UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Regla de los signos de Descartes óptima para circuitos multidimensionales

Anamilla Busca Canosa

Directora: Alicia Dickenstein

12 de diciembre de 2021

Agradecimientos

Escribí esta tesis como un cierre a una etapa como dicen siempre, pero que al menos para mí fue larguísima y que me tuvo muchos años convencida de que nunca iba a terminar. Sin embargo termina y estos últimos meses me hicieron repensar en lo que me permitió haber llegado hasta acá, en todo el esfuerzo detrás de bambalinas y agradecer mucho.

Quiero decir gracias a mis viejos. A mi mamá que, de alguna forma fantástica, hizo que nunca dudara que podía hacer lo que me proponía, me convenció que llegar es una cuestión de decisión y no de capacidad, la capacidad siempre esta. Porque además puso frente a mí la Universidad como el camino natural, aunque se olvidó de contarme sus matices. Creo que si elegí una carrera universitaria y me metí de lleno es sobre todo por ella en cada día de mi infancia y adolescencia. Y por mostrarme el feminismo, sin buscarlo, con experiencias que hubiéramos preferido ambas ahorrar y probablemente de más formas de las que somos conscientes pero acá estamos después de eso y mejores.

A mi viejo, que con todos sus tornasoles me mostró el camino de la computación. Me puso de chiquita frente a una maquina y me mostró la magia. El acercamiento a la matemática fue por otro lado, pero acercarme con una computadora a mano como una extensión de mi cuerpo creo que se lo debo sobre todo a él.

A mis hermanos, porque los hermanos que tocan son un poco una lotería y yo tengo una suerte increíble. A Camil, por todo, por hacerme pata siempre, por invitarme a que le haga pata, por cuidarme mientras la cuido, por festejarme cada logro, porque llegamos a la adultez un poco pateándonos para convertirnos en amigas, porque la vida es muy linda compartida con ella. A Lihue por esa infancia de niños, acompañándonos más de lo que decimos en voz alta, porque aprendimos juntos los límites, los golpes, pero por sobre todo a defendernos mutuamente. Por la adultez de a pares, cuestionándonos todo, preguntándonos todo y apoyándonos un montón. Porque te quiero más de lo que te lo digo. A Antual porque con él aprendí de cuidado, porque su curiosidad infranqueable me hizo preguntarme más de lo que ya lo hacía. Antu me acompañó de pequeño a cada evento matemático que se me antojo ir, para después convertirse en un adulto que me enseñó las habilidades que necesitaba para alcanzar mis objetivos. Porque siempre tiene un rato para explicarme como funcionan las cosas. Porque el tiempo pasa pero el amor no se quita. A Lienan, porque me trastocó entera. Él fue el primero en obligarme a pensar quien quiero ser y quien no, porque pensar en su infancia fue el motor de muchas de las cosas que hice y me definieron durante la mitad de mi vida. Porque hoy la infancia ya pasó y él es el que elige saltar las distancias de vuelta. Porque la relación que tenemos es también porque él se la puso al hombro y busco que fuera distinta, mejor. Porque me pide que le explique que cuernos dice en las siguientes 100 páginas con más paciencia de la que yo tendría.

A mis amigos. A Flor, porque llegó para amar a mi hermana de una forma hermosa y nos obligó a todos a querernos más, porque me volvió a unir a mi familia de una forma más sana y más feliz. Por las noches de filosofía barata, guiones de sit coms y jueguitos, siempre con una birra a mano, siempre festejando los pequeños logros. Porque ella pone la alegría a las reuniones a las que esta. Por los viajes y los planes infinitos, porque hace que todo sea un planazo. Porque mi plan para festejar esto es llamarlas. A Laura por estar siempre

y compartir todo, porque estamos a mano una de la otra desde siempre, porque a veces nos perdemos un rato, pero siempre volvemos. Porque siempre volvernos a ver es como si el tiempo no hubiera pasado. Gracias amiga. A Charly por las noches dándonos ánimos mientras estudiábamos, por los mensajes antes de rendir, porque nos acompañamos estas carreras infinitas hasta casi el final.

A mis suegros porque no hubiera jamás llegado tan lejos sin ellos. A Susana porque me cuidó en uno de los momentos más difíciles de mi vida, pero también de la de ella. Porque eligió activamente tener conmigo una relación hermosa y hoy eso se nota en cada día, porque eso también vuelve y mi futuro la incluye más de lo que ella quiere admitir. A Daniel por el cuidado y el cariño diarios, porque hoy que ya no es diario así que llega todo junto, por las charlas al azar y las risas. A ambos les debo el cuidado y el camino, me dieron las herramientas y el apoyo para no irme, para que esto termine. Gracias infinitas.

A Vibra en general y a Pablo en particular. Porque hace dos años llegar acá era imposible y llegué buscando un trabajo que no me impidiera avanzar. En cambio encontré a Pablo que primero me vio y después me enseñó muchas de las cosas que aplico en esta tesis, desarrollé la mayoría del código en paralelo mientras pensabamos jueguitos. Los últimos dos años los pasé agradeciendo esa entrevista, valorando este tiempo. Por la confianza, porque creyó en mí más que yo misma. Gracias. A mis compañeros que están siempre empujando por un poquito más, a Vibra que me dió el mejor ambiente posible para llegar hasta acá y las herramientas que me faltaban. A Saki, Tuchi y Franquito, porque llegué como pollito mojado y ellos me incluyeron en tres minutos, porque la matemática es un arte y ellos son grandes compañeros y un gran apoyo.

Por sobre todo a Javi porque él si que nunca dudo que iba a llegar hasta acá. Por la vida de a dos, porque los días no pueden ser tan malos si terminan con él y los días buenos son increíbles. Porque todo plan es una aventura fantástica. Porque nada de este camino hubiera sido tan hermoso ni valdría tanto la pena si no estuviera él para compartirlo. Porque te elijo cada día y ojala pudiera hacerte tan feliz como vos me haces a mi.

Por último quiero agradecer a Alicia, por trabajar en esta tesis conmigo, por la paciencia y la garra. A Gabriela y a Daniel por aceptar ser jurados, leer y evaluar este trabajo.

Índice general

Introducción

La Regla de los Signos de Descartes fue enunciada por primera vez por René Descartes en 1637 en *La Géométrie*, en un apéndice a su *Discours de la Méthode* [48]). Esta regla provee una cota superior para la cantidad de raíces positivas de un polinomio en $\mathbb{R}[x]$ que es fácil de computar. Específicamente, él establece que:

Regla de los Signos de Descartes *El número de raíces reales positivas contadas con multiplicidad de un polinomio $p \in \mathbb{R}[x]$ es, menor o igual al número de cambios de signo que se produzcan entre sus coeficientes ordenados eliminando los ceros.*

Si bien al momento de enunciarla, Descartes mostró con ejemplos la validez de la afirmación y la consideró cierta, en el trabajo original él no da una demostración formal sobre su veracidad. Más aún, en su enunciado original él habla de "variaciones" y "permanencias" de signos de una ecuación pero no las define precisamente y los problemas y contraejemplos ante esta falta de definiciones no tardaron en llegar. En 1828, Gauss da una definición apropiada para estos conceptos que permite interpretar la Regla de los Signos de Descartes como lo hacemos actualmente. Él da una demostración válida para la regla, pero además su definición da un marco general que permite validar los resultados obtenidos en los dos siglos anteriores. Además amplía la Regla de los Signos de Descartes mostrando que el número de raíces positivas y el número de variaciones de signos tienen la misma paridad [28]. Desde entonces, se han publicado varias pruebas distintas de la regla [1], [22], [51], [38],[36] y se han hecho varias generalizaciones en distintas direcciones. En 1999, Grabiner mostró que la cota de la Regla de los Signos de Descartes es ajustada, es decir, para una sucesión de signos dada existe un sucesión de coeficientes compatibles tales que el polinomio tiene el máximo posible de raíces positivas según la cota de Descartes [29].

En el Capítulo 1 haremos un repaso histórico de la Regla de los Signos de Descartes y sus interpretaciones desde el siglo XVII hasta la actualidad. Daremos el enunciado original de Descartes, estudiaremos las primeras definiciones para "variación" y "permanencia" de signos y las primeras demostraciones publicadas. Veremos como la definición de Gauss modificó el panorama llevando a la formulación actual (Teorema 1.7), sus aportes y las demostraciones que lo siguieron ampliando la regla original.

Luego daremos una demostración formal del Teorema 1.7 y de varios de sus corolarios directos. Enunciaremos el teorema de Boudan-Fourier que extiende la regla, permitiendo acotar las raíces en un intervalo arbitrario. Completaremos el capítulo con diversos resultados recientes referidos a la Regla de los Signos de Descartes en una variable.

No se conoce al presente ninguna generalización completa de la Regla de los Signos de Descartes en el caso multivariado, ni siquiera en forma de conjetura. Los esfuerzos por generalizar la Regla de los Signos de Descartes para múltiples variables se han enfocado en general en acotar la cantidad de soluciones en el ortante positivo de sistemas con n ecuaciones polinomiales en n variables, usando propiedades de signo de los coeficientes del sistema. Obtener una Regla de los Signos de Descartes es de especial importancia en el

estudio de soluciones positivas de modelos en biología y bioquímica donde las variables son concentraciones de especies químicas.

La primera conjetura para una cota como esta fue publicada en 1996 por Itenberg y Roy [31]. Ellos pudieron demostrar esta conjetura para algunos casos particulares. Los primeros en mostrar ejemplos no triviales que apoyen esta conjetura fueron Lagarias y Richardson [37] en 1997. Casi al mismo tiempo, Li y Wang dieron un contraejemplo para la conjetura de Itenberg-Roy [39].

Las únicas generalizaciones de la Regla de los Signos de Descartes para el caso multidimensional son las siguientes. La primera generalización fue dada recientemente e identificó sistemas con a lo sumo una solución en el ortante positivo [40] y [21]. Más adelante, en [10] y [11], Bihan, Dickenstein y Forsgård dieron una cota superior ajustada para sistemas polinomiales soportados en circuitos multidimensionales (ver Definición 2.2). En estos trabajos, la cota se da en términos de la variación de signos de los coeficientes de una sucesión asociada a los coeficientes y los exponentes del sistema. El trabajo presentado en esta tesis estará basado en estos artículos de Bihan, Dickenstein y Forsgård. Dedicaremos el Capítulo 2 a estudiar los sistemas polinomiales soportados en cicuitos, la definición de la sucesión asociada a los coeficientes y los exponentes del sistema y finalmente la cota para las soluciones en el primer ortante que brindan los autores.

La Regla de los Signos de Descartes permite una interpretación dual, abordada en el artículo reciente [27]. Dado $p \in \mathbb{R}[x_1 \ldots, x_n]$ un polinomio y $z(p)$ su conjunto de raíces, la Regla de los Signos de Descartes da una cota superior del número de componentes conexas de $V(p) := \mathbb{R}^n_{>0} \setminus z(p)$ en el caso $n = 1$. Además, si el signo del coeficiente principal está fijado, da una cota superior al número de componentes conexas donde las evaluaciones del polinomio son positivas o negativas. Otra generalización posible de la regla al caso multidimensional es la obtención de una cota superior optimal para el número de componentes conexas del conjunto $V(p) \subset \mathbb{R}^n_{>0}$. Hay una cota superior no ajustada en [8, Theorem 1] y en [5, Theorem 1.1] para el número de componentes conexas de un conjunto semi-algebraico. Las hipersuperficies de interés en la modelización de redes de reacciones bioquímicas tienen muchos monomios y variables y por lo tanto están fuera del alcance de los algoritmos estándar de geometría semi-algebraica, como los basados en la Descomposición Cilíndrica Algebraica, que tiene complejidad que puede ser doblemente exponencial en el número de variables [7, Remark 11.19]. En [27] se dan algunos primeros resultados en la acotación del número de componentes conexas de $V(p)$, obteniendo condiciones basadas en signos que permiten garantizar que el número de componentes conexas de $V(p)$ en las que p es negativo es a lo sumo 1 o 2 (que en el caso en que $n = 1$ corresponden al caso en que la variación de signos es 1).

En el Capítulo 3 introduciremos al lector en el uso de SINGULAR [24] para trabajar con polinomios y sistemas polinomiales. SINGULAR es un sistema de álgebra computacional para polinomios con énfasis en las necesidades del álgebra conmutativa, geometría algebraica y teoría de singularidades. Los objetos computacionales principales de SINGULAR son ideales y módulos sobre una gran cantidad de anillos distintos. Los anillos base pueden tomarse como anillos de polinomios y localizaciones sobre un cuerpo (cuerpos finitos, los racionales, extensiones algebraicas, extensiones trascendentes, etc), sobre un conjunto de anillos finitos dados o sobre anillos cociente respecto a un ideal. SINGULAR posee una de las más rápidas y más generales implementaciones de varios algoritmos para computar bases de Groebner y bases estandard. Más aún, provee algoritmos para factorizar polinomios, calcular resultantes, conjuntos característicos, máximo común divisor entre polinomios,

cálculos para resoluciones libres, etc.

El sistema SINGULAR está basado en el lenguaje de programación `C` y utiliza comando por consola para interactuar con el usuario. Además, la funcionalidad interna de SINGULAR puede ser extendida a partir de librerías en el lenguaje de programación SINGULAR. Una implementación general y eficiente de links permite a SINGULAR permitir que su funcionalidad esté disponible para otros programas.

Utilizaremos las librerías desarrolladas por Enrique Tobis en [50] para escribir un programa que permita calcular una cota a la cantidad de soluciones en el primer ortante de un sistema polinomial en varias variables utilizando los desarrollos que hicieron Bihan, Dickenstein y Forsgård en [10] y [11].

Más adelante, en el Capítulo 4 generalizaremos el programa detallado en el Capítulo 3 para dar un programa que calcule las cotas para gran cantidad de sistemas al mismo tiempo. Presentaremos los resultados que este cálculo masivo de cotas brinda con una aplicación web que permite filtrar por orden y configuración de potencias. Relacionaremos las cotas posibles y la cantidad de veces que se obtienen con los órdenes y la dimensión del sistema. Por último enunciaremos dos resultados propios (Teoremas 4.17 y 4.19) que surgen del análisis de los datos analizados. En el caso de n impar, obtenemos una cota precisa del número máximo de componentes abiertas disjuntas del espacio de coeficientes del sistema para las que la cota máxima $n+1$ puede alcanzarse para soportes $\mathcal{A}$ apropiados. Para el caso par, obtenemos una cota superior que podría mejorarse y por eso terminamos esta tesis proponiendo un problema.

1. La Regla de los signos de Descartes

En 1637 en su libro "La Géométrie" [48], Descartes enunció por primera vez su Regla de los Signos que permite hallar, de forma veloz e intuitiva, una cota para la cantidad de raíces de un polinomio. Para enunciarla en su versión actual comencemos tomando $p(x) = a_n x^n + \cdots + a_0 \in \mathbb{R}[x]$ un polinomio de grado n, es decir tal que $a_n \neq 0$ y $a_m = 0$ para todo $m > n$. Entonces podemos definir:

Definición 1.1. *Dado un elemento $a \in \mathbb{R}$ definimos la función $sg : \mathbb{R} \to \{-1, 0, 1\}$ como: $sgn(x) = 1$ si $x > 0$, $sgn(x) = -1$ si $x < 0$ y $sgn(x) = 0$ si $x = 0$.*

Definición 1.2. *Dada una sucesión de elementos ordenados $a = (a_n, \ldots, a_0) \in \mathbb{R}^{n+1}$, la variación de signos de a, $varsigns(a_n, \ldots, a_0)$ es la cantidad de pares distintos (i, j) tales que $0 \leq i < j \leq n$, $a_i \cdot a_j < 0$ y $a_k = 0$ para todo $i < k < j$. Es decir, es la cantidad de cambios de signo entre dos elementos consecutivos ignorando los elementos nulos.*

Notación 1.3. Si $p(x) = a_n x^n + \ldots a_0$ es un polinomio en $\mathbb{R}[x]$ notamos $z_+(p)$ a la cantidad de raíces positivas de p contadas con multiplicidad, $z_-(p)$ a la cantidad de raíces negativas de p contadas con multiplicidad y $z_0(p)$ a la cantidad de raíces nulas de p contadas con multiplicidad. Por último notemos $varsigns(p) = varsigns(a_n, \ldots a_0)$.

Teorema 1.4. *(La Regla de los Signos de Descartes versión moderna). Para cualquier polinomio $p \in \mathbb{R}[x]$ vale que:*

$$z_+(p) \leq varsigns(p).$$

Además $z_+(p) \equiv varsigns(p) \mod 2$.

Con la siguiente definición, el corolario es directo:

Definición 1.5. *Sea $p \in \mathbb{R}[x]$ un polinomio y tomemos $\tilde{p}(x) = p(-x)$, definimos la permanencia de signos de p, $permsigns(p)$ como la variación de signos de $\tilde{p}$: $permsigns(p) = varsigns(\tilde{p})$.*

Corolario 1.6. *Para cualquier polinomio $p \in \mathbb{R}[x]$ vale que:*

$$z_-(p) \leq permsigns(p).$$

Además $z_-(p) \equiv permsigns(p) \mod 2$.

Sin embargo para en el enunciado original de Descartes y para la matemática de la época no eran claras estas definiciones y por lo tanto el Teorema y su Corolario eran difíciles de estudiar y más aún, de poner en práctica.

En este capítulo comenzaremos estudiando el enunciado original de Descartes, sus primeras interpretaciones y demostraciones haciendo un recorrido histórico basado en el trabajo [9] de Bensimhoun. Veremos que la dificultad de los primeros años radicó en dar una definición apropiada para la variación y permanencia de signos cuando el polinomio tiene

coeficientes nulos. Justamente la falta de esta definición es la que hace que las pruebas antiguas no sean suficientemente amplias. En este contexto, estudiaremos las demostraciones de Segner y De Gua junto con las definiciones y convenciones que ellos hacen para llevar la regla a cabo.

