
                MatrizFinal=MatrizFinal+MatrizFinal-MatrizFinal;
                Factorizar[pol_]:=FactGalois[pol,Head[AnilloBase][[1]],Head[AnilloBase][[2]],varx];
                AnilloOk=True;
            ,
            (* Else *)
                Message[CargaFunciones::CoefsnoEnteros];
            ];
        ,
        (* Else *)
            Message[CargaFunciones::NoenAnilloPol,AnilloBase];
        ];
,
```

```
(* Else *)
    Message[CargaFunciones::pNoPrimo,Head[AnilloBase][[1]]]
]
,

{AnilloBase}===Cases[{AnilloBase},Z[_Integer]], (* Anillo Enteros módulo p con primo *)
(* Se comprueba que los coeficientes son todos números enteros *)
MatrizEntrada=Flatten[Matriz];
If[
    Length[Select[MatrizEntrada,Element[#1,Integers]&]]==Length[MatrizEntrada],
        If[
            PrimeQ[AnilloBase[[1]]],
                MaximoComunDivisor[a_,b_]:=a;
                MaximoComunDivisorEx[a_,b_]:={a,{1,0}};
                DividirQ[a_,b_]:=True;
                Cociente[a_,b_]:=a/b;
                Normaliza[A_,B_:0]:=NormalizaCuerpo[A,B];
                (* Se Reformatea la matriz *)
                MatrizFinal=GF[AnilloBase[[1]]][{1}] MatrizFinal;

                MatrizIdentidad[n_]:=GF[AnilloBase[[1]]][{1}] IdentityMatrix[n];
                UnidadQ[Valor_]:=UnidadCuerpo[Valor];
                AnilloOk=True;
        ,
        (* Else *)
            Message[CargaFunciones::pNoPrimo,AnilloBase[[1]]]
        ]
,
(* Else *)
    Message[CargaFunciones::MatrizNoModp,Matriz];
]
,
(* Anillo de los polinomios con coeficientes en el cuerpo de los enteros
módulo p *)
Cases[{AnilloBase},Z[_Integer,_Symbol]]=!={},
(* Se comprueba que todos los polinomios tienen todos sus coeficientes
en el anillo de los enteros *)
MatrizEntrada=Flatten[CoefficientList[Flatten[Matriz],AnilloBase[[2]]]];
If[
    Length[Select[MatrizEntrada,Element[#1,Integers]&]]==Length[MatrizEntrada],
```

```
                    If[
                        PrimeQ[AnilloBase[[1]]],
                            MaximoComunDivisor[a_,b_]:=EuclidesRC[a,b,AnilloBase[[2]]][[1]];
                            MaximoComunDivisorEx[a_,b_]:=EuclidesRC[a,b,AnilloBase[[2]]];
                            DividirQ[a_,b_]:=PolynomialRemainder[b,a,AnilloBase[[2]]]===0;
                            Cociente[a_,b_]:=PolynomialQuotient[a,b,AnilloBase[[2]]];
                            Normaliza[A_,B_:0]:=NormalizaPolinomios[A,AnilloBase[[2]],B];
                            (* Se Reformatea la matriz *)
                            MatrizFinal=GF[AnilloBase[[1]]][{1}] MatrizFinal;
                            MatrizFinal=MatrizFinal+MatrizFinal-MatrizFinal;

                            MatrizIdentidad[n_]:=GF[AnilloBase[[1]]][{1}] IdentityMatrix[n];
                            UnidadQ[Valor_]:=UnidadPolinomios[Valor,AnilloBase[[2]]];
                            (* Para el cálculo de formas canónicas *)
                            Factorizar[pol_]:=FactGalois[pol,AnilloBase[[1]],1,varx];
                            AnilloOk=True;
                    ,
                    (* Else *)
                        Message[CargaFunciones::pNoPrimo,AnilloBase[[1]]]
                    ]
            ,
            (* Else *)
                Message[CargaFunciones::MatrizNoModp,Matriz];
            ]
            ,
            True, (* Caso por defecto *)
            Message[CargaFunciones::AnilloIncorrecto,AnilloBase];
        ]
    ,
    (* Else *)
        Message[CargaFunciones::MatrizErronea,Matriz];
    ];
Return[{AnilloOk,MatrizFinal}];
];

(*
=============== BLOQUE de cálculo del Máximo Común Divisor de dos elementos de Z[(1+Sqrt[-19])/2].
Z[(1+Sqrt[-19])/2] es un dip que no es dominio euclídeo
*)
```

```
(*
=============== FUNCIÓN: dipnodeCociente
Realiza el cociente de dos elementos de Z[(1+Sqrt[-19])/2]. Como hay una parte que siempre es la
misma ((1+Sqrt[-19])/2) se trabaja con pares {a,b}, que en realidad quiere decir
a+b*(1+Sqrt[-19])/2
*)

dipnodeCociente[elalfa_, elbeta_] := Module[
{parterealnum, parteimnum,parterealdor, parteimdor, parterealres, parteimres, vaux},

    parteimnum = Simplify[2*Im[elalfa]/Sqrt[19]];
    parterealnum = Simplify[Re[elalfa] - 1/2*parteimnum];

    parteimdor = Simplify[2*Im[elbeta]/Sqrt[19]];
    parterealdor = Simplify[Re[elbeta] - 1/2*parteimdor];

    vaux =Simplify[parterealdor^2+parterealdor*parteimdor+5*parteimdor^2];

    parterealres =Simplify[(parterealnum*(parterealdor+parteimdor)+5*parteimnum*parteimdor)/vaux];

    parteimres =Simplify[(parteimnum*parterealdor-parterealnum*parteimdor)/vaux];

    Return[{parterealres, parteimres}];

];

(*
=============== FUNCIÓN:dipnodeUnpaso
Proporciona un elemento de Z[(1+Sqrt[-19])/2] de norma menor que los dos de entrada.
Se identifica a+b*(1+Sqrt[-19])/2 con {a,b}
*)

dipnodeUnpaso[elalfa_, elbeta_]:=Module[{elcociente, resultado, vaux, varaux},

    elcociente = dipnodeCociente[elalfa, elbeta];

    Which[
        Element[elcociente, Integers],
         resultado={{0, 0},{1, 0}};
        ,
```

```
Element[elcociente[[2]], Integers],
 resultado={{1,0},(-1)*{Round[elcociente[[1]]],elcociente[[2]]}};
,
Element[elcociente[[1]], Integers],
 If[
   Element[5*elcociente[[2]], Integers],
    resultado={{1,0},(-1)*{elcociente[[1]],Round[elcociente[[2]]]}}
 ,
 (* Else *)
    resultado={{1,-1},(-1)*{Round[elcociente[[1]]+5*elcociente[[2]]],-elcociente[[1]]}};
 ];
,
True,
 vaux=Element[2*elcociente[[1]],Integers];
 varaux=Element[2*elcociente[[2]],Integers];
 Which[
    vaux && !varaux,
      vaux=5*elcociente[[2]];
        Which[
            Element[vaux, Integers],
             resultado={{5,0},(-1)*{Round[5*elcociente[[1]]],vaux}};
            ,
            Element[2*vaux, Integers],
             resultado={{0,5},(-1)*{Round[-25*elcociente[[2]]],5*(Plus@@elcociente)}};
            ,
            True,
             resultado={{2,-2},(-1)*{Round[2*elcociente[[1]]+2*vaux],-2*elcociente[[1]]}};
         ];
     ,
      !vaux && varaux,
       resultado = {{2,0},(-1)*{Round[2*elcociente[[1]]],2*elcociente[[2]]}};
     ,
       vaux && varaux,
        resultado = {{0,1},(-1)*{Round[-5*elcociente[[2]]],Round[Plus@@elcociente]}};
     ,
       True,
          If[
            Abs[Round[elcociente[[2]]]-elcociente[[2]]] <= 1/3,
             resultado={{1,0},(-1)*{Round[elcociente[[1]]],Round[elcociente[[2]]]}};
          ,
```

```
                (* Else *)
                   resultado = {{2,0},(-1)*{Round[2*elcociente[[1]]],Round[2*elcociente[[2]]]}};
                ];
            ];
     ];
 Return[resultado];
];


(*
=============== FUNCIÓN: dipnodeMultiplicacion
Esta función multiplica dos elementos de Z[(1+Sqrt[-19])/2] (pensados como pares).
*)

dipnodeMultiplicacion[{a_, b_}, {c_, d_}] := Module[{vaux},
    vaux=Simplify[{a*c-5*b*d,a*d+b*c+b*d}];
    Return[vaux];
];

(*
=============== FUNCIÓN: dipnodeRecurhastaDiv
Calcula un elemento del ideal <elalfa,elbeta>, que divide a elalfa.
*)
dipnodeRecurhastaDiv[elalfa_,elbeta_]:=Module[{vaux,varaux,elgamma,resultado={0,{0,0}}},

    vaux=dipnodeUnpaso[elalfa,elbeta];
    If[
     vaux=!={{0,0},{1,0}},
      elgamma=Expand[(vaux[[1]].{1,(1+Sqrt[-19])/2})*elalfa+(vaux[[2]].{1,(1+Sqrt[-19])/2})*elbeta];
      varaux=dipnodeRecurhastaDiv[elalfa,elgamma];
      resultado[[1]]=varaux[[1]];

      resultado[[2,1]]=Expand[dipnodeMultiplicacion[varaux[[2,2]],vaux[[1]]]+varaux[[2,1]]];
      resultado[[2,2]]=dipnodeMultiplicacion[varaux[[2,2]],vaux[[2]]];
    ,
    (* Else *)
      resultado={elbeta,{{0,0},{1,0}}};
    ];

    Return[resultado];
```

```
];

(*
=============== FUNCIÓN: dipnodeNormaMinimal
Proporciona un elemento de Norma minimal en los elementos del ideal <elalfa,elbeta>.
*)
dipnodeNormaMinimal[elalfa_, elbeta_] :=Module[
{vaux, varaux, resllam, resultado = {0, {{0, 0}, {1, 0}}}, elgamma,blnContinuar},

    blnContinuar = True;
    elgamma = elbeta;
    While[
         blnContinuar,
          vaux=dipnodeRecurhastaDiv[elalfa,elgamma];
          resultado[[2,1]]=Expand[dipnodeMultiplicacion[vaux[[2,2]],resultado[[2,1]]]+vaux[[2,1]]];
          resultado[[2,2]]=dipnodeMultiplicacion[vaux[[2,2]],resultado[[2,2]]];
          vaux=dipnodeRecurhastaDiv[elbeta,vaux[[1]]];
          resultado[[2,1]]=dipnodeMultiplicacion[vaux[[2,2]],resultado[[2,1]]];
          resultado[[2,2]]=Expand[dipnodeMultiplicacion[vaux[[2,2]],resultado[[2,2]]]+vaux[[2,1]]];
          elgamma = vaux[[1]];
          If[
            Element[dipnodeCociente[elalfa, elgamma], Integers],
             blnContinuar = False;
             resultado[[1]] = elgamma;
          ];

    ];

    Return[resultado];

];

(*
=============== FUNCIÓN : dipnodeMCD
Retorna una lista {d,{a,b}}, tal que:
    · d es el máximo común divisor de elalfa y elbeta.
    · a*elalfa+b*elbeta=d
El anillo de trabajo sigue siendo Z[(1+Sqrt[-19])/2]
*)
```

```
dipnodeMCD[elalfa_, elbeta_] := Module[{resultado, vaux},

    Which[
      elalfa === 0,
       resultado = {elbeta, {{0, 0}, {1, 0}}};
      ,
      elbeta === 0,
       resultado = {elalfa, {{1, 0}, {0, 0}}};
      ,
      True,
       If[
        Simplify[Conjugate[elalfa]*elalfa-Conjugate[elbeta]*elbeta] >= 0,
        resultado=dipnodeNormaMinimal[elalfa,elbeta];
       ,
       (* Else *)
        resultado=dipnodeNormaMinimal[elbeta,elalfa];

        (* Hay que intercambiar el resultado *)
        vaux = resultado[[2,2]];
        resultado[[2,2]]=resultado[[2,1]];
        resultado[[2,1]]:=vaux;
       ];
    ];

    resultado[[2,1]]=resultado[[2,1]].{1,(1+Sqrt[-19])/2};
    resultado[[2,2]]=resultado[[2,2]].{1,(1+Sqrt[-19])/2};

    Return[resultado];

];


(* Las siguientes funciones:
            · PasoDecimalRacional (R,C)
            · ConvertirPolinomio  (R[x],C[x])
    formatean la matriz de entrada, cambiando los números decimales
    por sus fracciones generatrices.


