Ecuaciones polinomiales: una aproximación algorítmica

Gabriela Jeronimo[1] Juan Sabia[1]

Departamento de Matemática
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
y
CONICET

Índice

1 Introducción **2**

2 Algoritmos y complejidad **3**
2.1 Un primer ejemplo 3
2.2 Nociones básicas 4
2.3 Algoritmos de Álgebra Lineal 5

3 Sistemas de ecuaciones polinomiales: existencia de soluciones **9**
3.1 Resultantes 9
3.2 Un teorema de existencia de soluciones en varias variables 10
3.3 El teorema de los ceros efectivo 11
3.4 Codificación densa de polinomios 13
3.5 Un algoritmo para el teorema de los ceros 13

4 Geometría algebraica efectiva **14**
4.1 Algunas definiciones de geometría algebraica 14
4.2 Un algoritmo para calcular dimensiones 15
4.3 Resolución geométrica de variedades cero-dimensionales 17
4.4 Variedades equidimensionales de dimensión positiva 20
4.5 Descomposición equidimensional 21

5 Codificación de polinomios por straight-line programs **22**
5.1 Cotas inferiores de complejidad debidas a la representación densa 22
5.2 Straight-line programs 22
5.3 Algunos problemas 24

6 Algoritmos que utilizan straight-line programs **26**
6.1 Cálculo de la dimensión de una variedad 26
6.2 Operador de Newton-Hensel 28
6.3 Aplicación: resolución de sistemas paramétricos 31
6.4 Sistemas cero-dimensionales y straight-line programs 36
6.5 Teorema de los ceros 38
6.6 Algoritmos probabilísticos 38
6.7 Un algoritmo probabilístico para la descomposición equidimensional 39

Referencias **41**

[1]Parcialmente financiado por UBACyT (proyecto X112, 2004-2007) y CONICET (PIP 2461, 2000).

1 Introducción

La resolución de ecuaciones polinomiales es un tema que ha sido muy estudiado a lo largo de los años a causa de las distintas aplicaciones, provenientes de diversas áreas de la ciencia y de la tecnología, en las que este tipo de ecuaciones aparecen. Este tema ha recobrado importancia en las últimas décadas con el surgimiento de la computación que, en particular, ha introducido la necesidad del estudio de los aspectos *algorítmicos* de distintos problemas en esta área.

Este curso se concentrará en la resolución de ecuaciones polinomiales en varias variables desde el punto de vista algorítmico. Más precisamente, dados polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$, donde k es un cuerpo de característica cero (por ejemplo $k = \mathbb{Q}$), nos interesará estudiar el conjunto de las soluciones del sistema $f_1(x_1, \ldots, x_n) = 0, \ldots, f_s(x_1, \ldots, x_n) = 0$ en una clausura algebraica de k.

En primer lugar, es necesario precisar qué entendemos por "resolver algorítmicamente" un sistema de ecuaciones polinomiales. Una respuesta posible, que constituye el punto de vista que tendremos en este curso, es que esto consiste en determinar si el sistema tiene soluciones y, en caso de que las tenga, describirlas de alguna manera que sea "útil" en el contexto algorítmico.

Muchos de los algoritmos que se conocen para resolver problemas relacionados con ecuaciones polinomiales se basan en reducirlos a problemas de álgebra lineal. Es por esto que, antes de comenzar con nuestro estudio específico sobre sistemas de ecuaciones polinomiales, presentaremos algunos procedimientos del álgebra lineal efectiva.

Definiremos también la noción de algoritmo con la que vamos a trabajar. Un punto fundamental de nuestro enfoque es poder predecir cuánto tiempo requerirá un algoritmo para resolver un problema dado. Para esto, introduciremos la noción de *complejidad algebraica*, que nos dará una primera medida de este tiempo. Otro aspecto que se tendrá en cuenta es cómo representar a los polinomios en el desarrollo de un algoritmo.

En la primera parte del curso, nos concentraremos en el estudio de algoritmos que representan a cada polinomio como el vector de sus coeficientes. Comenzaremos (Sección 3) con el estudio del problema de decidir si un sistema de ecuaciones polinomiales tiene solución. Presentaremos el *Teorema de los ceros de Hilbert* y versiones efectivas de este resultado, que proveen una reformulación del problema que nos permitirá reducirlo al de la existencia de soluciones de sistemas de ecuaciones lineales. Posteriormente (Sección 4) consideraremos el problema de describir el conjunto de las soluciones de un sistema de ecuaciones polinomiales consistente. Introduciremos los conceptos básicos de geometría algebraica que utilizaremos y describiremos algunos algoritmos relacionados con distintos aspectos del problema.

El uso de vectores de coeficientes para representar los polinomios manipulados por un algoritmo puede hacer que el tiempo que requiere su ejecución sobre un input dado sea demasiado grande. Esta observación llevó a la búsqueda de formas alternativas más eficientes para representar polinomios, entre las cuales estudiaremos los *straight-line programs* (Sección 5). En pocas palabras, un straight-line program que representa un polinomio es un programa (constituido por una secuencia de instrucciones, cada una de las cuales es una suma, resta o producto) que calcula el valor del polinomio en cualquier punto dado.

La segunda parte del curso estará orientada al estudio de algoritmos para la resolución de ecuaciones polinomiales que utilizan straight-line programs para representar polinomios. Se considerarán nuevamente los problemas estudiados en las secciones anteriores y se presentarán algoritmos más eficientes para su resolución (Sección 6).

Se analizarán también algunas dificultades que surgen al utilizar la representación de poli-

nomios mediante straight-line programs (por ejemplo, el problema de determinar si un polinomio dado por un straight-line program es el polinomio nulo) y posibles acercamientos para solucionar estas dificultades. Esto nos llevará a la noción de *algoritmos probabilísticos*, que son algoritmos que resuelven un problema dado pero con cierta probabilidad de error, con la que concluiremos el curso.

2 Algoritmos y complejidad

2.1 Un primer ejemplo

Un primer ejemplo de resolución algorítmica de ecuaciones que podemos dar es el de las ecuaciones lineales. Un método conocido para la resolución de este tipo de ecuaciones es el método de Gauss, que consiste en triangular la matriz del sistema.
Vamos a describir un algoritmo que nos permite triangular cualquier matriz cuadrada en $\mathbb{Q}^{m\times n}$. Nos interesa estimar la cantidad máxima $x_{m,n}$ de comparaciones y operaciones $(+,-,\times,\div)$ entre elementos de $\mathbb{Q}$ que efectúa el algoritmo para triangular una matriz de $m \times n$ (no se contarán los cambios de lugar de coeficientes).

Dada la matriz $A = (a_{ij}) \in \mathbb{Q}^{m\times n}$, el algoritmo procede como sigue:

(i) Si $a_{i1} = 0$ para cada i $(1 \leq i \leq m)$, el problema se reduce a triangular la matriz de $(m-1) \times (n-1)$ que se obtiene al suprimir la primera fila y la primera columna de A.

(ii) Si $a_{i1} \neq 0$ para algún i $(1 \leq i \leq m)$, podemos suponer que $a_{11} \neq 0$ (si es cero, se elige algún elemento $a_{i1} \neq 0$, $2 \leq i \leq m$, y se intercambia la fila i con la fila 1).

Para cada $2 \leq i \leq m$: Se multiplica la primera fila por a_{i1}/a_{11} y se resta el resultado a la i-ésima fila, para lograr un cero en el lugar que ocupa a_{i1}.

Ahora, como en (i), el problema se reduce a triangular una matriz de $(m-1)\times(n-1)$.

¿Cuántas comparaciones y operaciones se realizan? En primer lugar, se efectúan a lo sumo m comparaciones (para determinar si algún elemento de la primer columna de A es no nulo). En el caso (i) no se hacen más operaciones ni comparaciones que las necesarias para triangular una matriz de $(m-1) \times (n-1)$. En el caso (ii), para cada $2 \leq i \leq m$, hay una división, $n-1$ multiplicaciones y $n-1$ restas (ya sabemos que en el primer lugar de la fila va a ir un cero), y a continuación se aplica el algoritmo para triangular una matriz de $(m-1) \times (n-1)$. Por lo tanto, tenemos que

$$x_{m,n} = m + (m-1)(1 + 2(n-1)) + x_{m-1,n-1} \quad \forall\, m,\, n \geq 2.$$

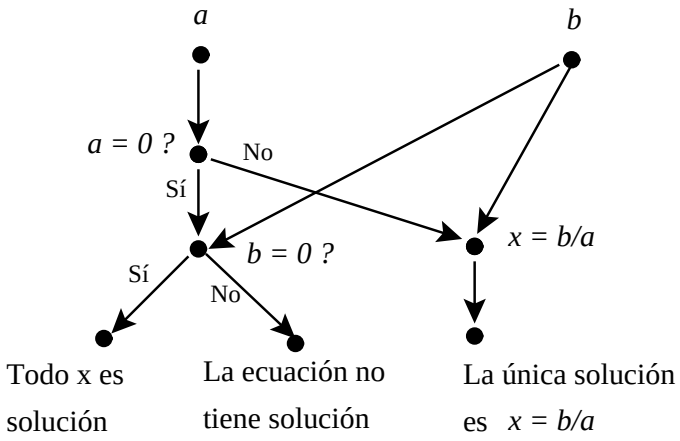
Teniendo en cuenta que $x_{1,n} = 0$ (puesto que una matriz de $1 \times n$ es triangular) y $x_{m,1} = m$, resulta que la sucesión $\{x_{m,n}\}_{m,n\in\mathbb{N}}$ queda definida por la recursión:

$$\begin{cases} x_{1,n} = 0 \\ x_{m,1} = m \\ x_{m,n} = 2mn - 2n + 1 + x_{m-1,n-1} \quad \forall\, m,n \geq 2. \end{cases}$$

A partir de esta recursión se puede probar que la máxima cantidad $x_{m,n}$ de comparaciones y operaciones que realiza el algoritmo para triangular una matriz de $m \times n$ está acotada superiormente por $2mn\min\{m,n\}$.

2.2 Nociones básicas

En esta sección formalizaremos las nociones intuitivas de algoritmo y complejidad que aparecieron en el ejemplo anterior.
La noción de algoritmo que vamos a utilizar puede explicarse brevemente como sigue: dados algunos datos (números, por ejemplo), un algoritmo que tiene esos datos como entrada será una secuencia de operaciones o comparaciones que comenzando con los datos de entrada termina en cierto "objeto" lógico o matemático que queremos calcular.
Por ejemplo, supongamos que queremos resolver la ecuación $ax = b$ con coeficientes $a, b \in \mathbb{Q}$ y que podemos trabajar con números racionales (es decir, realizar comparaciones y operaciones aritméticas) algorítmicamente. La figura siguiente muestra un algoritmo posible para resolver esta ecuación.



Más formalmente, nuestros algoritmos trabajarán con datos de entrada en un cuerpo k (o un dominio íntegro $\mathcal{R}$) de característica cero y *efectivo*, es decir, un cuerpo en el cual las operaciones aritméticas y comparaciones pueden ser realizadas algorítmicamente. Para nosotros, un *algoritmo* es un *grafo orientado acíclico* con una cantidad finita de nodos. Cada nodo del grafo representa un elemento de k, una operación o una comparación entre elementos de k. Una rama que llega a un nodo indica una condición o elemento previamente calculado que es necesario para realizar la siguiente operación.
Notar que, como queremos poder predecir el tiempo que tardará el algoritmo en ser ejecutado sobre un input dado, consideramos una familia restringida de algoritmos: no permitimos instrucciones que impliquen repetir un proceso mientras que se satisfaga una determinada condición; en los únicos casos en que se podrá repetir un proceso es cuando se conozca de antemano cuántas veces habrá que hacerlo. (En lenguaje de programación, no permitimos el uso del "WHILE", sino únicamente instrucciones del tipo "FOR ... FROM ... TO ...").

La idea de complejidad de un algoritmo está relacionada con el tiempo que requiere el algoritmo para realizar la tarea deseada. Una primera forma en que esto puede medirse es mediante la cantidad de nodos del grafo: cuanto mayor sea este número, la ejecución del algoritmo requerirá más tiempo. Ésta es la noción de *complejidad* que vamos a usar en este curso, la cual se conoce como *complejidad algebraica secuencial*.
Si bien esta medida de complejidad da una primera idea acerca de si el algoritmo en cuestión es "bueno" o no, no es del todo precisa. Por ejemplo, es mucho más fácil para una máquina calcular la suma $1 + 1$ que sumar dos números enormes, pero nuestra noción de complejidad no tiene en cuenta esto. Más aún, en general es más rápido calcular una suma que

un producto. Surgen entonces distintas nociones de complejidad (complejidad no escalar, complejidad bit, etc.), pero éstas no serán estudiadas en este curso.
Además de la complejidad secuencial, existen otras variables que pueden ser tenidas en cuenta para analizar la factibilidad de un algoritmo: el *espacio en memoria* necesario para el desarrollo del algoritmo o si el algoritmo puede ser *bien paralelizable* (es decir, si el algoritmo puede ejecutarse suficientemente rápido en caso de disponerse de una cantidad suficiente de procesadores donde desarrollar su ejecución simultáneamente), entre otras. Estos aspectos tampoco serán estudiados en este curso.

Para expresar órdenes de complejidad utilizaremos la notación usual O: dadas dos funciones $f : \mathbb{N} \to \mathbb{N}$ y $g : \mathbb{N} \to \mathbb{N}$, decimos que $f = O(g)$ si y sólo si existe $c \in \mathbb{N}$ tal que $f(n) \leq cg(n)$ para todo $n \in \mathbb{N}$.
Diremos que una complejidad es *polinomial* en un parámetro λ si es del orden $\lambda^{O(1)}$.

2.3 Algoritmos de Álgebra Lineal

Usando la complejidad del método de triangulación de la Sección 2.1 (que se puede aplicar a matrices con coeficientes en cualquier cuerpo k de característica 0) podemos calcular fácilmente la complejidad del procedimiento clásico de álgebra lineal para la resolución de sistemas de ecuaciones lineales:

Proposición 1 *Un sistema lineal de m ecuaciones con n incógnitas a coeficientes en un cuerpo k de característica 0 se puede resolver con complejidad $O(mn \min\{m, n\})$.*

A continuación mostraremos algunos algoritmos eficientes de álgebra lineal que también se usan en la resolución de ecuaciones polinomiales.

Polinomio característico

Una herramienta básica del álgebra lineal es el polinomio característico de una matriz. A continuación mostraremos un algoritmo *sin divisiones* para el cálculo del polinomio característico de una matriz con coeficientes en un dominio íntegro $\mathcal{R}$ (ver [1]).
La idea del algoritmo es relacionar los coeficientes del polinomio característico de $A \in \mathcal{R}^{n\times n}$ con los del polinomio característico de una submatriz de A de $(n-1) \times (n-1)$ y proceder recursivamente.

Observación 2 *Sea $A = (a_{ij}) \in \mathcal{R}^{n\times n}$ y sean $R \in \mathcal{R}^{1\times(n-1)}$, $S \in \mathcal{R}^{(n-1)\times 1}$ y $A' \in \mathcal{R}^{(n-1)\times(n-1)}$ tales que*

$$A = \begin{pmatrix} a_{11} & R \\ S & A' \end{pmatrix}.$$

Entonces, si $\mathcal{X}_A := \det(A - T.I_n)$ y $\mathcal{X}_{A'} := \det(A' - T.I_{n-1})$ son los polinomios característicos de A y A' respectivamente, se tiene que

$$\mathcal{X}_A(T) = (a_{11} - T)\mathcal{X}_{A'}(T) - R \cdot \operatorname{adj}(A' - T.I_{n-1}) \cdot S. \tag{1}$$

El paso siguiente será calcular una matriz adjunta del tipo $M - T.I_n$ usando los coeficientes del polinomio característico de M.

Observación 3 *Sea* $M \in \mathcal{R}^{n\times n}$ *y sea* $\mathcal{X}_M = \sum_{k=0}^{n} q_k T^{n-k}$ *el polinomio característico de* M. *Entonces*

$$\operatorname{adj}(M - T.I_n) = -\sum_{k=1}^{n} \Big(\sum_{j=0}^{k-1} q_j \, M^{k-1-j} \Big) T^{n-k}.$$

Aplicando la Observación 3 para el cálculo de la matriz adjunta que aparece en (1) se obtiene la relación entre los coeficientes de los polinomios característicos de A y A':
Si $\mathcal{X}_A(T) = \sum_{k=0}^{n} p_k T^{n-k}$ y $\mathcal{X}_{A'}(T) = \sum_{k=0}^{n-1} p'_k T^{n-1-k}$, entonces

$$\begin{cases} p_0 &= -p'_0 \\ p_1 &= a_{11}\, p'_0 - p'_1 \\ p_k &= \sum_{j=0}^{k-2} R\,(A')^{k-2-j} S\, p'_j + a_{11}\, p'_{k-1} - p'_k \qquad \text{para } k = 2, \ldots, n. \end{cases}$$

Esto puede reescribirse como $(p_0, \ldots, p_n)^t = C\,(p'_0, \ldots, p'_{n-1})^t$, donde $C \in \mathcal{R}^{n\times(n-1)}$ es la matriz de Toeplitz triangular inferior definida por

$$C_{i1} = \begin{cases} -1 & \text{si } i = 1 \\ a_{11} & \text{si } i = 2 \\ R\,(A')^{i-3} S & \text{si } i \geq 3 \end{cases}$$

es decir,

$$C = \begin{pmatrix} -1 & 0 & \cdots & \cdots & \cdots & 0 \\ a_{11} & -1 & \ddots & & & \vdots \\ RS & a_{11} & \ddots & \ddots & & \vdots \\ RA'S & RS & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & a_{11} & -1 \\ R(A')^{n-3}S & \cdots & \cdots & RA'S & RS & a_{11} \end{pmatrix}.$$

Recursivamente se obtiene:

Proposición 4 *Sea* $A \in \mathcal{R}^{n\times n}$ *y sea* $\mathcal{X}_A = \sum_{k=0}^{n} p_k\, T^{n-k}$ *el polinomio característico de* A. *Para cada* $1 \leq \ell \leq n-1$ *sean* $R_\ell \in \mathcal{R}^{1\times(n-\ell)}$, $S_\ell \in \mathcal{R}^{(n-\ell)\times 1}$ *y* $A_\ell \in \mathcal{R}^{(n-\ell)\times(n-\ell)}$ *las matrices*

$$R_\ell := (a_{\ell\,\ell+1} \ \ldots \ a_{\ell\, n}), \quad S_\ell := (a_{\ell+1\,\ell} \ \ldots \ a_{n\ell})^t, \quad A_\ell := (a_{ij})_{\ell+1\leq i,j\leq n}$$

y sea $C_\ell \in \mathcal{R}^{(n-\ell+1)\times(n-\ell)}$ *la matriz de Toeplitz triangular inferior definida por*

$$(C_\ell)_{i1} = \begin{cases} -1 & \textit{si } i = 1 \\ a_{\ell\ell} & \textit{si } i = 2 \\ R_\ell\, A_\ell^{i-3} S_\ell & \textit{si } i \geq 3 \end{cases}$$

Entonces $(p_0, \ldots, p_n)^t = C_1 C_2 \ldots C_{n-1}$.

Utilizando el resultado anterior, obtenemos un algoritmo de baja complejidad para calcular el polinomio característico de una matriz.

Teorema 5 *Sea $A \in \mathcal{R}^{n\times n}$. Entonces, los coeficientes del polinomio característico de A se pueden obtener por medio de un algoritmo (sin divisiones) con complejidad $O(n^4)$.*

Demostración. El algoritmo se reduce a realizar los cálculos descriptos en la fórmula de la Proposición 4. Ahora estimaremos su complejidad:

- Cálculo de las entradas de C_ℓ para cada $1 \leq \ell \leq n-1$:

  Para $k = 1, \ldots, n-\ell-2$ se calculan, inductivamente, los productos $A_\ell^k S_\ell = A_\ell(A_\ell^{k-1} S_\ell)$. La complejidad de este paso es $O((n-\ell)^3)$, puesto que cada uno de los productos (una matriz por una matriz columna) puede efectuarse con $O((n-\ell)^2)$ operaciones.

  Para $k = 1, \ldots, n-\ell-2$ se calculan $R_\ell.(A_\ell^k S)$. La complejidad de este paso es del mismo orden.

  Luego, la complejidad del cálculo de las $n-1$ matrices C_ℓ es $O(n^4)$.

- Cálculo del producto $C_1 \ldots C_{n-1}$.