Es Gauss en 1828 (recordemos que Descartes enunció su regla en 1637) quien consigue definir la variación y permanencia de signos de forma tal que la regla se vuelva aplicable naturalmente y su definición es la que se utiliza en la actualidad para enunciar la regla. Gauss además da una prueba algebraica, no inductiva y que permite visualizar profundamente las ideas de Descartes.

Estudiaremos luego las pruebas de Laguerre en 1883 y Krishnaiah en 1963. La prueba de Krishnaiah puede verse como una formulación inductiva de la prueba de Gauss, menos algebraica y quizás un poco más formal. Es interesante remarcar que en su prueba Krishnaiah plantea una inducción sobre la cantidad de variaciones de signos y, en cambio la idea original de Descartes, lo hace sobre el grado del polinomio que resulta más natural. Por último la prueba analítica de Laguerre se caracteriza por ser simple y proveer una extensión inmediata de la regla de Descartes a funciones racionales y series con un número finito de variaciones de signos.

1.1. Primeras interpretaciones

A partir de la traducción que Smith y Latham hacen en [46] podemos ver que Descartes no considera polinomios sino ecuaciones igualadas a cero, sobre las que asume (sin explicitar) que todos sus términos son no nulos. Como era natural en la época, considera "raíces" de la ecuación a las raíces positivas y llama "raíces falsas o menores a nada" a las raíces negativas de la ecuación. Él afirma, a partir de distribuir $(x-2)(x-3)(x-4)(x+5)=0$, que este polinomio es divisible por $(x-2)$, $(x-3)$, $(x-4)$ y $(x+5)$ pero no es divisible por $x+$ ni $x-$ ninguna otra cantidad. Luego enuncia:

> *"On connaît aussi de ceci combien il peut y avoir de vraies racines, et combien de fausses en chaque Équation. À savoir il y en peut avoir autant de vraies, que les signes + et – s'y trouvent de fois être changés ; et autant de fausses qu'il s'y trouve de fois deux signes + ou deux signes – qui s'entresuivent."*

Que puede traducirse como:

> *"Se conoce también de esto cuántas raíces verdaderas, y cuántas raíces falsas puede haber en cada Ecuación. A saber, puede haber tantas verdaderas, como la cantidad de veces que los signos + y − se encuentren cambiados; y tantas falsas como la cantidad de veces que dos signos + o dos signos − se sucedan."*

Podemos entonces separar la Regla de los Signos de Descartes original en dos partes bien diferenciadas:

Teorema 1.7 (Regla de los Signos de Descartes versión original)**.** *Si p es un polinomio con coeficientes reales.*

1. *La cantidad de raíces reales positivas de p es menor o igual a la cantidad de variaciones de signos entre coeficientes consecutivos.*

2. *La cantidad de raíces reales negativas de p es menor o igual a la cantidad de permanencias de signos entre coeficientes consecutivos.*

Despojados del contexto que proveen la Definición 1.2 y la Notación 1.3 es razonable preguntarse: ¿A qué refiere *coeficientes consecutivos*? ¿Cómo debieran contarse las variaciones y permanencias de signo entre los coeficientes de un polinomio?

Si el polinomio tiene todos sus coeficientes no nulos y los ordenamos en forma creciente respecto a las potencias de x, la forma de contar variaciones y permanencias de signo resulta clara:

Ejemplo 1.8. Tomemos $p(x) = x^4 - 4x^3 - 19x^2 + 106x - 120$. p tiene tres variaciones de signos: de x^4 a $-4x^3$, de $-19x^2$ a $106x$ y de $106x$ a -120. Además p tiene una permanencia de signos: de $-4x^3$ a $-19x^2$. Entonces el Teorema 1.7 afirma que $z_+(p) \leq 3$ y $z_-(p) \leq 1$. De hecho, este polinomio tiene tres raíces reales positivas $(2, 3, 4)$ y una negativa (-5), cumpliendo el enunciado de la Regla de los Signos de Descartes Original 1.7.

Sin embargo cuando el polinomio tiene coeficientes nulos esta forma de contar no era para nada obvia:

Ejemplo 1.9. Si tomamos el polinomio $p(x) = x^2 - 1$ y consideramos solamente los coeficientes no nulos tenemos que p tiene una variación de signo (de x^2 a -1) y ninguna permanencia de signos. Pero $p(x) = x^2 - 1 = (x + 1)(x - 1)$ tiene una raíz real positiva y una negativa, contradiciendo el enunciado de la Regla de los Signos de Descartes Original 1.7.

De todos modos, si le asignamos signos a los coeficientes nulos de p podemos llegar a un resultado verdadero. Por ejemplo, escribamos $p(x) = +x^2 + 0 \cdot x - 1$. Así p tiene una variación de signos (de $+0x$ a -1) y una permanencia de signos (de x^2 a $+0x$), que coincide con las raíces positivas y negativas del polinomio. Lo mismo vale si tomamos $p(x) = +x^2 - 0 \cdot x - 1$ aunque no es cierto en general que la alternancia y permanencia de los signos sea la misma si asignamos diferentes signos a los términos nulos.

Fue entonces necesario definir un método que permita contar las variaciones y permanencias de signos cuando hay coeficientes nulos para poder aplicar la regla en general.

1.1.1. Segner y De Gua: primeras demostraciones de la regla

Varios geómetras probaron la Regla de los signos de Descartes luego de que él mismo la enunciara, aunque no siempre de forma completamente válida o rigurosa. Al momento de enunciar la regla Descartes dio una idea de su demostración: las palabras "como resultado" luego de afirmar que un polinomio p con raíz en α es divisible por $(x-\alpha)$ sugieren que tenía una prueba por inducción, mostrando que multiplicar un polinomio por $(x - \alpha)$ con $\alpha > 0$ aumenta el número de variaciones de signo por al menos uno, mientras que multiplicarlo por $(x + \alpha)$ aumenta el número de permanencias de signo. Esta proposición es conocida como la Regla de Segner y no tuvo un enunciado y prueba correctos conocidos hasta 1828 en un trabajo de Gauss.

Segner publicó en 1728 en [45] una demostración de la regla de Descartes. Algunos autores sostienen en [6] y [49] que esta es la primer prueba rigurosa de la regla, aunque desafortunadamente es difícil encontrar este documento, originalmente escrito en latín. Gustaf Eneström escribió un resumen de esta prueba en alemán en Bibliotheca Mathematica [26] en 1906 y desde él podemos obtener las ideas principales de Segner. Él implícitamente supone que todos los coeficientes son no nulos y que todas las raíces son reales y utiliza

una versión del siguiente Teorema atribuido a Newton, la demostración se encuentra en [47, Th 2].

Teorema 1.10. *Sea p un polinomio con todos sus coeficientes y raíces reales, $p(x) = \sum_{i=0}^{n} a_n x^n$. Tomemos $\alpha_j = \frac{a_j}{\binom{n}{j}}$, entonces*

$$\alpha_j^2 \geq \alpha_{j-1} \cdot \alpha_{j+1}. \tag{1.1.1}$$

La condición (1.1.1) se conoce como *concavidad logarítmica*.

Eneström califica esta prueba de "esencialmente correcta", aunque solo es válida en el caso en el que todas las raíces del polinomio sean reales. Segner publicó una segunda demostración de la regla en 1756 [44] aparentemente no restringida a polinomios con todas sus raíces reales. Esta prueba fue luego explicada y ampliada en la Encyclopædia Britannica en 1824 [14]. Sin embargo la prueba original fallaba al no considerar cuidadosamente los coeficientes nulos del polinomio y este error no fue corregido en la versión de la Encyclopædia Britannica.

Por su parte, De Gua publicó en 1742 [23] una demostración rigurosa de la Regla de los Signos de Descartes para polinomios con todas sus raíces reales dividida en dos lemas. En uno de ellos De Gua afirma y demuestra una versión de lo que ahora se conoce como Lema de Segner para polinomios con todas sus raíces reales:

Lema 1.11 (Lema de Segner, 1756)**.** *Multiplicar un polinomio por $(x - \alpha)$ con $\alpha > 0$ aumenta en uno la variación de signos. Multiplicarlo por $(x + \alpha)$ con $\alpha > 0$ aumenta en uno la permanencia de signos.*

Las pruebas de Segner y De Gua siguen las mismas ideas pero no son igual de válidas, la diferencia radica en que la segunda se restringe a polinomios con todas sus raíces reales y la extensión a polinomios arbitrarios no es obvia. En sus pruebas ambos definen métodos equivalentes para contabilizar las variaciones y permanencias de signos entre los coeficientes del polinomio: De Gua reemplaza los coeficientes nulos por valores infinitesimales con signos arbitrarios [23, pág 83], mientras que Segner los reemplaza por ± 0 [44, pág 292]. Esta idea que, en un principio parece ser suficiente para la regla de Descartes, hace falso al Lema de Segner.

Ejemplo 1.12. Tomemos el polinomio $p(x) = x^3 - 1$. Asignando signos a los coeficientes nulos podemos considerar $p(x) = x^3 - 0x^2 + 0x - 1$, con lo que la variación de signos de sus coeficientes es 3. Si ahora multiplicamos al polinomio p por $(x - 1)$ obtenemos $q(x) = x^4 - x^3 - x + 1$. Asignando signos a sus coeficientes nulos podemos escribirlo como $q(x) = x^4 - x^3 - 0x - x + 1$ con variación de signos igual a 2. Esto contradice el Lema 1.11 pues debiera ser $varsigns(q) = varsigns(p) + 1$.

Podemos asumir entonces que la prueba de Segner no es rigurosa y el enunciado del lema, al menos con estas convenciones, es incorrecto. Sin embargo, si utilizamos las Definiciones 1.2 y 1.5 actuales el Lema de Segner es cierto. Por otro lado, como De Gua asume que el polinomio tiene todas sus raíces reales no nulas y esto implica necesariamente que no hay coeficientes nulos que sean consecutivos, este ejemplo no le aplica y su prueba es correcta.

De Gua publicó más tarde una segunda prueba analítica que no asume que las raíces del polinomio sean reales. Si bien no define explícita y correctamente las variaciones y permanencias de signo de los coeficientes, su prueba para la primera parte del Teorema

1.7 es correcta y muy bonita. De Gua comienza enunciando el siguiente Teorema previo. A continuación reproducimos la demostración que Bensimouhn hace en [9] y atribuye a De Gua, con algunas correcciones propias.

Teorema 1.13 (De Gua)**.** *Sea p un polinomio en $\mathbb{R}[x]$. Si los coeficientes de p tienen todos el mismo signo, entonces p no tiene raíces reales positivas. Si p tiene al menos una variación de signo, tomemos a_k y a_{k-j} dos coeficientes consecutivos no nulos de p en los que se produce la variación de signos, es decir tales que a_k y a_{k-j} tienen signos opuestos y $a_{k-i} = 0$ para $i = 1, \dots, j-1$. Definamos $f(x) = x^{-k}p(x)$ y $g(x) = x^{k+1}f'(x)$. Entonces:*

(a.) La variación de signos de g es uno menos que la de p: $varsigns(g) = varsigns(p) - 1$.

(b.) Si $p(x)$ tiene exactamente n raíces positivas contadas con multiplicidad entonces g tiene al menos $n-1$ raíces positivas.

Demostración. Sea $p(x) = \sum_{i=0}^{n} a_i x^i \in \mathbb{R}[x]$ de grado n tal que $a_n \neq 0$. Supongamos que $a_i \geq 0$ para todo i y que el polinomio no es nulo. Entonces si $\alpha > 0$ es un número real arbitrario, $p(\alpha) = \sum_{i=0}^{n} a_i \alpha^i > 0$ y por lo tanto p no tiene raíces positivas.

Supongamos ahora que p posee al menos una variación de signos y tomemos $k \leq n$ tal que a_k y a_{k-j} tienen signos opuestos. Defino $f(x) = x^{-k}p(x)$ y tomo $g(x) = x^{k+1}f'(x)$. Entonces:

$$\begin{aligned} g(x) =& x^{k+1}\left(\frac{p(x)}{x^k}\right)' = x^{k+1}\left(\frac{p'(x)\cdot x^k - k\cdot p(x)x^{k-1}}{(x^k)^2}\right) \\ =& x^{k+1}x^{k-1}\cdot\left(\frac{x\cdot p'(x) - k\cdot p(x)}{x^{2k}}\right) = x\cdot p'(x) - k\cdot p(x). \end{aligned}$$

Reemplazando $p(x) = \sum_{i=0}^{n} a_i x^i$ y $p'(x) = \sum_{i=0}^{n} i\cdot a_i x^{i-1}$ obtenemos:

$$\begin{aligned} g(x) =& x\cdot\sum_{i=0}^{n} i\cdot a_i x^{i-1} - k\cdot\sum_{i=0}^{n} a_i x^i \\ =& \sum_{i=0}^{n}(i\cdot a_i x^i - k\cdot a_i x^i) = \sum_{i=0}^{n}(i-k)\cdot a_i x^i. \end{aligned}$$

Tomando $varsigns(p)$ y $varsigns(g)$ como en la Definición 1.2 podemos ver que, si $k = n$:

$$\begin{aligned} varsigns(g) =& varsigns((-k)\cdot a_0, \dots, (-1)a_{n-j}) = varsigns(a_0, \dots a_{n-j}) \\ varsigns(p) =& varsigns(a_0, \dots a_{n-j}) + varsigns(a_{n-j}, a_n) = varsigns(a_0, \dots a_{n-j}) + 1. \end{aligned}$$

Con lo que $varsigns(p) = varsigns(g) + 1$.

En cambio, si $k \neq n$ y $a_{k+\tilde{j}}$ es el siguiente coeficiente no nulo de a_k, es decir $a_{k+\tilde{j}} \neq 0$ y $a_{k+i} = 0$ para $i = 1, \dots, \tilde{j} - 1$, tenemos que:

$$\begin{aligned} varsigns((-k)a_0, \dots, (-j)a_{k-j}) &= varsigns(a_0, \dots a_{k-j}) \\ varsigns((\tilde{j})a_{k+\tilde{j}}, \dots, (n-k)a_n) &= varsigns(a_{k+\tilde{j}}, \dots a_n). \end{aligned}$$

Y además

$$\begin{aligned} varsigns(p) =& varsigns(a_0, \dots a_{k-j}) + varsigns(a_{k-j}, a_k, a_{k+\tilde{j}}) + varsigns(a_{k+\tilde{j}}, \dots a_n) \\ varsigns(g) =& varsigns(a_0, \dots a_{k-j}) + varsigns(-a_{k-j}, 0, a_{k+\tilde{j}}) + varsigns(a_{k+\tilde{j}}, \dots a_n). \end{aligned}$$

Restando:

$$varsigns(p) - varsigns(g) = varsigns(a_{k-j}, a_k, a_{k+\tilde{j}}) - varsigns(-a_{k-j}, 0, a_{k+\tilde{j}}).$$

Por hipótesis sabemos que a_{k-j} y a_k tienen signos opuestos. Si a_k y $a_{k+\tilde{j}}$ tienen el mismo signo entonces $varsigns(a_{k-j}, a_k, a_{k+\tilde{j}}) = 1$ y $varsigns(-a_{k-j}, 0, a_{k+\tilde{j}}) = 0$. Si a_k y $a_{k+\tilde{j}}$ tienen signos distintos entonces $varsigns(a_{k-j}, a_k, a_{k+\tilde{j}}) = 2$ y $varsigns(-a_{k-j}, 0, a_{k+\tilde{j}}) = 1$. En cualquier caso conseguimos que: $varsigns(p) - varsigns(g) = 1$ con lo que concluimos que g tiene exactamente una alternancia de signo menos que p.

Recordemos ahora que $f(x) = x^{-k}p(x)$, entonces $z > 0$ es una raíz de f si y solo si es raíz de p. Además, por la regla de Leibniz, se puede ver que $f^{(j)}(x) = \sum_{i=0}^{j} q_{j,i}(x) \cdot p^{(i)}(x)$ con $q_{j,i}$ funciones racionales y $q_{j,j} = x^{-k}$ que no tiene raíces en $\mathbb{R} \setminus \{0\}$. Estas funciones $q_{i,j}$ son múltiplos de derivadas de x^-k, por lo tanto sus denominadores son potencias de x y no tiene problemas de dominio en $\mathbb{R}_{>0}$. Entonces si z es una raíz de multiplicidad m de p vale que $p^{(j)}(z) = 0$ para $j < m$ y $p^{(m)}(z) \neq 0$. Por lo tanto $f^{(j)}(z) = 0$ para $j < m$ y $f^{(m)}(z) \neq 0$ y z es una raíz de multiplicidad m de f.

Llamemos n a la cantidad de raíces positivas de p contadas con multiplicidad, N la cantidad de raíces positivas distintas y m_k la multiplicidad de la raíz z_k. Luego, por la expresión de f podemos afirmar que toda raíz positiva z_k de p es necesariamente raíz de f y viceversa. Además, por la expresión de $f^{(n)}$ podemos afirmar que las raíces de f y p tienen la misma multiplicidad, entonces si z_k es una raíz de p de multiplicidad m_k entonces es una raíz de multiplicidad $m_k - 1$ de f' (y de g). Por otro lado, entre dos raíces distintas de p la función f' debe anularse al menos una vez por el Teorema de Rolle. Entonces si f tiene N raíces positivas distintas, f' y g tendrán al menos $N - 1$. Así, la cantidad de raíces positivas de g es al menos:

$$\sum_{k=1}^{N}(m_k - 1) + (N - 1) = \sum_{k=1}^{N} m_k - N + N - 1 = n - 1.$$

De lo que concluimos que g tiene al menos $n - 1$ raíces positivas. □

Utilizando el Teorema 1.13, De Gua demuestra la siguiente versión de la Regla de los Signos de Descartes.

Teorema 1.14 (Regla de los signos de Descartes versión de De Gua)**.** *Un polinomio con todos sus coeficientes reales no tiene más raíces reales positivas que la cantidad de variaciones de signos entre coeficientes consecutivos.*

Demostración (De Gua). Sea $p \in \mathbb{R}[x]$, si p tiene todos sus coeficientes con el mismo signo entonces, por el Teorema 1.13, podemos afirmar que p no tiene raíces reales positivas.