============== FUNCIÓN: PasoDecimalRacional
```

```
*)
PasoDecimalRacional[n_]:=Module[{Resultado,partereal,parteim},

    Resultado=n;
    partereal=Re[Resultado];
    parteim=Im[Resultado];
    If[
        Head[partereal]===Real,
        If[
            partereal>=0,
            (* Se cambia la parte real por su fracción generatriz *)
            partereal=FromDigits[RealDigits[partereal]];
            Resultado=partereal+I*parteim;
        ,
        (* Else *)
            partereal=(-1)*FromDigits[RealDigits[partereal]];
            Resultado=partereal+I*parteim;
        ];
    ];

    If[
        Head[parteim]===Real,
        If[
            parteim>=0,
            (* Se cambia la parte imaginaria por su fracción generatriz *)
            parteim=FromDigits[RealDigits[parteim]];
            Resultado=partereal+I*parteim;
        ,
        (* Else *)
            parteim=(-1)*FromDigits[RealDigits[parteim]];
            Resultado=partereal+I*parteim;
        ];
    ];

    Return[Resultado];
];

Attributes[PasoDecimalRacional]={Listable};

(*
```

```
=============== FUNCIÓN: ConvertirPolinomio
*)
ConvertirPolinomio[Matriz_List,var_]:=Module[
{numfilas,numcolumnas,icontador,jcont,ListaCoefs,Resultado,grado,kconta},

    {numfilas,numcolumnas}=Dimensions[Matriz];
    Resultado=Matriz;
    Do[
        Do[
            ListaCoefs=CoefficientList[Resultado[[icontador,jcont]],var];
            grado=Length[ListaCoefs]-1;
            ListaCoefs=PasoDecimalRacional[ListaCoefs];
            Resultado[[icontador,jcont]]=ListaCoefs.Table[var^kconta,{kconta,0,grado}];
        ,
        {jcont,1,numcolumnas}];
    ,
    {icontador,1,numfilas}];

    Return[Resultado];
];


Begin["`Private`"]

(*
=============== FUNCIÓN: PrimerElemento
*)
PrimerElemento[x_]:=Module[{},
    If[
        Length[x]===0,
        Return[x];
    ,
    (* Else *)
        Return[x[[1]]];
    ];
];

(*
=============== FUNCIÓN: ListaEnteros
Dado AnilloTrabajo, que va a ser de la forma Galois[p,ListaCoefIrr], comprueba
```

```
si todos los elementos de ListaCoefIrr son enteros y están comprendidos entre
0 y p-1; de serlo, devuelve True, si no False.
*)
ListaEnteros[AnilloTrabajo_]:=Module[{Resultado,numelementos,icontador},

    Resultado=True;
    numelementos=Length[AnilloTrabajo[[2]]];
    Do[
        If[
            AnilloTrabajo[[2,icontador]]<0 ||
            AnilloTrabajo[[2,icontador]]>=AnilloTrabajo[[1]] ||
            !Element[AnilloTrabajo[[2,icontador]],Integers],
            Resultado=False;
            Break[];
        ];
    ,
    {icontador,1,numelementos}];

    Return[Resultado];
];

(*
=============== FUNCIÓN: Irreducible
Dado AnilloTrabajo, que va a ser de la forma Galois[p,ListaCoefIrr], comprueba si
ListaCoefIrr, son los coeficientes, ordenados de menor grado a mayor grado, de un
polinomio irreducible sobre el cuerpo de los enteros módulo p.
*)
Irreducible[AnilloTrabajo_]:=Module[{icontador,jcont,n,Resultado,valor},

    Resultado=True;
    n=Length[AnilloTrabajo[[2]]];
    Do[
        valor=Join[{1},Table[icontador^jcont,{jcont,1,n-1}]].AnilloTrabajo[[2]];
        valor=Mod[valor,AnilloTrabajo[[1]]];
        If[
            valor===0 && n>2,
            Resultado=False;
            Break[];
        ];
    ,
```

```
    {icontador,0,AnilloTrabajo[[1]]-1}];

    Return[Resultado];
];

(*
=============== FUNCIÓN: EuclidesRC
*)
EuclidesRC[f_,g_,var_]:=Module[{maximocd,alfa,bet,vaux},

    {maximocd,{alfa,bet}}=CalculoEuclides[f,g,var];

    (* Se hace mónico maximocd *)
    {maximocd,{alfa,bet}}=Simplify[
                                    Expand[
                                            1/Coefficient[maximocd,var,Exponent[maximocd,
                                                                                var]
                                                          ]*{maximocd,{alfa,bet}}
                                        ]
                                    ];
    Return[{maximocd,{alfa,bet}}];
];

(*
=============== FUNCIÓN: CalculoEuclides
PolynomialExtendedGCD no funciona sobre R[x] y C[x] (falla en los decimales), es por esto que
se implementa CalculoEuclides. Dados dos polinomios f,g en una variable sobre R ó C, se
devuelve una lista {d,{a,b}}, tal que:
            · d=máximocomúndivisor(f,g);
            · a·f+b·g=d
*)
CalculoEuclides[f_,g_,var_]:=Module[{cocien,resto,maxcd,pol1,pol2,vaux},

cocien=FullSimplify[PolynomialQuotient[f,g,var]];
resto=FullSimplify[PolynomialRemainder[f,g,var]];

If[
    resto===0,
    Return[{g,{0,MatrizIdentidad[1][[1,1]]}}];
    (*Return[{g,{0,1}}];*)
```

```
  ,
  (* Else *)
    {maxcd,{pol1,pol2}}=CalculoEuclides[g,resto,var];
    vaux=pol1;
    pol1=pol2;
    pol2=Simplify[Expand[pol2*cocien*(-1)+vaux]];
    Return[{maxcd,{pol1,pol2}}];
  ];
];

(*
=============== FUNCIÓN: MCD
MCD[{e1, ..., en}] calcula recursivamente el máximo común divisor de varios elementos no
nulos del dip seleccionado por el usuario. El formato de salida es {{f1,...,fn},g} donde:
        · g es el máximo común divisor de {e1, ...,en}.
        · {e1, ...,en}.{f1, ...,fn}=g
*)
MCD[Coefs_List]:=Module[{Resultado,Maxcd,longCoefs},
    longCoefs=Length[Coefs];
    Which[
            longCoefs > 1,
                Maxcd=MaximoComunDivisorEx[Coefs[[1]],Coefs[[2]]];
                Resultado=MCD[Join[{Maxcd[[1]]},Drop[Coefs,2]]];
                Resultado[[1,1]]=Expand[Resultado[[1,1]] Maxcd[[2]]];
                Resultado[[1]]=Flatten[Resultado[[1]]];
            ,
            longCoefs ==1,
                Resultado={{1},Coefs[[1]]};
            ,
            longCoefs==0,
                Resultado={};
    ];
    Return[Resultado];
];

(*
=============== FUNCIÓN: MCDenLista
Dada una lista m de elementos no nulos, MCDenLista busca si en m hay un elemento
que divida a todos los demás. De haberlo, devuelve su posición, si no lo hay
devuelve {}, y si hay más de uno devuelve la posición del primero.
```

```
*)
MCDenLista[m_List]:=Module[{longm,icontador,jcont,Resultado={},valorfinal=0},
    longm=Length[m];

    Do[
        Do[
            If[DividirQ[m[[icontador]],m[[jcont]]],
                        valorfinal=jcont,
                        Break[]
            ],
        {jcont,1,longm}];
        If[valorfinal==longm,
            Resultado={icontador};
            Break[]
        ],
    {icontador,1,longm}];

    Return[Resultado];
];

(*
=============== FUNCIÓN: Sicigia
Esta función calcula la sicigia de dos polinomios homogéneos cuya potencia principal
es var.
*)
Sicigia[f_,g_,var_]:=Module[{Maxcd,Resultado,coeff,coeficienteg},
    coeff=Coefficient[f,var];
    coeficienteg=Coefficient[g,var];

    Maxcd=MaximoComunDivisor[coeff,coeficienteg];

    (* Hay que utilizar la función Cociente, pues se dan casos en los que
    el comando Simplify por sí sólo no simplifica *)

    Resultado=Simplify[Expand[Cociente[coeficienteg,Maxcd]*f-Cociente[coeff,Maxcd]*g]];

    Return[Resultado];
];

(*
```

```
=============== FUNCIÓN: ReducirPolinomio
Este procedimiento es el encargado de la reducción de un polinomio homogéneo y de grado
1, respecto a un conjunto de polinomios homogéneos y de grado 1.
Las variables se suponen ordenadas según su posición en Vars.
*)
ReducirPolinomio[Dividendo_,Divisor_List,Vars_List]:=
Module[{pol,lp,cocientePol,iacum,contj,numvars,numdivisor,Q,CoefDivisor,CoefDividendo,
Maxcd,Posdivisor,varsdivisor,varspol,PolsEntre,varsnopresentes,longPosdivisor},

    pol=Dividendo;
    numvars=Length[Vars];
    numdivisor=Length[Divisor];
    Q=Table[0,{numdivisor}];
    varspol=Variables[pol];
    Do[
        (* lp contendrá la potencia principal del polinomio dividendo. Las variables
        se suponen ordenadas según su posición en Vars *)
        lp=Intersection[{Vars[[iacum]]},varspol];
        If[
            lp =!={},
                PolsEntre={};
                Posdivisor={};
                varsnopresentes=Take[Vars,iacum-1];
                Do[
                    varsdivisor=Variables[Divisor[[contj]]];
                    If[
                        Intersection[lp,varsdivisor]===lp &&
                        Intersection[varsnopresentes,varsdivisor]==={},
                        PolsEntre=Join[PolsEntre,{Divisor[[contj]]}];
                        Posdivisor=Join[Posdivisor,{contj}]
                    ],
                {contj,1,numdivisor}];
                If[
                    PolsEntre=!={},
                    CoefDivisor=Coefficient[PolsEntre,lp[[1]]];
                    CoefDividendo=Coefficient[pol,lp[[1]]];
                    Maxcd=MCD[CoefDivisor];
                    If[
                        DividirQ[Maxcd[[2]],CoefDividendo],
                        (* Se puede dividir *)
```

```
                    cocientePol=Cociente[CoefDividendo,Maxcd[[2]]];
                    pol=Simplify[Expand[pol-cocientePol*Maxcd[[1]].PolsEntre]];

                    longPosdivisor=Length[Posdivisor];
                    Do[
                        Q[[Posdivisor[[contj]]]]=Expand[Q[[Posdivisor[[contj]]]]+cocientePol*
                                                        Maxcd[[1,contj]]],
                    {contj,1,longPosdivisor}],
                (* Else *)
                    Break[];
                ],
            (* Else *)
                Break[]
            ];
        ],
    {iacum,1,numvars}];

    Return[{Q,pol}];
    ];


(*
=============== FUNCIÓN: MatrizEquivalente
Esta función calcula una matriz equivalente a una matriz dada, mediante el cálculo parcial
de una base de Groebner. Sólo se realizarán los cálculos que conducirán a polinomios lineales.
*)

MatrizEquivalente[Matriz_List]:=
Module[{numvars,numpols,varsIdeal,icontador,jcont,kconta,hcon,lp,longI,PolRed,Maxcd,Resultado,
Ideal},

    {numpols,numvars}=Dimensions[Matriz];
    varsIdeal=Table[x[icontador],{icontador,1,numvars}];
    Ideal=Expand[Matriz.varsIdeal];

    (* El papel que en la teoría juegan los Ik, aquí lo desempeña PolsI.
    Primero se inicializan a {} *)
    Do[
        PolsI[x[icontador]]={};
    ,
```

```
{icontador,1,numvars}];

(* Se construyen los PolsI *)
Do[
    lp=Intersection[Variables[Ideal[[icontador]]],varsIdeal];

    (* Esta condición es para prevenir en el caso de que en la matriz
        haya una fila de ceros. *)
    If[
        lp=!={},
        PolsI[lp[[1]]]=Join[PolsI[lp[[1]]],{Ideal[[icontador]]}];
    ],
{icontador,1,numpols}];

(* Se calculan las sicigias, para ello se buscan los PolsIk con dos ó más
elementos. Hay que procesar todos los bloques PolsIk excepto el último, y
dentro de cada uno, hay que calcular todas las sicigias.
El último bloque (PolsIn) no es necesario procesarlo, ya que está formado
por monomios, y la sicigia de dos monomios da siempre cero; de ahí que el
bucle vaya de 1 hasta numvars-1
*)