  Para $\ell = n-1, \ldots, 1$, se calcula $C_\ell.(C_{\ell+1} \ldots C_{n-1})$. Como $C_\ell \in \mathcal{R}^{(n-\ell+1)\times(n-\ell)}$ y $C_{\ell+1} \ldots C_{n-1} \in \mathcal{R}^{(n-\ell)\times 1}$, el producto de estas matrices se calcula con $O((n-\ell)^2)$ operaciones. Por lo tanto, la complejidad de este paso es $O(n^3)$.

En consecuencia, la complejidad total del algoritmo es $O(n^4)$. □

Se puede obtener una cota mejor para la complejidad del cálculo del polinomio característico modificando levemente este algoritmo (en el cálculo de $R_\ell A_\ell^{i-3} S_\ell$) y utilizando cotas más precisas para la complejidad del producto de dos matrices.

Observación 6 *El algoritmo descripto en el Teorema 5 calcula el determinante de una matriz $A \in \mathcal{R}^{n\times n}$ con complejidad $O(n^4)$ y sin efectuar divisiones.*

Rango de una matriz

El procedimiento para calcular polinomios característicos puede utilizarse para calcular algorítmicamente sin divisiones el rango de una matriz. Esto será de particular importancia en el caso de trabajar con matrices con coeficientes polinomios. (Observar que en caso de permitir divisiones puede utilizarse el método de Gauss presentado en la Sección 2.1.)

Sea $A \in k^{n\times n}$ una matriz que verifica $\mathrm{Nu}(A) \oplus \mathrm{Im}(A) = k^n$. Su rango puede calcularse a partir de su polinomio característico $\mathcal{X}_A$ como $\mathrm{rg}(A) = n - \mathrm{mult}(0, \mathcal{X}_A)$, donde $\mathrm{mult}(0, \mathcal{X}_A)$ denota la multiplicidad de 0 como raíz de $\mathcal{X}_A$. Este número puede obtenerse fácilmente conociendo los coeficientes de $\mathcal{X}_A$. Más precisamente:

$$\text{Si } \mathrm{Nu}(A) \oplus \mathrm{Im}(A) = k^n \text{ y } \mathcal{X}_A = \textstyle\sum_{i=0}^{n} p_i T^{n-i}, \text{ entonces } \mathrm{rg}(A) = \max\{i : p_i \neq 0\}. \qquad (2)$$

En el caso de una matriz arbitraria, la idea es construir una matriz C tal que $\mathrm{Nu}(C)$ e $\mathrm{Im}(C)$ estén en suma directa y cuyo rango nos permita calcular el rango de A.

Si $A \in \mathbb{R}^{n\times m}$, consideramos $A^t \in \mathbb{R}^{m\times n}$ la matriz transpuesta de A y definimos $C := A^t A \in \mathbb{R}^{m\times m}$. Es fácil ver que $\mathrm{Nu}(C) = \mathrm{Nu}(A)$, y por lo tanto, $\mathrm{rg}(C) = \mathrm{rg}(A)$. Además, siendo una matriz simétrica, C es diagonalizable y, en consecuencia verifica $\mathrm{Nu}(C) \oplus \mathrm{Im}(C) = \mathbb{R}^m$.

La matriz C se obtiene como el producto de $A^t \in \mathbb{R}^{m\times n}$ y $A \in \mathbb{R}^{n\times m}$ con $O(nm^2)$ operaciones en $\mathbb{R}$ y su polinomio característico puede calcularse con complejidad $O(m^4)$. Finalmente, si $\mathcal{X}_C = \sum_{i=0}^{m} p_i T^{m-i}$, se busca $\mathrm{rg}(C) = \max\{i : p_i \neq 0\}$, lo que requiere a lo sumo m comparaciones. Luego, la complejidad total para el cálculo del rango de A es $O(nm^2 + m^4)$.

Para un cuerpo k cualquiera, usaremos las ideas de [32]. Sea $A \in k^{n\times m}$. En primer lugar, construimos una matriz simétrica $M \in k^{(n+m)\times(n+m)}$ asociada a A:

$$M = \begin{pmatrix} 0_{n\times n} & A \\ A^t & 0_{m\times m} \end{pmatrix}.$$

Es claro que $\mathrm{rg}(M) = 2\,\mathrm{rg}(A)$, con lo cual calcularemos el rango de M.
Sea $N := n + m$. Puesto que el rango de una matriz no cambia al extender el cuerpo de base, podemos trabajar en una extensión de k. Sea x una indeterminada sobre k y sea $D \in k(x)^{N\times N}$ la matriz diagonal definida por $D_{ii} = x^{i-1}$ para $1 \leq i \leq N$. Sea $C \in k(x)^{N\times N}$ definida por $C := D.M$. Como D es no singular, se tiene que $\mathrm{rg}(C) = \mathrm{rg}(M)$.

Lema 7 *Con la notación anterior,* $\mathrm{Nu}(C) \oplus \mathrm{Im}(C) = k(x)^N$.

Demostración. Probaremos que $\mathrm{Nu}(C^2) = \mathrm{Nu}(C)$, de donde se deduce el resultado del lema. Ahora, $\mathrm{Nu}(C^2) = \mathrm{Nu}(DMDM) = \mathrm{Nu}(MDM)$ y $\mathrm{Nu}(C) = \mathrm{Nu}(M)$, por lo que basta ver que $\mathrm{Nu}(MDM) = \mathrm{Nu}(M)$.
Es claro que $\mathrm{Nu}(M) \subset \mathrm{Nu}(MDM)$. Supongamos que existe $u \in k(x)^N$ tal que $u \in \mathrm{Nu}(MDM)$, pero $u \notin \mathrm{Nu}(M)$. Sin pérdida de generalidad, podemos suponer que $u \in k[x]^N$. Sea $v := M.u \in k[x]^N$. Por la elección de u, se tiene que $v \neq 0$.
Sea z una nueva indeterminada sobre k. Teniendo en cuenta que M es una matriz simétrica y el hecho que $u \in \mathrm{Nu}(MDM)$ resulta que

$$v(z)^t D\, v(x) = u(z)^t M^t DMu(x) = u(z)^t MDMu(x) = 0. \tag{3}$$

Sean $\delta := \max\{\deg(v_i) : 1 \leq i \leq N\}$ y $\nu := \max\{i : \deg(v_i) = \delta\}$. Entonces, el monomio $z^\delta x^{\delta+\nu-1}$ aparece con coeficiente no nulo en el polinomio $v_\nu(z)v_\nu(x)x^{\nu-1}$ y no puede aparecer en ningún otro término de la suma $\sum_{i=1}^{N} v_i(z)v_i(x)x^{i-1} = v(z)^t D\, v(x)$. Luego, $z^\delta x^{\delta+\nu-1}$ tiene coeficiente no nulo en el polinomio $v(z)^t D\, v(x)$, lo que contradice (3). □

Como consecuencia de (2), por el lema anterior se tiene que $\mathrm{rg}(C) = \mathrm{mult}(0, \mathcal{X}_C)$.
En vez de calcular el polinomio $\mathcal{X}_C$ con x variable, lo calcularemos para suficientes especializaciones de x en elementos de k, para lo cual bastará calcular los polinomios característicos de las matrices que se obtienen a partir de C especializando x en esos elementos. Como $\mathcal{X}_C$ tiene grado en x acotado por $\frac{N(N-1)}{2}$, para determinar si uno de sus coeficientes es cero, basta evaluarlo en $\frac{N(N-1)}{2} + 1$ valores distintos en k. Por lo tanto, tomamos $\frac{N(N-1)}{2} + 1$ puntos y calculamos la multiplicidad del cero como raíz del polinomio característico de la matriz obtenida para cada punto usando el algoritmo del Teorema 5. El mínimo de estas multiplicidades es la multiplicidad del cero como raíz de $\mathcal{X}_C$, que es el rango de C.
La complejidad total de este algoritmo se calcula fácilmente teniendo en cuenta la complejidad dada por el Teorema 5 para el cálculo de polinomios característicos y la cantidad de veces ($O(N^2)$) que este algoritmo se aplica, y resulta ser del orden de $O(N^6)$.
En resumen, hemos demostrado el siguiente resultado:

Teorema 8 *Sea $A \in k^{n\times m}$. Entonces, el rango de A puede calcularse algorítmicamente (sin divisiones) con complejidad $O((n+m)^6)$.*

3 Sistemas de ecuaciones polinomiales: existencia de soluciones

En muchos contextos aparece la necesidad de resolver sistemas de ecuaciones polinomiales. En lo que sigue vamos a tratar de dar algunas aproximaciones algorítmicas para la resolución de estos sistemas.

Definición 9 *Sea k un cuerpo y sean $x_1, \ldots, x_n$ indeterminadas sobre k. Un* sistema de ecuaciones polinomiales con coeficientes en k *es*

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \vdots \\ f_s(x_1, \ldots, x_n) = 0 \end{cases}$$

donde $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$.

La primera pregunta que intentaremos responder es si un sistema de ecuaciones polinomiales dado tiene o no solución.

3.1 Resultantes

El primer caso que vamos a tratar es el de dos polinomios en una variable con coeficientes en un dominio íntegro $\mathcal{R}$. Para esto vamos a definir la resultante, herramienta que también utilizaremos más adelante en el caso general.

Definición 10 *Sea x una indeterminada sobre $\mathcal{R}$ y sean $f, g \in \mathcal{R}[x]$ polinomios de grado d, e respectivamente. Si $f = \sum_{i=0}^{d} a_i x^i$ y $g = \sum_{i=0}^{e} b_i x^i$, se define la* resultante entre f y g *como*

$$\mathrm{Res}(f,g) = \det \begin{pmatrix} a_d & 0 & \cdots & 0 & b_e & 0 & \cdots & 0 \\ a_{d-1} & a_d & \ddots & \vdots & b_{e-1} & b_e & \ddots & \vdots \\ \vdots & a_{d-1} & \ddots & 0 & \vdots & b_{e-1} & \ddots & 0 \\ \vdots & \vdots & \ddots & a_d & \vdots & \vdots & \ddots & b_e \\ \vdots & \vdots & & a_{d-1} & b_0 & \vdots & & b_{e-1} \\ a_0 & \vdots & & \vdots & 0 & b_0 & & \vdots \\ 0 & a_0 & & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots & & & \\ 0 & \cdots & \cdots & a_0 & 0 & \cdots & \cdots & b_0 \end{pmatrix}$$

(e columnas; d columnas)

A continuación enunciaremos algunas propiedades básicas de la resultante cuyas demostraciones pueden encontrarse, por ejemplo, en [39] o [6].

Proposición 11 *Con las notaciones anteriores, f y g tienen un factor en común en $\mathcal{R}[x]$ si y sólo si* $\mathrm{Res}(f,g) = 0$.

Corolario 12 *Sea k el cuerpo de cocientes de $\mathcal{R}$ y sea $\overline{k}$ una clausura algebraica de k. Entonces el sistema*

$$\begin{cases} f(x) = 0 \\ g(x) = 0 \end{cases}$$

tiene solución en $\overline{k}$ si y sólo si $\mathrm{Res}(f, g) = 0$.

Notar que el corolario anterior nos da un criterio algorítmico para decidir si un sistema de dos ecuaciones polinomiales en una variable tiene solución o no.

La siguiente propiedad va a ser usada más adelante.

Proposición 13 *Con las notaciones anteriores, existen polinomios p y q en $\mathcal{R}[x]$ de grados acotados por $e-1$ y $d-1$ respectivamente tales que* $\mathrm{Res}(f, g) = pf + qg$.

3.2 Un teorema de existencia de soluciones en varias variables

Sea $X = \{x_1, \ldots, x_n\}$ una familia de indeterminadas sobre un cuerpo k. Dados polinomios $f_1, \ldots, f_s \in k[X]$, vamos a probar un criterio para decidir si el sistema de ecuaciones asociado tiene soluciones o no en una clausura algebraica $\overline{k}$ de k.
En el caso en que todos los polinomios $f_1, \ldots, f_s$ tengan grado 1, se trata de un sistema lineal y, en este caso, el criterio está dado por una simple comparación de rangos de matrices que involucran los coeficientes del sistema:
Si el sistema se escribe $A.x^t = B$ (con $A \in k^{s\times n}$ y $B \in k^{s\times 1}$) entonces

$$\exists\, x \in \overline{k}^n \ / \ A.x^t = B \iff \mathrm{rg}(A) = \mathrm{rg}(A|B)$$

donde $(A|B)$ denota la matriz que se obtiene al agregar la columna B a la matriz A.

Un primer paso hacia una generalización de este resultado al caso de sistemas de ecuaciones polinomiales de grados arbitrarios está dado por el teorema siguiente, que relaciona la existencia de soluciones de un sistema polinomial con ciertos cálculos que involucran los coeficientes de los polinomios considerados:

Teorema 14 *(Teorema de los ceros de Hilbert) Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$. Las siguientes afirmaciones son equivalentes:*

i) $\{\xi \in \overline{k}^n \ / \ f_1(\xi) = \ldots = f_s(\xi) = 0\} = \emptyset$.

ii) Existen polinomios $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ tales que $1 = \sum_{1\le i\le s} g_i.f_i$.

Demostración.

ii) $\Rightarrow$ i) Es inmediato.

i) $\Rightarrow$ ii) Sea $I = (f_1, \ldots, f_s) \subset k[x_1, \ldots, x_n]$ el ideal generado por $f_1, \ldots, f_s$. Si $1 \notin I$, existe un ideal maximal $\mathcal{M}$ de $k[x_1, \ldots, x_n]$ tal que $I \subset \mathcal{M}$. Consideremos la extensión de cuerpos K/k donde $K = k[x_1, \ldots, x_n]/\mathcal{M}$.
Sea B una base de trascendencia de K/k (eventualmente $B = \emptyset$). Entonces $K/k(B)$ es una extensión algebraica, en particular, K es un $k(B)$-espacio vectorial finitamente generado. Sea $w_1, \ldots, w_m$ un sistema de generadores de K como $k(B)$-espacio vectorial que contiene a $\overline{x}_1, \ldots, \overline{x}_n$, y sean $r_l^{(ij)} \in k(B)$ tales que $w_i w_j = \sum_{l=1}^m r_l^{(ij)} w_l$. Entonces cada elemento

de $K = k[\overline{x}_1, \ldots, \overline{x}_n]$ se escribe como $\sum_{h=1}^{m} P_h(r_l^{(ij)}).\, w_h$, para ciertos polinomios P_h con coeficientes en k. Ahora, si $B \neq \emptyset$ y $r_l^{(ij)} = f_l^{(ij)}/g_l^{(ij)}$ con $f_l^{(ij)}$, $g_l^{(ij)} \in k[B]$ para cada i, j, l, todo elemento que puede representarse de esta forma tiene como denominador un polinomio en $k[B]$ tal que cada uno de sus factores irreducibles divide a algún $g_l^{(ij)}$, lo que contradice que todo elemento de $k(B) \subset K$ tiene una representación de este tipo.
Luego, K/k es una extensión algebraica y, por lo tanto, existe un morfismo de extensiones $\varphi : K/k \to \overline{k}/k$. Entonces $(\varphi(\overline{x}_1), \ldots, \varphi(\overline{x}_n)) \in \overline{k}^n$ resulta una solución del sistema de ecuaciones $f_1 = 0, \ldots, f_s = 0$. □

Este resultado ya era conocido por Kronecker y esencialmente muestra cómo un problema geométrico *"¿Es vacío el conjunto de los ceros comunes de los polinomios $f_1, \ldots, f_s$?"* es equivalente a uno algebraico *"¿Pertenece el elemento 1 al ideal $(f_1, \ldots, f_s)$?"*.

3.3 El teorema de los ceros efectivo

Un *teorema de los ceros efectivo* es un algoritmo que, tomando como entradas polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$, determina si existen polinomios $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ tales que

$$\sum_{1 \leq i \leq s} g_i.f_i = 1 \tag{4}$$

y calcula una solución particular $(g_1, \ldots, g_s)$ para esta identidad.

Un posible primer paso para conseguir un teorema de los ceros efectivos es encontrar cotas superiores para los grados de una solución $g_1, \ldots, g_s$ de la ecuación (4) en función de s, n y d, donde d es una cota superior para los grados de los polinomios $f_1, \ldots, f_s$. El primer resultado en este sentido fue obtenido por G. Hermann (ver [23]).

Teorema 15 *Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ tales que $\deg(f_i) \leq d$ para cada $1 \leq i \leq s$. Si $1 \in (f_1, \ldots, f_s)$, existen polinomios $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ con $\deg(g_i) \leq 2\,(2d)^{2^{n-1}}$ para cada $1 \leq i \leq s$ tales que $\sum_{1 \leq i \leq s} g_i.f_i = 1$.*

Demostración. Probaremos el teorema por inducción en n.
Para $n = 1$, sean $f_1, \ldots, f_s \in k[x]$ y supongamos que $\deg(f_1) = d \geq \deg(f_i)$ $(2 \leq i \leq s)$. Si $1 = \sum_{1 \leq i \leq s} h_i.f_i$, aplicando el algoritmo de división por f_1 en $k[x]$, tenemos que

$$h_i = f_1.q_i + r_i \qquad (2 \leq i \leq s),$$

y, reagrupando los términos en la suma, obtenemos

$$1 = f_1.(h_1 + \sum_{2 \leq i \leq s} q_i.f_i) + \sum_{2 \leq i \leq s} f_i.r_i.$$

Como $\deg(r_i) \leq d - 1$ $(2 \leq i \leq s)$, resulta que $\deg(f_1.(h_1 + \sum_{2 \leq i \leq s} q_i.f_i)) \leq 2d - 1$. Por lo tanto, tomando $g_1 = h_1 + \sum_{2 \leq i \leq s} q_i.f_i$ y $g_i = r_i$ $(2 \leq i \leq s)$ obtenemos $1 = \sum_{1 \leq i \leq s} g_i.f_i$ y $\deg(g_i) \leq d - 1$ $(1 \leq i \leq s)$.
Supongamos ahora que el resultado vale para n. Sean entonces $f_1, \ldots, f_s \in k[x_1, \ldots, x_{n+1}]$ tales que $\deg(f_i) \leq \deg(f_1) = d$ $(2 \leq i \leq s)$.
En primer lugar, con el objeto de obtener polinomios que sean mónicos con respecto a una variable, consideramos el siguiente cambio de variables (donde $\lambda_2, \ldots, \lambda_n$ son nuevos

parámetros): $x_1 = y_1$, $x_2 = y_2 + \lambda_2 y_1, \ldots, x_n = y_n + \lambda_n y_1$. Los polinomios que se obtienen al hacer este cambio de variables tienen grado máximo en y_1 y sus coeficientes principales en esta variable son las partes homogéneas de grado máximo de los polinomios originales evaluadas en $(1, \lambda_2, \ldots, \lambda_n)$. Eligiendo una $(n-1)$-upla conveniente, de manera que estas partes homogéneas no se anulen, obtenemos el cambio de variables deseado.
De esta manera, podemos suponer, sin pérdida de generalidad, que cada polinomio f_i es mónico en la variable x_1. Introduzcamos nuevas variables $U_1, \ldots, U_s, V_1, \ldots, V_s$ y consideremos los polinomios

$$F := \sum_{1 \leq i \leq s} U_i f_i \quad \text{y} \quad G := \sum_{1 \leq i \leq s} V_i f_i \quad \text{en} \quad \mathbb{Q}[U, V][x_1, \ldots, x_{n+1}].$$

La resultante de estos polinomios con respecto a la variable x_1,

$$\mathrm{Res}_{x_1}(F, G) \in \mathbb{Q}[U, V][x_2, \ldots, x_{n+1}]$$

es un polinomio bi-homogéneo en los grupos de variables (U, V) de bi-grado (d, d).
Veamos que, si escribimos

$$\mathrm{Res}_{x_1}(F, G) = \sum_{\alpha, \beta} h_{\alpha,\beta}(x_2, \ldots, x_{n+1}) U^\alpha V^\beta,$$

entonces $f_1, \ldots, f_s$ tienen un cero en común en $\overline{k}^{n+1}$ si y sólo si $(h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d}$ tienen un cero en común en $\overline{k}^n$: Si $(\xi_1, \ldots, \xi_{n+1}) \in \overline{k}^{n+1}$ es una raíz común de $f_1, \ldots, f_s$, entonces

$$\mathrm{Res}_{x_1}(F, G)(\xi_2, \ldots, \xi_{n+1})(U, V) = 0$$

y, por lo tanto, los polinomios $(h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d}$ tienen una raíz común en $\overline{k}^n$. Por otro lado, si $(\xi_2, \ldots, \xi_{n+1})$ es un cero común de $(h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d}$, los polinomios $F(x_1, \xi_2, \ldots, \xi_{n+1})$ y $G(x_1, \xi_2, \ldots, \xi_{n+1})$, considerados como polinomios en $k(U_1, \ldots, U_s, V_1, \ldots, V_s)[x_1]$ tienen una raíz común en $\overline{k(U, V)}$. Pero, como los ceros de $F(x_1, \xi_2, \ldots, \xi_{n+1})$ pertenecen a $\overline{k(U)}$ y los de $G(x_1, \xi_2, \ldots, \xi_{n+1})$ a $\overline{k(V)}$, la raíz común debe estar en $\overline{k}$. Es decir, existe $\xi_1 \in \overline{k}$ tal que $F(\xi_1, \ldots, \xi_{n+1}) = 0$ y $G(\xi_1, \ldots, \xi_{n+1}) = 0$. Puesto que las variables U, V son algebraicamente independientes sobre k, concluimos que $(\xi_1, \ldots, \xi_{n+1}) \in \overline{k}^{n+1}$ es una raíz común de $f_1, \ldots, f_s$.
Debido al teorema de los ceros de Hilbert (Teorema 14), resulta que

$$1 \in (f_1, \ldots, f_s) \iff 1 \in (h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d},$$

lo que reduce en uno la cantidad de variables.
Finalmente, veamos que la escritura del 1 como combinación lineal de $(h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d}$ nos permite deducir su escritura como combinación de los polinomios $f_1, \ldots, f_s$: Por la Proposición 13, existen polinomios R y S en $k[U, V][X]$, de grados acotados por $2d^2$ en las variables X tales que $\mathrm{Res}_{x_1}(F, G) = RF + SG$. Reescribiendo esta identidad en potencias de U y V, tenemos que

$$h_{\alpha,\beta} = \sum_{1 \leq i \leq s} p_i^{(\alpha,\beta)} f_i,$$

donde los polinomios $p_i^{(\alpha,\beta)}$ tienen grados acotados por $2d^2$.