Si p tiene variación de signos no nula tomemos k_1 una posición de cambio de signo de p, es decir tal que a_{k_1} y a_{k_1-j} son no nulos, tienen signos distintos y $a_{k_1-i} = 0$ para $i = 1, \dots j - 1$. Definimos f_1 y g_1 como en el Teorema 1.13: $f_1(x) = x^{-k_1}p(x)$ y $g_1(x) = x^{k_1+1}f_1'(x)$. Tomando $varsigns(p)$ y $z_+(p)$ como en la Definición 1.2 tenemos que $z_+(g_1) \geq z_+(p) - 1$ y $varsigns(g_1) = varsigns(p) - 1$.

Repetimos ahora el argumento para $p = g_1$: tomamos k_2 una posición de cambio de signo de g_1, defino f_2 y g_2 tales que $f_2(x) = x^{-k_2}g_1(x)$ y $g_2(x) = x^{k_2+1}f_2'(x)$. Entonces g_2 tiene exactamente una variación de signos menos y al menos $z_+(p) - 2$ raíces positivas. Inductivamente definimos g_i a partir de g_{i-1} que cumplirá que $varsigns(g_i) = varsigns(p) - i$

y $z_+(g_i) \geq z_+(p) - i$. Así, tomando $r = varsigns(p)$ tenemos

$$varsigns(g_r) = varsigns(p) - r = varsigns(p) - varsigns(p) = 0.$$

Así podemos afirmar que g_r tiene todos sus coeficientes con el mismo signo, por lo tanto no tiene raíces reales positivas, entonces:

$$z_+(p) - varsings(p) = z_+(p) - r \leq z_+(g_r) = 0.$$

Por lo tanto $z_+(p) \leq varsings(p)$ como queríamos ver. □

1.1.2. La versión de Gauss: un salto cualitativo

Casi dos siglos después del enunciado original de la Regla de los Signos de Descartes, en 1828, Gauss clarificó el enunciado y publicó una demostración de la regla completa válida para cualquier polinomio $p \in \mathbb{R}[x]$. La prueba es algebraica y sigue la estrategia original de Descartes, pero su particularidad radica en que reduce el enunciado a las variaciones de signo. Lo más importante es que, aparentemente por primera vez, la forma de considerar los coeficientes nulos en el cálculo de variaciones y permanencias de signo en la regla es exhibida formalmente.

Gauss define la variación de signos como la cantidad de cambios de signo entre los coeficientes del polinomio ignorando los coeficientes nulos y luego enuncia:

Definición 1.15 (Permanencia de signos de Gauss)**.** *Sea $p \in \mathbb{R}[x]$ un polinomio con sus términos ordenados de acuerdo a las potencias de x. La permanencia de signos del polinomio es la variación de signos de $p(-x)$:*

$$permsigns(p(x)) = varsigns(p(-x)).$$

Esto le permite enunciar la Regla de los Signos de Descartes y su Corolario exactamente como los enunciamos al principio del capítulo y dar una demostración para ambos casos. Gauss replica la demostración que expusimos para la Regla de los Signos de Descartes en 1.14 ahora extendiendo el resultado a las raíces negativas y la permanencia de signos. Puede encontrarse en [28].

El salto cualitativo para la Regla de los Signos de Descartes que da la intervención de Gauss radica en que él consigue dar una definición directa de la variación y permanencia de signos que solucionó los problemas de conteo con coeficientes nulos. A partir de su definición deja de ser necesario acudir a hipótesis sobre que todas sus raíces son reales o convenciones extrañas e infinitesimales para contar los cambios de signos. Esta forma de aplicar la Regla de los Signos de Descartes resulta natural y se ha convertido en estándar en la actualidad.

1.1.3. La prueba de Laguerre: una extensión de la regla

En 1883 Laguerre publicó una nueva prueba de la regla [38] que resulta simple y concisa. Su demostración utiliza las ideas de De Gua, pero en los tiempos de Laguerre era posible formularla en términos más claros lo que le permitió probar la regla para una extensión de los polinomios. Además el método de Gauss para contar las variaciones de signos ya era estándar y el conteo de variaciones de signos con coeficientes nulos ya no era un problema.

Laguerre enuncia:

Teorema 1.16. *Sea p una función definida en $\mathbb{R}_+$ de la forma:*

$$p(x) = a_{\lambda_1} x^{\lambda_1} + \cdots + a_{\lambda_N} x^{\lambda_N},$$

donde $\lambda_i \in \mathbb{R}$, $\lambda_i > \lambda_{i+1}$ y $a_{\lambda_i} \neq 0$. Entonces la cantidad de raíces positivas de p contadas con multiplicidad es menor o igual a la variación de signos de sus coeficientes.

La prueba de Laguerre utiliza las ideas que enuncia De Gua en el Teorema 1.13, pero su particularidad es que se extiende inmediatamente a series infinitas de la forma $\sum_i a_{\lambda_i} x^{\lambda_i}$ que posean una variación de signos de sus coeficientes finita.

1.1.4. Krishnaiah: una demostración moderna del Lema de Segner

La última prueba de nuestro interés es la formulada por Krishnaiah en 1963 [36] basada en la estrategia de Descartes y el Lema de Segner. En su trabajo, Krishnaiah llama a la prueba del lema que aparece en la literatura "una persuasión esquemática, cuya presentación rigurosa es bastante larga" y hace una demostración por inducción corta y concisa. Luego demuestra la Regla de los signos de Descartes como consecuencia.

Lema 1.17 (de Segner versión Krishnaiah)**.** *Si p es un polinomio en $\mathbb{R}[x]$ y $\alpha > 0$ un número real, entonces:*

$$varsigns((x-\alpha)p(x)) \geq varsigns(p(x)) + 1.$$

Demostración . Sea $p \in \mathbb{R}[x]$, vamos a proceder por inducción en $varsigns(p) = n$.

Si $varsigns(p) = 0$ entonces $p(x) = \sum_{k=0}^{m} a_k x^k$ tiene todos sus coeficientes a_k del mismo signo y por lo tanto

$$\begin{aligned}(x-\alpha)p(x) =& \sum_{k=0}^{m} a_k x^{k+1} - \sum_{k=0}^{m} \alpha \cdot a_k x^k \\ =& a_m x^{m+1} + \sum_{k=1}^{m+1} (a_{k-1} - \alpha \cdot a_k) x^k - \alpha \cdot a_0.\end{aligned}$$

Como a_m y $-\alpha \cdot a_0$ tienen signos opuestos entonces $varsigns((x-\alpha)p(x)) \geq 1$ como queríamos ver.

Supongamos que el lema es válido para cualquier polinomio con variación de signos estrictamente menor a n y tomemos $p(x) = \sum_{k=0}^{m} a_k x^k$ de grado m, con n variaciones de signo. Sea $s < m$ el último coeficiente en el que la variación de signos ocurre, es decir tal que $a_s \neq 0$ y $a_s \cdot a_k \leq 0$ para todo $k > s$. Definamos los polinomios $q(x) := \sum_{k=0}^{s} a_k x^k$ y $r(x) := \sum_{k=s+1}^{m} a_k x^k$ tales que $varsigns(q) = n-1$, $varsigns(r) = 0$ y $p(x) = q(x) + r(x)$. Sea $\alpha > 0$, entonces:

$$(x-\alpha)p(x) = (x-\alpha)q(x) + (x-\alpha)r(x)$$

Por hipótesis inductiva $varsigns((x-\alpha)q(x)) \geq n$ y $varsigns((x-\alpha)r(x)) \geq 1$.

Podemos escribir, como antes:

$$\begin{aligned}(x-\alpha)q(x) =& a_s x^{s+1} + \sum_{k=1}^{s} (a_{k-1} - \alpha \cdot a_k) x^k - \alpha \cdot a_0 \\ (x-\alpha)r(x) =& a_m x^{m+1} + \sum_{k=s+2}^{m} (a_{k-1} - \alpha \cdot a_k) x^k - \alpha \cdot a_{s+1} x^{s+1}.\end{aligned}$$

Entonces:

$$\begin{aligned}(x-\alpha)p(x) &= a_m x^{m+1} + \sum_{k=s+2}^{m}(a_{k-1}-\alpha\cdot a_k)x^k - \alpha\cdot a_{s+1}x^{s+1} \\ &\qquad + a_s x^{s+1} + \sum_{k=1}^{s}(a_{k-1}-\alpha\cdot a_k)x^k - \alpha\cdot a_0 \\ &= a_m x^{m+1} + \sum_{k=s+2}^{m}(a_{k-1}-\alpha\cdot a_k)x^k + (a_s-\alpha\cdot a_{s+1})x^{s+1} \\ &\qquad + \sum_{k=1}^{s}(a_{k-1}-\alpha\cdot a_k)x^k - \alpha\cdot a_0.\end{aligned}$$

Como a_s y a_{s+1} tenían signos opuestos, $sgn(a_s-\alpha\cdot a_{s+1}) = sgn(a_s) = -sgn(a_m)$. Así:

$$varsigns(a_m x^{m+1} + \sum_{k=s+2}^{m}(a_{k-1}-\alpha\cdot a_k)x^k + (a_s-\alpha\cdot a_{s+1})x^{s+1}) = varsigns((x-\alpha)r(x)) \geq 1.$$

$$varsigns((a_s-\alpha\cdot a_{s+1})x^{s+1} + \sum_{k=1}^{s+1}(a_{k-1}-\alpha\cdot a_k)x^k - \alpha\cdot a_0) = varsigns((x-\alpha)q(x)) \geq n.$$

Luego $varsigns((x-\alpha)p(x)) \geq n+1$. □

Teorema 1.18 (Regla de los signos de Descartes versión de Krishnaiah)**.** *Un polinomio con todos sus coeficientes reales no tiene más raíces reales positivas que la cantidad de variaciones de signos entre coeficientes consecutivos.*

Demostración (Krishnaiah). Sea p en $\mathbb{R}[x]$ tal que $z_+(p) = n$, vamos a hacer inducción en n: si $n = 0$, entonces como $varsigns(p) \geq 0$ la desigualdad se cumple trivialmente.

Si la desigualdad es cierta para cualquier polinomio con $n-1$ raíces positivas, veamos que vale para p con n raíces positivas. Sea $\alpha > 0$ una raíz de p, dividiendo por $(x-\alpha)$ tenemos que $p(x) = (x-\alpha)q(x)$ con $z_+(q) = z_+(p) - 1 = n-1$ y $varsigns(p) \geq varsigns(q)+1$ por el Lema de Segner. Por hipótesis inductiva tenemos que

$$\begin{aligned}z_+(q) &\leq varsigns(q) \\ z_+(p)-1 &\leq varsigns(q) \leq varsigns(p)-1.\end{aligned}$$

Entonces obtenemos que: $z_+(p) \leq varsigns(p)$ como queríamos ver.

□

1.2. Demostraciones y aplicaciones

Al comienzo del capítulo enunciamos

Teorema 1.4. *(La Regla de los Signos de Descartes versión moderna). Para cualquier polinomio $p \in \mathbb{R}[x]$ vale que:*

$$z_+(p) \leq varsigns(p).$$

Además $z_+(p) \equiv varsigns(p)$ mód 2.

y su Corolario inmediato:

Corolario 1.6. *Para cualquier polinomio $p \in \mathbb{R}[x]$ vale que:*

$$z_-(p) \leq permsigns(p).$$

Además $z_-(p) \equiv permsigns(p) \mod 2$.

Centraremos esta sección en completar sus demostraciones y enunciar otros corolarios de interés.

Demostración. (del Teorema 1.4). La desigualdad del Teorema ya fue probada en este capítulo, se pueden citar la primera parte las prueba de De Gua (ver 1.13) y la prueba de Krishnaiah (ver 1.17).

Resta ver que $z_+(p) - varsigns(p(x))$ es siempre un número par. Tomemos $p(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=k_0}^{n} a_i x^i$ con k_0 la menor potencia de x con coeficiente no nulo. Si a_{k_0} y a_n tienen el mismo signo entonces $varsigns(p)$ es par y, de la misma forma, si a_{k_0} y a_n tienen signos distintos, $varsigns(p)$ es impar. Además, si $x \in \mathbb{R}$ es un valor positivo cercano a 0, $p(x)$ tiene el mismo signo que a_{k_0} y si $x > 0$ es un valor suficientemente grande, $p(x)$ tiene el mismo signo que a_n.

Luego, si a_{k_0} y a_n tienen el mismo signo, $\lim_{x\to 0} p(x)$ y $\lim_{x\to+\infty} p(x)$ tienen el mismo signo. Como p es una función continua entonces debe tener una cantidad par de raíces. Luego tanto $z_+(p)$ como $varsigns(p)$ son cantidades pares. Aplicando un argumento análogo, si a_{k_0} y a_n tienen distinto signo, $\lim_{x\to 0} p(x)$ y $\lim_{x\to+\infty} p(x)$ tienen distinto signo y p tiene una cantidad impar de raíces. Entonces tanto $z_+(p)$ como $varsigns(p)$ son cantidades impares. □

Lema 1.19 (de Segner)**.** *Para cualquier polinomio $p \in \mathbb{R}[x]$ y $\alpha > 0$ vale que*

$$varsigns((x-\alpha)p(x)) \geq varsigns(p(x)) + 1.$$

Demostración. Ver la prueba de Krishnaiah en 1.17 □

Corolario 1.20. *Si $p \in \mathbb{R}[x]$ entonces*

$$varsigns(p(x)) + varsigns(p(-x)) \leq gr(p) - z_0(p).$$

Demostración. Si p es un polinomio con todos sus coeficientes no nulos entonces $z_0(p) = 0$. Además cada variación de signos en $p(x)$ corresponde a una permanencia en $varsigns(p(-x))$ y viceversa. Como en total hay $gr(p)+1$ coeficientes entonces entre ellos hay $gr(p)$ permanencias o variaciones de signos. Luego $varsigns(p(x)) + varsigns(p(-x)) = gr(p)$ como queríamos ver.

Si $p(x) = \sum_{i=0}^{n} a_i x^i$ es un polinomio arbitrario sin raíces nulas, defino

$$q(x) = \sum_{i=0}^{n} b_i x^i \text{ donde } b_i = a_{k_i} \text{ y } k_i = \text{máx}\{j \leq i \text{ tq } a_j \neq 0\}.$$

El polinomio q tiene el mismo grado que p pero sin coeficientes nulos, más aún vale que $varsigns(p(x)) = varsigns(q(x))$. Además observemos que

$$p(-x) = \sum_{i=0}^{n}(-1)^i a_i x^i \text{ y } q(-x) = \sum_{i=0}^{n}(-1)^i b_i x^i.$$

Entonces para cualquiera dos a_{j_1}, a_{j_2} coeficientes consecutivos no nulos de p vale que

$$\begin{aligned} varsigns((-1)^{j_1}a_{j_1}, (-1)^{j_2}a_{j_2}) &\leq varsigns((-1)^{j_1}a_{j_1}, (-1)^{j_1+1}a_{j_1}, \ldots, (-1)^{j_2-1}a_{j_1}, (-1)^{j_2}a_{j_2}) \\ &= varsigns((-1)^{j_1}b_{j_1}, (-1)^{j_1+1}b_{j_1+1}, \ldots, (-1)^{j_2}b_{j_2}). \end{aligned}$$

Y por lo tanto $varsigns(p(-x)) \leq varsigns(q(-x))$. Luego:

$$varsigns(p(x)) + varsigns(p(-x)) \leq varsigns(q(x)) + varsigns(q(-x)) \leq gr(q) = gr(p).$$

Lo que demuestra el Corolario para el caso en el que p no tiene raíces nulas.

Por último si p es de la forma $x^k q(x)$ con q un polinomio sin raíces nulas entonces:

$$\begin{aligned} varsigns(p(x)) &= varsigns(x^k q(x)) = varsigns(q(x)) \\ varsigns(p(-x)) &= varsigns((-x)^k q(-x)) = varsigns(q(-x)), \end{aligned}$$

y entonces

$$varsigns(p(x)) + varsigns(p(-x)) = varsigns(q(x)) + varsigns(q(-x)).$$

Como q es un polinomio sin raíces nulas entonces $varsigns(q(x)) + varsigns(q(-x)) = gr(q)$. Además $gr(q) = gr(p) - k = gr(p) - z_0(p)$, con lo que conseguimos

$$varsigns(p(x)) + varsigns(p(-x)) = gr(p) - z_0(p),$$

como queríamos ver. □

Corolario 1.21. *Si $p \in \mathbb{R}[x]$ es un polinomio con todas sus raíces reales, entonces:*

$$z_+(p) = varsigns(p(x)) \text{ y } z_-(p) = varsigns(p(-x)).$$

Demostración. Si p es un polinomio arbitrario, por el Corolario 1.20 tenemos que $varsigns(p(x)) + varsigns(p(-x)) \leq gr(p) - z_0(p)$. Además, utilizando el enunciado actual de la Regla de los Signos de Descartes 1.4, $z_+(p) + z_-(p) \leq varsigns(p(x)) + varsigns(p(-x))$. Unificando ambas ecuaciones conseguimos:

$$z_+(p) + z_-(p) \leq varsigns(p(x)) + varsigns(p(-x)) \leq gr(p) - z_0(p).$$

Si p tiene todas sus raíces reales, $z_+(p) + z_-(p) = gr(p) - z_0(p)$ entonces:

$$z_+(p) + z_-(p) = varsigns(p(x)) + varsigns(p(-x)) = gr(p) - z_0(p).$$

Luego $z_+(p) = varsigns(p(x))$ y $z_-(p) = varsigns(p(-x))$ como queríamos ver. □

Corolario 1.22. *Si $p \in \mathbb{R}[x]$ es un polinomio con k coeficientes no nulos entonces p tiene a lo sumo $2k-2$ raíces no nulas.*

Demostración. Si p tiene exactamente k coeficientes no nulos entonces p se escribe como:

$$p(x) = \sum_{i=1}^{k} a_{\lambda_i} x^{\lambda_i}.$$

Por lo tanto $varsigns(p(x)) = varsigns(a_{\lambda_1}, \ldots, a_{\lambda_k}) \leq k-1$ y $z_+(p) \leq k-1$. De la misma forma podemos asegurar que $varsigns(p(-x)) = k-1$ y $z_-(p) \leq k-1$. Luego p tiene a lo sumo $2k-2$ raíces no nulas. □

Observación 1.23. Notemos que esto tiene sentido cuando $\lambda_k \geq 2k-2$. De otro modo, como p es un polinomio de grado λ_k entonces la cantidad total de raíces contadas con multiplicidad $C_0(p)$ cumple:

$$z_+(p) + z_-(p) + z_0(p) \leq \#C_0(p) \leq gr(p) = \lambda_k \leq 2k-2$$

y el grado es una cota mucho mejor que la brindada por el Corolario 1.22.