Do[
    longI=Length[PolsI[x[icontador]]];
    If[
        longI>=2,
        (* Sólo se obtendrán polinomios lineales de los PolsIk con dos
        ó más elementos *)
        Do[
            Do[
                PolRed=ReducirPolinomio[
                        Sicigia[
                                PolsI[x[icontador]][[jcont]],
                                PolsI[x[icontador]][[kconta]],
                                x[icontador]
                                ],
                        Flatten[Table[PolsI[x[hcon]],{hcon,icontador,numvars}]],
                        varsIdeal];

                If[
```

```
                    PolRed[[2]]=!=0,
                    (* Se añade PolRed a su correspondiente PolsI *)
                    lp=Intersection[Variables[PolRed[[2]]],varsIdeal][[1]];
                    PolsI[lp]=Join[PolsI[lp],{PolRed[[2]]}];
                ],
            {kconta,jcont+1,longI}],
        {jcont,1,longI-1}]
    ],
{icontador,1,numvars-1}];

(* Cálculo de la Base de Groebner Fuerte *)
Do[
    longI=Length[PolsI[x[icontador]]];

    If[
        longI>=2,

        lp=Table[Coefficient[PolsI[x[icontador]][[jcont]],
                             {x[icontador]}],
                {jcont,1,longI}];

        lp=Flatten[lp];

        Resultado=MCDenLista[lp];

        If[
            Resultado===={},
            Maxcd=MCD[lp];

            PolsI[x[icontador]]=Simplify[Expand[Apply[Plus,
                                          Maxcd[[1]] PolsI[x[icontador]]]]],
        (* Else Restricción {C1} impuesta en la Teoría *)
            PolsI[x[icontador]]=PolsI[x[icontador]][[Resultado[[1]]]]
        ]
    ],
{icontador,1,numvars}];

(* Este procedimiento devuelve una matriz equivalente a la inicial *)
Resultado=Flatten[Table[PolsI[x[icontador]],{icontador,1,numvars}]];
```

```
    If[Length[Resultado]===0,
        (* Caso en el que la matriz equivalente es la matriz nula *)
        Resultado={{}};
    ,
    (* Else *)
        Resultado=Transpose[Table[Coefficient[Resultado,x[icontador]],
                            {icontador,1,numvars}]];
    ];

    (* Esta función debe devolver una matriz con las mismas dimensiones
    de la matriz de entrada. *)
    Resultado=PadRight[Resultado,{numpols,numvars}];

    Return[Resultado];

];

(*
=============== FUNCIÓN: CompruebaOpcion
Esta función comprueba que el argumento Opcion es de una de estas dos formas:
        Paso -> Sí ó Paso -> No
De ser así devuelve True, si no False.
Este procedimiento se utiliza en las funciones GroebnerEquivalente y Smith.
*)

CompruebaOpcion[Opcion_]:=Module[{},

    If[
        ToString[Opcion[[1]]]===ToString["Paso"] &&
        (ToString[Opcion[[2]]]===ToString["Sí"] ||
         ToString[Opcion[[2]]]===ToString["No"]),
        (* Opción Correcta *)
        Return[True];
    ,
    (* Else *)
        Return[False];
    ];
];

(*
```

```
============== FUNCIÓN: MatrizEquivalentePaso
Dada una matriz A, esta función devuelve un par {B,P}, donde:
        · B es una matriz equivalente a A.
        · P es una matriz inversible tal que B=P·A.
*)

MatrizEquivalentePaso[Matriz_List]:=Module[{MatrizNueva,numfilas,numcols},

    {numfilas,numcols}=Dimensions[Matriz];
    (* Se añade la matriz identidad de orden numfilas a Matriz *)
    MatrizNueva=Transpose[Join[Transpose[Matriz],MatrizIdentidad[numfilas]]];

    MatrizNueva=MatrizEquivalente[MatrizNueva];

    Return[{MatrizNueva[[All,Range[1,numcols]]] (* Matriz Equivalente a la dada *)
            ,
            MatrizNueva[[All,Range[numcols+1,numcols+numfilas]]]} (* Matriz Paso*)
    ];
];

(*
============== FUNCIÓN: GroebnerEquivalente
Calcula una matriz equivalente a una dada, mediante el cálculo parcial de una Base de Groebner.
    GroebnerEquivalente[Matriz,AnilloBase] devuelve sólo una matriz equivalente a la inicial.
    GroebnerEquivalente[Matriz,AnilloBase,Paso->Sí] devuelve {A,P}, siendo:
            ·A una matriz equivalente a la inicial
            ·P matriz inversible
            ·A=P.Matriz
*)

GroebnerEquivalente[Matriz_List,AnilloBase_,Opcion_:0]:=Module[
{AnilloOk,MatrizNueva},

    {AnilloOk,MatrizNueva}=CargaFunciones[AnilloBase,Matriz];
    If[AnilloOk,
        (* Se comprueba que la matriz sea no nula *)
        If[Length[Select[Flatten[Matriz],#1=!=0 &]]>0,
            If[Opcion=!=0,
                If[OptionQ[Opcion],
                    If[CompruebaOpcion[Opcion],
```

```
                    If[ToString[Opcion[[2]]]===ToString["Sí"],
                        Return[MatrizEquivalentePaso[MatrizNueva]];
                    ,
                    (* Else *)
                        Return[MatrizEquivalente[MatrizNueva]];
                    ]
                ,
                (* Else *)
                    Message[GroebnerEquivalente::noOpcionOk,Opcion];
                ]
            ,
            (* Else *)
                Message[GroebnerEquivalente::NoOpcion,Opcion];
            ]
        ,
        (* Else *)
            (* Opción por defecto: Paso-> No *)
            Return[MatrizEquivalente[MatrizNueva]];
        ]
    ,
    (* Else Caso de una Matriz Nula *)
        Return[Matriz];
    ]
,
(* Else *)
    Message[GroebnerEquivalente::ErrDatos];
]
];

(*
=============== FUNCIÓN: ParEquivalente
Dada una matriz, calcula su par equivalente.
*)

ParEquivalente[Matriz_List]:=Module[{Resultado},

    Resultado=MatrizEquivalente[Matriz];

    Resultado=MatrizEquivalente[Transpose[Resultado]];
```

```
    Resultado=Transpose[Resultado];
    Return[Resultado];
];

(*
=============== FUNCIÓN: ParEquivalentePaso
Dada una matriz A, calcula su par equivalente y las matrices de paso.
Esta función devuelve una lista {B,P,Q}, siendo:
        ·B una matriz equivalente a A
        ·P,Q matrices inversibles
        ·B=P.A.Q
*)

ParEquivalentePaso[Matriz_List]:=Module[{Resultado,MFila,MColumna},

    Resultado=MatrizEquivalentePaso[Matriz];
    MFila=Resultado[[2]];
    Resultado=Transpose[Resultado[[1]]];

    Resultado=MatrizEquivalentePaso[Resultado];
    MColumna=Transpose[Resultado[[2]]];
    Resultado=Transpose[Resultado[[1]]];

    Return[{Resultado,MFila,MColumna}];
];

(*
=============== FUNCIÓN: Diagonal
Dada una matriz A, calcula una matriz diagonal y equivalente a A.
*)

Diagonal[Matriz_List]:=Module[{numfilas,numcolumnas,minimo,numceros,diagonal,icontador,Resultado},

    {numfilas,numcolumnas}=Dimensions[Matriz];
    minimo=Min[numfilas,numcolumnas];
    numceros=Count[Flatten[Matriz],0];
    diagonal=Select[Table[Matriz[[icontador,icontador]],
                        {icontador,1,minimo}],
                    #1=!=0&];
    Resultado=Matriz;
```

```
    While[
            Length[diagonal]+numceros!=numfilas*numcolumnas,
            Resultado=ParEquivalente[Resultado];

            numceros=Count[Flatten[Resultado],0];
            diagonal=Select[Table[Resultado[[icontador,icontador]],
                                {icontador,1,minimo}],
                            #1=!=0&];
        ];

    Return[Resultado];
];

(*
=============== FUNCIÓN: DiagonalPaso
Dada una matriz A, esta función devuelve una lista {B,P,Q}, siendo:
        ·B una matriz Equivalente a A y Diagonal
        ·P,Q matrices inversibles
        ·B=P.A.Q
*)

DiagonalPaso[Matriz_List]:=Module[
{numfilas,numcolumnas,minimo,numceros,diagonal,icontador,Resultado,MFila,MColumna},

    {numfilas,numcolumnas}=Dimensions[Matriz];
    minimo=Min[numfilas,numcolumnas];
    (* Se inicializan MFila y MColumna *)
    MFila=MatrizIdentidad[numfilas];
    MColumna=MatrizIdentidad[numcolumnas];
    numceros=Count[Flatten[Matriz],0];
    diagonal=Select[Table[Matriz[[icontador,icontador]],
                        {icontador,1,minimo}],
                    #1=!=0&];
    Resultado=Matriz;

    While[
            Length[diagonal]+numceros!=numfilas*numcolumnas,
            Resultado=ParEquivalentePaso[Resultado];
            MFila=Expand[Resultado[[2]].MFila];
```

```
            MColumna=Expand[MColumna.Resultado[[3]]];
            Resultado=Resultado[[1]];
            numceros=Count[Flatten[Resultado],0];
            diagonal=Select[Table[Resultado[[icontador,icontador]],
                                {icontador,1,minimo}],
                            #1=!=0&];

        ];

    Return[{Resultado,MFila,MColumna}];
];

(*
=============== FUNCIÓN: NoesSmith
Comprueba si una matriz diagonal es la Forma Normal de Smith.
El argumento de entrada minimo es la menor de las dimensiones de la matriz.
*)

NoesSmith[Matriz_List,minimo_]:=Module[{diagonal,icontador,menor,Resultado},

    diagonal=Select[Table[Matriz[[icontador,icontador]],
                        {icontador,1,minimo}],
                    #1=!=0&];
    icontador=Length[diagonal];
    menor=MCDenLista[diagonal];

    While[
            menor=!={},
            icontador=icontador-1;
            diagonal=Drop[diagonal,menor[[1]]];
            menor=MCDenLista[diagonal];
    ];
    If[
        icontador!=0,
        (* La matriz no es la Forma Normal de Smith *)
        Resultado=True;
    ,
    (* Else *)
        Resultado=False;
    ];
```

```
    Return[Resultado];
];


(*
=============== FUNCIÓN: FormaNormal
Calcula la Forma Normal de Smith de una matriz
*)

FormaNormal[Matriz_List]:=Module[{Resultado,icont,jcontador,numfilas,numcolumnas,minimo,P},

    {numfilas,numcolumnas}=Dimensions[Matriz];
    minimo=Min[numfilas,numcolumnas];

    Resultado=Diagonal[Matriz];

    If[
        NoesSmith[Resultado,minimo],
        Do[
            (* Con este pequeño truco de igualar al elemento (1,1), se consigue tener la matriz P
            adecuada, con independencia de si se trabaja en característica cero ó no *)
            P=MatrizIdentidad[numcolumnas];
            Do[
                P[[jcontador,icont]]=P[[1,1]],
            {jcontador,icont+1,minimo}];
            Resultado=Expand[Resultado.P];
            Resultado=Diagonal[Resultado];

            If[
                !NoesSmith[Resultado,minimo],
                (* Si ya se alcanzó la Forma Normal de Smith, se sale del bucle *)
                Break[];
            ];
        ,{icont,1,minimo-1}];
    ];

    Return[Resultado];
];
```

```
(*
=============== FUNCIÓN: FormaNormalPaso
Dada una matriz A, esta función devuelve una lista {B,P,Q}, siendo:
        ·B la Forma Normal de Smith de A
        ·P,Q matrices inversibles
        ·B=P.A.Q
*)

FormaNormalPaso[Matriz_List]:=Module[
{MDiagonal,icont,jcontador,numfilas,numcolumnas,minimo,P,MFilas,MColumnas},

    {numfilas,numcolumnas}=Dimensions[Matriz];
    minimo=Min[numfilas,numcolumnas];

    {MDiagonal,MFilas,MColumnas}=DiagonalPaso[Matriz];