La cota de grado deseada para los polinomios que dan la escritura del 1 como combinación lineal de $f_1, \ldots, f_s$ se deduce ahora fácilmente aplicando la hipótesis inductiva a $(h_{\alpha,\beta})_{|\alpha|=d,|\beta|=d}$, cuyos grados están acotados por $2d^2$. □

Las cotas obtenidas en el teorema anterior no son buenas desde el punto de vista algorítmico. Existen otras cotas mejores, pero las herramientas usadas para demostrar su existencia se escapan al nivel de este curso.
En [2], se obtuvo la primera cota simplemente exponencial, del tipo $3\min\{n,s\}nd^{\min\{n,s\}}$, para cuerpos de característica cero. Más adelante, en [9] y [27] se obtuvieron las cotas más precisas conocidas hasta ahora para el caso general: $(\max\{3,d\})^n$.

3.4 Codificación densa de polinomios

Como nuestro objetivo es resolver sistemas de ecuaciones polinomiales *algorítmicamente*, debemos determinar una manera de codificar polinomios en varias variables para poder ingresarlos en un procesador como datos de entrada.
Una primera forma (la más ingenua) de codificar un polinomio es copiar la forma usual en que éste se escribe: como una suma de monomios. Para hacer esto de manera que lo pueda entender una computadora necesitamos conocer de antemano una cota superior d para el grado del polinomio y la cantidad de variables n que involucra. Dados d y n, damos un orden al conjunto de todos los monomios de grado menor o igual que d en n variables y codificamos al polinomio como el vector de sus coeficientes en este orden. Esta manera de codificar polinomios se llama *codificación densa*.
Por ejemplo, supongamos que el polinomio que queremos codificar es $f(x,y) = x^2 - 2xy + y^2 + x + 3$. Fijamos un orden en el conjunto de los monomios de grado 2 en 2 variables, por ejemplo $(1, x, y, x^2, xy, y^2)$. La codificación densa de f usando este orden será $(3, 1, 0, 1, -2, 1)$.
Nos interesa estimar el tamaño de la codificación que hemos definido, es decir, cuántos números será necesario almacenar para codificar un polinomio f de grado acotado por d en n variables (utilizando un orden preestablecido de los monomios). De acuerdo a nuestra definición, esta cantidad es exactamente el número de monomios de grado menor o igual que d en n variables, es decir, $\binom{d+n}{d}$.
Si consideramos que la cantidad de variables n con las que trabajamos está fija, pero los grados de los polinomios pueden cambiar, entonces si $d \geq 2$ se tiene que

$$\binom{d+n}{d} = \prod_{1 \leq i \leq n} \frac{d+i}{i} \leq 2d^n.$$

Más aún, asintóticamente en d, $\binom{d+n}{d}$ y d^n son del mismo orden, puesto que

$$\frac{d^n}{\prod_{1 \leq i \leq n} \frac{d+i}{i}} \leq n!.$$

En este sentido diremos que un polinomio de grado $d \geq 2$ en n variables tiene $O(d^n)$ coeficientes.

3.5 Un algoritmo para el teorema de los ceros

Supongamos que tenemos una función $\varphi : \mathbb{N}^3 \to \mathbb{N}$ que satisface:

Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ *tales que* $\deg(f_i) \leq d$ $(1 \leq i \leq s)$. *Si* $1 \in (f_1, \ldots, f_s)$, *existen polinomios* $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ *con* $\deg(g_i) \leq \varphi(n, s, d)$ $(1 \leq i \leq s)$ *tales que* $\sum_{1 \leq i \leq s} g_i . f_i = 1$.

Entonces podemos reducir el problema de decidir si un sistema de ecuaciones polinomiales tiene solución o no a un problema de álgebra lineal que puede ser resuelto algorítmicamente y por lo tanto tenemos un teorema de los ceros efectivo.
Para cada coeficiente de cada uno de los polinomios $g_1, \ldots, g_s$ hasta grado $\varphi(n, s, d)$ introducimos una nueva variable. Luego la igualdad $1 = \sum_{1 \leq i \leq s} g_i f_i$ se puede ver, igualando los coeficientes de ambos miembros, como un sistema de $O((\varphi(n, s, d) + d)^n)$ ecuaciones lineales en $O(s\varphi(n, s, d)^n)$ incógnitas.
Si este sistema no tiene solución, el sistema de ecuaciones polinomiales $f_1 = 0, \ldots, f_s = 0$ tendrá soluciones en $\overline{k}^n$. Por el contrario, si el sistema lineal tiene solución, cada una de sus soluciones provee polinomios que nos dan la escritura del 1.
La complejidad de este procedimiento será de orden $O(s^2(\varphi(n, s, d) + d)^{3n})$ (ver Proposición 1). Utilizando las mejores cotas conocidas para el teorema de los ceros (ver Sección 3.3) la complejidad de este algoritmo es de orden $(sd^{n^2})^{O(1)}$.

4 Geometría algebraica efectiva

4.1 Algunas definiciones de geometría algebraica

En lo que sigue daremos algunas definiciones y propiedades básicas de geometría algebraica, que pueden encontrarse por ejemplo en [36], [30] y [5].

Sea k un cuerpo y sea $\overline{k}$ una clausura algebraica de k.
Diremos que un subconjunto $V \subset \overline{k}^n$ es una *variedad algebraica afín* si V es el conjunto de los ceros comunes de una familia de polinomios en n variables con coeficientes en $\overline{k}$. Análogamente, diremos que un subconjunto $V \subseteq \overline{k}^n$ es una *variedad algebraica afín definible sobre* k o una *k-variedad algebraica* si V es el conjunto de los ceros comunes en $\overline{k}^n$ de una familia de polinomios en n variables con coeficientes en k. Si $f_1, \ldots, f_s$ son polinomios en $k[x_1, \ldots, x_n]$ notaremos por $V(f_1, \ldots, f_s)$ al conjunto de sus ceros comunes en $\overline{k}^n$.
Se define el espacio afín n-dimensional sobre $\overline{k}$, que notaremos por $\mathbb{A}^n(\overline{k})$ (o simplemente $\mathbb{A}^n$), como el conjunto $\overline{k}^n$ provisto de la topología cuyos cerrados son las variedades algebraicas afines de $\overline{k}^n$. Esta topología se llama la *topología de Zariski* de $\overline{k}^n$. Puesto que mediante uniones finitas e intersecciones arbitrarias de k-variedades se obtienen nuevamente k-variedades, las k-variedades pueden tomarse como los conjuntos cerrados de una topología en $\mathbb{A}^n$, que se llama la *topología de Zariski de* $\mathbb{A}^n$ *respecto de* k.

Diremos que una propiedad se verifica para $x \in \mathbb{A}^n$ *genérico* o simplemente que se verifica *genéricamente* si existe un abierto de Zariski $U \subseteq \mathbb{A}^n$ no vacío tal que la propiedad vale para todo $x \in U$.

Una k-variedad algebraica V se dice *irreducible* si verifica la siguiente propiedad: si para dos k-variedades V_1 y V_2 vale $V = V_1 \cup V_2$, entonces $V = V_1$ o $V = V_2$.
Para una k-variedad algebraica afín arbitraria $V \subseteq \mathbb{A}^n$, existe una familia finita $\{C_i : 1 \leq i \leq t\}$ de k-variedades irreducibles de $\mathbb{A}^n$, unívocamente determinada por V, tal que

$$V = \bigcup_{i=1}^{t} C_i, \qquad C_i \not\subseteq C_j \text{ si } i \neq j.$$

Esta representación se llama la *descomposición irreducible (minimal) de* V y las variedades C_i $(1 \leq i \leq t)$, las *componentes irreducibles* de V.

Uno de los parámetros más importantes asociados a una variedad algebraica es su dimensión. Sea $V \subseteq \mathbb{A}^n$ una variedad algebraica afín no vacía. Se define la *dimensión de* V, que será denotada por $\dim V$, como el máximo entero r tal que una variedad lineal genérica en $\mathbb{A}^n$ de dimensión $n-r$ interseca a V en un conjunto no vacío. Observamos que una variedad lineal de dimensión $n-r$ es la intersección de r hiperplanos y puede representarse entonces por medio de un elemento en $\mathbb{A}^{r\times(n+1)}$ (cada fila representa los coeficientes de una ecuación de uno de los hiperplanos), y en este sentido hablamos de genericidad. Se define también $\dim(\emptyset) = -1$.
Sea $V = \bigcup_{i=1}^{t} C_i$ la descomposición irreducible de V. Para cada $0 \leq \ell \leq n$, consideremos la unión de todas las componentes irreducibles de V de dimensión ℓ:

$$V_\ell = \bigcup_{\substack{1\leq i\leq t\\ \dim C_i=\ell}} C_i.$$

Observamos que, para cada $0 \leq \ell \leq r$, V_ℓ es o bien el conjunto vacío o bien una variedad *equidimensional*, es decir, una variedad cuyas componentes irreducibles tienen todas la misma dimensión. Llamamos la *descomposición equidimensional de* V a la representación

$$V = \bigcup_{\ell=0}^{n} V_\ell.$$

Las variedades V_ℓ se llaman las *componentes equidimensionales* de V.

Otro de los parámetros intrínsecos asociados a una variedad algebraica es su grado (ver [19]). La noción de grado de una variedad se define en primer lugar para el caso de una variedad afín irreducible y luego se extiende al caso general.
Si $V \subseteq \mathbb{A}^n$ es una variedad algebraica afín irreducible de dimensión r, el *grado* de V es

$$\begin{aligned}\deg V := \sup\{\#H_1 \cap \ldots \cap H_r \cap V \ ;\ & H_1, \ldots, H_r \text{ hiperplanos afines}\\ & \text{en } \mathbb{A}^n \text{ tales que } H_1 \cap \ldots \cap H_r \cap V \text{ es un conjunto finito}\}.\end{aligned}$$

Este supremo es finito. Para una variedad algebraica afín arbitraria $V \subseteq \mathbb{A}^n$, se define $\deg V$ como la suma de los grados de todas las componentes irreducibles de V.
Con esta definición vale la siguiente propiedad, conocida como *desigualdad de Bézout* (ver [19]): Si $V, W \subseteq \mathbb{A}^n$ son dos variedades algebraicas, entonces $\deg(V \cap W) \leq \deg V . \deg W$.
Como consecuencia de esta desigualdad, puede probarse que si $V \subset \mathbb{A}^n$ es una variedad definida por polinomios de grados acotados por d, entonces $\deg V \leq d^n$.

4.2 Un algoritmo para calcular dimensiones

De la definición de dimensión de variedades dada en la sección anterior se desprende inmediatamente el siguiente algoritmo para el cálculo de la dimensión de una variedad algebraica $V := V(f_1, \ldots, f_s) \subset \mathbb{A}^n$ donde $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$:
Sean T_{ij} $(1 \leq i \leq n, 0 \leq j \leq n)$ nuevas indeterminadas sobre k y sea K la clausura algebraica de $k(T_{ij})_{\substack{1\leq i\leq n\\ 0\leq j\leq n}}$. Para cada $1 \leq i \leq n$, sea $L_i := T_{i0} + T_{i1}x_1 + \cdots + T_{in}x_n$.

Por abuso de notación, seguiremos llamando V a la variedad definida por los polinomios $f_1, \ldots, f_s$ en K^n, que conserva la dimensión de la variedad original. Consideremos las variedades $V^{(i)} := V \cap V(L_1, \ldots, L_i)$ para $i = 1, \ldots, n$, y $V^{(0)} := V$. Entonces, si $V \neq \emptyset$, $\dim V = \max\{i : 0 \leq i \leq n, V^{(i)} \neq \emptyset\}$. Como consecuencia del teorema de los ceros de Hilbert, la condición $V^{(i)} = \emptyset$ es equivalente a

$$1 \in (f_1, \ldots, f_s, L_1, \ldots, L_i) \subset K[x_1, \ldots, x_n],$$

la cual puede verificarse mediante una versión efectiva de este teorema.
La complejidad asociada a este procedimiento es muy alta por la cantidad de variables libres que tenemos que agregar. Más adelante mencionaremos métodos más efectivos para resolver este problema usando esta misma idea (ver Sección 6.1).

A continuación daremos una definición alternativa de dimensión, más relacionada con un punto de vista algebraico que con uno geométrico, que será usada a su vez para obtener otro algoritmo para el cálculo de la dimensión. (Para la equivalencia de estas dos definiciones de dimensión, ver [36].)

Si $V = V(I)$ con I un ideal de $k[x_1, \ldots, x_n]$, entonces

$$\dim V = \max\{t \in \mathbb{N}_0 : \exists\, 1 \leq i_1, \ldots, i_t \leq n \text{ tales que } I \cap k[x_{i_1}, \ldots, x_{i_t}] = \{0\}\}. \qquad (5)$$

Teniendo en cuenta esta definición, vemos que para calcular la dimensión de la variedad basta verificar si $I \cap k[x_{i_1}, \ldots, x_{i_t}] = \{0\}$ para subconjuntos $\{i_1, \ldots, i_t\}$ de $\{1, \ldots, n\}$.
El siguiente resultado (ver [7, Proposition 1.7]) nos permite reducir esto a un problema de álgebra lineal sobre k:

Proposición 16 *Sea $V = V(f_1, \ldots, f_s)$ con $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por d y sea $I := (f_1, \ldots, f_s)$ el ideal de $k[x_1, \ldots, x_n]$ generado por estos polinomios. Las siguientes condiciones son equivalentes:*

i) $I \cap k[x_{i_1}, \ldots, x_{i_t}] \neq \{0\}$

ii) Existe $g \in k[x_{i_1}, \ldots, x_{i_t}]$ tal que $g \neq 0$ y $g = \sum_{i=1}^{s} b_i f_i$ con $b_i \in k[x_1, \ldots, x_n]$ y $\deg(b_i f_i) \leq d^n(\deg V + 1)$.

Utilizando esta equivalencia y teniendo en cuenta que si $V = V(f_1, \ldots, f_s)$ con $f_1, \ldots, f_s$ polinomios en $k[x_1, \ldots, x_n]$ de grados acotados por d, se tiene $\deg V \leq d^n$ (consecuencia de la desigualdad de Bézout), por comparación de coeficientes podemos reducir el problema de determinar si $I \cap k[x_{i_1}, \ldots, x_{i_t}] \neq \{0\}$ al de decidir si un sistema lineal homogéneo de tamaño $O(sd^{n^2}) \times O(sd^{n^2})$ tiene una solución con algunas coordenadas no todas nulas (las que corresponden a los coeficientes de g).
Esto puede hacerse mediante los algoritmos descriptos en la Sección 2.3 con complejidad de orden $s^{O(1)}d^{O(n^2)}$. Repitiendo esto para (a lo sumo) los 2^n subconjuntos de $\{x_1, \ldots, x_n\}$ se obtiene un conjunto maximal de variables $x_{i_1}, \ldots, x_{i_r}$ que satisfacen $I \cap k[x_{i_1}, \ldots, x_{i_r}] = \{0\}$, y por lo tanto la dimensión de V, con complejidad $s^{O(1)}d^{O(n^2)}$.

Teorema 17 *Sea $V \subset \mathbb{A}^n$ una variedad definida como el conjunto de los ceros comunes de s polinomios en $k[x_1, \ldots, x_n]$ de grados acotados por d. Entonces, la dimensión de V puede calcularse por medio de un algoritmo de complejidad $s^{O(1)}d^{O(n^2)}$.*

4.3 Resolución geométrica de variedades cero-dimensionales

En el caso de una variedad cero-dimensional, es posible dar una representación paramétrica de la variedad que resulta ser muy útil desde el punto de vista algorítmico. Esta manera de describir variedades (que ya era conocida por Kronecker, ver [29]) recibe el nombre de *resolución geométrica*. La idea es la siguiente (supongamos por el momento que k es algebraicamente cerrado):
Sea $V \subset \mathbb{A}^n$ una variedad cero-dimensional definida por polinomios en $k[x_1, \ldots, x_n]$ que consiste de D puntos

$$\xi^{(1)} = (\xi_1^{(1)}, \ldots, \xi_n^{(1)}), \ldots, \xi^{(D)} = (\xi_1^{(D)}, \ldots, \xi_n^{(D)}).$$

Eligiendo convenientemente $u_1, \ldots, u_n \in k$, es posible hallar una forma lineal $\ell(x) = u_1 x_1 + \ldots + u_n x_n$ en $k[x_1, \ldots, x_n]$ tal que los valores $\ell(\xi^{(i)})$ para $i = 1, \ldots D$ sean todos distintos. Entonces, existe un polinomio univariado no nulo y libre de cuadrados $p \in k[T]$ de grado D cuyos ceros son exactamente estos valores:

$$p = \prod_{1 \leq i \leq D} (T - \ell(\xi^{(i)})).$$

En este caso decimos que ℓ es un *elemento primitivo* de V o que ℓ *separa los puntos* de V y que p es el *polinomio minimal* de ℓ con respecto a V.
Además, para cada índice j con $1 \leq j \leq n$, existe un único polinomio $v_j \in k[T]$ de grado acotado por $D-1$ tal que $v_j(\ell(\xi^{(i)})) = \xi_j^{(i)}$ para cada $1 \leq i \leq D$ (el polinomio interpolador correspondiente a los puntos $(\ell(\xi^{(i)}), \xi_j^{(i)})$). Resulta entonces que

$$\begin{aligned} V &= \{\xi \in \mathbb{A}^n \ / \ p(\ell(\xi)) = 0,\ \xi_1 - v_1(\ell(\xi)) = 0,\ \xi_2 - v_2(\ell(\xi)) = 0, \ldots,\ \xi_n - v_n(\ell(\xi)) = 0\} \\ &= \{(v_1(t), v_2(t), \ldots, v_n(t)) \ / \ t \in k,\ p(t) = 0\}. \end{aligned}$$

Decimos que esta es una descripción paramétrica de V en el sentido que las coordenadas de los puntos de V están dadas en función de un parámetro t (que a su vez puede tomar finitos valores, que son los ceros del polinomio p).
Si realizamos esta construcción para una k-variedad cero-dimensional V, con k un cuerpo cualquiera, y una forma lineal ℓ con coeficientes en k resulta que los polinomios $p, v_1, \ldots, v_n$ también tienen coeficientes en el cuerpo de base k.
Esto da lugar a la siguiente definición de resolución geométrica de una variedad cero-dimensional:

Definición 18 *Sea $V = \{\xi^{(1)}, \ldots, \xi^{(D)}\} \subset \overline{k}^n$ una variedad cero-dimensional definida por polinomios en $k[x_1, \ldots, x_n]$. Una* resolución geométrica *de V consiste de una forma lineal $\ell(x) = u_1 x_1 + \ldots + u_n x_n \in k[x_1, \ldots, x_n]$, y polinomios $p, v_1, \ldots, v_n \in k[T]$ (donde T es una variable nueva) tales que:*

- $\ell(\xi^{(i_1)}) \neq \ell(\xi^{(i_2)})$ *si* $i_1 \neq i_2$.
- $p(T) = \prod_{1 \leq i \leq D} (T - \ell(\xi^{(i)}))$.
- *Para cada* $1 \leq j \leq n$, $\deg(v_j) \leq D - 1$ *y* $V = \{(v_1(t), \ldots, v_n(t)) \ / \ t \in \overline{k},\ p(t) = 0\}$.