Por último vamos a estudiar los Teoremas de Boudan - Fourier y de Sturm que dan generalizaciones a la Regla de los Signos de Descartes para hallar raíces en un intervalo. Para esto requeriremos introducir algunas definiciones y lemas previos. [7]

Definición 1.24. *Sea $\mathcal{P} = (P_0, P_1, \ldots, P_n)$ una sucesión ordenada de polinomios y sea $a \in \mathbb{R}$, definimos*

$$varsigns(\mathcal{P}, a) = varsigns(P_0(a), P_1(a), \ldots, P_n(a)),$$

y $varsigns(\mathcal{P}, a, b) = varsigns(\mathcal{P}, a) - varsigns(\mathcal{P}, b)$. Si además definimos $sgn(P(\infty)) = sgn(\lim_{x\to\infty} P(x))$ podemos extender la definición $varsigns(\mathcal{P}, a)$ para $a \in \mathbb{R} \cup \{-\infty, +\infty\}$.

Notación 1.25. Sea $p \in \mathbb{R}[x]$, notamos $z_{a,b}(p)$ a la cantidad de raíces de p en el intervalo $(a, b]$ contadas con multiplicidad.

Se puede encontrar la demostración al siguiente Teorema en [7, Th 2.35].

Teorema 1.26 (Boudan - Fourier, 1828)**.** *Sea $p \in \mathbb{R}[x]$ un polinomio de grado n, tomemos $\mathcal{P} = (p, p', \ldots p^{(n)})$ y sean $a, b \in \mathbb{R} \cup \{-\infty, \infty\}$. Entonces:*

$$z_{a,b}(p) \leq varsigns(\mathcal{P}, a, b).$$

Además $z_{a,b}(p) \equiv varsigns(\mathcal{P}, a, b) \mod 2$.

Observación 1.27. Notemos que $z_+(p) = z_{0,+\infty}(p)$ y que si $p(x) = a_n \cdot x^n + \cdots + a_0$ podemos expresar su $k-$ésima derivada como $p^{(k)}(x) = \frac{n!}{(n-k)!} \cdot a_n \cdot x^{n-k} + \cdots + k! \cdot a_k$. Entonces $sgn(\lim_{x\to+\infty} p^{(k)}(x)) = sgn(\frac{n!}{(n-k)!} a_n)$, $p(0) = k! a_k$. Por lo tanto $varsigns(\mathcal{P}, +\infty) = 0$ porque todos los polinomios de $\mathcal{P}$ tienen el mismo signo en $+\infty$ y $varsigns(\mathcal{P}, 0) = varsigns(a_0, a_1 \ldots, n! a_n) = varsigns(p)$. Luego el Teorema 1.26 afirma que:

$$z_+(p) = z_{0,+\infty}(p) \leq varsigns(\mathcal{P}, 0, +\infty) = varsigns(p).$$

Con lo que la Regla de los Signos de Descartes se desprende directamente del Teorema de Boudan - Fourier.

Definición 1.28. *Sea $p \in \mathbb{R}[x]$ de grado n definimos la sucesión de Sturm para p, $\hat{\mathcal{P}}$ como $\hat{\mathcal{P}} = (p_0, p_1 \dots p_n)$ dada por $p_0 = p$, $p_1 = p'$, e inductivamente tomemos p_{i+1} el resto de dividir p_i por p_{i-1}.*

El siguiente Teorema se encuentra demostrado en [7, Th 2.50]

Teorema 1.29 (Sturm, 1828)**.** *Sea $p \in \mathbb{R}[x]$ y $\hat{\mathcal{P}}$ su sucesión de Sturm, tomemos $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$, entonces $varsigns(\hat{\mathcal{P}}, a, b)$ es igual a la cantidad de raíces de p en el intervalo $(a, b]$.*

Observación 1.30. Las raíces en el Teorema de Sturm no se cuentan con multiplicidad.

Observación 1.31. Como consecuencia si $p \in \mathbb{R}[x]$ es un polinomio y $\hat{\mathcal{P}}$ su sucesión de Sturm entonces p tiene al menos una raíz real si y solo si $varsigns(\hat{\mathcal{P}}, -\infty, +\infty) > 0$.

Existe una generalización al Teorema de Sturm en varias variable espero que requiere trabajar en un anillo cociente y el uso de bases de Groebner o alguna herramienta similar, ver [41].

Ejemplo 1.32. Tomemos $p(x) = x^4 - 5x^2 + 4$, entonces:

$$p' = 4x^3 - 10x \qquad p'' = 12x^2 - 10$$
$$p''' = 24x \qquad p^{(iv)} = 24.$$

Y calculando la sucesión de Sturm nos queda:

$$p_0(x) = x^4 - 5x^2 + 4 \qquad p_1(x) = 4x^3 - 10x$$
$$p_2(x) = \frac{5}{2}x^2 - 4 \qquad p_3(x) = \frac{18}{5}x \qquad p_4(x) = 4.$$

La sucesión de signos de $\hat{\mathcal{P}}$ en el $+\infty$ es $+ + + + +$ con lo que la variación de signos es 0. En cambio en $-\infty$ la sucesión de signos de $\hat{\mathcal{P}}$ es $+ - + - +$ lo que devuelve variación de signos igual a 4. Entonces como $4 = varsigns(\hat{\mathcal{P}}, -\infty, +\infty) = z_{-\infty,+\infty}(p)$ el polinomio tiene todas sus raíces reales.

En efecto $x^4 - 5x^2 + 4 = 0$ tiene como soluciones $x = 1, -1, 2$ y -2.

1.3. Resultados en una variable

A continuación citamos los resúmenes de los resultados publicados más actuales relacionados con la Regla de los Signos de Descartes.

Una revisita a la regla de los signos de Descartes [3]
Los autores examinan la cota superior que brinda la regla de los signos de Descartes. En particular estudian las siguientes preguntas. Dada una sucesión de signos, ¿se puede hallar un polinomio con el máximo posible de raíces positivas y cuyos coeficientes cumplan la sucesión de signos? Más aún, la regla asegura que la cantidad de raíces positivas es menor o igual a la variación de signos y difiere en un número par. Para cada cantidad de raíces positivas permitida ¿es posible encontrar un polinomio con esta cantidad de raíces positivas y signos de sus coeficientes dados por la sucesión de signos? Los autores se limitan a considerar el caso en el que los polinomios tienen todos sus coeficientes no nulos, dejando como un problema abierto el caso de polinomios con coeficientes nulos. Muestran que la

cota siempre puede ser alcanzada para cualquier sucesión de signos que no contenga ceros y enuncian una fórmula explícita para hallar el polinomio. Muestran también que este polinomio puede ser modificado para reducir el número de raíces positivas por un número par sin afectar a los signos de los coeficientes. Entonces la regla de los signos de Descartes es "completa", al menos para sucesiones de signos sin ceros.

La regla de los signos de Descartes: una nueva construcción [29]
El autor presenta una fórmula explícita para un polinomio real de un grado dado n, cuyos coeficientes tienen signos preasignados y con la cantidad máxima de raíces reales positivas y negativas permitida por la regla de los signos de Descartes. El prueba que si θ_j $(j = 0, 1, \dots n)$ es una sucesión cualquiera de $0, 1, -1$ y $k > n$ entonces el polinomio $\sum_{j=0}^{n} \theta_j k^{-j^2} x^j$ tiene esta propiedad.

Polinomios, patrones de signos y la regla de los signos de Descartes [35]
Por la regla de los signos de Descartes, un polinomio p de grado d con todos sus coeficientes no nulos, c cambios y p permanencias de signo en sus coeficientes $(c+p = d)$ tiene $pos \leq c$ raíces positivas y $neg \leq p$ raíces negativas, donde $pos \equiv c(mod2)$ y $neg \equiv p(mod2)$. Para $1 \leq d \leq 3$, cualquier elección de una sucesión de signos de coeficientes de p (patrón de signos) y para cualquier (pos, neg) que satisface estas condiciones existe un polinomio p con exactamente pos raíces positivas y neg raíces negativas (todas simples). Para $d \geq 4$ esto no es cierto. Se observó que para $4 \leq d \leq 8$, en todos los casos no realizables $pos = 0$ o $neg = 0$. Fue conjeturado que esto sucedía para cualquier $d \geq 4$. El autor muestra un contraejemplo para esta conjetura con $d = 11$. Prueba que para el patrón de signos $(+, -, -, -, -, -, +, +, +, +, +, -)$ y el par $(1, 8)$, no existe un polinomio con 1 raíz positiva, 8 raíces negativas simples y un par de raíces conjugadas.

Polinomios, patrones de signos y la regla de Descartes [32]
La famosa regla de los signos de Descartes de 1637 que da una cota superior a la cantidad de raíces positivas de un polinomio real en una variable en términos de los cambios de signos de sus coeficientes, ha sido una fuente indispensable de inspiración para varias generaciones de matemáticos. Tratando de extender y afinar esta regla consideramos el conjunto de los polinomios reales en una variable de un grado dado, una sucesión de signos de sus coeficientes y cantidades dadas de raíces positivas y negativas. A pesar de la definición elemental del objeto principal de nuestro estudio, no es una cuestión trivial para qué patrones de signos y números de raíces positivas y negativas el conjunto correspondiente no es vacío. El principal resultado del presente artículo es el descubrimiento de una nueva familia infinita de combinaciones no realizables de patrones de signos y números de raíces positivas y negativas.

La regla de los signos de Descartes y el modulo de las raíces [33]
El paper trabaja con polinomios hiperbólicos con coeficientes no nulos. Un polinomio en $\mathbb{R}[x]$ es hiperbólico si todas sus raíces son reales. Un patrón de signos es una sucesión de "+" y "−" que empiezan con "+". Un patrón de signos de longitud $d+1$ es realizable si existe un polinomio hiperbólico $p(x) = x^d + \sum_{j=0}^{d-1} a_j x^j$, tal que $(+, sgn(a_{d-1}), \dots, sgn(a_0))$ coincide con el patrón de signos inicial.
En el paper el autor investiga como se relacionan entre sí los módulos de las raíces positivas y negativas de un polinomio hiperbólico, dependiendo del patrón de signos de sus coeficientes. Además obtiene una descripción exhaustiva en el caso en que el polinomio tenga exactamente un cambio de signo. Si un patrón de signos con un solo cambio de

signo: $(\underbrace{+,\cdots+}_{m \text{ veces}},\underbrace{-,\cdots-}_{n \text{ veces}})$ es realizable por un polinomio hiperbólico p, por la regla de los signos de Descartes, p tiene exactamente 1 raíz positiva y $d-1$ negativas. Esto es $p(x) = (x-\alpha)(x+\gamma_1)\dots(x+\gamma_{d-1})$ donde $\alpha > 0$ y $0 \leq \gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_{d-1}$. Llamemos $m^\star$ a la cantidad de raíces γ_i que son más grandes que α y $n^\star$ a la cantidad de raíces menores a α.
El resultado principal del paper entonces es: El patrón de signos con $1 \leq n \leq m$ es realizable solo por polinomios hiperbólicos con $n^\star \leq 2n-2$. En particular, si un polinomio hiperbólico tiene un solo coeficiente $a_0 < 0$ entonces $\gamma_j > \alpha$ para todo $j = 1, 2, \dots d-1$.
El autor estudia además varios casos importantes de patrones de signos con exactamente dos cambios de signo.

La regla de los signos de Descartes, el teorema de Rolle y sucesiones de pares compatibles [19]
Consideremos la sucesión s de signos de los coeficientes de un polinomio p en $\mathbb{R}[x]$ de grado d. La Regla de los signos de Descartes da condiciones de compatibilidad entre s y el par (r^+, r^-), donde r^+ es la cantidad de raíces positivas y r^- la cantidad de raíces negativas de p. Una pregunta reciente es si hay otras condiciones de compatibilidad, y como respuesta se brindó una lista de triples incompatibles (s, r^+, r^-) para funciones de grado mayor a 4 y conocida hasta grado 8. En este paper los autores se preguntan por las condiciones de compatibilidad para $(s, r_0^+, r_0^-, r_1^+, r_1^-, \dots, r_{d-1}^+, r_{d-1}^-)$ donde r_i^+ (respectivamente r_i^-) es la cantidad de raíces positivas (resp. negativas) de la $i-$ ésima derivada de p. Prueban que hasta grado 5 no hay otras condiciones de compatibilidad que las que provienen de la regla de los signos de Descartes, las condiciones mencionadas antes para cada i y las condiciones triviales que provienen del teorema de Rolle.

Polinomios de grado 5 y la regla de los signos de Descartes [18]
La regla de los signos de Descartes da condiciones necesarias para la cantidad de raíces positivas y negativas y por lo tanto determina configuraciones admisibles para los patrones de signos de los coeficientes y las cantidades de raíces positivas y negativas de un polinomio con todos sus coeficientes no nulos y grado d. Para $d \geq 4$ no todas las configuraciones admisibles son realizadas por un polinomio.
Los autores investigan el caso $d = 5$, en el que la escena completa se debe a A. Albouy and Y. Fu [2]. En el paper los autores explican la existencia y no existencia de polinomios que realizan las configuraciones admisibles mediante un análisis geométrico del discriminante de un conjunto (reducido) de polinomios universales de grado 5 y los hiperplanos determinados por los coeficientes nulos.

Modulo de raíces en polinomios hiperbólicos y la regla de los signos de Descartes [34]
Un polinomio hiperbólico es un polinomio en $\mathbb{R}[x]$ con todas sus raíces reales. Por la regla de los signos de Descartes para polinomios hiperbólicos (HP) con todos sus coeficientes no nulos, un HP con c cambios de signos y p preservaciones de signo en sus coeficientes tiene exactamente c raíces positivas y p raíces negativas. Para $c = 2$ y grado 6 el autor se pregunta: Cuando el modulo de las 6 raíces de un HP están ordenadas en forma creciente en la semirrecta real, ¿en qué lugares el modulo de sus dos raíces positivas depende de las posiciones de los dos cambios de signo en los coeficientes?

La regla de los signos de Descartes para polinomios con dos variaciones de signos[20]

Para sucesiones de $d+1$ signos $+$ y $-$ que empiezan con $+$ y tienen exactamente dos variaciones de signo, los autores dan condiciones suficientes para la no existencia de un polinomio de grado d en $\mathbb{R}[x]$ con signos de sus coeficientes iguales a la sucesión y con una cantidad de raíces definida compatible con la regla de los signos de Descartes.

Regla de los signos de Descartes, polígonos de Newton y polinomios sobre hipercuerpos (hyperfields)[4]
En este artículo los autores desarrollan una teoría sobre la multiplicidad de las raíces de polinomios sobre hipercuerpos y la usan para dar una prueba unificada y conceptual de la regla de los signos de Descartes y la regla de los polígonos de Newton.

2. La Regla de los signos de Descartes para sistemas soportados en circuitos

La Regla de los Signos de Descartes original fue ampliamente estudiada desde su formulación para polinomios en una variable como vimos en el Capítulo 1, sin embargo pocas generalizaciones se conocen para la Regla de los Signos de Descartes para polinomios en varias variables. En [10] y [11] los autores desarrollan una cota para la cantidad de soluciones en el ortante positivo de un sistema polinomial soportado sobre circuitos. En este capítulo estudiaremos la cota que brinda [11] y veremos que mejora la cota de [10].

Fijemos un conjunto ordenado de exponentes $\mathcal{A} = \{a_0, a_1, \dots a_{n+1}\} \subset \mathbb{Z}^n$ de cardinal $n+2$ y una matriz de coeficientes $C = (c_{ij}) \in \mathbb{R}^{n\times(n+2)}$. Consideremos entonces el sistema de ecuaciones asociado en las n variables $x = (x_1, \dots x_n)$ con soporte $\mathcal{A}$ definido como:

$$f_i(x) = \sum_{j=0}^{n+1} c_{ij} x^{a_j} = 0, \text{ donde } i = 1, \dots n \text{ y } x^{a_j} = \prod_{k=1}^{n} x_k^{a_{j,k}}. \tag{2.0.1}$$

Notemos $n_{\mathcal{A}}(C)$ a la cantidad de soluciones positivas del sistema contadas con multiplicidad. El objetivo del trabajo es encontrar (siguiendo las ideas de Descartes) una cota ajustada para $n_{\mathcal{A}}(C)$ cuando es finito. Frédéric Bihan y Alicia Dickenstein ya habían planteado en el Teorema 2.9 de [10] una cota superior para $n_{\mathcal{A}}(C)$ en el sentido de Descartes, sin embargo esta cota no es ajustada para cualquier soporte $\mathcal{A}$. El Teorema 2.18 es el principal de esta sección y da una cota para el sistema (2.0.1) que es ajustada para cualquier soporte $\mathcal{A}$.

2.1. Algunas condiciones sobre el sistema

Sean $\mathcal{A} \subset \mathbb{Z}^n$ un conjunto de $n+2$ elementos y $C \in \mathbb{R}^{n\times(n+2)}$, si $\mathcal{A}$ y C definen el sistema polinomial de ecuaciones (2.0.1), diremos que $\mathcal{A}$ es la configuración de potencias del sistema y C la matriz de coeficientes. A continuación enunciaremos condiciones necesarias sobre $\mathcal{A}$ y C para que $0 < n_{\mathcal{A}}(C) < \infty$.

Comencemos observando que si la matriz de coeficientes C no tiene rango n, el sistema (2.0.1) que define tendrá, a lo sumo, $n-1$ ecuaciones y n incógnitas. Asumiremos entonces que C tiene rango máximo, es decir, $rk(C) = n$ para simplificar las hipótesis y concentrarnos e el caso general.