    If[
        NoesSmith[MDiagonal,minimo],
        Do[
            (* Con este pequeño truco de igualar al elemento (1,1), se consigue tener la matriz
            adecuada, con independencia de si se trabaja en característica cero ó no *)
            P=MatrizIdentidad[numcolumnas];
            Do[
                P[[jcontador,icont]]=P[[1,1]];
            ,{jcontador,icont+1,minimo}];
            MDiagonal=Expand[MDiagonal.P];
            MColumnas=Expand[MColumnas.P];
            MDiagonal=DiagonalPaso[MDiagonal];
            MFilas=Expand[MDiagonal[[2]].MFilas];
            MColumnas=Expand[MColumnas.MDiagonal[[3]]];
            MDiagonal=MDiagonal[[1]];
            If[
                !NoesSmith[MDiagonal,minimo],
                (* Si ya se alcanzó la Forma Normal de Smith, se sale del bucle *)
                Break[];
            ],
        {icont,1,minimo-1}];
    ];
    Return[{MDiagonal,MFilas,MColumnas}];
```

```
];


(*
=============== FUNCIÓN: Smith
Calcula la Forma Normal de Smith de una matriz.
    Smith[Matriz,AnilloBase] devuelve sólo la Forma Normal de Smith.
    Smith[Matriz,AnilloBase,Paso->Sí] devuelve {S,P,Q}, siendo:
            ·S la Forma Normal de Smith de Matriz
            ·P,Q matrices inversibles
            ·S=P.Matriz.Q
*)

Smith[Matriz_List,AnilloBase_,Opcion_:0]:=Module[{AnilloOk,MatrizNueva,MFilas,MColumnas},

    {AnilloOk,MatrizNueva}=CargaFunciones[AnilloBase,Matriz];
    If[
        AnilloOk,
        (* Se comprueba que la matriz sea no nula *)
        If[Length[Select[Flatten[Matriz],#1=!=0 &]]>0,
            If[
                Opcion=!=0,
                If[
                    OptionQ[Opcion],
                    If[
                        CompruebaOpcion[Opcion],
                        If[
                            ToString[Opcion[[2]]]===ToString["Sí"],
                            {MatrizNueva,MFilas,MColumnas}=FormaNormalPaso[MatrizNueva];
                            {MatrizNueva,MColumnas}=Normaliza[MatrizNueva,MColumnas];
                            Return[{MatrizNueva,MFilas,MColumnas}];
                        ,
                        (* Else *)
                            Return[Normaliza[FormaNormal[MatrizNueva]]];
                        ]
                    ,
                    (* Else *)
                        Message[Smith::noOpcionOk,Opcion];
                    ]
                ,
```

```
                    (* Else *)
                        Message[Smith::NoOpcion,Opcion];
                    ]
                ,
                (* Else *)
                    (* Opción por Defecto: Paso -> No *)
                    Return[Normaliza[FormaNormal[MatrizNueva]]];
                ]
            ,
            (* Else Caso Matriz Nula *)
                Return[Matriz];
             ]
        ,
        (* Else *)
            Message[Smith::ErrDatos];
        ]
    ];

(*
=============== BLOQUE Normaliza_
Las rutinas FormaNormal y FormaNormalPaso, devuelven la Forma
Normal de Smith de una matriz dada. Para unificar su salida, antes
de mostrar el resultado al usuario, se procede a reformatearla.
Las siguientes cuatro funciones, se utilizan para este fin. Cada
anillo utiliza una función de las siguientes. *)
    (*
    =============== FUNCIÓN: NormalizaEnteros
    Esta función reformatea la Forma Normal de Smith para Z.
    La matriz que devuelve tiene todos los elementos diagonales positivos
    *)

    NormalizaEnteros[Matriz_List,MColumnas_:0]:=Module[
    {Resultado,MatrizCols,icont,minimo},

        minimo=Min@@Dimensions[Matriz];
        Resultado=Matriz;
        MatrizCols=MColumnas;

        If[
            MatrizCols===0,
```

```
        (* Caso en el que se escogió la opción Paso->No *)
        Resultado=Abs[Resultado];
    ,
    (* Else *)
        (* Caso en el que se escogió la opción Paso->Sí *)
        Do[
            If[
                Resultado[[icont,icont]]<0,
                (* Se cambia a positivo *)
                Resultado[[icont,icont]]=(-1)*Resultado[[icont,icont]];
                (* Se modifica MatrizCols, para llevar cuenta del
                cambio de signo *)
                MatrizCols[[All,icont]]=(-1)*MatrizCols[[All,icont]];
            ],
        {icont,1,minimo}];
        Resultado={Resultado,MatrizCols};
    ];
    Return[Resultado];
];


(*
=============== FUNCIÓN: NormalizaEnterosGauss
Esta función reformatea la Forma Normal de Smith para Z[i].
Se cambia I,-I,-1 por 1 en la matriz original y el resto de los elementos permanecen iguales.
*)

NormalizaEnterosGauss[Matriz_List,MColumnas_:0]:=Module[{Resultado,MatrizCols,icont,minimo},

    minimo=Min@@Dimensions[Matriz];
    Resultado=Matriz;
    MatrizCols=MColumnas;

    If[
        MatrizCols===0,
        (* Caso en el que se escogió la opción Paso->No *)
        Do[
            Which[
                    Resultado[[icont,icont]]===-1,
                    Resultado[[icont,icont]]=1;
```

```
                    ,
                    Resultado[[icont,icont]]===I,
                    Resultado[[icont,icont]]=1;
                    ,
                    Resultado[[icont,icont]]===-I,
                    Resultado[[icont,icont]]=1;
            ];
        ,{icont,1,minimo}];
    ,
    (* Else *)
        (* Caso en el que se escogió la opción Paso->Sí *)
        Do[
            Which[
                    Resultado[[icont,icont]]===-1,
                    Resultado[[icont,icont]]=1;
                    (* Se modifica MatrizCols, para llevar cuenta del cambio *)
                    MatrizCols[[All,icont]]=(-1)*MatrizCols[[All,icont]];
                    ,
                    Resultado[[icont,icont]]===I,
                    Resultado[[icont,icont]]=1;
                    (* Se modifica MatrizCols, para llevar cuenta del cambio *)
                    MatrizCols[[All,icont]]=(-I)*MatrizCols[[All,icont]];
                    ,
                    Resultado[[icont,icont]]===-I,
                    Resultado[[icont,icont]]=1;
                    (* Se modifica MatrizCols, para llevar cuenta del cambio *)
                    MatrizCols[[All,icont]]=I*MatrizCols[[All,icont]];
            ];
        ,{icont,1,minimo}];
        Resultado={Resultado,MatrizCols};
    ];
    Return[Resultado];
];

(*
=============== FUNCIÓN: NormalizaCuerpo
Esta función reformatea la Forma Normal de Smith para Q,R,C y Z[p].
Se cambia cada elemento no nulo de la diagonal principal por la unidad del anillo.
*)
```

```
NormalizaCuerpo[Matriz_List,MColumnas_:0]:=Module[{Resultado,MatrizCols,icont,minimo},

    minimo=Min@@Dimensions[Matriz];
    Resultado=Matriz;
    MatrizCols=MColumnas;

    If[
        MatrizCols===0,
        (* Caso en el que se escogió la opción Paso->No *)
        Do[
            If[
                Resultado[[icont,icont]]=!=0,
                (* Se cambia por la unidad.*)
                Resultado[[icont,icont]]= Resultado[[icont,icont]]/Resultado[[icont,icont]];
                (* Se pone Resultado[[icont,icont]]/Resultado[[icont,icont]] en lugar de 1, pues
                esta función también se utilizará para Z[p]. *)
            ],
        {icont,1,minimo}];
    ,
    (* Else *)
        (* Caso en el que se escogió la Opción Paso->Sí *)
        Do[
            If[
                Resultado[[icont,icont]]=!=0,
                (* Se modifica MatrizCols, para llevar cuenta del cambio *)
                MatrizCols[[All,icont]]=1/Resultado[[icont,icont]]*MatrizCols[[All,icont]];
                (* Se cambia por la unidad *)
                Resultado[[icont,icont]]=Resultado[[icont,icont]]/Resultado[[icont,icont]];
                (* Se pone Resultado[[icont,icont]]/Resultado[[icont,icont]] en lugar de 1, pues
                esta función también se utilizará para Z[p]. *)
            ];
        ,{icont,1,minimo}];
        Resultado={Resultado,MatrizCols};
    ];

    Return[Resultado];
];

(*
============== FUNCIÓN: NormalizaPolinomios
```

```
Esta función reformatea la Forma Normal de Smith para Q[x],R[x], C[x] y Galois[...][...][x].
Se cambian las constantes no nulas de la diagonal principal por 1 y se hacen mónicos los polinomios.
*)

NormalizaPolinomios[Matriz_List,Var_,MColumnas_:0]:=Module[
{Resultado,MatrizCols,icont,minimo,coefprincipal,elemneutro},

    minimo=Min@@Dimensions[Matriz];
    Resultado=Matriz;
    MatrizCols=MColumnas;
    elemneutro=MatrizIdentidad[1][[1,1]];

    If[
        MColumnas===0,
        (* Caso en el que se escogió la opción Paso->No *)
        Do[

            (* Exponent no expande las expresiones antes
            de actuar. Exponent[(10-3 x) GF[3][{1}],x]
            devuelve 1, lo cual no es correcto. *)
            Resultado[[icont,icont]]=Expand[Resultado[[icont,icont]]];

            Which[
                    Resultado[[icont,icont]]===elemneutro,
                    Continue[];
                    ,
                    Exponent[Resultado[[icont,icont]],Var]===0,
                    (* Resultado[[icont,icont]] = constante *)
                    Resultado[[icont,icont]]=elemneutro;
                    ,
                    Exponent[Resultado[[icont,icont]],Var]>0,
                    (* Resultado[[icont,icont]]=polinomio de grado mayor ó igual que uno *)
                    coefprincipal=Coefficient[Resultado[[icont,icont]],
                                              Var,
                                              Exponent[Resultado[[icont,icont]],Var]
                                             ];
                    Resultado[[icont,icont]]=Expand[1/coefprincipal*Resultado[[icont,icont]]];
            ],
        {icont,1,minimo}];
    ,
```

```
    (* Else *)
        (* Caso en el que se escogió la Opción Paso->Sí *)
        Do[
            (* Exponent no expande las expresiones antes
            de actuar. Exponent[(10-3 x) GF[3][{1}],x]
            devuelve 1, lo cual no es correcto. *)
            Resultado[[icont,icont]]=Expand[Resultado[[icont,icont]]];

            Which[
                    Resultado[[icont,icont]]===elemneutro,
                    Continue[];
                    ,
                    Exponent[Resultado[[icont,icont]],Var]===0,
                    (* Resultado[[icont,icont]]=constante *)
                    (* Se modifica MatrizCols, para llevar cuenta del cambio *)
                    MatrizCols[[All,icont]]=Expand[1/Resultado[[icont,icont]]*MatrizCols[[All,icont]]];
                    Resultado[[icont,icont]]=elemneutro;
                    ,
                    Exponent[Resultado[[icont,icont]],Var]>0,
                    (* Resultado[[icont,icont]]=polinomio de grado mayor ó igual que 1 *)
                    coefprincipal=Coefficient[Resultado[[icont,icont]],
                                              Var,
                                              Exponent[Resultado[[icont,icont]],Var]
                                             ];
                    MatrizCols[[All,icont]]=Expand[1/coefprincipal*MatrizCols[[All,icont]]];
                    Resultado[[icont,icont]]=Expand[1/coefprincipal*Resultado[[icont,icont]]];
            ],
        {icont,1,minimo}];
        Resultado={Resultado,MatrizCols};
    ];

    Return[Resultado];
];

(*
=============== Unidad___
Para calcular la inversa de una matriz utilizando las matrices de paso, es necesario saber si un
elemento de un dip dado, es unidad ó no. Este es el objetivo de las funciones de este bloque. El
criterio para saber si un elemento es unidad ó no varía de un anillo a otro; es por esto que se
programaron estas cuatro funciones:
```

```
*)

(*
=============== FUNCIÓN: UnidadEnteros
Función booleana que devuelve True si Valor es una unidad en Z.
*)

UnidadEnteros[Valor_]:=Module[{},
    If[
        Valor===-1 || Valor===1,
        Return[True];
    ,
    (* Else *)
        Return[False];
    ];
];


(*
=============== FUNCIÓN: UnidadEnterosGauss
Función booleana que devuelve True si Valor es una unidad en Z[i].
*)

UnidadEnterosGauss[Valor_]:=Module[{},
    If[
        Valor===1 || Valor===-1 ||
        Valor===I || Valor===-I,
        Return[True];
    ,
    (* Else *)
        Return[False];
    ];
];