Como esta descripción de V está unívocamente determinada por la forma lineal ℓ, la llamaremos *la* resolución geométrica de V asociada a ℓ.

Observamos que la definición de resolución geométrica fue dada a partir de las coordenadas de los puntos de V. El problema es entonces cómo hallar un elemento primitivo ℓ para V y los polinomios asociados $p, v_1, \ldots, v_n$ sin conocer dichos puntos, sino a partir de polinomios $f_1, \ldots, f_s$ cuyo conjunto de ceros comunes es la variedad V. Los dos ingredientes fundamentales para esto son poder encontrar resoluciones geométricas de ciertas variedades cero-dimensionales en $\mathbb{A}^2$ (lo que se estudiará más adelante) y resolver el siguiente problema:

Dada una forma lineal $\ell \in k[x_1, \ldots, x_n]$, hallar un polinomio no nulo $p \in k[T]$ libre de cuadrados cuyos ceros sean exactamente los valores de ℓ en los puntos de la variedad cero-dimensional $V = V(f_1, \ldots, f_s) \subset \mathbb{A}^n$, donde $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ son polinomios de grados acotados por d.

Haciendo un cambio lineal de variables, podemos suponer que $\ell = x_1$.
Sea $p \in k[T]$ el polinomio mónico cuyos ceros son las primeras coordenadas de los puntos de V. Entonces $p \in \operatorname{rad}(f_1, \ldots, f_s)$, y por lo tanto (ver [7, Remark 1.6]), existen polinomios $p_1, \ldots, p_s \in k[x_1, \ldots, x_n]$ tales que

$$p^{d^n} = \sum_{i=1}^{s} p_i f_i, \qquad \deg(p_i f_i) \leq d^n(\deg p + 1) \leq d^n(d^n + 1).$$

Podemos encontrar un polinomio no nulo $q \in k[T]$ cuyos ceros coinciden con los ceros de p. Para esto basta hallar el mínimo $d_0 \leq d^n(d^n + 1)$ tal que el sistema lineal no homogéneo determinado por las condiciones:

- $q \in k[T]$, mónico de grado d_0;
- $q(x_1) = \sum_{i=1}^{s} p_i f_i, \quad \deg(p_i f_i) \leq d^n(d^n + 1)$

sea consistente, y hallar una solución para el sistema asociado a ese grado d_0.
El polinomio p buscado es

$$p := \frac{q}{\gcd(q, q')}.$$

Teniendo en cuenta que el cálculo del gcd y la división exacta de polinomios en una variable pueden efectuarse con complejidad polinomial en el grado de los polinomios involucrados, hemos demostrado el siguiente:

Lema 19 *Sea $V := V(f_1, \ldots, f_s) \subset \mathbb{A}^n$ una variedad cero-dimensional definida por polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ de grados acotados por d, y sea $\ell \in k[x_1, \ldots, x_n]$ una forma lineal. Existe un algoritmo de complejidad polinomial en sd^{n^2} que calcula un polinomio $p \in k[T]$ libre de cuadrados cuyos ceros son los valores de ℓ en los puntos de V.*

Observar que si ℓ es un elemento primitivo de V, el polinomio p que obtenemos es el asociado a ℓ en la resolución geométrica de V correspondiente.

El siguiente lema nos permite encontrar algorítmicamente una resolución geométrica de una variedad cero-dimensional en $\mathbb{A}^2$ definida por polinomios en variables separadas.

Lema 20 *Sean $f(x) \in k[x]$, $g(y) \in k[y]$ polinomios no nulos libres de cuadrados. Sea $V = \{(\xi, \eta) \in \mathbb{A}^2 : f(\xi) = 0, g(\eta) = 0\}$. Sea $\delta := \deg V = \deg(f).\deg(g)$. Entonces existe un algoritmo de complejidad $\delta^{O(1)}$ que halla un elemento $\lambda \in k$ tal que la forma lineal $\ell = y + \lambda\, x$ es un elemento primitivo de V y calcula una resolución geométrica de V asociada a ℓ.*

Demostración. Consideremos el polinomio $\hat{g} := g(T - \Lambda x) \in k[\Lambda][T, x]$ con Λ y T nuevas variables, y sea $h := \mathrm{Res}_x(f, \hat{g}) \in k[\Lambda][T]$. Observamos que $\deg_T(h) = \deg(f)\deg(g)$ y, para cada $\lambda \in k$, si $\ell_\lambda := y + \lambda x$, las raíces de $h_\lambda(T) := h(\lambda, T) \in k[T]$ son exactamente los elementos $\ell_\lambda(\xi, \eta) = \eta + \lambda\,\xi$ con $(\xi, \eta) \in V$.
Entonces, ℓ_λ separa los ceros de V si y sólo si h_λ es libre de cuadrados. Por lo tanto, basta elegir $\lambda \in k$ de manera que $\mathrm{Res}_T(h, h')(\lambda) \neq 0$, donde h' es la derivada de h respecto de T. Observar que, como $\deg V = \delta$, hay a lo sumo $\binom{\delta}{2}$ valores de λ tales que ℓ_λ no separa los puntos de V. Luego, basta buscar λ en el conjunto $\{1, \ldots, \binom{\delta}{2} + 1\}$.
Para un valor de λ tal que $\mathrm{Res}_T(h, h')(\lambda) \neq 0$, el polinomio h_λ resulta ser el polinomio p que forma parte de la resolución geométrica de V asociada a ℓ.
Para hallar los polinomios v_1 y v_2, basta observar que $\mathcal{B}_1 := \{\ell_\lambda^j : 0 \leq j < \deg(f)\deg(g)\}$ y $\mathcal{B}_2 := \{x^\alpha y^\beta : 0 \leq \alpha < \deg f,\ 0 \leq \beta < \deg g\}$ son dos bases de $k[x, y]/(f, g)$ como k-espacio vectorial y utilizar la matriz de cambio de base.
Notar que todos los pasos que se efectúan son cálculos de coeficientes de $\hat{g}$ y cálculos de determinantes de matrices de tamaño $O(\delta)$. Por lo tanto, la complejidad total de este algoritmo es de orden $\delta^{O(1)}$. □

Ahora vamos a utilizar los resultados anteriores recursivamente para calcular la resolución geométrica de una variedad cero-dimensional cualquiera a partir de un conjunto de polinomios que la define.

Proposición 21 *Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por d que definen una variedad cero-dimensional V. Existe un algoritmo de complejidad polinomial en sd^{n^2} que calcula una resolución geométrica de V.*

Demostración. La demostración se basa en un algoritmo recursivo (ver [15, Sec. 3.4.6]):
Para $i = 1, \ldots, n$ se calculan una forma lineal $y_i := u_1x_1 + \cdots + u_ix_i$ y polinomios $v_1^{(i)}, \ldots, v_i^{(i)} \in k[T]$ tales que para cada punto $\xi := (\xi_1, \ldots, \xi_n) \in V$ vale $\xi_j = v_j^{(i)}(y_i(\xi))$ para $j = 1, \ldots, i$.

- Para $i = 1$, tomamos $y_1 := x_1$ y $v_1^{(1)} := T$.
- Sea $i \geq 2$, y supongamos dados la variable y_{i-1} y los polinomios $v_j^{(i-1)}$ $(1 \leq j \leq i-1)$ que verifican las condiciones requeridas.

  Calculamos dos polinomios $f(x_i) \in k[x_i]$ y $g(y_{i-1}) \in k[y_{i-1}]$ que se anulan sobre V (ver Lema 19). Sea $Z := \{(\eta_1, \eta_2) \in \mathbb{A}^2 : f(\eta_1) = 0,\ g(\eta_2) = 0\}$. Aplicando el Lema 20 a Z obtenemos un elemento $u_i \in k$ y polinomios p, $v_1, v_2 \in k[T]$ que forman una resolución geométrica de Z asociada a $y_i := y_{i-1} + u_ix_i$.

  Ahora, para la forma lineal y_i, podemos calcular el polinomio $p_i \in k[T]$ dado por el Lema 19 cuyos ceros son exactamente los valores de y_i en los puntos de V. El polinomio $v_i^{(i)}$ se obtiene entonces como el resto de la divisón de v_1 por p_i. Los polinomios $v_1^{(i)}, \ldots, v_{i-1}^{(i)}$ se obtienen reemplazando la variable T por v_2 en los polinomios $v_1^{(i-1)}, \ldots, v_{i-1}^{(i-1)} \in k[T]$ y reduciendo el resultado módulo p_i.

Al terminar el paso n se obtiene una forma lineal $y_n := u_1x_1+u_2x_2+\cdots+u_nx_n$, el polinomio $p_n \in k[T]$ cuyos ceros son los valores de y_n en los puntos de V y polinomios $v_1^{(n)}, \ldots, v_n^{(n)}$ tales que para cada $\xi := (\xi_1, \ldots, \xi_n) \in V$ vale $\xi_j = v_j^{(n)}(y_n(\xi))$ para todo $1 \leq j \leq n$. Observar que la validez de estas últimas igualdades que dan los valores de las coordenadas de ξ a partir de $y_n(\xi)$ implica que y_n es un elemento primitivo de V.
Teniendo en cuenta la elección de la forma lineal en el Lema 20, y que los polinomios $f(x_i)$ y $g(y_{i-1})$ que aparecen en cada paso de la recursión tienen grados acotados por d^n, concluimos que los coeficientes u_i pueden elegirse en el conjunto $\{1, \ldots, d^{4n}\}$.
En cuanto a la complejidad del algoritmo, nótese que en cada paso el cálculo de los polinomios univariados por medio del Lema 19 tiene complejidad polinomial en sd^{n^2}, mientras que la complejidad de cada uno de los cálculos restantes es de orden $d^{O(n)}$. Luego, la complejidad total del algoritmo es polinomial en sd^{n^2}. □

4.4 Variedades equidimensionales de dimensión positiva

Posición de Noether

La siguiente noción nos permite, entre otras cosas, generalizar la noción de resolución geométrica para variedades equidimensionales de cualquier dimensión.

Definición 22 *Sea $V \subset \mathbb{A}^n$ y sea $r := \dim V$. Se dice que las variables $x_1, \ldots, x_n$ están en* posición de Noether con respecto a V *si:*

- $k[x_1, \ldots, x_r] \cap I(V) = \{0\}$ *(en este caso se dice que $x_1, \ldots, x_r$ son variables* libres *con respecto a V).*
- *Para cada $r+1 \leq j \leq n$ existe un polinomio en $k[x_1, \ldots, x_r, x_j]$ que es mónico en x_j y se anula sobre V (entonces se dice que $x_{r+1}, \ldots, x_n$ son* enteras *con respecto a $x_1, \ldots, x_r$ y V).*

Dada $V \subset \mathbb{A}^n$ una variedad de dimensión r y $\{x_1, \ldots, x_r\}$ un conjunto de variables libres con respecto a V, se tiene que para cada $r+1 \leq j \leq n$, existe un polinomio no nulo $q_j \in k[x_1, \ldots, x_r, x_j]$ que se anula sobre V. Este polinomio en general no es mónico, es decir, x_j no es entera con respecto a $x_1, \ldots, x_r$ y V, pero haciendo un cambio lineal de variables es posible obtener esta condición.
Un posible algoritmo para el cálculo de una posición de Noether para una variedad $V = V(f_1, \ldots, f_s) \subset \mathbb{A}^n$, con $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ de grados acotados por d, es el siguiente:

- Hallar un conjunto maximal de variables libres para V (ver Sección 4.2). Supongamos que éstas son $x_1, \ldots, x_r$.
- Para $j = r+1, \ldots, n$:
  - Hallar un polinomio no nulo $q_j \in k[x_1, \ldots, x_r, x_j]$ que se anula sobre V.
    Este polinomio se puede calcular en forma análoga a la vista para el caso cero-dimensional, puesto que existe un polinomio $q \in k[x_1, \ldots, x_r, x_j]$ tal que

$$q = \sum_{i=1}^{s} p_i f_i, \quad \deg(p_i f_i) \leq d^n(d^n + 1).$$

– Hacer un cambio de variables $x_i' := x_i - \lambda_i x_j$ con $\lambda_i \in k$ $(1 \leq i \leq r)$ de manera que $q_j(x_1' + \lambda_1 x_j, \ldots, x_r' + \lambda_r x_j, x_j)$ sea mónico en la variable x_j, en forma similar a como se hizo en la demostración del Teorema 15.

Teniendo en cuenta estos pasos y efectuando el cálculo de la complejidad tenemos:

Proposición 23 *Sea $V \subset \mathbb{A}^n$ una variedad definida por s polinomios en $k[x_1, \ldots, x_n]$ de grados acotados por d. Existe un algoritmo de complejidad polinomial en sd^{n^2} que calcula una posición de Noether para V.*

Resolución geométrica de variedades equidimensionales

Para variedades equidimensionales en general podemos dar una descripción paramétrica similar a la dada en la Definición 18 para variedades cero-dimensionales.
Sea $V \subset \mathbb{A}^n$ una variedad equidimensional de dimensión r definida por los polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$. Supongamos que las variables $x_1, \ldots, x_n$ están en posición de Noether y que $x_1, \ldots, x_r$ son las variables libres. Entonces, si consideramos el cuerpo $K = k(x_1, \ldots, x_r)$, la variedad $\widetilde{V}$ definida por $f_1, \ldots, f_s$ en una clausura algebraica $\overline{K}$ de K resulta ser cero-dimensional. Dada una forma lineal $\ell \in k[x_{r+1}, \ldots, x_n]$ que separa los puntos de $\widetilde{V}$ podemos considerar la resolución geométrica $(p, v_{r+1}, \ldots, v_n) \in (k(x_1, \ldots, x_r)[T])^{n-r+1}$ de $\widetilde{V}$ asociada a ℓ.
La hipótesis de posición de Noether nos asegura que p está en $k[x_1, \ldots, x_r][T]$ y es un polinomio mónico. Se puede probar que existe $\rho \in k[x_1, \ldots, x_r]$ tal que $\widetilde{v}_{r+1} := \rho v_{r+1}, \ldots, \widetilde{v}_n := \rho v_n \in k[x_1, \ldots, x_r][T]$ y

$$V \cap \{\rho \neq 0\} = \{\xi \in \mathbb{A}^n : p(\ell(\xi)) = 0, \rho(\xi_1, \ldots, \xi_r).\xi_{r+1} - \widetilde{v}_{r+1}(\xi_1, \ldots, \xi_r, \ell(\xi)) = 0, \ldots,$$
$$\rho(\xi_1, \ldots, \xi_r).\xi_n - \widetilde{v}_n(\xi_1, \ldots, \xi_r, \ell(\xi)) = 0\} \cap \{\rho \neq 0\}.$$

Bajo estas condiciones diremos que $p, \rho, \widetilde{v}_{r+1}, \ldots, \widetilde{v}_n$ es una *resolución geométrica* de V asociada a ℓ.

4.5 Descomposición equidimensional

Un problema fundamental relacionado con la descripción de una variedad algebraica de dimensión positiva consiste en el cálculo efectivo de su descomposición equidimensional. El problema es, dados polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ que definen una variedad $V \subset \mathbb{A}^n$ cuya descomposición equidimensional es $V = \bigcup_{\ell=0}^{n} V_\ell$, hallar una descripción de cada una de las componentes equidimensionales V_ℓ de V.

Entre los primeros algoritmos que se construyeron para el cálculo la descomposición equidimensional de una variedad podemos mencionar el de A. Chistov y D. Grigor'ev (ver [4]) y el que presentan M. Giusti y J. Heintz en [14]. En el primero se describe cada componente de la variedad de dos formas: primero por medio de un punto genérico (resolución geométrica) y luego como el conjunto de los ceros comunes de una familia de polinomios. En [14], se usa sólo esta segunda descripción. En ambos casos, si la variedad está dada como el conjunto de los ceros comunes de s polinomios en n variables de grados acotados por d, la complejidad del algoritmo es de orden $s^{O(1)}d^{O(n^2)}$.
Más aún, el algoritmo descripto en [4] calcula la descomposición de la variedad en componentes irreducibles utilizando un algoritmo de factorización de polinomios multivariados con coeficientes en el cuerpo de base k.

Si bien no vamos a entrar en detalles acerca de cómo funcionan estos algoritmos, más adelante describiremos otro algoritmo que encuentra la descomposición equidimensional de una variedad (ver Sección 6.6) con herramientas que permiten obtener una mejor complejidad.

5 Codificación de polinomios por straight-line programs

5.1 Cotas inferiores de complejidad debidas a la representación densa

En esta sección, vamos a mostrar que el orden de las mejores cotas superiores que se conocen para el teorema de los ceros efectivo (mencionadas al final de la Sección 3.3) es óptimo. Para esto exhibiremos un ejemplo muy conocido, debido a Masser y Philippon (ver [2]).

Ejemplo 24 Sean $f_1, \ldots, f_n \in \mathbb{Q}[x_1, \ldots, x_n]$ los polinomios:

$$f_1 = x_1^d,\ f_2 = x_1 - x_2^d, \ldots,\ f_{n-1} = x_{n-2} - x_{n-1}^d,\ f_n = 1 - x_{n-1}x_n^{d-1}.$$

Es fácil ver que estos polinomios no tienen ceros comunes en una clausura algebraica de $\mathbb{Q}$. Supongamos que $g_1, \ldots, g_n \in \mathbb{Q}[x_1, \ldots, x_n]$ son polinomios tales que $1 = \sum_{1 \leq i \leq n} g_i f_i$. Consideremos una nueva variable T y evaluemos los polinomios en el vector

$$\xi_T := (T^{(d-1)d^{n-2}}, \ldots, T^{d-1}, 1/T) \in \mathbb{Q}(T)^n.$$

Observamos que para $i = 2, \ldots, n$, el polinomio f_i se anula en ξ_T y entonces se tiene que

$$1 = g_1(T^{(d-1)d^{n-2}}, \ldots, T^{d-1}, 1/T)\, T^{(d-1)d^{n-1}}.$$

Esta igualdad implica que $\deg_{x_n}(g_1) \geq (d-1)d^{n-1}$ y por lo tanto $\deg(g_1) \geq (d-1)d^{n-1}$.

Como consecuencia de este ejemplo, se deducen las siguientes cotas inferiores para la complejidad de algoritmos que usan la representación densa de polinomios:

Proposición 25 *Todo algoritmo* general *que, tomando como input un conjunto de s polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ de grados acotados por d, calcula (si es que existen) polinomios $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ tales que $1 = \sum_{1 \leq i \leq s} g_i . f_i$ y los codifica en* forma densa *tiene complejidad de orden al menos $O(d^{n^2})$.*

5.2 Straight-line programs

Lo visto en la sección anterior muestra que es imposible obtener algoritmos *generales* más eficientes que los conocidos que trabajen con polinomios representados en *forma densa*. Así, dos enfoques posibles en la búsqueda de mejores algoritmos son los siguientes: cambiar la manera en que se codifican los polinomios (es decir, encontrar una forma más corta de representar los polinomios con los que trabajan los algoritmos) o diseñar algoritmos no generales que puedan resolver sólo problemas específicos pero con mejores cotas de complejidad. Nos concentraremos en el primero de estos enfoques.

Una manera alternativa de codificar polinomios es la denominada *representación rala*, que consiste en representar un polinomio $P \in k[x_1, \ldots, x_n]$ (con "pocos" monomios en relación a su grado) por medio de un vector de $(n+1)$-uplas que especifican cuáles son los monomios que aparecen con coeficiente no nulo en P y cuál es este coeficiente. Por ejemplo, el polinomio

$P = 2x^{15}y^4 + 2x^7y^3 - 3x^2 + 1$ puede codificarse por medio de un vector de cuatro ternas, cada una de las cuales está asociada a uno de los cuatro monomios que aparecen en P. En cada terna, el primer coeficiente indica el grado del monomio en la variable x; el segundo, su grado en y; y el tercero, el coeficiente con que aparece dicho monomio en P. En definitiva, P puede codificarse con la representación rala

$$P := ((15, 4, 2); (7, 3, 2); (2, 0, -3); (0, 0, 1)),$$

en lugar de un vector de $\binom{21}{2} = 210$ coordenadas que requeriría su representación en forma densa.
Este tipo de codificación ha demostrado ser eficiente para trabajar con familias particulares de polinomios y se ha desarrollado una gran cantidad de teoría y diversos algoritmos que utilizan la representación rala de polinomios. Para el estudio de esta teoría (incluyendo resultantes ralas, polítopos de Newton, variedades tóricas y el teorema de Bernstein, entre otras nociones), ver [6], [12] y [11].
Sin embargo, no es claro que este tipo de representación lleve a buenas cotas de complejidad para algoritmos *generales*: el output del algoritmo podría contener demasiados monomios. Otro problema que surge al utilizar esta representación es que no presenta un comportamiento adecuado en relación a cambios lineales de coordenadas, en el sentido que un polinomio "corto" en forma rala puede transformarse en uno muy "largo" mediante un cambio lineal de variables: observar, por ejemplo, que $(x + y)^{100} = \sum_{0 \leq i \leq 100} \binom{100}{i} x^i y^{100-i}$ (es decir, un único monomio se puede transformar en un polinomio con muchos coeficientes).