Definición 2.1. *Un conjunto $\{v_1, \dots, v_r\}$ es afinmente independiente si cada vez que $\sum_{i=1}^r \alpha_i v_i = 0$ con $\alpha_i \in \mathbb{R}$, $i = 1, \dots, r$ y $\sum_{i=1}^r \alpha_i = 0$, vale que $\alpha_i = 0$.*

Definición 2.2. *Un conjunto $\mathcal{A} = \{a_0, \dots, a_{n+1}\} \in \mathbb{Z}^n$ de $n+2$ elementos es un circuito si es un conjunto mínimamente afinmente dependiente, es decir que todo subconjunto de $n+1$ elementos es afinmente independiente.*

Por otro lado, si multiplicamos todos los polinomios por un monomio x^b las soluciones en el toro $(\mathbb{C}^*)^n$ no se modifican. Más aún, el número de soluciones es invariante por cambios de variables monomiales inversibles en el toro. Entonces $n_{\mathcal{A}}(C)$ es un invariante afín de $\mathcal{A}$. Es útil en este contexto considerar la matriz $A \in \mathbb{Z}^{(n+1)\times(n+2)}$ definida como:

$$A = \begin{pmatrix} 1 & \dots & 1 \\ a_0 & \dots & a_{n+1} \end{pmatrix}. \tag{2.1.1}$$

Entonces $n_{\mathcal{A}}(C)$ es un invariante afín de A. Además $n_{\mathcal{A}}(C)$ debe depender de los menores maximales de A, entendiendo por menores maximales a los determinantes de las submatrices cuadradas de A de tamaño máximo. Más aún se puede observar que, $n_{\mathcal{A}}(C)$ es invariante por reenumeraciones de $\mathcal{A}$ si la misma permutación se aplica a las columnas de C.

Tomemos $b = (b_0, \dots, b_{n+1})$ en el núcleo de A, definido como:

$$b_j = (-1)^j \det(A(j)). \tag{2.1.2}$$

donde $A(j)$ es la matriz cuadrada que se obtiene de A eliminando la columna j. Como $rank(A) = n+1$ entonces $\dim(\ker(A)) = 1$ y b genera al núcleo de A.

Notemos que $\mathcal{A}$ es un circuito si y solo si todos los menores maximales de A son no nulos. Por lo tanto, deducimos que:

Lema 2.3. *Un conjunto $\mathcal{A} \subset \mathbb{Z}^n$ de $n+2$ elementos es un circuito si y solo si para todo $b \in \ker(A)$ no nulo, $b_j \neq 0$ para $j = 0, \dots, n+1$.*

Si la configuración de potencias $\mathcal{A} = \{a_0, \dots, a_{n+1}\} \subset \mathbb{Z}^n$ no es un circuito, C tiene rango máximo y $rk(A) < n+1$ entonces el sistema (2.0.1) puede ser reescrito como un sistema con n ecuaciones y menos de n variables, así obtendremos en general un sistema sobredeterminado. Este sistema en general resulta incompatible y no tiene soluciones. Si la configuración de potencias $\mathcal{A} = \{a_0, \dots, a_{n+1}\} \subset \mathbb{Z}^n$ no es un circuito y $rk(A) = n+1$ entonces el sistema (2.0.1) se puede reducir al caso de un circuito en tamaño menor: si $\#\mathcal{A} = n+1$ entonces el número de soluciones positivas de un sistema soportado en $\mathcal{A}$, en caso de que sean finitas, es a lo sumo 1 por la Proposición [12, Prop 3.3]:

Proposición 2.4. *Sean $\mathcal{A} = \{a_0, \dots, a_n\} \subset \mathbb{Z}^n$ afinmente independiente y $C \in \mathbb{R}^{n\times(n+1)}$ de rango n. Consideremos el sistema de ecuaciones asociado en las n variables $x = (x_1, \dots x_n)$ con soporte $\mathcal{A}$ definido como:*

$$f_i(x) = \sum_{j=0}^{n} c_{ij} x^{a_j} = 0, \text{ donde } i = 1, \dots n \text{ y } x^{a_j} = \prod_{k=1}^{n} x_k^{a_{j,k}} \tag{2.1.3}$$

tiene a lo sumo una solución positiva no degenerada y tiene exactamente una si y solo si para todo $i = 1, \dots n+1$ vale que $(-1)^i \det(C(i))$ tiene siempre el mismo signo.

Recordemos que una solución x es no degenerada si el jacobiano del sistema tiene rango máximo en x.

Asumiremos siempre que $\mathcal{A}$ es un circuito.

Por otro lado, si multiplicamos a izquierda la matriz de coeficientes C por una matriz inversible $M \in \mathbb{R}^{n\times n}$ obtenemos un sistema equivalente a (2.0.1) que cumple que $n_{\mathcal{A}}(C) = n_{\mathcal{A}}(MC)$. Luego $n_{\mathcal{A}}(C)$ depende solamente del espacio generado linealmente por las filas

de C. En este sentido, podemos enunciar una condición necesaria para la existencia de al menos una solución positiva del sistema (2.0.1). Sean $C_0, \dots, C_{n+1} \in \mathbb{R}^n$ los vectores columna de la matriz de coeficientes C y llamemos

$$\mathcal{C}^0 = \mathbb{R}_{>0}C_0 + \cdots + \mathbb{R}_{>0}C_{n+1}, \tag{2.1.4}$$

al cono positivo generado por estos vectores. Dada una solución $x \in \mathbb{R}^n_{>0}$, el vector $(x^{a_0}, \dots, x^{a_{n+1}})$ es positivo y por lo tanto

$$0 = x^{a_0} \cdot C_0 + \cdots + x^{a_{n+1}} \cdot C_{n+1} \in \mathcal{C}^0. \tag{2.1.5}$$

Luego el origen $0 \in \mathbb{R}^n$ pertenece al cono $\mathcal{C}^0$.

Definición 2.5. *Sea $C \in \mathbb{R}^{n\times(n+2)}$ definimos el Gale dual de C como la matriz $P \in \mathbb{R}^{(n+2)\times 2}$ cuyas columnas son una base del núcleo de C. En este caso denotaremos $P_0, \dots, P_{n+1} \in \mathbb{R}^2$ a los vectores fila de P. Llamaremos configuración Gale dual a las columnas de C a los vectores $\{P_0, \dots, P_{n+1}\}$. La configuración Gale dual es única salvo isomorfismos lineales, es decir salvo multiplicación a derecha de P por una matriz inversible $D \in \mathbb{R}^{2\times 2}$.*

Podemos reescribir la condición (2.1.5) en estos términos: Tomemos $\mathcal{B} = \{u, w\}$ una base del núcleo de C y P la matriz Gale dual de C que tiene a los vectores u y w como columnas. Puedo definir entonces los vectores P_j de la configuración Gale dual de C como $P_j = (u_j, w_j)$. Sea x un vector en el núcleo de C, como u y v lo generan, existen α y β tales que $x = \alpha u + \beta w$. Luego tenemos que $x_j = \langle P_j, (\alpha, \beta)\rangle$ para todo $j = 0, \dots n+1$ y por lo tanto

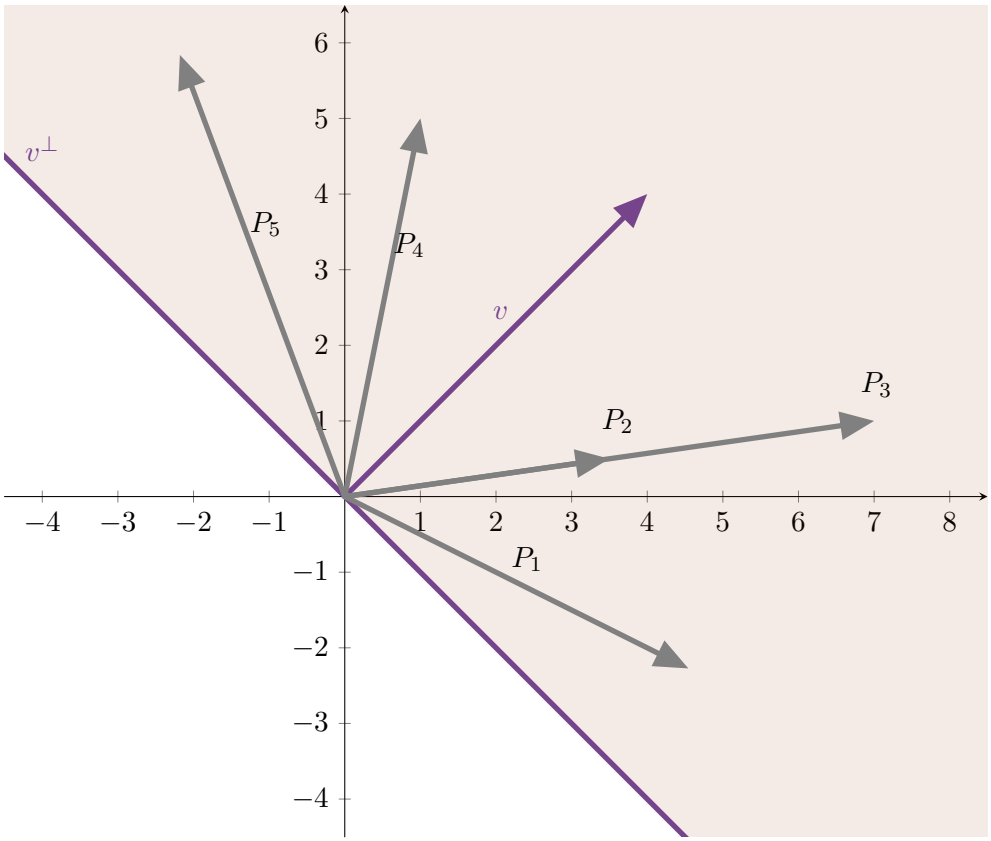
$$x = P \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{2.1.6}$$

Podemos afirmar así que todo vector del núcleo de C puede escribirse de la forma $P \cdot v$ para algún $v \in \mathbb{R}^2$ y por lo tanto la existencia de un vector positivo que es anulado por C es equivalente a la existencia de un vector v tal que:

$$\langle P_j, v\rangle > 0, \text{ para todo } j. \tag{2.1.7}$$

Además, como $\|P_j\|\|v\| \cos(\theta_j) = \langle P_j, v\rangle > 0$ con θ_j el ángulo que queda definido entre los vectores P_j y v, tenemos que $\theta_j \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. Luego, si tomamos $v^\perp$ un vector ortogonal a v, podemos ver que todos los vectores P_j caen en el semiplano abierto que pasa por el origen y esta definido por $v^\perp$, ver al fin de la Sección 2.1.

Más aún, se puede ver que los menores maximales de C se corresponden con los menores maximales de P con índices complementarios. Entonces las dependencias lineales de la configuración Gale dual reflejan las dependencias lineales de la configuración de las columnas de C: dos vectores P_i, P_j son colineales si y solo si el menor maximal de C que resulta de eliminar las columnas i y j es cero. Para una descripción más acabada de este hecho ver Lema 2.8



En lo que sigue asumiremos las siguientes hipótesis

Hipótesis 2.6. Asumiremos siempre que $\mathcal{A}$ es un circuito y que C es de rango máximo. Luego:

$$rk(A) = n + 1, \quad rk(C) = n. \tag{2.1.8}$$

Por otro lado, asumiremos que 0 pertenece al cono positivo generado por las columnas de C. Esto implica que, por (2.1.7) si $P_0, \dots P_{n+1} \in \mathbb{R}^2$ forman un Gale dual para C, existe un vector $v \in \mathbb{R}^2$ tal que

$$\langle P_j, v\rangle > 0, \text{ para todo } j,$$

y todos los vectores P_i pertenecen a un semiplano que pasa por el origen.

2.2. Resultados previos

Vamos a usar la notación $[k] := \{0, 1, \dots, k-1\}$ para cualquier $k \in \mathbb{N}$. Además, dados $i, j \in [n+2], i \neq j$ llamemos $C(i)$ a la submatriz de C con columnas indexadas en los índices $[n+2] \setminus \{i\}$ y $C(i,j)$ a la submatriz con columnas en los índices $[n+2] \setminus \{i, j\}$.

Llamaremos $\mathbb{S}_n$ al conjunto de permutaciones de los elementos $\{1, \dots, n\}$. Presentaremos los elementos $\sigma \in \mathbb{S}_n$ a partir de su rango, por ejemplo $\sigma \in \mathbb{S}_4$ dada por $\sigma = (4, 2, 1, 3)$ es la permutación tal que $\sigma(1) = 4$, $\sigma(2) = 2$, $\sigma(3) = 1$ y $\sigma(4) = 3$. Las permutaciones son siempre biyectivas.

Definición 2.7. *Sea C una matriz con rango maximal y $\{P_0, \dots, P_{n+1}\}$ una elección para su configuración Gale dual que satisfacen (2.1.7). Definimos la relación de equivalencia $\sim$*

en $[n+2]$ *como:*

$$i \sim j \Leftrightarrow det(P_i, P_j) = 0$$

Entonces podemos definir sobre las clases de equivalencia, de cardinal $k \leq n+1$*:*

$$[n+2]/\sim = \{K_0, \ldots, K_{k-1}\}$$

un orden canónico, eligiendo una orientación este orden es único. Eligiendo un orden arbitrario dentro de cada clase de equivalencia obtenemos una permutación $\sigma \in \mathbb{S}_{n+2}$ *tal que:*

$$\varepsilon \cdot \det(P_{\sigma_i}, P_{\sigma_j}) \geq 0, \text{ si } \sigma_i < \sigma_j$$

donde $\varepsilon \in \{-1, 1\}$ *depende de la orientación elegida.*

Sea $K \subset [n+2]$ *un conjunto de representantes de las clases de equivalencia* $K_0, \ldots, K_{k-1}$*. Notemos por* $\overline{\sigma}$ *a la biyección inducida por* σ*:*

$$\overline{\sigma} : [k] \to K$$

Entonces $\varepsilon \cdot \det(P_{\overline{\sigma}_i}, P_{\overline{\sigma}_j}) > 0$ *si* $\overline{\sigma}_i < \overline{\sigma}_j$*.*

Decimos que σ *es un orden y que* $\overline{\sigma}$ *es un orden estricto para* C*.*

Notemos además que estas definiciones son independientes de la elección de la configuración Gale dual dado que es única salvo transformaciones lineales. Además podemos trasladar un orden σ en los vectores del Gale dual de C a condiciones de signo en los menores maximales de C a través de los siguientes resultados:

Lema 2.8 (Lemma 2.10, [40])**.** *Sean* $C \in \mathbb{R}^{n\times(n+2)}$ *y* $P \in \mathbb{R}^{(n+2)\times 2}$ *matrices de rango máximo tales que* $Im(P) = \ker(C)$ *(luego los vectores fila de* P*,* $P_0, \ldots P_{n+1}$ *forman una configuración Gale dual de* C*). Entonces existe un número real* $\delta \neq 0$ *tal que:*

$$\delta \cdot \det(C(j_1, j_2)) = (-1)^{j_1+j_2} \det(P_{j_1}, P_{j_2}),$$

para cualquier subconjunto $j_1, j_2 \in [n+2]$ *tal que que* $j_1 < j_2$*.*

Se deduce inmediatamente del lema anterior la siguiente proposición.

Proposición 2.9. *Sea* $C \in \mathbb{R}^{n\times(n+2)}$ *una matriz de rango máximo que satisface* (2.1.5)*. Una biyección* $\sigma : [n+2] \to [n+2]$ *es un orden para* C *si y solo si existe* $\varepsilon \in \{-1, 1\}$ *tal que*

$$\varepsilon \cdot (-1)^{\sigma_i+\sigma_j} \cdot \det(C(\sigma_i, \sigma_j)) \cdot \frac{\sigma_i - \sigma_j}{i-j} \geq 0, \quad \forall i, j \in [n+2],\ i \neq j.$$

Definición 2.10. *Manteniendo las notaciones de la Definición 2.7, para cada* $l = 0, \ldots, k-1$ *tomamos:*

$$L_l = K_{\overline{\sigma}_0} \cup \cdots \cup K_{\overline{\sigma}_l}.$$

Tomemos b *un vector no nulo en el núcleo de* A *y consideremos las sucesiones* $\lambda = \{\lambda_l\}_{l\in[k]}$ *y* $\mu = \{\mu_l\}_{l\in[k]}$ *definidas por:*

$$\lambda_l = \sum_{j \in K_{\overline{\sigma}_l}} b_j \quad y \quad \mu_l = \sum_{j \in L_l} b_j.$$

Ambas sucesiones dependen del orden estricto $\overline{\sigma_l}$, pero no se incorporó a la notación por claridad.

Observación 2.11. Notemos que:

$$\mu_l = \lambda_0 + \dots \lambda_l.$$

En particular,

$$\mu_{k-1} = \sum_{l=0}^{k-1} \lambda_l = \sum_{j=0}^{n+1} b_j = 0$$

ya que el vector $(1, \dots 1)$ es una fila de A.

Observación 2.12. Sean b, $b' \in \ker(A)$ no nulos tales que $b \neq b'$. Como $\ker(A)$ tiene dimensión 1, existe $\alpha \in \mathbb{R}$ tal que $b' = \alpha \cdot b$ y reemplazando en su definición 2.10 tenemos que

$$\lambda_b = \alpha \cdot \lambda_{b'} \quad \text{y} \quad \mu_b = \alpha \cdot \mu_{b'}.$$

Definición 2.13. *Dadas $(h_1, h_2, \dots, h_s)$ una sucesión de funciones reales analíticas definidas en un intervalo abierto $I \subset \mathbb{R}$, diremos que satisfacen la regla de los signos de Descartes en I si, para cualquier sucesión de números reales $(a_1, a_2, \dots, a_s)$ no nula, la cantidad de raíces de la función $f_a = a_1 h_1 + \dots a_s h_s$ en I contadas con multiplicidad es siempre menor o igual a $varsigns(a)$.*

Observemos que la regla de Descartes en su enunciado clásico afirma que la base de monomios de $\mathbb{R}[x]$ satisface la regla de los signos de Descartes (en el formato enunciado en la Definición 2.13) en el intervalo abierto $(0, +\infty)$.