(*
=============== FUNCIÓN: UnidadCuerpo
Función booleana que devuelve True si Valor es una unidad y False si no lo es.
Cuerpos sobre los que opera Q,R,C,Z[p],Galois[p,{...}],Galois[p,d].
*)
```

```
UnidadCuerpo[Valor_]:=Module[{varaux},

    (* Para conseguir el cero entero *)
    varaux=Valor+Valor-Valor;

    If[
        varaux=!=0,
        Return[True];
    ,
    (* Else *)
        Return[False];
    ];
];


(*
=============== FUNCIÓN: UnidadPolinomios
Función booleana que devuelve True si Valor es una unidad en Q[x],R[x],C[x] ó
Galois[...][...][x].
*)

UnidadPolinomios[Valor_,Var_]:=Module[{},
    If[
        Exponent[Valor,Var]===0 && Valor=!=0,
        (* Constante no nula *)
        Return[True];
    ,
    (* Else *)
        Return[False];
    ];
];

(*
=============== FUNCIÓN: CalculoInversaPaso
*)

CalculoInversaPaso[Matriz_List,numfilas_]:=Module[{Resultado,MFila,MColumna,icont},

    {Resultado,MFila,MColumna}=FormaNormalPaso[Matriz];
```

```
    Do[
        If[
            UnidadQ[Resultado[[icont,icont]]],
            (* Es unidad *)
            Resultado[[icont,icont]]=1/Resultado[[icont,icont]];
        ,
        (* Else *)
            Resultado=0;
            Break[];
        ],
    {icont,1,numfilas}];
    Return[{Resultado,MFila,MColumna}];
];


(*
=============== FUNCIÓN: InversaPaso
Calcula la inversa de una matriz dada.
*)

InversaPaso[Matriz_List,AnilloBase_]:=Module[{AnilloOk,Resultado,numfilas,numcolumnas,MFila,
MColumna},

    {AnilloOk,Resultado}=CargaFunciones[AnilloBase,Matriz];
    If[
        AnilloOk,
        If[
            Length[Select[Flatten[Resultado],#1=!=0 &]]>0,
            {numfilas,numcolumnas}=Dimensions[Resultado];
            If[
                numfilas===numcolumnas,

                {Resultado,MFila,MColumna}=CalculoInversaPaso[Resultado,numfilas];

                If[
                    Resultado=!=0,
                    (* Matriz Inversible.*)
                    Resultado=Simplify[Expand[MColumna.Resultado.MFila]];
                    Return[Resultado];
```

```
                ,
                (* Else *)
                    Message[InversaPaso::noInversible,AnilloBase];
                ];
            ,
            (* Else *)
                Message[InversaPaso::noCuadrada,numfilas,numcolumnas];
            ];
        ,
        (* Else *)
            Message[InversaPaso::matrizNula];
        ];
    ,
    (* Else *)
        Message[InversaPaso::ErrDatos];
    ];
];


(*
=============== FUNCIÓN: InversaMinimo
Calcula la inversa de una matriz con coeficientes sobre C
*)
InversaMinimo[Matriz_List]:=Module[
{Resultado,AnilloOk,numfilas,numcolumnas,polminimo,InversaMatriz,termind,gradopol,icont},

    {AnilloOk,Resultado}=CargaFunciones[C[varaux],Matriz];

    If[
        AnilloOk,
        If[
            Length[Select[Flatten[Matriz],#1=!=0 &]]>0,
            {numfilas,numcolumnas}=Dimensions[Resultado];

            If[
                numfilas===numcolumnas,
                Resultado=FormaNormal[Resultado-varaux*IdentityMatrix[numfilas]];
                polminimo=Resultado[[numfilas,numcolumnas]];
                termind=Coefficient[polminimo,varaux,0];
                If[
```

```
                    termind===0,
                    Message[InversaMinimo::matrizSingular,Matriz];
                ,
                (* Else *)
                    gradopol=Exponent[polminimo,varaux];
                    InversaMatriz=Coefficient[polminimo,varaux,1] IdentityMatrix[numfilas];
                    Resultado=Matriz;
                    Do[
                        InversaMatriz=InversaMatriz+Coefficient[polminimo,varaux,icont] Resultado;
                        Resultado=Resultado.Resultado;
                    ,{icont,2,gradopol}];
                    InversaMatriz=FullSimplify[Expand[-1/termind InversaMatriz]];
                    Return[InversaMatriz];
                ];
            ,
            (* Else *)
                Message[InversaMinimo::noCuadrada,numfilas,numcolumnas];
            ];
        ,
        (* Else *)
            Message[InversaMinimo::matrizNula];
        ];
    ,
    (* Else *)
        Message[InversaMinimo::ErrDatos];
    ];
];

(*
=============== BLOQUE DE CÁLCULO DE FORMAS CANÓNICAS
Este paquete porporciona, dada una matriz sobre un cuerpo K,
cuatro formas canónicas, que son:
        · La forma canónica Racional.
        · La segunda forma canónica.
        · La forma canónica de Jacobson.
        · La forma canónica de Jordan.
*)

(*
=============== FUNCIÓN: GenPrimDesc
```

```
Dada una matriz A de orden n, GenPrimDesc calcula la descomposición en
K[x]-módulos cíclicos del K[x]-módulo K^n (recuérdese que la estructura
de K[x]-módulo de K^n está inducida por la matriz A.
Esta función no está pensada para ser utilizada sóla.
*)
GenPrimDesc[A_List]:=Module[{mFila,mColumna,matSmith,InvP,nounidades,
unidad,norden,icontador,diferencia,vauxiliar,MatIdentidad,generadoresz,
factoresd},

    norden=Dimensions[A][[1]];
    MatIdentidad=MatrizIdentidad[norden];
    unidad=MatIdentidad[[1,1]];
    matSmith=A-varx*MatIdentidad;

    (* Se calcula la Forma Normal de Smith *)
    {matSmith,mFila,mColumna}=FormaNormalPaso[matSmith];
    {matSmith,mColumna}=Normaliza[matSmith,mColumna];
    nounidades=Length[Select[Table[matSmith[[icontador,icontador]],
                                   {icontador,1,norden}
                                   ]
                             , #1=!=unidad &]];

    {InvP,mFila,mColumna}=CalculoInversaPaso[mFila,norden];
    InvP=Simplify[Expand[mColumna.InvP.mFila]];
    diferencia=norden-nounidades;
    generadoresz={};
    factoresd={};
    Do[
        vauxiliar=diferencia+icontador;
        generadoresz=Join[generadoresz,{InvP[[All,vauxiliar]]}];

        factoresd=Join[factoresd,{matSmith[[vauxiliar,vauxiliar]]}];

    ,{icontador,1,nounidades}];

    Return[{generadoresz,factoresd,nounidades}];
];


(*
```

```
=============== FUNCIÓN: CalculoVector
Convierte un elemento del K[x]-Módulo K^n en un elemento del
K-espacio vectorial K^n (multiplica en el K[x]-módulo K^n).
*)
CalculoVector[vector_,potenciasmatA_]:=Module[
{Resultado,kconta,maxgrado},

    Resultado=Coefficient[vector,varx,0];
    maxgrado=Max@@Exponent[vector,varx];

    Do[
        Resultado=Resultado+Plus@@(Coefficient[
                                            vector,
                                            varx,
                                            kconta]*
                                  Transpose[potenciasmatA[[kconta+1]]]);
    ,
    {kconta,1,maxgrado}];

    Return[Resultado];
];

(*
=============== FUNCIÓN: BaseRacional
Dado un vector u y un número natural gtotal, BaseRacional calcula una lista
de la forma {u,x u,x^2 u,...,x^(gtotal-1) u}.
Los argumentos polinomio y girred no se utilizan en la función. El motivo
de que se incluyan es que, posteriormente, en la función CalculoFormaCanonica,
se utilizará un nombre genérico (funcionBase), para referirse a Base Racional
ó BaseJacobson; y esta última sí necesita estos argumentos.
*)
BaseRacional[polinomio_,u_,girred_,gtotal_,potmatA_]:=Module[
{icontador,Resultado},

    Resultado={};
    Do[
        Resultado=Join[Resultado,{potmatA[[icontador+1]].u}];
    ,
    {icontador,0,gtotal-1}];
    Return[Resultado];
```

```
];

(*
=============== FUNCIÓN: BaseJacobson
Dado un vector u, un polinomio, dos números naturales girred y gtotal,
BaseJacobson calcula una lista de la forma
{u,x u,x^2 u,...,x^(girred-1) u,pol u,x pol u,x^2 pol u,...,
x^(girred-1) pol u,...,pol^(gtotal/girred-1) u,x pol^(gtotal/girred-1) u,
x^2 pol^(gtotal/girred-1) u,...,x^(girred-1) pol^(gtotal/girred-1) u}.
*)
BaseJacobson[polinomio_,u_,girred_,gtotal_,potmatA_]:=Module[
{icontador,Resultado,vaux,vectoraux},

    vaux=gtotal/girred-1;
    Resultado={};
    Do[
        vectoraux=CalculoVector[polinomio^icontador u,potmatA];
        Resultado=Join[Resultado,BaseRacional[0,vectoraux,girred,girred,potmatA]];
    ,
    {icontador,0,vaux}];
    Return[Resultado];
];


(*
=============== FUNCIÓN: MatrizCompañera
Calcula la matriz compañera de pol^n (se supone que pol es mónico).
*)
MatrizCompañera[pol_,n_,tamañomat_,girred_]:=Module[
{Resultado,jcont,polinomio,unidad},

    Resultado=Table[Table[0,{tamañomat}],{tamañomat}];
    polinomio=Expand[pol^n];
    unidad=MatrizIdentidad[1][[1,1]];
    Do[
        Resultado[[jcont+1,tamañomat]]=(-1)*Coefficient[polinomio,varx,jcont];
        Resultado[[jcont+2,jcont+1]]=unidad;
    ,
    {jcont,0,tamañomat-2}];
    Resultado[[tamañomat,tamañomat]]=(-1)*Coefficient[polinomio,varx,tamañomat-1];
```

```
Return[Resultado];
];

(*
=============== FUNCIÓN: MatrizHipercompañera
Calcula la matriz hipercompañera de pol^n (se supone que pol es mónico).
*)
MatrizHipercompañera[pol_,n_,tamañomat_,girred_]:=Module[
{matcomp,Resultado,icontador,unidad},

    Resultado=Table[Table[0,{tamañomat}],{tamañomat}];
    unidad=MatrizIdentidad[1][[1,1]];
    matcomp=MatrizCompañera[pol,1,girred,girred];

    Do[
        Resultado[[Range[girred*(icontador-1)+1,girred*icontador],
                    Range[girred*(icontador-1)+1,girred*icontador]]]=matcomp;
        Resultado[[girred*icontador+1,girred*icontador]]=unidad;
    ,
    {icontador,1,n-1}];

    Resultado[[Range[girred*(n-1)+1,n*girred],
                Range[girred*(n-1)+1,n*girred]]]=matcomp;
    Return[Resultado];
];

(*
=============== FUNCIÓN: CalculoFormaCanonica
Calcula, según los argumentos de entrada, una de las cuatro formas canónicas.
*)
CalculoFormaCanonica[Matriz_List,funcionMatriz_,funcionFactorizar_,funcionBase_]:=
Module[{generadoresz,factoresd,nounidades,norden,matrixaux,gradogen,maxgrado,
icontador,jcont,divelementales,vectoru,kconta,MatC,indicadorbloque,gradoirr,
gradototal,BaseMC,Sumaux,maxgradogen,maxgradofact},

{generadoresz,factoresd,nounidades}=GenPrimDesc[Matriz];

(* Se obtienen las potencias de A *)
gradogen=Exponent[generadoresz,varx];
```

```
gradofact=Exponent[factoresd,varx];

maxgradogen=Max[gradogen];
maxgradofact=Max[gradofact];
maxgrado=Max[maxgradogen*maxgradofact,maxgradogen,maxgradofact]+1;

norden=Dimensions[Matriz][[1]];
matrizaux={MatrizIdentidad[norden]};
Do[
    matrizaux=Join[matrizaux,{matrizaux[[icontador-1]].Matriz}];
,
{icontador,2,maxgrado}];

MatC=Table[Table[0,{norden}],{norden}];
BaseMC={};
indicadorbloque=1;
divelementales=Map[funcionFactorizar,factoresd];