Otra forma de codificar polinomios (que es la que vamos a estudiar en lo sucesivo) se basa en la idea de considerar un polinomio como una función que puede ser evaluada en lugar de una "expresión formal" (para cuerpos de característica cero, las funciones polinomiales y los polinomios pueden considerarse como los "mismos" objetos). Si definimos una función polinomial dando su valor exacto en cada punto (es decir, describiendo cómo evaluarla), estaremos definiendo un polinomio. Por ejemplo, el polinomio $(x + y)^{100} - 1$ es el único polinomio en $\mathbb{Q}[x, y]$ que se evalúa en un punto (x, y) calculando la suma de x e y, elevando el resultado a la potencia 100 y restándole luego 1. Llamaremos un *straight-line program* a esta manera de representar un polinomio. Más precisamente:

Definición 26 *Sean $x_1, \ldots, x_n$ indeterminadas sobre k y sea $M \in \mathbb{N}$. Un* straight-line program *(slp) es un elemento $\beta := (Q_1, \ldots, Q_M) \in k(x_1, \ldots, x_n)^M$ tal que cada Q_ρ satisface una de las condiciones siguientes:*

- *$Q_\rho \in k \cup \{x_1, \ldots, x_n\}$ o*
- *Existen $\rho_1, \rho_2 < \rho$ y $* \in \{+, -, \cdot, \div\}$ tales que $Q_\rho = Q_{\rho_1} * Q_{\rho_2}$.*

Decimos que β es un straight-line program sin divisiones *si $Q_\rho = Q_{\rho_1} \div Q_{\rho_2} \Rightarrow Q_{\rho_2} \in k - \{0\}$.*

Vamos a trabajar solamente con straight-line programs *sin divisiones*. Observamos que, en este caso, cada elemento Q_ρ es un polinomio en $k[x_1, \ldots, x_n]$. Si $f \in \{Q_\rho \ / \ 1 \leq \rho \leq M\}$, diremos que β *calcula* f.
Dado un straight-line program β es posible asociarle distintas medidas de complejidad. Por ejemplo:

- La *longitud total* de β (denotada por $L(\beta)$) es la cantidad de operaciones realizadas en el desarrollo de β (más precisamente, es la cantidad de coordenadas Q_ρ que se obtienen como el resultado de una operación entre dos coordenadas anteriores).

- La *longitud aditiva* de β ($L_{\pm}(\beta)$) es la cantidad de sumas y restas realizadas durante el slp.
- La *longitud no escalar* de β ($L_k(\beta)$) es la cantidad de productos entre dos elementos que no pertenecen a k que se efectúan durante el slp.

Dado un polinomio $f \in k[x_1, \ldots, x_n]$ definimos su *longitud total* (también llamada *complejidad total*) como

$$L(f) := \min\{L(\beta) \ / \ \beta \text{ es un slp que calcula } f\}.$$

En forma análoga se definen $L_{\pm}(f)$ y $L_k(f)$.
De ahora en más, a menos que se especifique lo contrario, consideraremos solamente la longitud total de un slp y de un polinomio, a la que llamaremos simplemente su *longitud*.

En general, dado un polinomio $f \in k[x_1, \ldots, x_n]$ de grado d, podemos construir un straight-line program que evalúe a f a partir de su representación densa: se calculan todos los monomios de grado menor o igual que d en $x_1, \ldots, x_n$, se multiplica cada uno por el coeficiente con el que aparece en f y se suman los resultados. Este slp tendrá longitud acotada por $3\binom{d+n}{n}$. Por lo tanto, siempre existe un slp que calcula un polinomio dado cuya longitud es del mismo orden que la complejidad de la representación densa del polinomio. Sin embargo, para algunos polinomios pueden darse straight-line programs de longitud sensiblemente menor que los evalúan.
Como ejemplo, vamos a mostrar un straight-line program "corto" que calcula el polinomio $f = 1 + x + x^2 + x^3 + \ldots + x^{2^j-1}$. Una primera forma de evaluar este polinomio sería calcular cada una de las potencias de x y luego efectuar la suma, pero esto llevaría a un slp de longitud $2^{j+1} - 3$. Un slp más corto para f, basado en la representación binaria de enteros, es el siguiente:

$$\begin{aligned} \beta \ := \ & \Big(1, x, x^2, x^4, \ldots, x^{2^{j-1}}, 1+x, 1+x^2, 1+x^4, \ldots, 1+x^{2^{j-1}}, \\ & (1+x)(1+x^2), (1+x)(1+x^2)(1+x^4), \ldots, \prod_{0 \le i \le j-1} (1+x^{2^i})\Big), \end{aligned}$$

que verifica $L(\beta) = 3j - 2$. (Comparar con el tamaño 2^j de la representación densa.)
Otro ejemplo conocido de slp para la codificación de polinomios en una variables es la regla de Horner:

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_d x^d = (a_0 + x(a_1 + x(a_2 + x(\ldots(a_{d-1} + a_d x)\ldots)))).$$

La longitud de este slp es $2d$ e involucra d productos y d sumas. Se puede probar que, si los elementos $x, a_0, \ldots, a_d$ son algebraicamente independientes, las cantidades de sumas y de productos involucrados en *cualquier* slp que calcule este polinomio son mayores o iguales que d (ver, por ejemplo, [3]).
Finalmente observamos que, si un polinomio $f \in k[x_1, \ldots, x_n]$ está dado por un slp de longitud L y se conoce una cota d para su grado, es posible obtener su representación densa con complejidad de orden $d^{O(n)}L$ interpolándolo en un conjunto de tipo $\mathcal{A}^n$, donde $\mathcal{A}$ es un subconjunto de k con $d+1$ elementos.

5.3 Algunos problemas

Un problema fundamental que surge al trabajar con polinomios codificados por straight-line programs es que un mismo polinomio puede ser codificado por medio de distintos slp's y, en consecuencia, podría ser difícil verificar una igualdad entre polinomios.

Supongamos dado un slp de longitud L que calcula un polinomio f en n variables de grado acotado por d. Si queremos saber si $f \equiv 0$, una forma sería hallar los coeficientes de f por medio de un proceso de interpolación, pero esto requeriría evaluar f en demasiados puntos, llevando entonces a una complejidad demasiado alta (del mismo orden que el tamaño de la representación densa de f). Otra forma de hacer esto consiste en hallar un conjunto de puntos particular, más chico, con la propiedad que un polinomio de longitud y grado acotados es el polinomio nulo si y sólo si su evaluación en cada punto del conjunto da cero. Un resultado debido a Heintz y Schnorr establece la existencia de conjuntos de este tipo:

Teorema 27 *(ver [21, Theorem 4.4]) Sea $\widetilde{W}(d,n,L) \subset k[x_1,\ldots,x_n]$ el conjunto de los polinomios de grado acotado por d en n variables que pueden calcularse por medio de straight-line programs de longitud L. Sea $\Gamma \subset k$ un conjunto de $2L(1+d)^2$ elementos. Entonces existe un conjunto de puntos $\{\alpha_1,\ldots,\alpha_\mu\} \subset \Gamma^n$ con $\mu = 6(L+n)(L+n+1)$ que satisface:*

$$f \in \widetilde{W}(d,n,L) \text{ tal que } f(\alpha_i) = 0 \ \forall\, 1 \le i \le \mu \implies f \equiv 0.$$

A un conjunto $\{\alpha_1,\ldots,\alpha_\mu\}$ con la propiedad enunciada en el teorema anterior lo llamaremos una *correct test sequence* para $\widetilde{W}(d,n,L)$.
Desafortunadamente, no se conocen algoritmos de complejidad "razonable" que calculen correct test sequences. En la práctica, las correct test sequences necesarias en el desarrollo de un algoritmo se consideran obtenidas mediante un pre-procesamiento cuyo costo no se incluye en la complejidad del algoritmo en cuestión, o bien se las elige al azar en un conjunto de cardinal apropiado, dando lugar a algoritmos probabilísticos (esta noción será estudiada en la Sección 6.6).

El siguiente tema que analizaremos es cuántos polinomios son "fáciles de evaluar", es decir, pueden calcularse por medio de slp's cortos. Como veremos, la respuesta no es muy alentadora (ver [34] y [22]).
Para n, d y L fijos, consideremos el conjunto de todos los polinomios $f \in k[x_1,\ldots,x_n]$ con $\deg(f) \le d$ y longitud no escalar $L_k(f) \le L$. Cada uno de estos polinomios puede ser evaluado por un slp "no escalar" (teniendo en cuenta sólo las coordenadas del slp que son productos entre elementos no escalares) de tipo

$$\beta := (\beta_{-n+1},\ldots,\beta_0,\beta_1,\ldots,\beta_L),$$

donde $\beta_{-n+i} = x_i$ $(1 \le i \le n)$ y, si definimos $\beta_{-n} := 1$,

$$\beta_\ell = \Big(\sum_{-n\le j\le \ell-1} a_j^{(\ell)}.\beta_j\Big).\Big(\sum_{-n\le j\le \ell-1} b_j^{(\ell)}\beta_j\Big) \qquad \ell = 1,\ldots,L,$$

para algunos $a_j^{(\ell)}, b_j^{(\ell)} \in k$.
Consideremos nuevas variables $A_j^{(h)}$ y $B_j^{(h)}$ $(1 \le h \le L;\ -n \le j \le h-1)$. Entonces existen polinomios $Q_\alpha^{(\ell)} \in \mathbb{Z}[A_j^{(h)}, B_j^{(h)}]$ tales que los coeficientes de cualquier polinomio f_ℓ que se pueda calcular en el ℓ-ésimo paso de β son las especializaciones de estos polinomios en ciertos vectores a, b con coordenadas en k, es decir

$$f_\ell = \sum_\alpha Q_\alpha^{(\ell)}(a,b)\, X^\alpha.$$

Tenemos entonces el siguiente resultado:

Proposición 28 *Dados L, $n \in \mathbb{N}$, si $m := (L+n)(L+n+1)$, existen polinomios $Q_\alpha \in \mathbb{Z}[T_1, \ldots, T_m]$ para $\alpha \in (\mathbb{N}_0)^n$, $|\alpha| \leq 2^L$, con $\deg Q_\alpha \leq 2|\alpha|L$, que satisfacen: para todo $f \in k[x_1, \ldots, x_n]$ con $L_k(f) \leq L$,*

$$f = \sum_\alpha Q_\alpha(t) X^\alpha \quad \textit{para algún } t \in k^m.$$

Sea $(Q_\alpha : |\alpha| \leq d) : \overline{k}^{(L+n)(L+n+1)} \to \overline{k}^{\binom{n+d}{n}}$ el morfismo inducido por la evaluación de esta familia de polinomios. Entonces, si $f := \sum_{|\alpha| \leq d} c_\alpha X^\alpha \in k[x_1, \ldots, x_n]$ es cualquier polinomio con $\deg(f) \leq d$ y $L_k(f) \leq L$, considerándolo como el vector $(c_\alpha)_{|\alpha| \leq d} \in k^{\binom{n+d}{n}}$, resulta que $f \in \mathrm{Im}(Q_\alpha : |\alpha| \leq d)$.
Como consecuencia tenemos que, para $d, n, L \in \mathbb{N}$ fijos, todos los vectores de coeficientes de polinomios $f \in k[x_1, \ldots, x_n]$ con $\deg(f) \leq d$ y $L_k(f) \leq L$ pertenecen al conjunto

$$W(n,d,L) := \overline{\mathrm{Im}(Q_\alpha : \ \alpha \in (\mathbb{N}_0)^n, \ |\alpha| \leq d)} \subset \overline{k}^{\binom{n+d}{n}},$$

que, siendo la clausura de la imagen de un espacio de dimensión $(L+n)(L+n+1)$ por una aplicación polinomial, satisface $\dim(W(n,d,L)) \leq (L+n)(L+n+1)$.
Esto puede interpretarse de la manera siguiente: los polinomios en n variables de grado acotado por d y longitud no-escalar acotada por L pueden verse como elementos de una subvariedad de dimensión $(L+n)(L+n+1)$ del espacio $\binom{n+d}{d}$-dimensional de todos los polinomios de grado d en n variables. Así, para L tal que $(L+n)(L+n+1) < \binom{n+d}{n}$, existen "muy pocos" polinomios fáciles de evaluar, puesto que el complemento del conjunto de estos polinomios contiene un abierto de Zariski no vacío. En otras palabras, la mayoría de los polinomios son difíciles de evaluar.

Teniendo en cuenta las observaciones hechas en esta sección, es posible preguntarse si será útil trabajar con la codificación de polinomios por straight-line programs para resolver ecuaciones polinomiales. Como veremos en la próxima sección, la respuesta a esta pregunta es afirmativa.

6 Algoritmos que utilizan straight-line programs

6.1 Cálculo de la dimensión de una variedad

La codificación de polinomios por medio de straight-line programs se utilizó por primera vez en el contexto de la resolución algorítmica de ecuaciones polinomiales en [15]. En ese trabajo, Giusti y Heintz presentan un algoritmo que calcula la dimensión de una variedad algebraica V, dada como el conjunto de los ceros comunes de una familia finita de polinomios, con complejidad *polinomial* en el tamaño del input (que se supone codificado en forma densa). En particular, este algoritmo resuelve en tiempo polinomial en el tamaño del input el problema de decidir si un sistema de ecuaciones polinomiales es consistente.
El algoritmo se basa en una subrutina que, dados polinomios que definen una variedad V, calcula una resolución geométrica de una variedad cero-dimensional o vacía incluida en V y que contiene a la componente equidimensional V_0 (puntos aislados) de V.
La herramienta clave para conseguir un algoritmo con las cotas de complejidad mencionadas está dada por el siguiente resultado (comparar con el Lema 19):

Lema 29 *([15, Sec. 3.4.5]) Sea $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por d y sea $V = V(f_1, \ldots, f_s) \subset \mathbb{A}^n$. Existe un algoritmo de complejidad polinomial en sd^n que, dada una forma lineal $\ell \in k[x_1, \ldots, x_n]$, calcula un polinomio $p \in k[T]$ de grado acotado por d^n tal que $p(\ell)$ se anula en los puntos aislados de V.*

No daremos aquí la demostración de este lema, puesto que utiliza técnicas no presentadas en este curso. El algoritmo se basa en una deformación homotópica del sistema $f_1, \ldots, f_s$ con el objeto de obtener una sucesión auxiliar de n polinomios homogéneos $G_1, \ldots, G_n \in k(\varepsilon)[x_0, \ldots, x_n]$ (donde ε es el parámetro de deformación), con buenas propiedades de intersección, a partir de cuyos ceros se pueden obtener las soluciones del sistema original $f_1, \ldots, f_s$. Los polinomios $G_1, \ldots, G_n$ definen una variedad cero-dimensional sin ceros en el infinito en el espacio proyectivo n-dimensional $\mathbb{P}^n(\overline{k(\varepsilon)})$, lo que permite hallar un polinomio $q(x_0, \ell)$ con coeficientes en $k[\varepsilon]$ que se anula en los puntos de esta variedad utilizando el polinomio característico de una transformación lineal conveniente asociada a ℓ. A partir de este polinomio q, se obtiene el polinomio p buscado mediante un proceso de paso al límite ($\varepsilon \to 0$) y deshomogeneización ($x_0 = 1$).

Utilizando el procedimiento de este lema para calcular los polinomios univariados que aparecen en la Proposición 21, se obtiene un algoritmo que calcula una resolución geométrica de una variedad cero-dimensional que contiene a V_0 con complejidad polinomial en sd^n.
Finalmente, se utilizan las ecuaciones que definen a V, junto con el polinomio asociado a la forma lineal en la resolución geométrica hallada, para "descartar" los puntos de esta variedad que no pertenecen a V y obtener una descripción de una variedad V_0', cero-dimensional o vacía, tal que $V_0 \subset V_0' \subset V$.
Observamos que, si V no posee componentes equidimensionales de dimensión positiva, este resultado permite determinar si V es vacía o no. Este es el hecho fundamental que se utiliza para el desarrollo del algoritmo que calcula la dimensión de una variedad algebraica.

La idea del algoritmo de [15] para el cálculo de la dimensión es la descripta al comienzo de la Sección 4.2:
Sea $V := V(f_1, \ldots, f_s) \subset \mathbb{A}^n$ con $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por d. Introducimos nuevas indeterminadas T_{ij} $(1 \leq i \leq n, 0 \leq j \leq n)$ sobre k y una clausura algebraica K de $k(T_{ij}; 1 \leq i \leq n, 0 \leq j \leq n)$. Para cada $1 \leq i \leq n$, sea $L_i := T_{i0} + T_{i1}x_1 + \cdots + T_{in}x_n$. Si notamos $V^{(i)} := V(f_1, \ldots, f_s, L_1, \ldots, L_i) \subset \mathbb{A}^n(K)$ para cada $0 \leq i \leq n$, entonces, si $V \neq \emptyset$,

$$\dim V = \max\{i : 0 \leq i \leq n,\ V^{(i)} \neq \emptyset\}.$$

El algoritmo comienza con la variedad $V^{(n)}$, que es vacía o cero-dimensional, y determina si es vacía o no. Si $V^{(n)} \neq \emptyset$, entonces $\dim V = n$. De lo contrario, continúa con $V^{(n-1)}$, que resulta ser vacía o cero-dimensional dado que $V^{(n-1)} \cap V(L_n) = V^{(n)} = \emptyset$, y procede de la misma manera. El proceso se repite hasta obtener el primer valor r tal que $V^{(r)} \neq \emptyset$, que será la dimensión de V, o bien hasta llegar a que $V^{(0)} = \emptyset$, en cuyo caso $V = \emptyset$.
El resultado de complejidad que se prueba es el siguiente:

Teorema 30 *([15, Sec. 3.5]) Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por d, y sea $V := V(f_1, \ldots, f_s) \subset \mathbb{A}^n$. Existe un algoritmo que calcula la dimensión de V con complejidad polinomial en sd^n.*

Los resultados intermedios del algoritmo son polinomios en las n^2+n variables T_{ij} ($1 \leq i \leq n$, $0 \leq j \leq n$) de grados acotados por $d^{O(n)}$ codificados por straight-line programs de longitud polinomial en sd^n. En algunos pasos, es necesario chequear igualdades de estos polinomios a cero, lo cual se efectúa por medio de correct test sequences con complejidad polinomial en sd^n (ver Teorema 27). Es la utilización de straight-line programs (y el hecho que existan straight-line programs "cortos" para codificar los polinomios calculados por el algoritmo) lo que permite obtener las cotas de complejidad enunciadas (observar que la representación de polinomios en forma densa en este algoritmo llevaría a complejidades de orden al menos d^{n^3}).
Queremos destacar que el uso de straight-line programs es también esencial para obtener las cotas de complejidad enunciadas en el Lema 29, debido también a la introducción de variables auxiliares durante la ejecución del algoritmo.

6.2 Operador de Newton-Hensel

El uso de straight-line programs para codificar polinomios posibilitó adaptar algorítmicamente en el contexto del cálculo simbólico una herramienta del análisis numérico muy conocida, el método de Newton-Hensel, que puede considerarse una versión particular del teorema de la función impícita.
A continuación daremos el marco teórico algebraico en el que utilizaremos el método de Newton-Hensel y luego mostraremos una aproximación algorítmica.

Sean $T_1, \ldots, T_m, x_1, \ldots, x_n$ indeterminadas sobre k. Dado $\tau \in \overline{k}^n$, notaremos $T - \tau := (T_1 - \tau_1, \ldots, T_m - \tau_m)$ y $\overline{k}[[T-\tau]]$ al anillo de series formales en las variables $T-\tau$ con coeficientes en $\overline{k}$.
Dados polinomios $f_1, \ldots, f_n \in k[T, X]$, notaremos $f := (f_1, \ldots, f_n)$, Df a la matriz jacobiana $Df := (\partial f_i / \partial x_j)_{1 \leq i,j \leq n}$ del sistema f con respecto a las variables X, y Jf al determinante de Df.