Definición 2.14. *Dadas $h_1, \dots h_s$ funciones reales, definimos el Wronskiano de $h_1, \dots h_s$ como:*

$$W(h_1, \dots, h_s) = \det \begin{pmatrix} h_1 & h_2 & \dots & h_s \\ h_1' & h_2' & \dots & h_s' \\ \vdots & \vdots & \dots & \vdots \\ h_1^{(s-1)} & h_2^{(s-1)} & \dots & h_s^{(s-1)} \end{pmatrix}. \tag{2.2.1}$$

Observación 2.15. Si un conjunto de funciones $h_1, \dots, h_s$ es linealmente dependiente en un intervalo, esto implica obligatoriamente que el Wronskiano correspondiente es uniformemente cero en el intervalo. Si existe k tal que h_k es combinación lineal de las funciones h_i entonces derivando a ambos lados de la igualdad se ve que las derivadas de h_k son combinación lineal de las derivadas de h_i y por lo tanto el Wronskiano es cero. Sin embargo, si el Wronskiano es identicamente nulo esto no implica en general que las funciones sean linealmente independientes. Por ejemplo si las funciones $h_1, \dots, h_s$ son no nulas, C^∞, de soporte compacto y disjuntos dos a dos entonces su Wronskiano es cero para cualquier evaluación. Sin embargo, si $h_1, \dots, h_s$ son analíticas y su Wornskiano es identicamente nulo entonces las funciones son linealmente independientes.

Proposición 2.16 (Part 5, items 87 and 90 en [42])**.** *Una sucesión de funciones $h_1, \dots, h_s$ satisface la regla de los signos de Descartes en $I \subset \mathbb{R}$ si y solo si se satisfacen las siguientes dos condiciones:*

(i) Para cualquier subconjunto de índices $1 \leq j_1 < j_2 < \dots < j_k \leq s$ vale que

$$W(h_{j_1}(x), \dots, h_{j_k}(x)) \neq 0.$$

(ii) Para cualquiera dos subconjuntos de índices $1 \leq j_1 < j_2 < \cdots < j_k \leq s$ *y* $1 \leq j'_1 < j'_2 < \cdots < j'_k \leq s$ *del mismo tamaño se tiene que*

$$W(h_{j_1}(x), \ldots, h_{j_k}(x)) \cdot W(h_{j'_1}(x), \ldots, h_{j'_k}(x)) > 0.$$

Es particular, si una sucesión de funciones analíticas $h_1, \ldots, h_s$ satisface la regla de los signos de Descartes en $I \subset \mathbb{R}$ entonces $h_1, \ldots, h_s$ no se anulan, tienen el mismo signo en I y no hay dos de ellas que sean múltiplos.

Demostración. Sean primero $h_1, \ldots, h_s$ funciones que cumplen la Regla de los Signos de Descartes en I y veamos que cumplen las propiedades (i) y (ii).

Para ver el ítem (i), tomemos un subconjunto de índices cualquiera $1 \leq j_1 < j_2 < \cdots < j_k \leq s$ y supongamos que $W(h_{j_1}(x), \ldots, h_{j_k}(x)) = 0$ para cierto valor de $x = x_0 \in I$. Tomemos

$$W(x) = \begin{pmatrix} h_{j_1}(x) & h_{j_2}(x) & \ldots & h_{j_k}(x) \\ h'_{j_1}(x) & h'_{j_2}(x) & \ldots & h'_{j_k}(x) \\ \vdots & \vdots & \ddots & \vdots \\ h_{j_1}^{(k-1)}(x) & h_{j_2}^{(k-1)}(x) & \ldots & h_{j_k}^{(k-1)}(x) \end{pmatrix} \tag{2.2.2}$$

y entonces $W(h_{j_1}(x), \ldots, h_{j_k}(x)) = \det(W(x))$. Como $\det(W(x_0)) = 0$ por hipótesis, $W(x_0)$ es una matriz con núcleo de dimensión no nula y entonces podemos tomar $a = (a_{j_1}, \ldots, a_{j_k}) \in \ker(W(x_0))$, $a \neq \overline{0}$. Luego a verifica el siguiente sistema:

$$\begin{cases} a_{j_1}h_{j_1}(x_0) + & a_{j_2}h_{j_2}(x_0) + & \cdots + & a_{j_k}h_{j_k}(x_0) = 0 \\ a_{j_1}h'_{j_1}(x_0) + & a_{j_2}h'_{j_2}(x_0) + & \cdots + & a_{j_k}h'_{j_k}(x_0) = 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{j_1}h_{j_1}^{(k-1)}(x_0) + & a_{j_2}h_{j_2}^{(k-1)}(x_0) + & \cdots + & a_{j_k}h_{j_k}^{(k-1)}(x_0) = 0 \end{cases} \tag{2.2.3}$$

Así, x_0 es una raíz de multiplicidad al menos k de $F(x) = a_{j_1}h_{j_1}(x) + a_{j_2}h_{j_2}(x) + \cdots + a_{j_k}h_{j_k}(x)$. Como $a \in \mathbb{R}^k$ tenemos que $varsigns(a) \leq k-1$ y por lo tanto la cantidad de raíces de F en I contadas con multiplicidad es mayor que la variación de signos de a. Esto contradice el hecho de que $h_1(x), \ldots h_s(x)$ cumplen la Regla de los Signos de Descartes en I, luego no existe x_0 tal que $W(h_{j_1}(x_0), \ldots, h_{j_k}(x_0)) = 0$

Para ver el ítem (ii), sea $k \leq s$ y veamos que todos los Wronskianos con $k-1$ columnas tienen el mismo signo. Sea $x_0 \in I$, por el ítem (i) sabemos que

$$W(h_{j_1}(x_0), \ldots, h_{j_k}(x_0)) = \det(W(x_0)) \neq 0$$

Por lo tanto $W(x_0)$ es una matriz inversible y podemos tomar $a = (a_{j_1}, \ldots, a_{j_k})$ tal que

$$W(x_0) \cdot \begin{pmatrix} a_{j_1} \\ \vdots \\ a_{j_{k-1}} \\ a_{j_k} \end{pmatrix} = (0, \ldots, 0, 1) \tag{2.2.4}$$

Así x_0 es una raíz de multiplicidad $k-1$ de $F(x) = a_{j_1}h_{j_1}(x) + a_{j_2}h_{j_2}(x) + \cdots + a_{j_k}h_{j_k}(x)$ y como $h_1, \ldots, h_s$ cumplen la Regla de los Signos de Descartes en I, $k-1 \leq varsigns(a)$. Además, como $a \in \mathbb{R}^k$, $varsigns(a) \leq k-1$ con lo que podemos concluir que $varsigns(a) = k-1$ y las coordenadas a_i son todas alternadas.

Notemos W_i al determinante de la matriz que surge de quitar la columna i y la fila k a la matriz $W(x_0)$, W_i corresponde al Wronskiano de las k funciones originales $h_{j_1}, \ldots, h_{j_s}$ descartando la función h_{j_i}. Aplicando la Regla de Cramer a la Ecuación (2.2.4), definimos V_i la matriz que resulta de reemplazar la columna i de W por el vector $(0, \ldots, 0, 1) \in \mathbb{R}^k$ y podemos ver que para cada $i = j_1, \ldots, j_k$ vale que

$$a_i = \frac{\det(V_i)}{\det(W)}.$$

Desarrollando el determinante por la columna $i-$ésima obtenemos que $\det(V_i) = (-1)^{i+1} W_i$. Luego

$$(-1)^{i+1} a_i \det(W) = W_i$$

Como los a_i son todos alternados, $(-1)^{i+1} a_i \det(W)$ tiene siempre el mismo signo para todo i y en conclusión todos los W_i Wronskianos de $k-1$ elementos tienen el mismo signo. Como k era arbitrario podemos concluir que

$$W(h_{j_1}(x), \ldots, h_{j_k}(x)) \cdot W(h_{j'_1}(x), \ldots, h_{j'_k}(x)) > 0$$

para cualquier elección de índices $1 \leq j_1 < j_2 < \cdots < j_k \leq s$ y $1 \leq j'_1 < j'_2 < \cdots < j'_k \leq s$ del mismo tamaño.

Tomemos ahora funciones $h_1, \ldots, h_s$ que cumplen (i) y (ii), $a = (a_1, \ldots, a_s) \in \mathbb{R}^s \setminus \{0\}$, z la cantidad de raíces en I de $F(x) = a_1 h_1(x) + a_2 h_2(x) + \cdots + a_s h_s(x)$ y $c = varsigns(a)$. Queremos ver que $z \leq c$.

Supongamos primero que $c = 0$ y que existe $x_0 \in I$ tal que $F(x_0) = 0$. Como $varsigns(a) = c = 0$, las coordenadas de a tienen todas el mismo signo. Además, aplicando (ii) con $k = 1$ tenemos que $h_i(x) = W(h_i)(x) \neq 0$ y tiene el mismo signo para todo $i = 1, \ldots s$.

Ahora como $F(x_0) = 0$ tenemos que:

$$\begin{aligned} 0 =& a_1 h_1(x_0) + a_2 h_2(x_0) + \cdots + a_s h_s(x_0) \\ -a_s h_s(x_0) =& a_1 h_1(x_0) + a_2 h_2(x_0) + \cdots + a_{s-1} h_{s-1}(x_0) \end{aligned}$$

Pero por lo anterior $sgn(a_1 h_1(x_0) + a_2 h_2(x_0) + \cdots + a_{s-1} h_{s-1}(x_0)) = sgn(a_s h_s(x_0))$. Luego $a_s h_s(x_0) = 0$ y como $h_s(x_0) \neq 0$ tenemos que $a_s = 0$. Además $a_1 h_1(x_0) + a_2 h_2(x_0) + \cdots + a_{s-1} h_{s-1}(x_0) = 0$. Repitiendo el argumento inductivamente concluimos que $a = 0$, lo que contradice las hipótesis.

Ahora si $c \neq 0$, supongamos que si $h_1, \ldots, h_k$ cumplen (i) y (ii) entonces satisfacen la Regla de los Signos de Descartes en I para cualquier $k < s$ y veamos que vale para $k = s$. Tomemos j_α una posición de cambio de signo de a, $j_{\alpha-1}$ y $j_{\alpha+1}$ las coordenadas siguiente y posterior a j_α que son no nulas. Es decir, tomemos $j_{\alpha-1}$ y $j_{\alpha+1}$ tales que $a_j = 0$ si $j_{\alpha-1} < j < j_\alpha$ o $j_\alpha < j < j_{\alpha+1}$, $a_{j_{\alpha-1}}, a_{j_\alpha}, a_{j_{\alpha+1}} \neq 0$ y $a_{j_\alpha} \cdot a_{j_{\alpha+1}} < 0$. Definamos

$$F^*(x) = \left(\frac{F(x)}{h_{j_\alpha}(x)}\right)' = a_1 \left(\frac{h_1(x)}{h_{j_\alpha}(x)}\right)' + a_2 \left(\frac{h_2(x)}{h_{j_\alpha}(x)}\right)' + \cdots + a_s \left(\frac{h_s(x)}{h_{j_\alpha}(x)}\right)'.$$

Como toda raíz de F es raíz de $\dfrac{F(x)}{h_{j_\alpha}(x)}$, tenemos que $z^* \geq z - 1$ con $z^\star$ la cantidad de raíces de F^* en I.

Definamos ahora:

$$H_1(x) = -\left(\frac{h_1(x)}{h_{j_\alpha}(x)}\right)', \; H_2(x) = -\left(\frac{h_2(x)}{h_{j_\alpha}(x)}\right)', \; \dots \; H_{j_\alpha-1}(x) = -\left(\frac{h_{j_\alpha-1}(x)}{h_{j_\alpha}(x)}\right)',$$

$$H_{j_\alpha}(x) = \left(\frac{h_{j_\alpha+1}(x)}{h_{j_\alpha}(x)}\right)', \; \dots \; H_{s-1}(x) = \left(\frac{h_s(x)}{h_{j_\alpha}(x)}\right)'.$$

Entonces

$$F^*(x) = -a_1H_1(x) - \cdots - a_{j_\alpha-1}H_{j_\alpha-1}(x) + a_{j_\alpha+1}H_{j_\alpha}(x) + \cdots + a_sH_{s-1}(x).$$

Llamando c^* a la variación de signos de los coeficientes de las funciones H_i en F^* tenemos que

$$\begin{aligned}
c^* &= varsigns(-a_1, \dots, -a_{j_{\alpha-1}}, a_{j_{\alpha+1}}, \dots, a_s) \\
&= varsigns(-a_1, \dots, -a_{j_{\alpha-1}}) + varsigns(-a_{j_{\alpha-1}}, a_{j_{\alpha+1}}) + varsigns(a_{j_{\alpha+1}}, \dots, a_s) \\
&= varsigns(a_1, \dots, a_{j_{\alpha-1}}) + varsigns(a_{j_{\alpha-1}}, a_{j_\alpha}, a_{j_{\alpha+1}}) - 1 + varsigns(a_{j_{\alpha+1}}, \dots, a_s) \\
&= varsigns(a_1, \dots, a_{j_{\alpha-1}}, a_{j_\alpha}, a_{j_{\alpha+1}}, \dots, a_s) - 1 \\
&= c - 1,
\end{aligned}$$

donde para ver la igualdad

$$varsigns(-a_{j_{\alpha-1}}, a_{j_{\alpha+1}}) = varsigns(a_{j_{\alpha-1}}, a_{j_\alpha}, a_{j_{\alpha+1}}) - 1$$

recordemos que $a_{j_\alpha} \cdot a_{j_{\alpha+1}} < 0$ y entonces

$$\begin{aligned}
varsigns(a_{j_{\alpha-1}}, a_{j_\alpha}, a_{j_{\alpha+1}}) &= varsigns(a_{j_{\alpha-1}}, a_{j_\alpha}) + varsigns(a_{j_\alpha}, a_{j_{\alpha+1}}) \\
&= varsigns(a_{j_{\alpha-1}}, a_{j_\alpha}) + 1 \\
&= varsigns(a_{j_{\alpha-1}}, -a_{j_{\alpha+1}}) + 1 \\
&= varsigns(-a_{j_{\alpha-1}}, a_{j_{\alpha+1}}) + 1.
\end{aligned}$$

Veamos ahora que las funciones $H_1, \dots, H_n$ cumplen (i) y (ii). Tomemos una elección de índices arbitraria que no incluya a j_α, $1 \le j_1 < \cdots < j_d < j_\alpha < j_{d+1} < \cdots < j_k \le s$. Notemos que para una función ϕ, por propiedades del determinante y la regla de Leibniz se tiene que:

$$W(\phi h_{j_1}, \dots, \phi h_{j_k}) = \phi^k W(h_{j_1}, \dots, h_{j_k}).$$

Entonces:

$$W(h_{j_1}, \dots, h_{j_d}, h_{j_\alpha}, h_{j_{d+1}}, \dots, h_{j_k}) = h_{j_\alpha}^{k+1} \cdot W\left(\frac{h_{j_1}}{h_{j_\alpha}}, \dots, \frac{h_{j_d}}{h_{j_\alpha}}, 1, \frac{h_{j_{d+1}}}{h_{j_\alpha}}, \dots, \frac{h_{j_k}}{h_{j_\alpha}}\right)$$

Desarrollando el determinante por la coordenada $(d+1)-$ésima:

$$\begin{aligned}
&= h_{j_\alpha}^{k+1} \cdot (-1)^{d+2} \cdot W\left(\left(\frac{h_{j_1}}{h_{j_\alpha}}\right)', \dots, \left(\frac{h_{j_d}}{h_{j_\alpha}}\right)', \left(\frac{h_{j_{d+1}}}{h_{j_\alpha}}\right)', \dots, \left(\frac{h_{j_k}}{h_{j_\alpha}}\right)'\right) \\
&= h_{j_\alpha}^{k+1} \cdot (-1)^{d+2} \cdot (-1)^d \cdot W\left(H_{j_1}, \dots H_{j_k-1}\right)
\end{aligned}$$

$$= h_{j_\alpha}^{k+1} \cdot W\left(H_{j_1}, \dots H_{j_k - 1}\right)$$

Como $h_1, \dots h_s$ cumplen (i) y (ii) tenemos que $H_1, \dots, H_{s-1}$ también y, por hipótesis inductiva, $H_1, \dots, H_{s-1}$ verifican la Regla de los Signos de Descartes en I.

Recapitulando, para $F^*(x) = -a_1 H_1(x) - \cdots - a_{j_{\alpha-1}} H_{j_{\alpha-1}}(x) + a_{j_{\alpha+1}} H_{j_{\alpha+1}}(x) + \cdots + a_s H_{s-1}(x)$ debe valer

$$\begin{aligned} z^* &\leq c^* \\ z - 1 \leq z^* &\leq c^* = c - 1 \\ z &\leq c, \end{aligned}$$

como queríamos ver. □

Tomemos $\{P_0, \dots, P_{n+1}\}$ una configuración Gale Dual para la matriz C de coeficientes definida para el sistema (2.0.1). Sabemos que existe un vector $v \in \mathbb{R}^2$ tal que $\langle P_j, v \rangle > 0$ para todo j por (2.1.7). Como $(v_1, v_2) \neq (0, 0)$, podemos asumir sin pérdida de generalidad que $v_1 = 1$ o $v_2 = 1$. Supongamos $v = (1, y)$ para algún valor $y \in \mathbb{R}$. Podemos definir entonces las funciones lineales p_j asociadas a la configuración Gale Dual como:

$$p_j(y) = \langle P_j, (1, y) \rangle$$

y el segmento abierto $I_P \subset \mathbb{R}$ como:

$$I_P = \{y \in \mathbb{R} \ / \ p_j(y) > 0 \ \forall j = 0, \dots, n+1\}$$

En este contexto podemos enunciar la siguiente proposición cuyo desarrollo y demostración se encuentra en [13].

Proposición 2.17. *Consideremos los polinomios p_j como antes y definamos la función $g : I_P \to \mathbb{R}$ como*

$$g(y) = \prod_{j \in [n+2]} p_j(y)^{b_j}.$$

Entonces la cantidad de raíces contadas con multiplicidad de la ecuación $g(y) = 1$ en I_P coincide con $n_{\mathcal{A}}(C)$.