If[FreeQ[divelementales,SalirRutina],
    Do[

        Do[
            vectoru=PolynomialQuotient[factoresd[[icontador]],
                        Power@@divelementales[[icontador,1,jcont]],
                        varx]*generadoresz[[icontador]];
            vectoru=CalculoVector[vectoru,matrizaux];
            gradoirred=Exponent[divelementales[[icontador,1,jcont,1]],varx];
            gradototal=gradoirred*divelementales[[icontador,1,jcont,2]];

            BaseMC=Join[BaseMC,funcionBase[divelementales[[icontador,1,jcont,1]],
                                           vectoru,gradoirred,gradototal,
                                           matrizaux]];

            Sumaux=indicadorbloque+gradototal-1;

            MatC[[Range[indicadorbloque,Sumaux],Range[indicadorbloque,Sumaux]]]=
            funcionMatriz[divelementales[[icontador,1,jcont,1]],
                        divelementales[[icontador,1,jcont,2]],
                        gradototal,
                        gradoirred];
```

```
                indicadorbloque=Sumaux+1;
            ,
            {jcont,1,divelementales[[icontador,-1]]}];
        ,
        {icontador,1,nounidades}];
        BaseMC=Transpose[BaseMC];
    ,
    (* Else *)
        (* Si no se pudieron calcular los divisores elementales
            se sale de la función *)
        Message[CalculoFormaCanonica::NoDivisores];
    ];
    Return[{MatC,BaseMC}];
    ];

(*
=============== BLOQUE DE FACTORIZACIÓN DE UN POLINOMIO


=============== FUNCIÓN: FactRacional
Esta función reformatea la entrada. Esto se hace para que poder utilizar
la función CalculoFormaCanonica para calcular la forma Racional.
*)
FactRacional[pol_]:=Module[{},
    Return[{{{pol,1}},1}];
];


(*
=============== FUNCIÓN: FactQ
Esta función factoriza un polinomio dado sobre Q[x]. La salida de esta
función es una lista de la forma {{{pol1,n1},...{polp,np}},p}, tal que
pol=pol1^n1*...*polp^np
*)
FactQ[pol_]:=Module[{Lista,nlong,icontador},
    lista=Factor[pol];

    If[
        Head[lista]===Times,
```

```
        (* Caso en el que Factor devuelve un producto de dos ó más factores *)
        lista=List@@lista;
    ,
    (* Else *)
        (* Caso en el que el polinomio es irreducible sobre Q ó Zp *)
        lista={lista};
    ];

    nlong=Length[lista];
    Do[
        If[
            Head[lista[[icontador]]]===Power,
            (* Caso de potencia de irreducible con exponente > 1 *)
            lista[[icontador]]={lista[[icontador,1]],lista[[icontador,2]]};
        ,
        (* Else *)
            lista[[icontador]]={lista[[icontador]],1};
        ];
    ,
    {icontador,1,nlong}];

    Return[{lista,Length[lista]}];
];

(*
=============== FUNCIÓN: FactReal
Esta función factoriza un polinomio dado sobre R[x]. La salida de esta
función es una lista de la forma {{{pol1,n1},...{polp,np}},p}, tal que
pol=pol1^n1*...*polp^np (pol1,...,polp son todos distintos).
*)
FactReal[pol_]:=Module[{raicespol,raicespolreales,raicespolcomplejas,
vaux,resultado,numraices,conjugado,icontador,nlong},

    raicespol=Simplify[Solve[pol==0,varx]/.(valorvarx_->valor_):>(valorvarx->ComplexExpand[valor])];

    If[
        FreeQ[raicespol,Root],
        raicespol=Flatten[raicespol];
        raicespolreales=Select[raicespol,Im[#1[[2]]]===0&];
        raicespolcomplejas=Select[raicespol,Im[#1[[2]]]=!=0&];
```

```
raicespolreales=Apply[Plus,
                    raicespolreales/.
                    {(varx->valorraiz_)->(varx->(-1)*valorraiz)}
                    ,{1}];

raicespolcomplejas=Apply[Plus,
                    raicespolcomplejas/.
                    {(varx->valorraiz_)->(varx->(-1)*valorraiz)}
                    ,{1}];

(* Se agrupan las raíces reales *)
resultado={};
numraices=0;

While[
    raicespolreales=!={},
    vaux=Length[Select[raicespolreales,
                    #1===raicespolreales[[1]]&]];
    resultado=Join[resultado,
                {{raicespolreales[[1]],vaux}}];
    numraices=numraices+1;
    vaux=raicespolreales[[1]];
    raicespolreales=DeleteCases[raicespolreales,vaux];
];

While[
    raicespolcomplejas=!={},
    vaux=Length[Select[raicespolcomplejas,
                    #1===raicespolcomplejas[[1]]&]];
    (* Las raíces complejas se presentan por pares *)
    conjugado=varx+(Conjugate[raicespolcomplejas[[1]]-varx]);
    resultado=Join[resultado,
                {{Expand[raicespolcomplejas[[1]]*conjugado
                        ]
                    ,vaux}}];
    numraices=numraices+1;
    vaux=raicespolcomplejas[[1]];
    raicespolcomplejas=DeleteCases[raicespolcomplejas,vaux];
    (* Se identifican los conjugados para borrarlos de la lista *)
```

```
            nlong=Length[raicespolcomplejas];
            icontador=1;
            While[icontador<=nlong,
            (* Al probar esta función se topó con el problema de la
            equivalencia con cero, es por esto que se utiliza Developer`ZeroQ *)
                If[
                    Developer`ZeroQ[raicespolcomplejas[[icontador]]-conjugado],
                    raicespolcomplejas=Drop[raicespolcomplejas,{icontador}];
                    nlong=nlong-1;
                ,
                (* Else *)
                    icontador=icontador+1;
                ];
            ];

        ];

        Return[{resultado,numraices}];
    ,
    (* Else *)
        Return[SalirRutina];
    ];
];


(*
=============== FUNCIÓN: FactCompleja
Esta función factoriza un polinomio dado sobre C[x]. La salida de esta
función es una lista de la forma {{{pol1,n1},...{polp,np}},p}, tal que
pol=pol1^n1*...*polp^np
*)
FactCompleja[pol_]:=Module[{raicespol,raicespolreales,raicespolcomplejas,
vaux,resultado,numraices},

    raicespol=Simplify[Solve[pol==0,varx]/.(valorvarx_->valor_):>(valorvarx->ComplexExpand[valor])];

    If[
        FreeQ[raicespol,Root],
        raicespol=Flatten[raicespol];
        raicespolreales=Select[raicespol,Im[#1[[2]]]===0&];
```

```
    raicespolcomplejas=Select[raicespol,Im[#1[[2]]]=!=0&];

    raicespolreales=Apply[Plus,
                        raicespolreales/.
                        {(varx->valorraiz_)->(varx->(-1)*valorraiz)}
                        ,{1}];

    raicespolcomplejas=Apply[Plus,
                        raicespolcomplejas/.
                        {(varx->valorraiz_)->(varx->(-1)*valorraiz)}
                        ,{1}];

    (* Se agrupan las raíces reales *)
    resultado={};
    numraices=0;

    While[
        raicespolreales=!={},
        vaux=Length[Select[raicespolreales,
                        #1===raicespolreales[[1]]&]];
        resultado=Join[resultado,
                    {{raicespolreales[[1]],vaux}}];
        numraices=numraices+1;
        vaux=raicespolreales[[1]];
        raicespolreales=DeleteCases[raicespolreales,vaux];
    ];

    While[
        raicespolcomplejas=!={},
        vaux=Length[Select[raicespolcomplejas,
                        #1===raicespolcomplejas[[1]]&]];
        resultado=Join[resultado,
                    {{raicespolcomplejas[[1]],vaux}}];
        numraices=numraices+1;
        vaux=raicespolcomplejas[[1]];
        raicespolcomplejas=DeleteCases[raicespolcomplejas,vaux];
    ];

    Return[{resultado,numraices}];
,
```

```
    (* Else *)
        Return[SalirRutina];
    ];
];


(*
====== Factorización de polinomios en una variable con coeficientes en un cuerpo
cuerpo finito (Implementación de los algoritmos del libro de Geddes-Czapor-Labahn,
consúltese la referencia bibliográfica).
*)

(*
=============== FUNCIÓN: FactLibreCuadrados
Dado un polinomio mónico pol, que pertenece a GF[p^d][x], FactLibreCuadrados[pol,
p,GF[p^d][{1}]] proporciona la factorización libre de cuadrados de pol.
*)
FactLibreCuadrados[pol_,p_,unidad_,var_]:=Module[{icontador=1,resultado=1,derpol,polmcd,
prodfactl2,polinomioy,factorlibre2,jcontad,gradopolmcd},

    derpol=Expand[unidad*D[pol,var]];
    If[
        derpol=!=0,
        polmcd=PolynomialExtendedGCD[pol,derpol][[1]];
        prodfactl2=PolynomialQuotient[pol,polmcd,var];
        While[
            prodfactl2=!=unidad && prodfactl2=!=1,
            polinomioy=PolynomialExtendedGCD[
                                    prodfactl2,
                                    polmcd][[1]];
            factorlibre2=PolynomialQuotient[
                                    prodfactl2,
                                    polinomioy,
                                    var];
            resultado=resultado*factorlibre2^icontador;
            icontador=icontador+1;
            prodfactl2=polinomioy;
            polmcd=PolynomialQuotient[
                                polmcd,
                                polinomioy,
```

```
                                    var];
        ];

        If[
            polmcd=!=unidad && polmcd=!=1,
            polmcd=polmcd/.{var->var^(1/p)};
            gradopolmcd=Exponent[polmcd,var];
            polmcd=(CoefficientList[polmcd,var]^(1/p)).
                    Table[var^jcontad,{jcontad,0,gradopolmcd}];
            resultado=resultado*FactLibreCuadrados[polmcd,p,unidad,var]^p;
        ];
    ,
    (* Else *)
            polmcd=pol/.{var->var^(1/p)};
            gradopolmcd=Exponent[polmcd,var];
            polmcd=(CoefficientList[polmcd,var]^(1/p)).
                    Table[var^jcontad,{jcontad,0,gradopolmcd}];
            resultado=FactLibreCuadrados[polmcd,p,unidad,var]^p;
    ];
    Return[resultado]
];

(*
=============== FUNCIÓN: ConstruirMatriz
Uno de los pasos que hay que dar para aplicar el algoritmo de Berlekamp es resolver
un sistema lineal, ConstruirMatriz[pola,nordenq] calcula la matriz de ese sistema,
siendo pola un polinomio del anillo de polinomios en una variable con coeficientes
en el cuerpo finito de nordenq elementos.
*)
ConstruirMatriz[pola_,nordenq_,var_]:=Module[{ngradopola,matresultado,icontador,unidad,
vaux,filanueva},

    ngradopola=Exponent[pola,var];
    matresultado=MatrizIdentidad[ngradopola];
    unidad=matresultado[[1,1]];
    vaux=ngradopola-1;

    Do[
        filanueva=CoefficientList[
                    PolynomialRemainder[
```

```
                            unidad*var^(icontador*nordenq),
                            pola,
                            var],
                        var];
            matresultado[[icontador+1,All]]=
                Join[
                    filanueva,
                    Table[0,{ngradopola-Length[filanueva]}]];
        ,
        {icontador,1,vaux}];
        Return[matresultado];
];

(*
=============== FUNCIÓN: ReducirMatriz
ReducirMatriz[Matriz,numfilas] obtiene a partir de Matriz, otra, que es triangular
inferior, con unos y ceros en la diagonal principal. Además, si en la posición (i,i)
aparece un 1, entonces la fila i tiene en las demás posiciones un cero.
*)
ReducirMatriz[Matriz_List,numfilas_]:=Module[{icontador,resultado,kcont,blnContinuar,
columnaux},

    resultado=Matriz;
    Do[
        (* Búsqueda del pivote *)
        icontador=kcont;
        blnContinuar=True;
        While[
            blnContinuar && icontador<= numfilas,
            If[
                resultado[[kcont,icontador]]===0,
                icontador=icontador+1;
            ,
            (* Else *)
                blnContinuar=False;
            ];
        ];

        If[
            icontador<= numfilas,
```

```
            (* Se normaliza la columna i *)
            resultado[[All,icontador]]=
                resultado[[All,icontador]]*1/resultado[[kcont,icontador]];
            (* Se intercambian las columnas *)
            columnaux=resultado[[All,icontador]];
            resultado[[All,icontador]]=resultado[[All,kcont]];
            resultado[[All,kcont]]=columnaux;

            Do[
                resultado[[All,icontador]]=
                    Expand[
                        resultado[[All,icontador]]-resultado[[All,kcont]]*resultado[[kcont,icontador]]];
            ,
            {icontador,1,kcont-1}];