Lema 31 *Sean $f_1, \ldots, f_n \in k[T, X]$, y sea $(\tau, \xi) \in \mathbb{A}^m \times \mathbb{A}^n$ tal que*

$$f_1(\tau, \xi) = 0, \ldots, f_n(\tau, \xi) = 0 \quad y \quad Jf(\tau, \xi) \neq 0.$$

Entonces existe una n-upla $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_n) \in \overline{k}[[T-\tau]]^n$ que verifica:

- $f_1(T, \mathcal{R}) = 0, \ldots, f_n(T, \mathcal{R}) = 0$.
- $\mathcal{R}(\tau) := (\mathcal{R}_1(\tau), \ldots, \mathcal{R}_n(\tau)) = \xi$.

Demostración. Dado $f(X) := (f_1(T, X), \ldots, f_n(T, X))$ se define el operador de Newton asociado a f como

$$N_f(X)^t := X^t - Df(X)^{-1}.\, f(X)^t = \frac{Jf(X).\, X^t - \mathrm{adj}(Df)(X)\,.\, f(X)^t}{Jf(X)}, \tag{6}$$

donde $\mathrm{adj}(Df)$ denota la matriz adjunta de la matriz Df. Observamos que $Jf(X) \not\equiv 0$, puesto que $Jf(\tau, \xi) \neq 0$.
Se define la siguiente sucesión de vectores de funciones racionales:

$$\begin{cases} R^{(0)} := \xi \\ R^{(\kappa)} := N_f(R^{(\kappa-1)}) = N_f^{\kappa}(\xi) \quad \text{para } \kappa \in \mathbb{N}. \end{cases}$$

Veamos en primer lugar que esta sucesión está bien definida. Para esto, basta ver que $Jf(R^{(\kappa)}) \neq 0$ para cada $\kappa \in \mathbb{N}_0$, lo que puede probarse inductivamente junto con el hecho que $R^{(\kappa)}(\tau) = \xi$ para todo $\kappa \in \mathbb{N}_0$: Supongamos que $R^{(\kappa)} \in \overline{k}(T)^n$ y que $R^{(\kappa)}(\tau) = \xi$ (es claro que vale en el caso $\kappa = 0$). Entonces $Jf(R^{(\kappa)}) \not\equiv 0 \in \overline{k}(T)$, puesto que nuestras hipótesis implican que $Jf(R^{(\kappa)})(\tau) = Jf(\tau, \xi) \neq 0$.
En consecuencia, se puede definir $R^{(\kappa+1)} := N_f(R^{(\kappa)}) \in \overline{k}(T)^n$ y, del hecho que $R^{(\kappa)}(\tau) = \xi$ y $f(\tau, \xi) = 0$, se deduce que

$$R^{(\kappa+1)}(\tau) = N_f(R^{(\kappa)})(\tau) = (\xi^t - Df(\tau, \xi)^{-1}.\, f(\tau, \xi)^t)^t = \xi.$$

En particular, vemos que los denominadores de las coordenadas de cada uno de los vectores $R^{(\kappa)}$, $\kappa \in \mathbb{N}$, no se anulan en τ y por lo tanto, son inversibles como elementos de $\overline{k}[[T - \tau]]$. Luego, $R^{(\kappa)} \in \overline{k}[[T - \tau]]^n$ para cada $\kappa \in \mathbb{N}_0$.
Finalmente, veremos que para cada $\kappa \in \mathbb{N}_0$ valen las siguientes condiciones:

(i) $f_j(T, R^{(\kappa)}) \in (T - \tau)^{2^\kappa} \subset \overline{k}[[T - \tau]]$ para cada $1 \leq j \leq n$,

(ii) $R_i^{(\kappa+1)} - R_i^{(\kappa)} \in (T - \tau)^{2^\kappa} \subset \overline{k}[[T - \tau]]$ para cada $1 \leq i \leq n$,

donde $(T - \tau)$ denota el ideal de $\overline{k}[[T - \tau]]$ generado por $T_1 - \tau_1, \ldots, T_m - \tau_m$.
Notar que la condición (ii) implica que para cada $1 \leq i \leq n$ la sucesión $(R_i^{(\kappa)})_{\kappa \in \mathbb{N}}$ converge a un elemento $\mathcal{R}_i \in \overline{k}[[T - \tau]]$. Sea $\mathcal{R} := (\mathcal{R}_1, \ldots, \mathcal{R}_n)$. Es claro que $\mathcal{R}(\tau) = \xi$ y, como consecuencia de la condición (i), que $f_j(T, \mathcal{R}) = 0$ para cada $1 \leq j \leq n$, con lo que $\mathcal{R}$ satisface las condiciones del enunciado.

Ahora probaremos la validez de las condiciones (i) y (ii) por inducción en κ.
Para $\kappa = 0$, dado que $f_j(\tau, \xi) = 0$ para $j = 1, \ldots, n$, $f_j(T, R^{(0)}) = f_j(T, \xi) \in (T - \tau)$ $(1 \leq j \leq n)$ y, como $(R^{(1)} - \xi)^t = -Df(\xi)^{-1}.\, f(\xi)^t$, vale $R_i^{(1)} - \xi_i \in (T - \tau)$ para cada $1 \leq i \leq n$.
Supongamos que las condiciones (i) y (ii) valen para $\kappa \in \mathbb{N}_0$. Para cada $1 \leq j \leq n$, considerando el desarrollo de Taylor centrado en $R^{(\kappa)}$ del polinomio f_j (visto como polinomio en las variables X) se tiene que

$$f_j(T, R^{(\kappa+1)}) - f_j(T, R^{(\kappa)}) - \sum_{i=1}^{n} \frac{\partial f_j}{\partial x_i}(T, R^{(\kappa)}).\, (R_i^{(\kappa+1)} - R_i^{(\kappa)}) \;\in\; (R^{(\kappa+1)} - R^{(\kappa)})^2, \quad (7)$$

donde $(R^{(\kappa+1)} - R^{(\kappa)})$ es el ideal $(R^{(\kappa+1)} - R^{(\kappa)}) := (R_i^{(\kappa+1)} - R_i^{(\kappa)} : 1 \leq i \leq n) \subseteq \overline{k}[[T - \tau]]$.
Ahora, de la definición de $R^{(\kappa+1)}$ se deduce que $Df(R^{(\kappa)}).(R^{(\kappa+1)} - R^{(\kappa)})^t = -f(R^{(\kappa)})^t$ y por lo tanto la expresión en (7) es simplemente $f_j(T, R^{(\kappa+1)})$. Por hipótesis inductiva, $R_i^{(\kappa+1)} - R_i^{(\kappa)} \in (T - \tau)^{2^\kappa}$ para todo $1 \leq i \leq n$, de donde se deduce que $(R^{(\kappa+1)} - R^{(\kappa)})^2 \subset (T - \tau)^{2^{\kappa+1}}$. Luego $f_j(T, R^{(\kappa+1)}) \;\in\; (T - \tau)^{2^{\kappa+1}}$, lo que prueba (i).
Para probar la validez de la condición (ii), basta observar que, por definición,

$$(R^{(\kappa+2)} - R^{(\kappa+1)})^t = -Df(R^{(\kappa+1)})^{-1}.f(R^{(\kappa+1)})^t$$

y tener en cuenta que vale (i) para $\kappa + 1$. □

Dados polinomios $f_1, \ldots, f_n \in k[T, X]$ tales que el determinante de la matriz jacobiana Df es un polinomio no nulo, es posible definir el operador de Newton $N_f(X)$ asociado a $f := (f_1, \ldots, f_n)$ como se hizo en la demostración del Lema 31.

Observamos que N_f está dado por un vector de n funciones racionales en $k(T,X)$, al igual que N_f^κ, la κ-ésima iteración de N_f, para cada $\kappa \in \mathbb{N}$. En otras palabras, para cada $\kappa \in \mathbb{N}$ existen numeradores $g_1^{(\kappa)}, \ldots, g_n^{(\kappa)} \in k[T,X]$ y un denominador no nulo $h^{(\kappa)} \in k[T,X]$ tales que

$$N_f^\kappa = \Big(\frac{g_1^{(\kappa)}}{h^{(\kappa)}}, \ldots, \frac{g_n^{(\kappa)}}{h^{(\kappa)}}\Big) \ \in \ k(T,X).$$

En el siguiente lema se describe un straight-line program sin divisiones que evalúa estos polinomios.

Lema 32 *([13, Lemma 30]) Con las mismas hipótesis y notaciones que en el Lema 31, supongamos además que $f_1, \ldots, f_n$ tienen grados acotados por d y están dados por un straight-line program de longitud L. Sea $\nu \in \mathbb{N}$. Entonces existe un straight-line program de longitud $O(\nu\, d^2 n^7 L)$ que evalúa polinomios $g_1^{(\nu)}, \ldots, g_n^{(\nu)}, h^{(\nu)}$ en $k[T,X]$ con $h^{(\nu)}(\tau,\xi) \neq 0$ que representan numeradores y un denominador de las funciones racionales que se obtienen en la ν-ésima iteración del operador de Newton.*

Demostración. Consideremos la escritura de N_f dada por el miembro derecho de (6). Entonces, si a_{ij} $(1 \leq i,j \leq n)$ son las entradas de la matriz $\mathrm{adj}(Df)$, para cada $1 \leq i \leq n$, el numerador de la i-ésima coordenada de $N_f(X)$ es $g_i := Jf.x_i - \sum_{j=1}^n a_{ij} f_j$, mientras que el denominador de cada coordenada es $h := Jf$. Como $\deg(a_{ij}) \leq (n-1)(d-1)$ y $\deg(Jf) \leq n(d-1)$, se tiene que $nd+1$ es una cota superior para los grados de los polinomios $g_1, \ldots, g_n, h \in k[T,X]$.
Para cada $1 \leq i \leq n$, sea $G_i \in k[T][x_0, \ldots, x_n]$ el polinomio que se obtiene al homogeneizar a grado $n(d-1)+1$ el polinomio g_i con una nueva variable x_0. Análogamente, sea $H \in k[T][x_0, \ldots, x_n]$ el homogeneizado a grado $n(d-1)+1$ de Jf. Más precisamente,

$$\begin{aligned} G_i(x_0, \ldots, x_n) &:= x_0^{n(d-1)+1}.\, g_i\Big(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\Big) \qquad (1 \leq i \leq n) \\ H(x_0, \ldots, x_n) &:= x_0^{n(d-1)+1}.\, Jf\Big(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\Big). \end{aligned}$$

Comenzando con $g_i^{(1)} := G_i(1, x_1, \ldots, x_n) = g_i$ $(1 \leq i \leq n)$ y $h^{(1)} := H(1, x_1, \ldots, x_n) = h$, definimos recursivamente para $\kappa \geq 2$,

$$\begin{aligned} g_i^{(\kappa)} &:= G_i(h^{(\kappa-1)}, g_1^{(\kappa-1)}, \ldots, g_n^{(\kappa-1)}) \qquad (1 \leq i \leq n) \\ h^{(\kappa)} &:= H(h^{(\kappa-1)}, g_1^{(\kappa-1)}, \ldots, g_n^{(\kappa-1)}). \end{aligned}$$

Entonces, para cada $\kappa \in \mathbb{N}$, los polinomios $g_1^{(\kappa)}, \ldots, g_n^{(\kappa)}$ son numeradores y $h^{(\kappa)}$ es un denominador para la κ-ésima iteración de N_f, y teniendo en cuenta que

$$h^{(\kappa)} = (h^{(\kappa-1)})^{n(d-1)+1}.\, Jf\Big(\frac{g_1^{(\kappa-1)}}{h^{(\kappa-1)}}, \ldots, \frac{g_n^{(\kappa-1)}}{h^{(\kappa-1)}}\Big),$$

se deduce que $h^{(\kappa)}(t,\xi) \neq 0$.
Siguiendo las reglas de derivación, se puede obtener un straight-line program de longitud $O(nL)$ que calcula las derivadas parciales de los polinomios $f_1, \ldots, f_n$. Entonces, aplicando el algoritmo de Berkowitz a Df, tenemos que las entradas de la matriz $\mathrm{adj}(Df)$ y el polinomio Jf pueden ser evaluados por medio de un straight-line program de longitud $O(n^5 + nL)$

y, por lo tanto, obtenemos un straight-line program de esta misma longitud que calcula $g_1, \ldots, g_n, h$.
Para homogeneizar hasta grado D un polinomio f dado por un straight-line program de longitud $\mathcal{L}$, podemos reemplazar cada coordenada del slp por una secuencia de instrucciones que calcule las componentes homogéneas de grado menor o igual que D de esa coordenada. Una vez hecho esto para todas las coordenadas, se obtiene un slp de longitud $(D+1)^2\mathcal{L}$ que calcula todas las componentes homogéneas del polinomio f, a partir del cual es inmediato obtener un slp para el homogeneizado de f hasta grado D con complejidad del mismo orden. Aplicando este procedimiento al slp que calcula $g_1, \ldots, g_n, h$ se obtiene un straight-line program de longitud de orden $O(d^2(n^7 + n^3 L))$ para los polinomios $G_1, \ldots, G_n$ y H.
Fijado $\nu \in \mathbb{N}$, iterando ν veces este straight-line program, se obtiene un straight-line program de longitud $O(\nu\, d^2 n^7 L)$ que evalúa los polinomios $g_1^{(\nu)}, \ldots, g_n^{(\nu)}, h^{(\nu)}$. □

6.3 Aplicación: resolución de sistemas paramétricos

Desde el punto de vista algorítmico, el operador de Newton introducido en la demostración del Lema 31 se utilizará con el objeto de aproximar soluciones en $\overline{k}[[T-\tau]]^n$ (para un valor de τ conveniente) de sistemas de ecuaciones polinomiales en $k[T][x_1, \ldots, x_n]$.
A continuación mostraremos cómo se aplica este método para la resolución de sistemas paramétricos de ecuaciones polinomiales (ver [20]).

Sean $f_1, \ldots, f_n \in k[T_1, \ldots, T_m, x_1, \ldots, x_n]$. Para cada $\tau \in \mathbb{A}^m$, sea

$$V_\tau = \{\xi \in \mathbb{A}^n : f_1(\tau, \xi) = 0, \ldots, f_n(\tau, \xi) = 0\}.$$

El problema que se plantea es obtener información sobre V_τ para $\tau \in \mathbb{A}^m$ arbitrario a partir de una descripción de V_{τ_0} para $\tau_0 \in \mathbb{A}^m$ fijo. Vamos a tratar este problema bajo las siguientes hipótesis sobre el sistema paramétrico:

(S1) $(f_1, \ldots, f_n)$ es un ideal radical de $k[T, X]$.

(S2) La variedad $V := V(f_1, \ldots, f_n) \subset \mathbb{A}^{n+m}$ es equidimensional de dimensión m y las variables $T_1, \ldots, T_m, x_1, \ldots, x_n$ están en posición de Noether para V, siendo $T_1, \ldots, T_m$ las variables libres.

La condición (S2) implica que las variedades V_τ son cero-dimensionales para todo $\tau \in \mathbb{A}^m$ y entonces, una forma posible de describirlas es por medio de una resolución geométrica.
Dada una forma lineal $\ell = u_1 x_1 + \cdots + u_n x_n \in k[X]$ existe un polinomio $P \in k[T_1, \ldots, T_m, Y]$, mónico en la variable Y y de grado total acotado por $\deg V$, tal que $P(T_1, \ldots, T_m, \ell)$ se anula en todo punto de V. Consideraremos el polinomio de grado mínimo con estas propiedades, que se llama el *polinomio minimal de ℓ con respecto a V*. Nuestro objetivo es calcular este polinomio minimal.
Para cada $\tau \in \mathbb{A}^m$, el polinomio $P(\tau, Y)$ es un polinomio no nulo (ya que P es mónico en la variable Y) que se anula en $\ell(\xi)$ para todo $\xi \in V_\tau$. Notar que un procedimiento que calcule estos polinomios minimales para distintas formas lineales ℓ se puede aplicar para obtener los polinomios univariados utilizados en el cálculo de una resolución geométrica de V_τ (ver Proposición 21 y comparar con Lema 19 y Lema 29).
Por simplicidad, supondremos que la forma lineal ℓ satisface:

(L) ℓ separa los puntos de la variedad $V_T := \{\mathcal{R} \in \mathbb{A}^n(\overline{k(T)}) : f_1(\mathcal{R}) = 0, \ldots, f_n(\mathcal{R}) = 0\}$.

Supondremos conocido un punto $\tau_0 \in k^m$ tal que:

(P1) $\#V_{\tau_0} = \#V_T$.

(P2) $(f_1(\tau_0, X), \ldots, f_n(\tau_0, X))$ es un ideal radical de $k[X]$.

(P3) La forma lineal ℓ separa los puntos de V_{τ_0}.

Dada una posición de Noether $T_1, \ldots, T_m, x_1, \ldots, x_n$ para $V \subset \mathbb{A}^{m+n}$, un punto $\tau_0 \in \mathbb{A}^m$ que satisface las condiciones (P1) y (P2) se llama un *punto de levantamiento* para V.

Bajo las hipótesis (S1), (S2) y (L) se tiene que el polinomio minimal buscado es

$$P = \prod_{\mathcal{R} \in V_T} (Y - \ell(\mathcal{R})). \tag{8}$$

Para simplificar nuestra descripción del algoritmo, supondremos conocidos los elementos de V_{τ_0} y que sus coordenadas pertenecen al cuerpo de base k.
Para cada $\xi \in V_{\tau_0}$, como V_{τ_0} es cero-dimensional y vale (P2), por el criterio del Jacobiano vale que $Jf(\tau_0, \xi) \neq 0$. Luego, por el Lema 31 existe una n-upla $\mathcal{R}_\xi \in \overline{k}[[T - \tau_0]]^n$ que verifica: $f_1(T, \mathcal{R}_\xi) = 0, \ldots, f_n(T, \mathcal{R}_\xi) = 0$ y $\mathcal{R}_\xi(\tau_0) = \xi$. De esta manera se obtienen $\#V_{\tau_0}$ puntos de V_T que, por la hipótesis (P1), resultan ser todos los puntos de esta variedad.
Ahora, para cada $\xi \in V_{\tau_0}$, el elemento $\mathcal{R}_\xi$ puede obtenerse iterando "infinitamente" el operador de Newton comenzando en ξ, pero esto no puede hacerse algorítmicamente. Sin embargo, teniendo en cuenta que el objeto que queremos calcular es un polinomio, resulta que basta calcular aproximaciones "suficientemente buenas" de $\mathcal{R}_\xi := (\mathcal{R}_{\xi 1}, \ldots, \mathcal{R}_{\xi n})$ para cada ξ.
Al iterar κ veces el operador de Newton se obtiene un vector $\mathcal{R}_\xi^{(\kappa)} := (\mathcal{R}_{\xi 1}^{(\kappa)}, \ldots, \mathcal{R}_{\xi n}^{(\kappa)})$ que satisface $\mathcal{R}_{\xi j} - \mathcal{R}_{\xi j}^{(\kappa)} \in (T - \tau_0)^{2^\kappa}$ para cada $1 \leq j \leq n$. En consecuencia, los coeficientes (en la variable Y) del polinomio

$$P^{(\kappa)} := \prod_{\xi \in V_{\tau_0}} (Y - \ell(\mathcal{R}_\xi)) \tag{9}$$

son elementos de $k[[T - \tau_0]]$ cuyos desarrollos coinciden hasta grado $2^\kappa - 1$ con los desarrollos de Taylor alrededor de τ_0 de los coeficientes de P. Como $\deg P \leq D := \deg V$, el polinomio P coincide con la parte de grado menor o igual que D del polinomio $P^{(\lceil \log(D+1) \rceil)}$.
Esto permite diseñar un algoritmo que, a partir de los puntos de V_{τ_0}, calcula un straight-line program para el polinomio minimal P buscado.
Dado $\kappa := \lceil \log(D+1) \rceil$, en un primer paso, se calcula un straight-line program que codifica los polinomios que representan la κ-ésima iteración del operador de Newton asociado a $f_1, \ldots, f_n$ (ver Lema 32). En un segundo paso, evaluando los polinomios en cada $\xi \in V_{\tau_0}$ se obtienen las aproximaciones $\mathcal{R}_\xi^{(\kappa)} := N_f^\kappa(\xi)$ que nos permiten calcular el polinomio $P^{(\kappa)} := \prod_{\xi \in V_{\tau_0}} (Y - \ell(\mathcal{R}_\xi))$. Finalmente se aplica un algoritmo que calcula un straight-line program para la suma de todos los monomios de grado menor o igual que D de $P^{(\kappa)}$, que resulta ser el polinomio P buscado.