2.3. El Teorema principal

Antes de enunciar el Teorema 2.18 haremos un resumen de las hipótesis y construcciones que hicimos en las secciones anteriores indicando donde fueron introducidas.

(*i.*) Tomamos $\mathcal{A} = \{a_0, \dots, a_{n+1}\} \subset \mathbb{Z}^n$ la configuración de potencias para el sistema de ecuaciones (2.0.1) y $C = (c_{ij}) \in \mathbb{R}^{n \times (n+2)}$ la matriz de coeficientes.

(*ii.*) Asumimos que $\mathcal{A}$ es un circuito.

(*iii.*) Definimos la matriz $A \in \mathbb{R}^{(n+1) \times (n+2)}$ asociada a $\mathcal{A}$ como (2.1.1)

$$A = \begin{pmatrix} 1 & \dots & 1 \\ a_0 & \dots & a_{n+1} \end{pmatrix}.$$

(*iv.*) Asumimos que las matrices A y C tienen rango máximo, es decir (2.1.8)

$$rk(A) = n+1,\ rk(C) = n.$$

(*v.*) Si tomo $b \in \ker(A)$, $b \in \mathbb{R}^{n+2}$ entonces b genera al núcleo de A y todas sus coordenadas son no nulas. (2.3)

(*vi.*) El 0 pertenece al cono positivo generado por las columnas de C. (2.1.5)

(*vii.*) Tomo $P \in \mathbb{R}^{(n+2)\times 2}$ la matriz cuyas columnas son una base del núcleo de C y notemos por $P_0, \ldots, P_{n+1}$ a los vectores fila de P, $P_j \in \mathbb{R}^2$ para todo $j \in [n+2]$.

(*viii.*) Existe un vector en el núcleo de C que tiene todas sus coordenadas positivas y por lo tanto existe $v \in \mathbb{R}^2$ tal que $\langle P_j, v\rangle > 0$ para todo $j \in [n+2]$. Este vector v define un semiplano en $\mathbb{R}^2$ que contiene a todos los vectores P_j. (2.1.7)

(*ix.*) En $[n+2]$ definimos la relación de equivalencia dada por $i \sim j \Leftrightarrow \det(P_i, P_j) = 0$ y que divide a $[n+2]$ en k clases de equivalencia $K_0, \ldots, K_{k-1}$. Eligiendo un orden arbitrario dentro de cada clase de equivalencia podemos definir $\sigma \in \mathbb{S}_{n+2}$ tal que $\sigma_i \leq \sigma_j \Leftrightarrow \varepsilon \cdot \det(P_{\sigma_i}, P_{\sigma_j}) \geq 0$ para algún $\varepsilon \in \{-1, 1\}$ fijo que depende de la orientación. Tomamos además $\overline{\sigma} \in \mathbb{S}_k$ la biyección inducida por σ en las clases de equivalencia. Llamamos a σ un orden para C y a $\overline{\sigma}$ un orden estricto para C. (2.7)

(*x.*) Definimos las sucesiones $\lambda = \{\lambda_l\}_{l\in[k]}$ y $\mu = \{\mu_l\}_{l\in[k]}$ como: (2.10)

$$\lambda_l = \sum_{j \in K_{\overline{\sigma}_l}} b_j \quad \text{y} \quad \mu_l = \sum_{i\leq l} \sum_{j\in K_i} b_j = \sum_{i\leq l} \lambda_i.$$

Utilizando estas definiciones y notaciones enunciamos y demostramos el Teorema principal de este capítulo.

Teorema 2.18 (La regla de los signos de Descartes para circuitos)**.** *Sean $C = (c_{ij}) \in \mathbb{R}^{n\times(n+2)}$ y $\mathcal{A} = \{a_0, \ldots, a_n\} \subset \mathbb{Z}^n$ un circuito tales que se cumplen (2.1.8) y (2.1.5). Sea $x = (x_1, \ldots, x_n)$, tomemos $x^{a_j} := \prod_{k=1}^n x_k^{a_{j,k}}$ y consideremos el sistema polinomial de n ecuaciones en n variables:*

$$f_i(x) = \sum_{j=0}^{n+1} c_{ij} x^{a_j} = 0,\ \textit{para } i = 1, \ldots n.$$

Sea σ un orden para C y sea μ la sucesión de sumas parciales asociada al par de órdenes $(\sigma, \overline{\sigma})$ para C. Si $n_{\mathcal{A}}(C)$ es la cantidad de soluciones positivas del sistema y es finito entonces:

$$n_{\mathcal{A}}(C) \leq 1 + varsigns(\mu).$$

Demostración. Sean $\mathcal{A}$ la configuración de potencias y C la matriz de coeficientes para el sistema (2.0.1). Elijamos una configuración Gale Dual para C, $\{P_0, \ldots P_n\}$ que satisfaga (2.1.7), definamos un orden σ y un orden estricto $\overline{\sigma}$ para esta configuración acordes a la Definición 2.7. Consideremos las formas lineales p_j y la función racional $g : I_P \to \mathbb{R}$ definida en la Proposición 2.17. Para demostrar el teorema es suficiente acotar la cantidad

de soluciones de la ecuación $g(y) = 1$ en I_P. Como asumimos que $n_{\mathcal{A}}(C)$ es finito tenemos en particular que $g(y) \not\equiv 1$.

Tomemos la partición de $[n+2] = \{K_0, \dots K_k\}$ dada por la relación de equivalencia definida en 2.7: un índice $j \in K_l$ para $l = 0, \dots, k$ si y solo si existe $e_{jl} \in \mathbb{R} \setminus \{0\}$ tal que $P_j = e_{jl} P_{\overline{\sigma}_l}$. Existe entonces una constante $e \in \mathbb{R} \setminus \{0\}$ tal que:

$$\begin{aligned} g(y) =& e \prod_{l \in [k]} (p_{\overline{\sigma}_l}(y))^{\lambda_l} \\ g(y) =& e \prod_{l \in [k]} (p_{\overline{\sigma}_l}(y))^{\mu_l} \frac{1}{(p_{\overline{\sigma}_l}(y))^{\mu_{l-1}}} \\ g(y) =& e \prod_{l \in [k-1]} \left(\frac{p_{\overline{\sigma}_l}(y)}{p_{\overline{\sigma}_{l+1}}(y)} \right)^{\mu_l}. \end{aligned}$$

Definamos ahora $G : I_P \to \mathbb{R}$ como $G(y) = \log(g(y))$. Como cada $p_j(y) > 0$ sobre I_P, G esta bien definida y cumple que $g(y) = 1$ si y solo si $G(y) = 0$. Podemos escribir la función G como:

$$\begin{aligned} G(y) =& \log(g(y)) = \log(e) + \sum_{l \in [k-1]} \mu_l \log \left(\frac{p_{\overline{\sigma}_l}(y)}{p_{\overline{\sigma}_{l+1}}(y)} \right) \\ =& \log(e) + \sum_{l \in [k-1]} \mu_l (\log (p_{\overline{\sigma}_l}(y)) - \log(p_{\overline{\sigma}_{l+1}}(y))) . \end{aligned}$$

Luego, derivando obtenemos que:

$$G'(y) = \sum_{l \in [k-1]} \mu_l \left(\frac{p'_{\overline{\sigma}_l}(y)}{p_{\overline{\sigma}_l}(y)} - \frac{p'_{\overline{\sigma}_{l+1}}(y)}{p_{\overline{\sigma}_{l+1}}(y)} \right).$$

Definamos $q_l(y) := \frac{p'_{\overline{\sigma}_l}(y)}{p_{\overline{\sigma}_l}(y)}$ para cada $l \in [k-1]$ y veamos que la colección de funciones $\{q_l(y) - q_{l+1}(y) / l \in [k-1]\}$ cumple la Regla de los Signos de Descartes en I_P. Para esto veamos que estas funciones cumplen las dos propiedades de la Proposición 2.16. Notemos primero que:

$$\begin{aligned} (q_l(y))' = \left(\frac{p'_l(y)}{p_l(y)} \right)' &= \frac{p''_l(y) p_l(y) - p'_l(y) p'_l(y)}{(p_l(y))^2} \\ &= - \left(\frac{p'_l(y)}{p_l(y)} \right)^2 = -(q_l(y))^2, \end{aligned}$$

porque $p'_l(y) = (\langle P_l, (1, y) \rangle)' = (P_{l,1} + y P_{l,2})' = P_{l,2}$ y $p''_l(y) = 0$. Entonces

$$q_l^{(i)}(y) = \left(\frac{p'_l(y)}{p_l(y)} \right)^{(i)} = (-1)^i i! \left(\frac{p'_l(y)}{p_l(y)} \right)^{i+1} = (-1)^i \cdot i! \cdot q_l^{i+1}(y).$$

Para ver que las funciones $\{q_l(y) - q_{l+1}(y)\}_{l \in [k-1]}$ cumplen (i) y (ii) en la Proposición 2.16 debemos calcular el Wronskiano de cualquier elección de l índices en $[k-1]$. Podemos

asumir sin pérdida de generalidad que estos índices son los primeros l y definamos V la matriz de Vandermonde con entradas en las potencias de $q_i(y)$. Entonces:

$$\prod_{i=1}^{l-1}((-1)^{i-1}\cdot i!)\det(V) = \prod_{i=1}^{l-1}((-1)^{i-1}\cdot i!)\det\begin{pmatrix} 1 & 1 & \dots & 1 \\ q_1(y) & q_2(y) & \dots & q_l(y) \\ \vdots & \vdots & \ddots & \vdots \\ (q_1(y))^{l-1} & (q_2(y))^{l-1} & \dots & (q_l(y))^{l-1} \end{pmatrix}$$

$$= \det\begin{pmatrix} 1 & \dots & 1 \\ q_1(y) & \dots & q_l(y) \\ \vdots & \ddots & \vdots \\ (-1)^{l-2}(l-1)!(q_1(y))^{l-1} & \dots & (-1)^{l-2}(l-1))!(q_l(y))^{l-1} \end{pmatrix}$$

$$= \det\begin{pmatrix} 1 & 1 & \dots & 1 \\ q_1(y) & q_2(y) & \dots & q_l(y) \\ \vdots & \vdots & \ddots & \vdots \\ (q_1(y))^{(l-1)} & (q_2(y))^{(l-1)} & \dots & (q_l(y))^{(l-1)} \end{pmatrix}.$$

Restando la columna $i-1$ a la columna i y colocando el resultado en la $i-$ésima columna en orden decreciente para $i = l, \dots, 2$ conseguimos:

$$= \det\begin{pmatrix} 1 & 0 & \dots & 0 \\ q_1(y) & q_2(y) - q_1(y) & \dots & q_l(y) - q_{l-1}(y) \\ \vdots & \vdots & \ddots & \vdots \\ (q_1(y))^{(l-1)} & (q_2(y))^{(l-1)} - (q_1(y))^{(l-1)} & \dots & (q_l(y))^{(l-1)} - (q_{l-1}(y))^{(l-1)} \end{pmatrix}.$$

Desarrollando el determinante por la primer fila:

$$= \det\begin{pmatrix} q_2(y) - q_1(y) & \dots & q_l(y) - q_{l-1}(y) \\ \vdots & \ddots & \vdots \\ (q_2(y))^{(l-1)} - (q_1(y))^{(l-1)} & \dots & (q_l(y))^{(l-1)} - (q_{l-1}(y))^{(l-1)} \end{pmatrix}$$
$$= W(q_2(y) - q_1(y), \dots, q_l(y) - q_{l-1}(y))$$
$$= (-1)^{l-1} W(q_1(y) - q_2(y), \dots, q_{l-1}(y) - q_l(y)).$$

Tomando $\gamma_l = \prod_{i=1}^{l-1}((-1)^{i-1}i!)$ tenemos que el desarrollo anterior se resume en:

$$\begin{aligned} W(q_1(y) - q_2(y), \dots, q_{l-1}(y) - q_l(y)) &= (-1)^{l-1}\gamma_l \det(V) \\ &= (-1)^{l-1}\gamma_l \prod_{1\le i<j\le l} (q_i(y) - q_j(y)). \end{aligned}$$

Ahora

$$\begin{aligned} q_i(y) - q_j(y) &= \frac{p'_{\overline{\sigma}_i}(y)}{p_{\overline{\sigma}_i}(y)} - \frac{p'_{\overline{\sigma}_j}(y)}{p_{\overline{\sigma}_j}(y)} \\ &= \frac{p_{\overline{\sigma}_j}(y)\cdot p'_{\overline{\sigma}_i}(y) - p_{\overline{\sigma}_i}(y)\cdot p'_{\overline{\sigma}_j}(y)}{p_{\overline{\sigma}_i}(y)\cdot p_{\overline{\sigma}_j}(y)} \end{aligned}$$

$$
\begin{aligned}
&= \frac{(P_{\overline{\sigma}_j,1} + yP_{\overline{\sigma}_j,2})P_{\overline{\sigma}_i,2} - (P_{\overline{\sigma}_i,1} + yP_{\overline{\sigma}_i,2})P_{\overline{\sigma}_j,2}}{p_{\overline{\sigma}_i}(y) \cdot p_{\overline{\sigma}_j}(y)} \\
&= \frac{P_{\overline{\sigma}_j,1}P_{\overline{\sigma}_i,2} + yP_{\overline{\sigma}_j,2}P_{\overline{\sigma}_i,2} - P_{\overline{\sigma}_i,1}P_{\overline{\sigma}_j,2} - yP_{\overline{\sigma}_i,2}P_{\overline{\sigma}_j,2}}{p_{\overline{\sigma}_i}(y) \cdot p_{\overline{\sigma}_j}(y)} \\
&= \frac{P_{\overline{\sigma}_j,1}P_{\overline{\sigma}_i,2} - P_{\overline{\sigma}_i,1}P_{\overline{\sigma}_j,2}}{p_{\overline{\sigma}_i}(y) \cdot p_{\overline{\sigma}_j}(y)} \\
&= \frac{\det(P_{\overline{\sigma}_j}, P_{\overline{\sigma}_i})}{p_{\overline{\sigma}_i}(y) \cdot p_{\overline{\sigma}_j}(y)}.
\end{aligned}
$$

Concluimos que:

$$
W(q_1(y) - q_2(y), \ldots, q_{l-1}(y) - q_l(y)) = (-1)^{l-1}\gamma_{l-1} \prod_{1 \le i < j \le l} \frac{\det(P_{\overline{\sigma}_j}, P_{\overline{\sigma}_i})}{p_{\overline{\sigma}_i}(y) \cdot p_{\overline{\sigma}_j}(y)}.
$$

Por la definición del orden σ y $\overline{\sigma}$ tenemos que existe $s \in \{-1, 1\}$ tal que para cualquier $i < j \in [k-1]$, $s \cdot \det(P_{\overline{\sigma}_j}, P_{\overline{\sigma}_i}) > 0$. Además $p_{\overline{\sigma}_i}(y)p_{\overline{\sigma}_j}(y) > 0$ para todo $i < j \in [k-1]$. Entonces $W(q_1(y) - q_2(y), \ldots, q_{l-1}(y) - q_l(y))$ no se anula y su signo depende únicamente de l. Luego la familia de funciones $\{q_l - q_{l+1}\}_{l \in [k-1]}$ cumple las condiciones (i) y (ii) y por lo tanto satisfacen la Regla de los Signos de Descartes en I_P.

Recordando que:

$$
\begin{aligned}
G'(y) &= \sum_{l \in [k-1]} \mu_l \left(\frac{p'_{\overline{\sigma}_l}(y)}{p_{\overline{\sigma}_l}(y)} - \frac{p'_{\overline{\sigma}_{l+1}}(y)}{p_{\overline{\sigma}_{l+1}}(y)} \right) \\
&= \sum_{l \in [k-1]} \mu_l \left(q_l(y) - q_{l+1}(y) \right),
\end{aligned}
$$

podemos afirmar que la cantidad de raíces de $G'(y)$ en I_P contadas con multiplicidad es menor o igual que $varsigns(\mu)$. Como la cantidad de raíces de G en I_P coincide con $n_{\mathcal{A}}(C)$ podemos concluir finalmente que $n_{\mathcal{A}}(C) \le varsigns(\mu) + 1$. □

Observación 2.19. De la definición 2.10 se desprende que μ depende solo del orden estricto $\overline{\sigma}$ y no del orden σ asociado. Además, en la definición 2.7 definimos el orden estricto para C y dijimos que era único salvo orientación. Tomemos $\overline{\sigma} \in \mathbb{S}_k$ un orden estricto para C que divide $[n+2]$ en k clases de equivalencia y definamos $\overline{\sigma}^{op}$ el orden con orientación inversa, es decir $\overline{\sigma}^{op}(j) = \overline{\sigma}(k-1-j)$ para $j \in [k]$. Si $\{\lambda_l\}_{l \in [k]}$ es la sucesión definida en 2.10 para $\overline{\sigma}$, entonces la sucesión $\{\lambda_l^{op}\}_{l \in [k]}$ que define $\overline{\sigma}^{op}$ es exactamente la misma pero con orden inverso porque las clases de equivalencia que ambos órdenes definen son las mismas. Vale que $\lambda_j = \lambda^{op}_{k+1-j}$. Veamos que si tomamos μ el vector de sumas parciales definido por $\overline{\sigma}$ y μ^{op} definido por $\overline{\sigma}^{op}$, las sucesiones son las mismas con orden inverso.