            Do[
                resultado[[All,icontador]]=
                    Expand[
                        resultado[[All,icontador]]-resultado[[All,kcont]]*resultado[[kcont,icontador]]];
            ,
            {icontador,kcont+1,numfilas}];
        ]
    ,
    {kcont,1,numfilas}];

    Return[resultado];
];


(*
=============== FUNCIÓN: BaseNucleo
BaseNucleo[Matriz,unidad] calcula una base del subespacio vectorial v.Matriz=0
*)
BaseNucleo[Matriz_,unidad_]:=Module[{nordenmat,kcont,icontador,blnContinuar,resultado,
columnaux,listaceros,Base={}},

    nordenmat=Dimensions[Matriz][[1]];
    resultado=ReducirMatriz[Matriz,nordenmat];

    (* Hay que asegurarse que las columnas correspondientes a elementos diagonales
```

```
nulos, son nulas *)

Do[
    If[
        resultado[[icontador,icontador]]===0,
        (* Hay que comprobar que la columna está formada sólo por ceros *)
        Do[
            If[
                resultado[[kcont,icontador]]=!=0,
                (* Se le suma a la columna kcont-ésima la icontador-ésima *)
                resultado[[All,kcont]]=Expand[resultado[[All,kcont]]+
                    1/resultado[[kcont,icontador]]*resultado[[All,icontador]]];
            ]
        ,
        {kcont,icontador+1,nordenmat}];
    ];
,
{icontador,1,nordenmat-1}];

resultado=ReducirMatriz[resultado,nordenmat];

(* Se pasa de M a M-I *)
Do[
    resultado[[icontador,icontador]]=-1*unidad+resultado[[icontador,icontador]];
,
{icontador,1,nordenmat}];

(* Las filas no nulas son los vectores buscados *)
kcont=1;
listaceros=Table[0,{nordenmat}];
While[
    kcont<=nordenmat,
    blnContinuar=True;
    While[
        blnContinuar && kcont<=nordenmat,
        If[
            resultado[[kcont,All]]===listaceros,
            kcont=kcont+1;
        ,
        (* Else *)
```

```
                blnContinuar=False;
            ];
        ];
        If[
            kcont<=nordenmat,
            Base=Join[Base,{resultado[[kcont,All]]}];
            kcont=kcont+1;
        ];
    ];
    Return[Base];
];


(*
=============== FUNCIÓN: Berlekamp
Dado un polinomio pola mónico, libre de cuadrados y que pertenece al anillo de
polinomios en la variable var, con coeficientes en el cuerpo finito de
ordcuerpo elementos; Berlekamp[pola,ordcuerpo,var] calcula la factorización en
irreducibles de pola. El formato de salida es una lista compuesta por los factores
irreducibles de pola.
*)
Berlekamp[pola_,ordcuerpo_,var_]:=Module[{MatQ,Base,icontador,nordQ,Factores,indicer,
numirreducibles,numFactores,jcontad,elemK,poling,polinomioU,blnContinuar,vaux},

    blnContinuar=True;
    MatQ=ConstruirMatriz[pola,ordcuerpo,var];

    nordQ=Dimensions[MatQ][[1]];
    Do[
        MatQ[[icontador,icontador]]=GF[ordcuerpo][{-1}]+MatQ[[icontador,icontador]];
    ,
    {icontador,1,nordQ}];
    Base=BaseNucleo[MatQ,GF[ordcuerpo][{1}]];
    numirreducibles=Length[Base];

    (* Si el polinomio fuese irreducible ya no se sigue *)
    Factores={pola};
    numFactores=1;

    If[
```

```
numirreducibles=!=1,
(* Se pasa a polinomios los elementos de la Base *)
vaux=nordQ-1;
Do[
    Base[[jcontad]]=Expand[
                        Base[[jcontad]].
                        Table[GF[ordcuerpo][{1}]*var^icontador,
                                {icontador,0,vaux}]];
,
{jcontad,1,numirreducibles}];

indicer=2;
While[
    numFactores < numirreducibles && blnContinuar,
    icontador=1;
    While[
        icontador <= numFactores && blnContinuar,
        polinomioU=Factores[[icontador]];
        (* El caso cero se trata a parte *)
        poling=PolynomialExtendedGCD[Base[[indicer]],
                                    polinomioU][[1]];
        If[
            Expand[GF[ordcuerpo][{1}]*poling+GF[ordcuerpo][{-1}]]=!=0 &&
            (Expand[GF[ordcuerpo][{1}]*poling+GF[ordcuerpo][{-1}]*polinomioU]=!=0),
            polinomioU=PolynomialQuotient[polinomioU,poling,var];
            Factores=Join[{polinomioU},Factores,{poling}];
            Factores=Drop[Factores,{icontador+1}];
            numFactores=numFactores+1;
        ];
        If[
            numFactores===numirreducibles,
            blnContinuar=False;
        ,
        (* Else *)
            (* Caso no nulo *)
            elemK=GF[ordcuerpo][{1}];
            Do[
                poling=PolynomialExtendedGCD[Base[[indicer]]-elemK,polinomioU][[1]];
                If[
                    Expand[GF[ordcuerpo][{1}]*poling+GF[ordcuerpo][{-1}]]=!=0 &&
```

```
                        (Expand[GF[ordcuerpo][{1}]*poling+GF[ordcuerpo][{-1}]*polinomioU]=!=0),
                        polinomioU=PolynomialQuotient[polinomioU,poling,var];
                        Factores=Join[{polinomioU},Factores,{poling}];
                        Factores=Drop[Factores,{icontador+1}];
                        numFactores=numFactores+1;
                    ];
                    elemK=Successor[elemK];

                    If[
                        numFactores===numirreducibles,
                        blnContinuar=False;
                        Break[];
                    ]
                ,
                {jcontad,1,ordcuerpo-1}];
                ];
                icontador=icontador+1
            ];
            indicer=indicer+1;
        ];
    ];
    Return[Factores];

];


(*
================ FUNCIÓN: FactGalois
Esta función proporciona la factorización en irreducibles de un polinomio pola, en una
variable con coeficientes en el cuerpo finito de q=p^d elementos.
*)
FactGalois[pola_,p_,d_,var_]:=Module[{q,factpol,icontador,kcont,factpollib2,nlong,nlonglib2},

    q=p^d;
    factpol=FactLibreCuadrados[pola,p,GF[q][{1}],var];

    Which[
        Expand[GF[q][{1}]*factpol-GF[q][{1}]*var]===0,
        (* factpol=1*x *)
        factpol={{GF[q][{1}]*var,1}}
```

```
,
    Head[factpol]===Plus,
    (* factpol=1*x+a *)
    factpol={{GF[q][{1}]*factpol,1}}
,
    Head[factpol]===Power,
    (* factpol=(1*x+a)^n *)
    factpol={List@@factpol}
,
    Head[factpol]===Times,
    factpol=List@@factpol;
    (* Se quitan los términos constantes del producto *)
    icontador=1;
    While[icontador<=Length[factpol],
        If[
            Intersection[{var},Variables[factpol[[icontador]]]]==={},
            factpol=Drop[factpol,{icontador}];
        ,
        (* Else *)
            icontador=icontador+1;
        ];
    ];
    nlong=Length[factpol];
    Do[
        If[
            Head[factpol[[icontador]]]===Power,
            (* Caso de una potencia de exponente > 1 *)
            factpol[[icontador]]={factpol[[icontador,1]],factpol[[icontador,2]]};
        ,
        (* Else *)
            (* Caso de potencia de exponente = 1 *)
            factpol[[icontador]]={factpol[[icontador]],1};
        ]
    ,
    {icontador,1,nlong}]
];


(* Se aplica el algoritmo de Berlekamp a los factores libres de cuadrados *)
nlong=Length[factpol];
```

```
    Do[
        factpollib2=Berlekamp[factpol[[icontador,1]],q,var];
        nlonglib2=Length[factpollib2];
        factpol=Join[factpol,Table[{factpollib2[[kcont]],factpol[[icontador,2]]},{kcont,1,nlonglib2}]];
    ,
    {icontador,1,nlong}];
    (* Se borran los primeros *)
    factpol=Drop[factpol,nlong];

    Return[{factpol,Length[factpol]}]
]


(*
=============== FUNCIÓN: ValidaCanonica
Valida y calcula en su caso, cualquiera de las formas canónicas (depende de los
datos de entrada, qué forma se calcula).
*)
ValidaCanonica[A_List,K_,fMatriz_,fFactorizacion_,fBase_]:=Module[
{AnilloOk,matriznueva,nfilas,ncols},

    (* Hay que distinguir el caso de Zp -> Z[p,x] *)
    If[Cases[{K},Z[_Integer]]===={},
        (* Caso de un cuerpo que no es el de los enteros módulo p *)
        {AnilloOk,matriznueva}=CargaFunciones[K[varx],A];
    ,
    (* Else *)
        (* Zp *)
        {AnilloOk,matriznueva}=CargaFunciones[Z[K[[1]],varx],A];
    ];

    If[
        AnilloOk,
            (* Se comprueba que la matriz es no nula *)
            If[
                Length[Select[Flatten[matriznueva],#1=!=0 &]]>0,
                (* Se comprueba que la matriz sea cuadrada *)
                {nfilas,ncols}=Dimensions[matriznueva];
                If[
                    nfilas===ncols,
```

```
                    Return[CalculoFormaCanonica[matriznueva,fMatriz,
                            fFactorizacion,fBase]];
                ,
                (* Else *)
                    Message[ValidaCanonica::NoCuadrada];
                ];
            ,
            (* Else *)
                Return[matriznueva];
            ];
    ,
    (* Else *)
        Message[ValidaCanonica::ErrDatos];
    ];
];

(*
=============== FUNCIÓN: Frobenius
Obtiene la forma Racional de la matriz dada.
*)
Frobenius[A_List,K_]:=Module[{},
    Return[ValidaCanonica[A,K,MatrizCompañera,FactRacional,BaseRacional]];
];

(*
=============== FUNCIÓN: SegundaFormaC
Obtiene la Segunda forma canónica de la matriz dada, sobre el cuerpo K.
*)
SegundaFormaC[A_List,K_]:=Module[{},
    Return[ValidaCanonica[A,K,MatrizCompañera,Factorizar,BaseRacional]];
];

(*
=============== FUNCIÓN: Jacobson
Obtiene la forma de Jacobson de la matriz dada, sobre el cuerpo K.
*)
Jacobson[A_List,K_]:=Module[{},
    Return[ValidaCanonica[A,K,MatrizHipercompañera,Factorizar,BaseJacobson]];
];
```

```
    End[]