El algoritmo anterior puede ser modificado para calcular el polinomio P a partir de una resolución geométrica de V_{τ_0}: se reemplazan los puntos de V_{τ_0} por matrices cuyos autovalores son las coordenadas de estos puntos (las cuales se obtienen a partir de la resolución geométrica de V_{τ_0}) y se opera con estas matrices utilizando las mismas técnicas. Puede demostrarse el siguiente resultado (ver [20, Theorem 2]):

Teorema 33 *Sean $f_1, \ldots, f_n \in k[T_1, \ldots, T_m, x_1, \ldots, x_n]$ que cumplen las hipótesis (S1) y (S2), sea $\ell \in k[x_1, \ldots, x_n]$ una forma lineal que cumple (L) y sea $\tau_0 \in k^m$ que verifica (P1), (P2) y (P3). Supongamos que $f_1, \ldots, f_n$ son polinomios de grado acotado por d dados por un straight-line program de longitud L. Entonces existe un algoritmo que calcula un straight-line program para los coeficientes en la variable Y del polinomio minimal de ℓ a partir de una resolución geométrica de V_{τ_0} con complejidad $d^2 n^7 (\deg(V))^{O(1)} L$.*

Observaciones:

- La hipótesis de que la forma lineal ℓ separe los puntos de la variedad V_T no es necesaria. El algoritmo descripto puede modificarse para calcular el polinomio minimal de *cualquier* polinomio en $k[T, X]$ (ver [20]). En particular, se puede hallar el polinomio minimal de una forma lineal arbitraria con la complejidad enunciada en el Teorema 33.
- La hipótesis de posición de Noether puede reemplazarse por una más débil: que las variables $T_1, \ldots, T_m$ sean libres con respecto a V. En este caso, el polinomio P de la ecuación (8) tendrá como coeficientes funciones racionales en las variables $T_1, \ldots, T_m$ en lugar de polinomios. Así, una vez obtenido el polinomio $P^{(\kappa)}$ para un κ apropiado, se utiliza un algoritmo para reconstruir funciones racionales a partir de aproximaciones de sus desarrollos en serie de potencias (ver [35]).

Concluimos esta sección presentando un ejemplo donde aplicamos el algoritmo descripto a un sistema paramétrico particular.

Ejemplo. Consideramos los polinomios en $\mathbb{Q}[T_1, T_2, T_3, T_4, x_1, x_2]$

$$\begin{aligned} f_1 &= x_1^2 + T_1\, x_2 + T_2 \\ f_2 &= x_2^2 + T_3\, x_1 + T_4 \end{aligned}$$

Vamos a calcular dos polinomios $P, Q \in \mathbb{Q}[T_1, T_2, T_3, T_4][Y]$ asociados a dos formas lineales ℓ_1 y ℓ_2 en $\mathbb{Q}[x_1, x_2]$, tales que para cada $\tau \in \mathbb{A}^4$, si $P_\tau := P(\tau, Y)$ y $Q_\tau := Q(\tau, Y)$,

$$V_\tau := \{\xi \in \mathbb{A}^2 : f_1(\tau, \xi) = 0,\, f_2(\tau, \xi) = 0\} \subset \{\xi \in \mathbb{A}^2 : P_\tau(\ell_1(\xi)) = 0,\ Q_\tau(\ell_2(\xi)) = 0\}.$$

Estos polinomios permiten obtener información sobre V_τ, para cada $\tau \in \mathbb{A}^4$, considerando la siguiente variedad cero-dimensional:

$$W_\tau := \{(\eta_1, \eta_2) \in \mathbb{A}^2 : P_\tau(\eta_1) = 0,\, Q_\tau(\eta_2) = 0\},$$

que está definida por polinomios en variables separadas. Más precisamente, para cada $\eta \in W_\tau$, existe un único $\xi \in \mathbb{A}^2$ tal que $\ell_1(\xi) = \eta_1$, $\ell_2(\xi) = \eta_2$, y los puntos de V_τ se hallan entre las soluciones de estos sistemas.

Observamos que si $\tau_0 := (0, -1, 0, -1) \in \mathbb{Q}^4$, el conjunto de soluciones V_{τ_0} del sistema asociado es

$$V_{\tau_0} = \{(\xi_1, \xi_2) \in \overline{\mathbb{Q}}^2 :\ \xi_1^2 - 1 = 0,\, \xi_2^2 - 1 = 0\} = \{(1,1), (1,-1), (-1,1), (-1,-1)\}.$$

Las formas lineales $\ell_1 := x_1 + 2x_2$ y $\ell_2 := 2x_1 + x_2$ separan los puntos de V_{τ_0}, y por lo tanto los de $V_T = \{(\mathcal{R}_1, \mathcal{R}_2) \in \mathbb{A}^2(\overline{\mathbb{Q}(T)}) : f_1(\mathcal{R}_1, \mathcal{R}_2) = 0,\ f_2(\mathcal{R}_1, \mathcal{R}_2) = 0\}$.

Luego, podemos calcular los polinomios minimales de ℓ_1 y ℓ_2 a partir de los puntos de V_{τ_0}. A continuación calculamos estos polinomios por medio del algoritmo descripto al comienzo de esta sección, utilizando el sistema Maple para efectuar los cálculos intermedios.
Aplicaremos subrutinas de Maple para cálculos de álgebra lineal y desarrollos de Taylor:

```
> with(linalg):
> readlib(mtaylor):
```

En primer lugar, definimos el sistema de polinomios con el que vamos a trabajar:

```
> f[1]:= x[1]^2+T[1]*x[2]+T[2];
```

$$f_1 := {x_1}^2 + T_1\, x_2 + T_2$$

```
> f[2]:= x[2]^2+T[3]*x[1]+T[4];
```

$$f_2 := {x_2}^2 + T_3\, x_1 + T_4$$

El primer paso del algoritmo consiste en obtener numeradores g_1 y g_2 y un denominador g_0 para las funciones racionales que dan el operador de Newton N_f asociado a f_1 y f_2 de acuerdo a la fórmula (6):

```
> DF:=jacobian([f[1],f[2]],[x[1],x[2]]):
> J:=det(DF):
> A:=adj(DF):
> for i from 1 to 2 do g[i]:=expand(J*x[i]-A[i,1]*f[1]-A[i,2]*f[2]) od;
```

$$g_1 := 2\,{x_1}^2\, x_2 - T_1\,{x_2}^2 - 2\, x_2\, T_2 + T_1\, T_4$$
$$g_2 := 2\, x_1\,{x_2}^2 - T_3\,{x_1}^2 + T_3\, T_2 - 2\, x_1\, T_4$$

```
> g[0]:=J;
```

$$g_0 := 4\, x_1\, x_2 - T_1\, T_3$$

Con el objeto de calcular las iteraciones sucesivas de N_f por medio del procedimiento descripto en el Lema 32, homogeneizamos estos polinomios a grado 3:

```
> for i from 0 to 2 do
  G[i]:=expand(x[0]^3*eval(g[i], [x[1]=x[1]/x[0], x[2]=x[2]/x[0]])) od;
```

$$G_0 := 4\, x_0\, x_1\, x_2 - {x_0}^3\, T_1\, T_3$$
$$G_1 := 2\,{x_1}^2\, x_2 - x_0\, T_1\,{x_2}^2 - 2\,{x_0}^2\, x_2\, T_2 + {x_0}^3\, T_1\, T_4$$
$$G_2 := 2\, x_1\,{x_2}^2 - x_0\, T_3\,{x_1}^2 + {x_0}^3\, T_3\, T_2 - 2\,{x_0}^2\, x_1\, T_4$$

Como la variedad $V := \{(\tau, \xi) \in \mathbb{A}^4 \times \mathbb{A}^2 : f_1(\tau, \xi) = 0,\ f_2(\tau, \xi) = 0\}$ está definida por dos polinomios de grado 2, se tiene que $\deg V \leq 4$. Por otro lado, intersecándola con $T_1 = 0$, $T_2 = -1$, $T_3 = 0$, $T_4 = -1$ se obtiene V_{τ_0}, que es un conjunto finito con 4 puntos. Concluimos entonces que $\deg V = 4$.
Luego, el polinomio minimal de una forma lineal arbitraria con respecto a V tiene grado total acotado por 4 y, en consecuencia, puede obtenerse a partir de los desarrollos en series de potencias hasta grado 4 de las coordenadas de los puntos de V_T, las que por el Lema 31 pueden considerarse como elementos en $\overline{\mathbb{Q}}[[T - \tau_0]]$. De acuerdo a la demostración de este lema, para hallar estas aproximaciones basta aplicar N_f^3 a los puntos de V_{τ_0}.
Calculamos entonces numeradores y un denominador para N_f^3: Comenzamos por N_f, que está dado por

```
> for i from 0 to 2 do N[i] := g[i] od:
```

y hacemos dos iteraciones más del operador

```
> for j from 1 to 2 do
    for i from 0 to 2 do Nn[i]:= N[i] od:
    for i from 0 to 2 do
    N[i]:=eval(G[i],[x[1]= Nn[1], x[2]= Nn[2], x[0]= Nn[0]]) od:
  od:
```

Introducimos ahora las soluciones del sistema particular asociado a $\tau_0 := (0, -1, 0, -1)$ (los cuatro puntos de V_{τ_0}):

```
> V:=array(1..4,1..2, [[1,1],[1,-1],[-1,1],[-1,-1]]):
```

Aplicando la tercera iteración del operador de Newton a cada uno de estos cuatro puntos obtenemos las aproximaciones deseadas de las coordenadas de cada una de las cuatro raíces del sistema original, a partir de las cuales calculamos sus desarrollos en serie de potencias centradas en $\tau_0 = (0, -1, 0, -1)$ hasta grado 4:

```
> for j from 1 to 4 do
    for i from 1 to 2 do
      R[j,i]:= mtaylor(eval(N[i]/N[0],[x[1]=V[j,1], x[2]=V[j,2]]),
                [T[1],T[2]=-1,T[3],T[4]=-1], 5)
    od:
  od:
```

Finalmente, calculamos los polinomios minimales P y Q de las formas lineales $\ell_1 = x_1 + 2x_2$ y $\ell_2 = 2x_1 + x_2$ como la suma de las partes homogéneas de grado menor o igual que 4 (desarrollos de Taylor hasta grado 4) centradas en $(\tau_0, 0)$ de los polinomios aproximantes que se obtienen, de acuerdo a (9), a partir de los cuatro puntos calculados en el paso anterior:

```
> P:=mtaylor(product(Y-R[k,1]-2*R[k,2], k=1..4),
     [T[1],T[2]=-1,T[3],T[4]=-1,Y], 5):
> Q:=mtaylor(product(Y-2*R[k,1]-R[k,2], k=1..4),
     [T[1],T[2]=-1,T[3],T[4]=-1,Y], 5):
```

Los coeficientes en la variable Y de los polinomios P y Q son:

```
> for i from 0 to 4 do p[i]:=expand(coeff(P, Y, i)) od;
```

$$p_0 := 16\,{T_3}^2\,T_2 + 16\,{T_4}^2 + {T_1}^2\,T_4 + {T_2}^2 + 2\,T_3\,T_1\,T_2 + 8\,T_1\,T_4\,T_3 - 8\,T_4\,T_2$$

$$p_1 := -16\,T_3\,T_2 + {T_1}^2\,T_3 + 8\,T_1\,{T_3}^2 - 8\,T_1\,T_4$$

$$p_2 := -6\,T_1\,T_3 + 2\,T_2 + 8\,T_4$$

$$p_3 := 0$$

$$p_4 := 1$$

```
> for i from 0 to 4 do q[i]:=expand(coeff(Q, Y, i)) od;
```

$$q_0 := {T_3}^2\,T_2 + {T_4}^2 + 16\,{T_1}^2\,T_4 + 16\,{T_2}^2 + 8\,T_3\,T_1\,T_2 + 2\,T_1\,T_4\,T_3 - 8\,T_4\,T_2$$

$$q_1 := -8\,T_3\,T_2 + 8\,{T_1}^2\,T_3 + T_1\,{T_3}^2 - 16\,T_1\,T_4$$

$$q_2 := -6\,T_1\,T_3 + 8\,T_2 + 2\,T_4$$
$$q_3 := 0$$
$$q_4 := 1$$

Una vez obtenidos los polinomios P y Q, podemos hallar V_τ para un elemento $\tau \in \mathbb{A}^4$ arbitrario procediendo como indicamos al comienzo del ejemplo.
Consideremos, por ejemplo $\tau = (1, -1, -4, -1)$. Entonces

$$V_\tau = \{(\xi_1, \xi_2) \in \overline{\mathbb{Q}}^2 : \ \xi_1^2 + \xi_2 - 1 = 0, \ \xi_2^2 - 4\xi_1 - 1 = 0\}.$$

Sean P_τ y Q_τ en $\mathbb{Q}[Y]$ los polinomios $P_\tau = P(1, -1, -4, -1, Y) = Y^4 + 14Y^2 + 68Y - 208$ y $Q_\tau = Q(1, -1, -4, -1, Y) = Y^4 + 14Y^2 - 32Y + 17$, que se factorizan como sigue:

$$\begin{aligned} P_\tau &= (Y-2)(Y+4)(Y-1-5i)(Y-1+5i) \\ Q_\tau &= (Y-1)^2(Y+1-4i)(Y+1+4i) \end{aligned}$$

Teniendo en cuenta que $V_\tau \subset \{(\xi_1, \xi_2) \in \mathbb{A}^2 : P_\tau(\xi_1 + 2\xi_2) = 0, \ Q_\tau(2\xi_1 + \xi_2) = 0\}$, obtenemos los puntos de la variedad resolviendo los 12 sistemas lineales

$$\left\{ \begin{array}{rcl} x_1 + 2x_2 &=& \eta_1 \\ 2x_1 + x_2 &=& \eta_2 \end{array} \right. \qquad \text{para } \eta_1 \in \{2, -4, 1+5i, 1-5i\}, \ \eta_2 \in \{1, -1+4i, -1-4i\}$$

y verificando cuáles de las soluciones halladas son ceros de los polinomios $x_1^2 + x_2 - 1$ y $x_2^2 - 4x_1 - 1$, que definen V_τ. De esta manera, resulta que

$$V_\tau = \{(2, -3), \ (0, 1), \ (-1+i, 1+2i), \ (-1-i, 1-2i)\}.$$

6.4 Sistemas cero-dimensionales y straight-line programs

En esta sección presentaremos esquemáticamente un algoritmo que calcula una resolución geométrica para una variedad cero-dimensional definida por polinomios codificados por straight-line programs. Este algoritmo fue introducido en [17] y [16], donde se diseñaron los primeros algoritmos que admiten como entrada polinomios codificados por straight-line programs. Sin embargo, estos algoritmos requerían cálculos con números algebraicos. Posteriormente, en [13] se dio una versión "racional" del algoritmo.

El algoritmo toma como entrada una familia de polinomios $f_1, \ldots, f_n \in k[x_1, \ldots, x_n]$ codificados por un straight-line program tales que

(H1) $V := V(f_1, \ldots, f_n)$ es una variedad cero-dimensional de $\mathbb{A}^n$.

(H2) Para cada $1 \leq i \leq n-1$:

- El ideal $(f_1, \ldots, f_i) \subset k[x_1, \ldots, x_n]$ es radical.
- La variedad $\mathcal{V}_i := V(f_1, \ldots, f_i)$ es equidimensional de dimensión $n - i$.

Observamos que, por los teoremas de Bertini (ver [26]), las hipótesis (H2) se pueden obtener para cualquier sistema cero-dimensional tomando combinaciones lineales genéricas de los polinomios involucrados.

Para calcular una resolución geométrica de V se aplica un procedimiento recursivo que, comenzando con $\mathcal{V}_0 := \mathbb{A}^n$ y $V_{p_0} := \{(0, \ldots, 0)\}$, calcula en cada paso, para $i = 1, \ldots, n$:

i) una posición de Noether para $\mathcal{V}_i$;

ii) un punto de levantamiento $p_i \in k^{n-i}$ para $\mathcal{V}_i$ (ver Sección 6.3);

iii) una resolución geométrica de

$$V_{p_i} = V(f_1(p_i, x_{n-i+1}, \ldots, x_n), \ldots, f_i(p_i, x_{n-i+1}, \ldots, x_n)) \subset \mathbb{A}^i;$$

a partir de los datos hallados en el paso anterior. La resolución geométrica hallada al concluir el n-ésimo paso es la resolución geométrica de la variedad $V = V(f_1, \ldots, f_n)$.

Cada paso de la recursión consta de tres etapas (para una descripción detallada de cada una de ellas ver [16] y [13]):
En la primera etapa, se calcula una resolución geométrica de $\mathcal{V}_{i-1}$ a partir de la resolución geométrica de $V_{p_{i-1}}$. La herramienta clave para esto es el proceso de levantamiento via el operador de Newton que hemos descripto en la sección anterior para el cálculo de polinomios minimales. Combinando esto con argumentos similares a los del Lema 20 y la Proposición 21 se obtiene la resolución geométrica buscada.
La segunda etapa consiste en calcular algorítmicamente la intersección $\mathcal{V}_{i-1} \cap V(f_i) = \mathcal{V}_i$ con el objeto de hallar un cambio lineal de variables que produzca una posición de Noether para $\mathcal{V}_i$.
Finalmente, se halla un punto de levantamiento p_i para esta variedad y se calcula una resolución geométrica de V_{p_i}.

El resultado de complejidad que se obtiene es el siguiente (ver [13, Theorem 19]):

Teorema 34 *Sean $f_1, \ldots, f_n \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por d dados por un straight-line program de longitud L que satisfacen las hipótesis (H1) y (H2). Sea $\delta := \max\{\deg(V(f_1, \ldots, f_i)) : 1 \leq i \leq n\}$. Entonces existe un algoritmo que calcula una resolución geométrica de la variedad cero-dimensional $V := V(f_1, \ldots, f_n) \subset \mathbb{A}^n$ con complejidad $(n\, d\, \delta)^{O(1)} L$.*

Desde el punto de vista de la complejidad, el algoritmo difiere de los conocidos anteriormente para la resolución de sistemas cero-dimensionales en que su medida de complejidad no depende del llamado *número de Bézout* del sistema de polinomios que define V (es decir, el producto de los grados de los polinomios), sino de los grados de las variedades que estos polinomios definen sucesivamente y de la longitud del slp que codifica estos polinomios. Esto se debe por un lado a la codificación por slp's de *todos* los polinomios con los que trabaja el algoritmo, incluyendo los polinomios de entrada, lo que evita que la complejidad dependa del tamaño de la codificación densa del input. Por otra parte, el uso del operador de Newton en los pasos intermedios de la recursión hace que la complejidad dependa de los grados de las variedades $V(f_1, \ldots, f_i)$ (ver Teorema 33) y no del número de Bézout del sistema.
Notar que la complejidad del Teorema 34 siempre tiene orden menor o igual que $(nd^n)^{O(1)} L$, ya que el número de Bézout es una cota superior para δ. Por otro lado, si $f_1, \ldots, f_n$ están codificados en forma densa, L puede tomarse de orden nd^n y, por lo tanto, la complejidad es de orden $(nd^n)^{O(1)}$.

6.5 Teorema de los ceros

Como vimos en la Sección 6.1, el uso de straight-line programs para representar polinomios dio lugar al diseño de un algoritmo de complejidad polinomial en el tamaño del input que resuelve el problema de la consistencia para sistemas de ecuaciones polinomiales (ver [15]). Estas técnicas se aplicaron también para obtener un teorema de los ceros efectivo: En [18], se presenta un algoritmo de complejidad polinomial en sd^n que, dados polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$, determina si $1 \in (f_1, \ldots, f_s)$ utilizando el algoritmo de [15] y en caso afirmativo, produce un straight-line program de longitud $s^{O(1)}d^{O(n)}$ que calcula polinomios $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ tales que $1 = \sum_{1 \leq i \leq s} g_i f_i$. Observamos que los polinomios $g_1, \ldots, g_s$ hallados por este algoritmo tienen grados acotados por $d^{O(n^2)}$, mientras que los mejores resultados conocidos establecen la existencia de polinomios de grados acotados por d^n (si $d > 2$) que permiten representar al 1 en el ideal.
Por este motivo, el problema fue considerado nuevamente en [10], donde se construyó otro algoritmo basado en técnicas más sofisticadas de álgebra conmutativa (más precisamente, en la teoría clásica de dualidad) que resuelve el problema. Bajo las hipótesis anteriores, el algoritmo calcula con complejidad $s^{O(1)}d^{O(n)}$ un straight-line program de longitud del mismo orden que representa polinomios $g_1, \ldots, g_s \in k[x_1, \ldots, x_n]$ de grados acotados por $d^{O(n)}$ que dan la escritura del 1.
El uso de estas técnicas posibilitó también obtener cotas superiores para los grados de los polinomios $g_1, \ldots, g_s$ (ver [33]). Asímismo, este enfoque dio lugar a la obtención de cotas superiores para estos grados que no dependen del factor exponencial d^n sino de los grados de las variedades definidas sucesivamente por combinaciones lineales genéricas de los polinomios $f_1, \ldots, f_s$ (ver [16] y [28]). Cotas del mismo tipo pueden hallarse también en [38].
Finalmente, las técnicas algorítmicas de [16] basadas en el operador de Newton vistas en las secciones previas, combinadas con las herramientas mencionadas en el párrafo anterior, se aplicaron siguiendo las ideas de [10] para obtener un algoritmo para el teorema de los ceros de complejidad polinomial en $nd\delta L$, donde L es la longitud de un straight-line program que codifica a los polinomios de entrada y δ es el máximo de los grados de las variedades definidas sucesivamente por combinaciones lineales genéricas de estos polinomios. El algoritmo produce, a partir de un straight-line program que codifica polinomios $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ de grados acotados por d sin ceros comunes en $\overline{k}^n$, un straight-line program de longitud polinomial en $nd\delta L$ que calcula polinomios $g_1, \ldots, g_s$ de grados acotados por $3n^2d\delta$ tales que $1 = \sum_{1 \leq i \leq s} g_i f_i$.

6.6 Algoritmos probabilísticos

El desarrollo de un algoritmo puede depender de elecciones de elementos que satisfagan ciertas condiciones (por ejemplo, elegir un punto en el que un polinomio no se anule), que pueden ser muy costosas desde el punto de vista algorítmico. Más aún, en ciertos casos estas elecciones podrían involucrar cálculos que no podemos o no sabemos efectuar (en el caso mencionado, por ejemplo, que no podamos calcular el polinomio en cuestión). Entonces, una posibilidad es elegir el objeto aleatoriamente y continuar con el algoritmo, pero los resultados obtenidos podrían ser erróneos. En este caso hablamos de un *algoritmo probabilístico*.
En la mayoría de los algoritmos probabilísticos relacionados con la resolución de ecuaciones polinomiales, la condición que debe cumplir el objeto que se elige aleatoriamente se traduce en que un punto no anule a cierto polinomio $f \in k[x_1, \ldots, x_n]$ para el cual se conocen cotas de grado. Este punto se elige tomando cada una de sus coordenadas al azar de un

subconjunto finito de k suficientemente grande. La probabilidad de error del algoritmo se estima entonces por medio del siguiente resultado (ver [37] y [40]):

Lema 35 *Sea $f \in k[x_1, \ldots, x_n]$ un polinomio no nulo y sea $\mathcal{A} \subset k$ un conjunto finito. Si se eligen aleatoriamente $\xi_1, \ldots, \xi_n \in \mathcal{A}$, entonces* $\text{Prob}(f(\xi_1, \ldots, \xi_n) = 0) \leq \dfrac{\deg(f)}{\#\mathcal{A}}$.

Un algoritmo que trabaja con straight-line programs puede transformarse en un algoritmo probabilístico si no se dispone de una correct test sequence para la clase de polinomios involucrados. Para esto se puede proceder de dos formas distintas: Una de ellas es tomar un conjunto de tantos puntos como el cardinal de una correct test sequence y correr el algoritmo como si este conjunto fuese en efecto una correct test sequence. La segunda es, cada vez que querramos determinar si un straight-line program representa el polinomio 0, evaluarlo en un punto elegido al azar. En ambos casos, si el resultado de alguna evaluación no es cero, sabemos que el polinomio es no nulo; en cambio, si todos los resultados son cero, podemos suponer que el polinomio es 0 pero hay una probabilidad de que no lo sea.

Para la construcción de un algoritmo probabilístico se supone dado un proceso aleatorio para seleccionar elementos de un subconjunto fijo del cuerpo de base k, el cual se utiliza para elegir los valores de los parámetros involucrados. Este tipo de algoritmos se representan mediante grafos en forma análoga a lo descripto en la Sección 2.2: la única modificación al modelo anterior consiste en agregar nodos para representar cada una de las elecciones aleatorias efectuadas durante el algoritmo. Estos nodos no serán tenidos en cuenta para el cálculo de la complejidad.

En los últimos años, fueron diseñados distintos algoritmos probabilísticos para problemas relacionados con la resolución de ecuaciones polinomiales. Como ejemplo de un algoritmo probabilístico en este contexto, en la próxima sección describiremos el algoritmo de descomposición equidimensional presentado en [25]. También se utilizaron algoritmos probabilísticos para resolver otros problemas, como por ejemplo en [35], donde se resuelven sistemas paramétricos como los ya mencionados en la Sección 6.3 bajo hipótesis más generales.

6.7 Un algoritmo probabilístico para la descomposición equidimensional

Mediante el uso de algoritmos probabilísticos y la codificación de polinomios por straight-line programs ha sido posible resolver el problema del cálculo de la descomposición equidimensional de una variedad algebraica con complejidad polinomial en el tamaño del input:

Teorema 36 *([25, Theorem 9]) Sean $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$ polinomios de grados acotados por un entero $d \geq n$, y sea $V := V(f_1, \ldots, f_s) \subset \mathbb{A}^n$. Existe un algoritmo de complejidad polinomial en sd^n que calcula la descomposición equidimensional $V = \bigcup_{j=0}^{n} V_j$: el algoritmo produce un straight-line program de longitud polinomial en sd^n que calcula polinomios $g_i^{(j)}$ $(0 \leq j \leq \dim V,\ 1 \leq i \leq n+1)$ tales que $V_j = V(g_1^{(j)}, \ldots, g_{n+1}^{(j)})$ para cada $0 \leq j \leq \dim V$.*

El algoritmo se basa en el hecho que toda variedad equidimensional en $\mathbb{A}^n$ puede describirse como el conjunto de los ceros comunes de $n+1$ polinomios:
Sea $W \subset \mathbb{A}^n$ una variedad equidimensional de dimensión $r < n$ definible por polinomios en $k[x_1, \ldots, x_n]$ y supongamos que las variables $x_1, \ldots, x_n$ están en posición de Noether con

respecto a W. Entonces, existen $n+1$ formas lineales $\ell_1, \ldots, \ell_{n+1} \in k[x_{r+1}, \ldots, x_n]$ tales que

$$W = \{\xi \in \mathbb{A}^n : m_{\ell_1}(\xi_1, \ldots, \xi_r, \ell_1(\xi)) = 0, \ldots, m_{\ell_{n+1}}(\xi_1, \ldots, \xi_r, \ell_{n+1}(\xi)) = 0\},$$

donde m_ℓ denota el polinomio minimal de la forma lineal ℓ con respecto a W. Más aún, la igualdad anterior vale para $n+1$ formas lineales genéricas (ver [25, Lemma 7], [19, Lemmas 2–3]).

A continuación daremos una breve descripción de los distintos pasos que sigue el algoritmo del Teorema 36. Para el cálculo detallado de la complejidad y de la probabilidad de éxito del algoritmo, ver [25].

• *Cálculo de la dimensión de V.*

El algoritmo comienza calculando $r := \dim V$, para lo cual se aplica el algoritmo de [15] descripto en la Sección 6.1.

• *Preparación de los datos de entrada.*

Por medio de una elección aleatoria de $n+1$ combinaciones lineales de los polinomios de entrada, se puede suponer que V está definida por $n+1$ polinomios $f_1, \ldots, f_{n+1}$ de grados acotados por d tales que para cada $0 \leq j \leq n$, la dimensión de cada componente irreducible de $V(f_1, \ldots, f_{n-j})$ no incluida en V es j. Esto implica que, para cada $0 \leq j \leq r$, se tiene que $V(f_1, \ldots, f_{n-j}) = V_r \cup V_{r-1} \cup \cdots \cup V_j \cup Z_j$, donde $Z_j = \emptyset$ o Z_j es la unión de las componentes irreducibles de la variedad no incluidas en V.
Haciendo un cambio lineal aleatorio de variables, se puede suponer que las variables están en posición de Noether con respecto a $V(f_1, \ldots, f_{n-r})$ y a $Z_j \cap V(f_{n-j+1})$ (si esta variedad es no vacía) para cada $1 \leq j \leq r$.

• *Descomposición auxiliar de V.*

En esta etapa el algoritmo obtiene recursivamente, para $j = r, r-1, \ldots, 0$, polinomios que definen una subvariedad V'_j de V de dimensión j que contiene a V_j.

– Se obtienen polinomios que definen V_r y polinomios que definen Z_r.

Sea K_r una clausura algebraica de $k(x_1, \ldots, x_r)$. La hipótesis de posición de Noether con respecto a $V(f_1, \ldots, f_{n-r}) = V_r \cup Z_r$ implica que el conjunto de los ceros comunes en $\mathbb{A}^{n-r}(K_r)$ de los polinomios $f_1, \ldots, f_{n-r} \in k(x_1, \ldots, x_r)[x_{r+1}, \ldots, x_n]$ es una variedad cero-dimensional. Dada una forma lineal ℓ, se calcula el polinomio minimal de ℓ con respecto a esta variedad (que coincide con el polinomio minimal de ℓ con respecto a $V_r \cup Z_r$) aplicando el algoritmo de [15] y a partir de este polinomio se recuperan los minimales de ℓ con respecto a V_r y Z_r respectivamente teniendo en cuenta que $V_r \subset V(f_{n-r+1})$, pero ninguna componente irreducible de Z_r está incluida en $V(f_{n-r+1})$. Esto se aplica para $n+1$ formas lineales elegidas aleatoriamente a partir de cuyos minimales, si se satisfacen ciertas condiciones genéricas, se obtienen ecuaciones para Z_r y para V_r.

– Para $j = r-1, \ldots, 0$, se calculan polinomios que definen Z_j y polinomios que definen una subvariedad V'_j de $\bigcup_{h=j}^{r} V_h$ de dimensión j que contiene a V_j.

Sea j con $0 \leq j \leq r-1$. Teniendo en cuenta que $V(f_1, \ldots, f_{n-j}) = V_r \cup \cdots \cup V_{j+1} \cup V_j \cup Z_j$ y que $Z_{j+1} \cap V(f_{n-j}) = Z_j \cup V_j \cup \widehat{V}_j$ donde $\widehat{V}_j$ es una subvariedad j-equidimensional de V, la hipótesis de posición de Noether para $Z_{j+1} \cap V(f_{n-j})$ nos permite reducirnos a una situación cero-dimensional considerando a $f_1, \ldots, f_{n-j}$ como elementos de $k(x_1, \ldots, x_j)[x_{j+1}, \ldots, x_n]$:

la información necesaria sobre $V_j \cup Z_j$ puede obtenerse a partir del conjunto de puntos aislados de la variedad definida por estos polinomios sobre una clausura algebraica de $k(x_1, \ldots, x_j)$.
Dada una forma lineal ℓ, se calcula un polinomio que evaluado en ℓ se anula en estos puntos aislados. El polinomio minimal de ℓ con respecto a Z_j se recupera, como en el primer paso, utilizando el polinomio f_{n-j+1}. Sin embargo, como $V(f_{n-j+1})$ contiene a todas las componentes de V, este polinomio no nos permite "separar" V_j de las restantes componentes irreducibles incluidas en V. Para tener cierto control sobre las componentes irreducibles que puedan aparecer, se utiliza entonces el hecho que $V_j \subset Z_{j+1}$ y por lo tanto todo punto de V_j verifica las ecuaciones para Z_{j+1} halladas en el paso recursivo anterior. Esto permite calcular un múltiplo (por un factor controlado) del polinomio minimal de ℓ con respecto a V_j.
Aplicando el procedimiento a $n+1$ formas lineales elegidas aleatoriamente se obtienen polinomios que definen Z_j y polinomios que definen una subvariedad V'_j de $\bigcup_{h=j}^{r} V_h$ de dimensión j que contiene a V_j.

• *Cálculo de la descomposición equidimensional de V.*

Recursivamente, para $j = r-1, \ldots, 0$, el algoritmo calcula $n+1$ polinomios que definen V_j.

Fijado j, para cada una de las $n+1$ formas lineales consideradas en el paso j de la etapa anterior, el algoritmo recupera el minimal de la forma lineal con respecto a V_j a partir del múltiplo calculado previamente. Para esto, utiliza los polinomios que definen $V'_{j+1}, \ldots, V'_{r-1}$ y V_r, los que permiten "separar" el factor que se anula sobre V_j.

Otros ejemplos de algoritmos probabilísticos que obtienen la descomposición equidimensional de una variedad algebraica con distintos enfoques pueden encontrarse en [8], [31] y [24].

Referencias

[1] Berkowitz, S., *On computing the determinant in small parallel time using a small number of processors.* Inform. Process. Lett. **18** (1984), No. 3, 147–150.

[2] Brownawell, W.D., *Bounds for the degrees in the Nullstellensatz.* Ann. of Math. (2) **126** (1987), 577–591.

[3] Bürgisser, P.; Clausen, M.; Shokrollahi, M.A., *Algebraic complexity theory.* Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences] **315**, Springer-Verlag, Berlin, 1997.

[4] Chistov, A.L.; Grigor'ev, D. Yu, *Subexponential time solving systems of algebraic equations.* LOMI preprint E-9-83, E-10-83, Steklov Institute, Leningrad (1983).

[5] Cox, D.; Little, J.; O'Shea, D., *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra.* Second edition. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1997.

[6] Cox, D.; Little J.; O'Shea, D., *Using algebraic geometry.* Graduate Texts in Mathematics **185**. Springer-Verlag, New York, 1998.

[7] Dickenstein, A.; Fitchas, N.; Giusti, M.; Sessa, C., *The membership problem for unmixed polynomial ideals is solvable in single exponential time.* Applied algebra, algebraic algorithms, and error-correcting codes (Toulouse, 1989). Discrete Appl. Math. **33** (1991), No. 1-3, 73–94.

[8] Elkadi, M.; Mourrain, B., *A new algorithm for the geometric decomposition of a variety.* Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC), 9–16 (electronic), ACM, New York, 1999.

[9] Fitchas, N.; Galligo, A., *Nullstellensatz effectif et conjecture de Serre (Théorème de Quillen-Suslin) pour le calcul formel.* Math. Nachr. **149** (1990), 231–253.

[10] Fitchas, N.; Giusti, M.; Smietanski, F., *Sur la complexité du théorème des zéros.* Approx. Optim. **8**, Approximation and optimization in the Caribbean, II (Havana, 1993), 274–329, Lang, Frankfurt am Main, 1995.

[11] Fulton, W., *Introduction to toric varieties.* Annals of Mathematics Studies **131**. The William H. Roever Lectures in Geometry. Princeton University Press, Princeton, NJ, 1993.

[12] Gelfand, I.; Kapranov, M.; Zelevinsky, A., *Discriminants, Resultants and Multidimensional Determinants.* Mathematics: Theory & Applications. Birkhäuser Boston, Inc., Boston, MA, 1994.

[13] Giusti, M.; Hägele, K.; Heintz, J.; Montaña, J.L.; Morais, J.E., Pardo, L.M., *Lower bounds for diophantine approximation.* J. Pure Appl. Algebra **117 & 118** (1997), 277–317.

[14] Giusti, M.; Heintz, J., *Algorithmes—disons rapides—pour la décomposition d'une variété algébrique en composantes irréductibles et équidimensionnelles.* Effective methods in algebraic geometry (Castiglioncello, 1990), 169–194, Progr. Math. **94**, Birkhäuser Boston, Boston, MA, 1991.

[15] Giusti, M.; Heintz, J., *La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial.* Computational algebraic geometry and commutative algebra (Cortona, 1991), 216–256, Sympos. Math., XXXIV, Cambridge Univ. Press, Cambridge, 1993.

[16] Giusti, M.; Heintz, J.; Morais, J.E.; Morgenstern, J.; Pardo, L.M., *Straight-line programs in geometric elimination theory.* J. Pure Appl. Algebra **124** (1998), No. 1-3, 101–146.

[17] Giusti, M.; Heintz, J.; Morais, J.E.; Pardo, L.M., *When polynomial equation systems can be "solved" fast?* Applied algebra, algebraic algorithms and error-correcting codes (Paris, 1995), 205–231, Lecture Notes in Comput. Sci. **948**, Springer, Berlin, 1995.

[18] Giusti, M.; Heintz, J.; Sabia, J., *On the efficiency of effective Nullstellensätze.* Comput. Complexity **3** (1993), No. 1, 56–95.

[19] Heintz, J., *Definability and fast quantifier elimination over algebraically closed fields.* Theoret. Comput. Sci. **24** (1983), 239–277.

[20] Heintz, J.; Krick, T.; Puddu, S.; Sabia, J.; Waissbein, A., *Deformation techniques for efficient polynomial equation solving.* J. Complexity **16** (2000), No. 1, 70–109.

[21] Heintz, J.; Schnorr, C.-P., *Testing polynomials which are easy to compute.* Logic and algorithmic (Zurich, 1980), pp. 237–254, Monograph. Enseign. Math. **30**, Univ. Genève, Geneva, 1982.

[22] Heintz, J.; Sieveking, M., *Lower bounds for polynomials with algebraic coefficients.* Theoret. Comput. Sci. **11** (1980), 321–330.

[23] Hermann, G., *Zur Frage der endlich vielen Schritte in der Theorie der Polynomideale.* Math. Ann. **95** (1926), 736–788.

[24] Jeronimo, G.; Krick, T.; Sabia, J.; Sombra, M., *The computational complexity of the Chow form.* Found. Comput. Math. **4** (2004), No. 1, 41–117.

[25] Jeronimo, G., Sabia, J., *Effective equidimensional decomposition of affine varieties.* J. Pure Appl. Algebra **169** (2002), No. 2-3, 229–248.

[26] Jouanolou, J.-P., *Théorèmes de Bertini et applications.* Progress in Math. **42**, Birkhäuser Boston, Inc., Boston, MA, 1983.

[27] Kollár, J., *Sharp effective Nullstellensatz.* J. Amer. Math. Soc **1** (1988), No. 4, 963–975.

[28] Krick, T.; Sabia, J.; Solernó, P., *On intrinsic bounds in the Nullstellensatz.* Appl. Algebra Engrg. Comm. Comput. **8** (1997), No. 2, 125–134.

[29] Kronecker, L., *Grundzüge einer arithmetischen Theorie de algebraischen Grössen.* J. reine angew. Math. **92** (1882), 1–122.

[30] Kunz, E., *Introduction to commutative algebra and algebraic geometry.* Birkhäuser Boston Inc., Boston, MA, 1985.

[31] Lecerf, G., *Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions.* Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation (St. Andrews), 209–216 (electronic), ACM, New York, 2000.

[32] Mulmuley, K., *A fast parallel algorithm to compute the rank of a matrix over an arbitrary field.* Combinatorica **7** (1987), No. 1, 101–104.

[33] Sabia, J.; Solernó, P., *Bounds for traces in complete intersections and degrees in the Nullstellensatz.* Appl. Algebra Engrg. Comm. Comput. **6** (1995), No. 6, 353–376.

[34] Schnorr, C.-P., *Improved lower bounds on the number of multiplications/divisions which are necessary to evaluate polynomials.* Theoret. Comput. Sci. **7** (1978), No. 3, 251–261.

[35] Schost, E., *Computing parametric geometric resolutions.* Appl. Algebra Engrg. Comm. Comput. **13** (2003), No. 5, 349–393.

[36] Shafarevich, I.R., *Basic Algebraic Geometry.* Springer-Verlag, New York-Heidelberg, 1974.

[37] Schwartz, J.T., *Fast probabilistic algorithms for verification of polynomial identities.* J. Assoc. Comput. Mach. **27** (1980), No. 4, 701–717.

[38] Sombra, M., *Bounds for the Hilbert function of polynomial ideals and for the degrees in the Nullstellensatz.* Algorithms for algebra (Eindhoven, 1996). J. Pure Appl. Algebra **117/118** (1997), 565–599.

[39] Walker, R.J., *Algebraic curves.* Dover Publications, Inc., New York, 1962.

[40] Zippel, R., *Probabilistic algorithms for sparse polynomials.* Symbolic and algebraic computation (EUROSAM '79, Internat. Sympos., Marseille, 1979), pp. 216–226, Lecture Notes in Comput. Sci. **72**, Springer, Berlin-New York, 1979.