EndPackage[]
```

Bibliografía

[AdL94] W. Adams, P. Loustaunau, *An introduction to Gröbner Bases*. Graduate Studies in Mathematics, Vol 3, AMS Press (1994).

[ApL88] J. Apel, W. Lassner, *An Extension of Buchberger's Algorithm and Calculations in Enveloping Fields of Lie Algebras*. J. Symb. Comput. 6, 361-370 (1988).

[Ape00] J. Apel, *Computational Ideal Theory in Finitely Generated Extension Rings*. Theoretical Comput. Sci. 244 (2000).

[AyG03] M. Aymon, P-P. Grivel, *Un Théorème de Poincaré-Birkhoff-Witt pour les Algèbres de Leibniz*. Comm. Algebra 31, 527-544 (2003).

[AyO98] S. Ayupov, B. Omirov, *On Leibniz Algebras*. Algebra and operator Theory (Tashkent, 1997), 1-12, Kluwer Acad. Publ., Dordrecht (1998).

[Bar96] S. Barnett, *Matrices: Methods and Applications*. Oxford Applied Mathematics and Computing Science Series. Clarendon Press - Oxford (1996).

[BeW93] T. Becker, V. Weispfenning, *Gröbner Bases. A computational approach to commutative algebra*. Springer-Verlag, New York (1993).

[Ber78] G. M. Bergman, *The diamond lemma for ring theory*. Adv. Math. 29, 178-218 (1978).

[Ber67] E. Berlekamp, *Factoring polynomials over finite fields*. Bell System Tech. J. 46, 1853-1859 (1967).

[Ber70] E. Berlekamp, *Factoring polynomials over large finite fields*. Math. Comp. 24, 713-735 (1970).

[Ber72] E. Berlekamp, *Factoring Polynomials*, pp. 1-7 in F. Hoffman, R.B. Levow, and R.S.D. Thomas (eds.), Proc. of the 3rd Southeastern Conference on Combinatorics, GRAPH THEORY AND COMPUTING, Atlantic Univ., Boca Raton, Florida (1972).

[Buc65] B. Buchberger, *An algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Ideal.* Ph. D. Thesis, Univ. of Innsbruck, Math. Inst. (1965).

[BuW98] B. Buchberger, F. Winkler, *Gröbner bases and applications*. Papers from the Conference on 33 Years of Gröbner Bases held at the University of Linz, Linz, February 2-4 (1998). London Mathematical Society Lecture Note Series, 251. Cambridge University Press, Cambridge (1998)

[BGC98] J. L. Bueso, J. Gómez Torrecillas, F. J. Lobillo, F. J. Castro, *An introduction to effective calculus in quantum groups*. Lecture Notes in Pure and Appl. Math. 197, Dekker, New York, pp. 55-83 (1998).

[Cam88] O. Cámpoli, *A Principal Ideal Domain That Is Not a Euclidean Domain*. Amer. Math. Monthly 95, 868-871 (1988).

[CaZ81] D. Cantor, H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*. Math. Comp. 36, 587-592 (1981).

[ChC82] T. Chou, G. Collins, *Algorithms for the solution of systems of linear Diophantine equations*. SIAM J. Computing 11, 687-708 (1982).

[Coh74] P. Cohn, *Algebra*. John Wiley & Sons (1974).

[CLO92] D. Cox, J. Little, D. O'Shea, *Ideals, Varieties and Algorithms*. Springer-Verlag, New York (1992).

[Czi95] G. Czichowski, *A Note on Gröbner Bases and Integration of Rational Functions*. J. Symb. Comput. 20, 163-167 (1995).

[Der99] H. Derksen, *Computation of invariants for reductive groups*. Adv. Math. 141, 366-384 (1999).

[Dix96] J. Dixmier, *Enveloping Algebras*. Graduate Studies in Mathematics, Vol. 11, AMS (1996).

[DuF99] D. Dummit, R. Foote, *Abstract Algebra*, Second Edition. John Wiley & Sons, Inc., New York (1999).

[F1879] G. Frobenius, L. Stickelberger, *Über Gruppen von Vertauschbaren Elementen*. J. de Crelle LXXXVI, 217 (1879).

[GaS92] J. von zur Gathen, V. Shoup, *Computing Frobenius maps and factoring polynomials.* Comput. Complexity 2, 187-224 (1992).

[GaG99] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*. Cambridge University Press (1999).

[GCL92] K. Geddes, S. Czapor, G. Labahn, *Algorithms for Computer Algebra*. Kluwer Academic Publishers (1992).

[Gie93] M. Giesbrecht, *Nearly optimal algorithms for canonical matrix forms*. Ph. D Thesis, Departament of Computer Science, University of Toronto (1993).

[GRZ02] M. Giesbrecht, G. Reid, Y. Zang, *Non-Conmutative Gröbner Bases in Poincaré-Birkhoff-Witt Extensions.* Proceedings of the Fifth International Workshop on Computer Algebra in Scientific Computing, September 22-27, Yalta, Ukraine, pp. 97-106 (2002).

[GiS02] M. Giesbrecht, A. Storjhann, *Computing Rational Forms of Integer Matrices*. J. Symb. Comput. 34, 157-172 (2002).

[G1899] P. Gordan, Newer Beweis des Hilbertschen Satzes über homogene Funktionen. Nachrichten König. Ges. Der. Wiss. Zu Gött. 240-242 (1899).

[Gra00] W. Graaf, *Lie Algebras: Theory and Algorithms*. Elsevier Science B. V., Amsterdam (2000).

[Gre94] E. L. Green, *An Introduction to Noncommutative Gröbner bases*. Computational algebra. Lect. Notes in Pure and Appl. Marcel Dekker, Inc., New York pp. 167-190 (1994).

[Gre97] J. Greene, *Principal Ideal Domains Are Almost Euclidean*. Amer. Math. Monthly 104, 154-156 (1997).

[Grö39] Gröbner W., *Über die algebraischen Eigenschaften der Integrale von linearen Differentialgleichungen mit konstanten Koeffizienten.* Monatsh. der Math. 47, 247-284 (1939).

[HaH83] B. Hartley, T. Hawkes, *Rings, Modules and Linear Algebra*. Chapman and Hall (1983).

[Hea95] A. Hearn, *Reduce User´s Manual 3.6.* Rand (1995).

[Her26] G. Hermann, *Der Frage der endlich vielen Schritte in der Theorie der Polynomideale*, Math. Ann. 95, 736-788 (1926).

[H1890] D. Hilbert, *Über die Theorie del algebraischen Formen*, Math. Ann. 36, 473-534 (1890). Reprinted in Gesammelte Abhandlungen, Volume II, Chelsea, New York (1965).

[H1893] D. Hilbert, *Über die vollen Invariantensysteme*. Math. Ann. 42, 313-373 (1893). Reprinted in: *Theory of Algebraic Invariants*, Cambridge University Press, Cambridge (1993).

[Hir64] H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero*. Ann. Math. 79, 109-326 (1964).

[HiSt97] P. Hilton, U. Stammbach, *A Course in Homological Algebra*, Second Edition. Springer Verlag (1997).

[Jac74] N. Jacobson, *Basic Algebra I*. W. H. Freeman and Company (1974).

[Jea99] C-P. Jeannerod, *Formes Normales de Matrices et valeurs propres en calcul formel*. Preprint (1999).

[KaS98] E. Kaltofen, V. Shoup, *Subquadratic-time factoring of polynomials over finite fields*. Math. Comp. 67, 1179-1197 (1998).

[KaW90] A. Kandri-Rody, V. Weispfenning, *Noncommutative Gröbner bases in algebras of solvable type*. J. Symb. Comput. 9(1), 1-26 (1990).

[KaB79] R. Kannan, A. Bachem, *Polynomial time algorithms to compute Hermite and Smith normal forms of an integer matrix*. SIAM J. Computing 8, 499-507 (1979).

[KnB70] D. Knuth, P. Bendix, *Simple word problems in universal algebras*. In: Leech, J. (ed.), Computational Problems in Abstract Algebra, Pergamon Press, 263-297 (1970).

[Kre92] H. Kredel, *Solvable polynomial rings*. PhD. Thesis, FMI, Universität Passau (1992).

[Laz85] D. Lazard, *Ideal Bases and Primary Decomposition: Case of Two Variables*. J. Symb Comput.1, 261-270 (1985).

[LaR90] D. Lazard, R. Rioboo, *Integration of Rational Functions: Rational Computation of the Logarithmic Part*. J. Symb Comput. 9(2), 113-116 (1990).

[Lak96] F. Lazebnik, *On systems of linear diophnatine equations.* Mathematics Magazine, Vol. 69, No. 4, October, 261-266 (1996).

[Li02] H. Li, *Noncommutative Gröbner bases and filtered-graded transfer.* Lecture Notes in Math. 1795. Springer-Verlag, Berlin (2002).

[LiN97] R. Lidl, H. Niederreiter, *Finite Fields.* Encyclopedia of Mathematics and its applications 20. Second Edition, Cambridge University Press (1997).

[Lod93] J-L. Loday, *Une version non commutative des algèbres de Lie: les algèbres de Leibniz.* Enseign. Math. 39, 269-293 (1993).

[Lod97] J-L. Loday, *Overview on Leibniz algebras, dialgebras and their homology.* Fields Inst. Comm. 17, 91-102 (1997).

[Lod01] J-L. Loday, *Dialgebras* in *Dialgebras and related operads.* Lecture Notes in Math. 1763, Springer, Berlin, 7-66 (2001).

[LoP93] J-L. Loday, T. Pirashvili, *Universal enveloping algebras of Leibniz algebras and (co)homology.* Math. Ann. 296, 139-158 (1993).

[LoP98] J-L. Loday, T. Pirashvili, *The Tensor Category of Linear Maps and Leibniz Algebras.* Georgian Math. J. Vol. 5, No. 3, 263-276 (1998).

[M1916] F. Macaulay, *The algebraic theory of modular systems.* Revised reprint of the 1916 original. With an introduction by Paul Roberts. Cambridge Mathematical Library. Cambridge University Press, Cambridge (1994).

[M1927] F. Macaulay, *Some properties of enumeration in the theory of modular systems.* Proc. London Math. Soc. 26, 531-555 (1927).

[Mal72] A. Máltsev, *Fundamentos de Álgebra Lineal.* Editorial Mir Moscú (1972).

[Mol88] H. Möller, *On the Construction of Gröbner Basis Using Syzygies*. J. Symb. Comput. 6, 345-359 (1988).

[Mor94] T. Mora, *An introduction to commutative and non conmutative Gröbner bases*. Theor. Comp. Sci. 134, 131-173 (1994).

[NaS92] R. Nagle, E. Saff, *Fundamentos de Ecuaciones Diferenciales*, Segunda Edición. Addison-Wesley Iberoamericana (1992).

[NaG94] G. Nakos, N. Glinos, *Computing Gröbner Bases over the Integers*. The Mathematica Journal, Vol. 4 (3), 70-75 (1994).

[Nie94] H. Niederreiter, *Factoring polynomials over finite fields using differential equations and Normal Bases*. Math. Comp. 62, 819-830 (1994).

[Nor01] P. Nordbeck, *On the Finiteness of Gröbner Bases Computation in Quotients of the Free Algebra*. Appl. Algebra Engrg. Comm. Comput. 11, 157-180 (2001).

[NoY02] M. Noro, K. Yokoyama, *Yet Another Practical Implementation of Polynomial Factorization over Finite Fields*. Proc. ISAAC'2002, ACM Press, New York, pp. 200-206 (2002).

[Pan88] L. Pan, *On D-bases of Polynomial Ideals Over Principal Ideal Domains*. J. Symb. Comput. 7, 55-69 (1988).

[Pes94] M. Pesch, *Gröbner bases in rings of substitution operators*. Proc. Workshop Comp. Algebra. Univ. Karlsruhe, 138-143 (1994).

[Rei95] B. Reinert, *On Gröbner Bases in Monoid and Group Rings.* Ph. D. Thesis, University of Kaiserslautern (1995).

[Rom92] S. Roman, *Advanced Linear Algebra*. Graduate Texts in Mathematics, Springer-Verlag (1992).

[Rot76] M. Rothstein, *Aspects of Symbolic Integration and Simplification of Exponential and Primitive Functions*. Ph. D. Thesis, University of Wisconsin (1976).

[Sch80] F. O. Schreyer, *Die Berechnung von Syzygien mit dem verallgemeintern Weierstrass'schen Divisionssatz*. Diplomarbeit, Hamburg (1980).

[Sei74] A. Seidenberg, *Constructions in Algebra.* Trans. Amer. Math. Soc. 197, 272-313 (1974).

[Ser02] D. Serre, *Matrices, Theory and Applications*. Graduate Texts in Mathematics. Springer (2002).

[Shi62] A. Shirshov, *Some algorithmic problems for Lie algebras*, Siberian Math. J. 3, 292-296 (1962).

[S1861] H. Smith, *On systems of indeterminate equations and congruences*. Philos. Trans. 151, 293-326 (1861).

[S1851] J. Sylvester, *An Essay on Canonical Forms, Supplement to a Sketch of a Memoir on Elimination, Transformation and Canonical Forms*. London (1851). Reprinted in J. Sylvester's Collected Mathematical Papers, Vol. 1. Cambridge, England: At the University Press, 209 (1904).

[Tra76] B. Trager, *Algebraic Factoring and Rational Function Integration*. Proc. SYMSAC 76, NY ACM Press, 219-226 (1976).

[Vil94] G. Villard, *Fast Parallel computation of the Smith Normal Form of Polynomial matrices*. In International Symposium on Symbolic and Algebraic Computation, Oxford, UK, 312-317, ACM Press (1994).

[Vil95] G. Villard, *Generalized subresultants for computing the Smith Normal Form of polynomial matrices*. J. Symb. Comput. 20, 269-286 (1995).

[Vil97] G. Villard, *Fast Parallel Algorithms for Matrix Reduction to Normal Forms*. AAECC 8, 511-537 (1997).

[Wei89] V. Weispfenning, *Gröbner bases for polynomial ideals over commutative regular rings*. Proc. EUROCAL'87, Springer, LNCS 378, pp. 336-347 (1989).

[Wei92] V. Weispfenning, *Finite Gröbner bases in non-Noetherian skew polynomials rings*. Proc. ISAAC'92, ACM Press, New York, pp. 329-334 (1992).

Índice alfabético