Como $\mu_{k-1} = 0$ tenemos que:

$$
\begin{aligned}
0 = \lambda_0 + \cdots + \lambda_{k-3} + \lambda_{k-2} + \lambda_{k-1} &= \mu_{k-1} \\
-\mu_0^{op} = -\lambda_{k-1} = \lambda_0 + \cdots + \lambda_{k-3} + \lambda_{k-2} &= \mu_{k-2} \\
-\mu_1^{op} = -\lambda_{k-1} - \lambda_{k-2} = \lambda_0 + \cdots + \lambda_{k-3} &= \mu_{k-3}
\end{aligned}
$$

$$\begin{aligned}
&\vdots = \vdots \\
-\mu^{op}_{k-2} &= -\lambda_1 - \cdots - \lambda_{k-1} = \lambda_0 = \mu_0 \\
-\mu^{op}_{k-1} &= -\lambda_0 - \cdots - \lambda_{k-1} = 0,
\end{aligned}$$

de lo que se desprende que $\mu_i = -\mu^{op}_{k-i-2}$ para i entre 0 y $k-2$, por lo que

$$varsigns(\mu) = varsigns(\mu^{op}).$$

En el trabajo [10], los autores dan una cota para la cantidad de soluciones positivas del sistema (2.0.1) en el sentido de la Regla de los Signos de Descartes con un espíritu similar al Teorema 2.18:

Teorema 2.20. *Sea $\mathcal{A}$ en $\mathbb{Z}^n$ un circuito, sea $C \in \mathbb{R}^{n\times(n+2)}$ una matriz de coeficientes que cumple (2.1.5), σ un orden para C y b un vector en el núcleo de A. Tomemos la partición en clases de equivalencia dada por el orden σ: $[n+2]/\sim = \{K_0, \ldots, K_{k-1}\}$ y definamos $\lambda \in \mathbb{R}^k$ tal que $\lambda_j = \sum_{i\in K_j} b_i$ para cada $j \in [k]$. Entonces, si $n_{\mathcal{A}}(C)$ es finito tenemos que:*

$$n_{\mathcal{A}}(C) \leq varsigns(\lambda).$$

Además, si λ_0 y λ_{k-1} son no nulos vale que:

$$n_{\mathcal{A}}(C) \equiv varsigns(\lambda) \ (mod\ 2).$$

Sin embargo, gracias a la siguiente propiedad, podemos ver que el resultado que demostramos mejora la cota que brinda este teorema:

Proposición 2.21. *Sea $\lambda = (\lambda_0, \ldots, \lambda_{k-1})$ una sucesión finita de números reales no nula tal que $\sum_{l=0}^{k-1} \lambda_l = 0$ y $\mu = (\mu_0, \ldots, \mu_{k-1})$ tal que $\mu_l = \lambda_0 + \cdots + \lambda_l$ para $l \in [k]$. Entonces:*

$$1 + varsigns(\mu) \leq varsigns(\lambda) \tag{2.3.1}$$

y más aún:

$$varsigns(\lambda) \equiv 1 + varsigns(\mu) \ (mod\ 2) \tag{2.3.2}$$

Además si λ es una sucesión de enteros,

$$1 + varsigns(\mu) \leq \sum_{\substack{\lambda_l > 0 \\ l\in[k]}} \lambda_l. \tag{2.3.3}$$

Para demostrar esta proposición vamos a necesitar la siguiente definición.

Definición 2.22. *Sea $\lambda = (\lambda_0, \ldots, \lambda_{k-1})$ una sucesión finita de números reales tal que $varsigns(\lambda) = m$. Definimos los índices de cambio de signo de λ, $\{l_0, \ldots l_m\} \in [k]$ recursivamente como:*

$$\begin{aligned}
l_0 =& min\{l \in [k]/\lambda_l \neq 0\}, \\
l_j =& min\{j \in \{l_{i-1}, \ldots k-1\}/\lambda_j \cdot \lambda_{l_{i-1}} < 0\}.
\end{aligned}$$

Demostración. (de la Proposición 2.21) Sea $m = varsigns(\mu)$ y $l_0 < l_1 < \cdots < l_m$ los índices de cambio de signo de μ. Asumamos sin pérdida de generalidad que las primeras entradas no nulas de λ son positivas, por lo tanto $\mu_{l_0} > 0$ y tenemos que

$$sgn(\mu_{l_s}) = (-1)^s, \text{ para } s = 0, \ldots, m.$$

Además $\lambda_{l_s} = \mu_{l_s} - \mu_{l_s-1}$, entonces

$$sgn(\lambda_{l_s}) = sgn(\mu_{l_s} - \mu_{l_s-1}) = (-1)^s,$$

porque $\mu_{l_s-1} = 0$ o $sgn(\mu_{l_s-1}) = -sgn(\mu_{l_s})$. Definamos

$$r := varsigns(\lambda) - m = varsigns(\lambda) - varsigns(\mu) \geq 0.$$

Para demostrar (2.3.1) y (2.3.2) basta ver que r es un número impar.

Por la definición de λ sabemos que $\mu_{k-1} = 0$, supongamos que μ_{k-2} es el último coeficiente no nulo de μ, si no la demostración es análoga. Como $\mu_{k-1} = \mu_{k-2} + \lambda_{k-1} = 0$ podemos concluir que $\lambda_{k-1} \neq 0$ y que $\mu_{k-2} \cdot \lambda_{k-1} < 0$. Entonces μ_{k-2} y λ_{k-1} son las últimas entradas no nulas de las sucesiones μ y λ. En particular:

- $sgn(\mu_{k-2}) = sgn(\mu_{l_m}) = (-1)^m$.
- $sgn(\lambda_{k-1}) = (-1)^{varsigns(\lambda)} = (-1)^{m+r}$.

Por lo tanto

$$(-1) = sgn(\mu_{k-2} \cdot \lambda_{k-1}) = (-1)^{2m+r},$$

de lo que se deduce que r es impar.

Para probar (2.3.3) asumimos que λ es entero y como $\lambda_{l_0} > 0$, tenemos que $\lambda_{l_0} \geq 1$. Más aún para $s = 1, \ldots, m$, $|\mu_{l_s}| \geq 1$ y

$$2 \leq (-1)^s(\mu_{l_s} - \mu_{l_{s-1}})$$

dado que μ_{l_s} y $\mu_{l_{s-1}}$ son enteros no nulos con signos opuestos. En particular, para cada s par tenemos que:

$$2 \leq (\mu_{l_s} - \mu_{l_{s-1}}) = \sum_{j=l_{s-1}+1}^{l_s} \lambda_j \leq \sum_{j=l_{s-1}+1}^{l_s} max(\lambda_j, 0)$$

y para cada s impar tenemos que:

$$2 \leq -(\mu_{l_s} - \mu_{l_{s-1}}) = \sum_{j=l_{s-1}+1}^{l_s} -\lambda_j \leq \sum_{j=l_{s-1}+1}^{l_s} -min(\lambda_j, 0).$$

Se sigue por un lado que

$$V := \sum_{j=0}^{k-1} max(\lambda_j, 0) \geq \sum_{\substack{s \text{ par} \\ 1<s\leq m}} \sum_{j=l_{s-1}+1}^{l_s} max(\lambda_j, 0) + \lambda_0 \geq 2 \cdot \left\lfloor \frac{m}{2} \right\rfloor + 1,$$

lo que devuelve la desigualdad buscada para m par. Por otro lado, como $V = \sum_{\lambda_j>0} \lambda_j = -\sum_{\lambda_j<0} \lambda_j$ tenemos que:

$$V = \sum_{j=0}^{k-1} -min(\lambda_j, 0) \geq \sum_{\substack{s \text{ impar} \\ 1\leq s\leq m}} \sum_{j=l_{s-1}+1}^{l_s} -min(\lambda_j, 0) \geq 2 \cdot \left\lceil \frac{m}{2} \right\rceil,$$

con lo que concluimos que $V \geq m + 1$ como queríamos ver. □

Podemos enunciar entonces el siguiente corolario que se desprende directamente del Teorema 2.20 y la Proposición 2.21.

Proposición 2.23. *Sean $\mathcal{A}$ y C como en el Teorema 2.18. Tomemos $\lambda_0, \ldots, \lambda_{k-1}$ correspondientes a un orden estricto de C. Asumamos además que λ_0 y λ_{k-1} son no nulas. Entonces:*

$$1 + varsigns(\mu) \equiv n_{\mathcal{A}}(C) \ (mod\ 2).$$

En particular $n_{\mathcal{A}}(C) > 0$ si $varsigns(\mu)$ es par.

Por último, utilizando las notaciones del Teorema 2.18 y a partir de [25, Th. 1.1], podemos afirmar que dado un cicuito $\mathcal{A}$ existe una matriz C tal que $n_{\mathcal{A}}(C) = n + 1$ si y solo si $k = n+2$ (donde k refiere a la cantidad de clases de equivalencia que define C sobre $[n + 2]$) y existe $\sigma \in \mathbb{S}_{n+2}$ tal que $varsigns(\mu) = n$. Este resultado puede ser recuperado como un caso particular del Teorema 2.18 y [11, Th. 3.4]:

Teorema 2.24. *Sea $\mathcal{A}$ un circuito en $\mathbb{Z}^n$ y $\{K_0, \ldots, K_{k-1}\}$ una partición de $[n + 2]$. Consideremos $\overline{\sigma} \in \mathbb{S}_k$ y μ como en la Definición 2.10. Entonces existe una matriz $C \in \mathbb{R}^{n\times(n+2)}$ de rango máximo que satisface (2.1.5), tal que $\overline{\sigma}$ es un orden estricto para C y*

$$n_{\mathcal{A}}(C) = 1 + varsigns(\mu).$$

3. Implementación para el cálculo de la cota

En el Capítulo 2 definimos en (2.0.1) para un conjunto de potencias $\mathcal{A} = \{a_0, \ldots, a_{n+1}\} \subset \mathbb{Z}^n$ y una matriz de coeficientes $C \in \mathbb{R}^{n\times(n+2)}$, el sistema de ecuaciones en n variables $x = (x_1, \ldots, x_n)$ asociado como:

$$f_i(x) = \sum_{j=0}^{n+1} c_{ij} x^{a_j} = 0, \text{ donde } i = 1, \ldots n \text{ y } x^{a_j} = \prod_{k=1}^{n} x_k^{a_{j,k}}.$$

En el Teorema 2.18 dimos una cota para la cantidad de soluciones positivas del sistema. El objetivo de este capítulo es presentar y desarrollar un programa que calcule automáticamente esta cota para cualquier sistema de ecuaciones de la forma (2.0.1) que cumpla las condiciones (2.1.8) y (2.1.5).

El programa está ordenado en tres funciones implementadas para SINGULAR [24], dos de ellas se usan para decidir si las hipótesis del teorema se cumplen. La función principal las usa y calcula la cota en caso de que las hipótesis se cumplan.

Todos los programas de este capítulo se encuentran citados completos con instrucciones para cargarlos en el sistema y utilizarlos en el Apéndice A.

En este capítulo utilizaremos los entornos `verbatim` y `listing` para explicitar el código del programa y dentro de él usaremos `>` para las respuestas de SINGULAR a través de la consola. Además el código insertado directamente desde el programa estudiado tendrá los números de línea correspondientes que permite luego rastrear en el código final.

3.1. Librerías y funciones previas

SINGULAR es un sistema desarrollado especialmente para realizar operaciones con polinomios arbitrarios, por lo tanto los cálculos más relevantes en SINGULAR requieren de la definición de un anillo. Los anillos más relevantes en este contexto son anillos polinomiales sobre un cuerpo, localizaciones o anillos cocientes de los anteriores módulo un ideal. La definición de un anillo consiste de tres partes: la primera determina el cuerpo base, la segunda determina los nombres de las variables y la tercera el orden monomial usado. Por ejemplo,

```
ring r = 0,(x,y,z),lp;
```

define un anillo `r` sobre un cuerpo de característica `0` (i.e. los números racionales), con variables `x, y, z` y `lp` indica que el orden monomial es el lexicográfico. Una lista completa de los ordenes disponibles con sus correspondientes definiciones puede encontrarse en [30, pág 503].

El sistema SINGULAR posee funciones, comandos y variables especiales propias del kernel de SINGULAR (comandos incorporados) y una librería `standard.lib` que carga automáticamente al iniciar que extienden la funcionalidad del kernel. Explicaremos a continuación las funciones que utilizaremos para el desarrollo del programa. Algunas de ellas

poseen múltiples definiciones y se comportan acorde al contexto en el que son utilizadas, para estas describiremos solo la funcionalidad que corresponde a la utilización que le daremos.

Entre las funciones propias del kernel y las cargadas por `standard.lib` utilizaremos

- `size(v)` con `v` un vector, una lista o una matriz devuelve la cantidad de entradas del elemento.

```
intvec v=1,2;
size(v);
> 2
matrix mm[2][2];
size(mm);
> 4
list l={ (1,2), (3,4), (0,3)};
size(l);
> 3
```

- `max (i_1, ..., i_k)` y `min (i_1, ..., i_k)` devuelven respectivamente máximo y el mínimo de los elementos `i_1, ..., i_k` siempre que `>` esté definido sobre ellos.

```
 // biggest int
max(2,3);
==> 3
max(1,4,3);
==> 4
// lexicographically biggest intvec
max(intvec(1,2),intvec(0,1),intvec(1,1));
==> 1,2
// polynomial with biggest leading monomial
ring r = 0,x,dp;
max(x+1,x2+x);
==> x2+x
```

- `nrows(A)` y `ncols(A)` devuelven respectivamente la cantidad de filas y columnas que tiene una matriz `A` definida (`matrix` o `intmatrix`).

```
ring r;
matrix m[5][6];
ncols(m);
> 6
nrows(m);
> 5
```

- `print(A)` muestra en pantalla la expresión `A` cualquiera sea ella. En particular muestra en pantalla las matrices por filas y columnas.

```
ring R=0,x,dp;
matrix m[2][3]=1,2,3,4,5,6;
print(m);
```

```
> 1,2,3,
> 4,5,6
```

- `transpose(A)` con `A` una matriz (`matrix` o `intmatrix`), devuelve la transpuesta de `A`.

```
ring R=0,x,dp;
matrix m[2][3]=1,2,3,4,5,6;
print(m);
> 1,2,3,
> 4,5,6
print(transpose(m));
> 1,4,
> 2,5,
> 3,6
```

- `syz(M)` con `M` un módulo o un ideal, calcula las relaciones en el módulo (resp ideal) de los generadores.

```
ring R=0,(x,y),(c,dp);
ideal i=x,y;
syz(i);
> _[1]=[y,-x]
```

En particular, SINGULAR puede leer una matriz M (`matrix` pero no `intmatrix`) como el módulo generado por su imagen y por lo tanto `syz(M)` calcula el núcleo de la matriz.

```
ring r=0,x,lp;
matrix C[2][4] = 1,0,1,-3,0,1,1,-4;
syz(C);
> [1]=gen(3)-gen(2)-gen(1)
> [2]=gen(4)+4*gen(3)-gen(1)
```

Además de la librería standard de SINGULAR necesitaremos cargar dos librerías extras: `matrix.lib` y `rootsur.lib`.

La librería `matrix.lib` es propia de SINGULAR y permite definir y realizar operaciones entre matrices. De esta librería utilizaremos las funciones:

- `concat(A1,A2,..)` con `A1,A2,..` matrices permite concatenar. Por ejemplo:

```
LIB "matrix.lib";
ring r=0,(x,y,z),ds;
matrix A[3][3]=1,2,3,x,y,z,x2,y2,z2;
matrix B[2][2]=1,0,2,0;
print(A);
> 1, 2, 3,
> x, y, z,
> x2,y2,z2
print(B);
```

```
> 1,0,
> 2,0
print(concat(A,B));
> 1, 2, 3, 1,0
> x, y, z, 2,0
> x2,y2,z2,0,0
```

- `diag(A)` con `A` matriz de dimensión $n \times m$, calcula una matriz de $nm \times nm$ con las entradas de `A` en la diagonal y el resto 0. Por ejemplo:

```
LIB "matrix.lib";
ring r = 0,(x,y,z),ds;
matrix A[3][2] = 1,2,3,4,5,6;
print(A);
> 1,2,
> 3,4,
> 5,6
print(diag(A));
> 1,0,0,0,0,0,
> 0,2,0,0,0,0,
> 0,0,3,0,0,0,
> 0,0,0,4,0,0,
> 0,0,0,0,5,0,
> 0,0,0,0,0,6
```

La librería `rootsur.lib` fue desarrollada por Enrique Tobis [50] para SINGULAR para contar raíces reales. De esta librería vamos a utilizar la función:

- `varsigns(l)` con `l` una lista, devuelve la cantidad de cambios de signos de `l`.

```
LIB "rootsur.lib";
ring r=0,x,dp;
list l=1,2,3;
varsigns(l);
> 0
l=1,-1,2,-2,3,-3;
varsigns(l);
> 5
```

3.2. Ingreso de datos y el estructura general de la función

Como dijimos anteriormente, el objetivo de este capítulo es presentar el programa desarrollado para SINGULAR : `OptimalDescartesRuleOfSigns.sing`. Este programa carga tres funciones en el sistema:

- `boundpoPolSyst(list configA, matrix C)`, la función principal de este trabajo.
- `CheckHypothesis(matrix A, matrix C, vector kerA)`, función auxiliar. Decide si el sistema dado cumple o no las hipótesis del teorema:(2.1.8) y (2.1.5).

- `posLinearComb(vector v, vector w)`, función auxiliar a `CheckHypothesis`. Decide si existe z en el subespacio generado por v y w que tenga todas sus coordenadas positivas.

Para utilizar la función `boundpoPolSyst(list configA, matrix C)` es necesario describir el sistema (2.0.1) en función de $\mathcal{A}$ y C. Es decir, a partir de su configuración de potencias y matriz de coeficientes.

Ejemplo 3.1. El sistema:

$$\begin{cases} x + x^2y^3 - 3xy^4 = 0 \\ y^2 + x^2y^3 - 4xy^4 = 0 \end{cases} \tag{3.2.1}$$

es descripto por la matriz de coeficientes $C = \begin{pmatrix} 1 & 0 & 1 & -3 \\ 0 & 1 & 1 & -4 \end{pmatrix}$ y la configuración de potencias $\mathcal{A} = \{(1,0),(0,2),(2,3),(1,4)\}$. Para cargar este sistema en SINGULAR definiremos un anillo y las dos variables `configA` y `C` necesarias:

```
ring r=0,(x,y),lp;
list configA;
configA=configA+list([1,0])+list([0,2])+list([2,3])+list
   ([1,4]);
matrix C[2][4] = 1,0,1,-3,0,1,1,-4;
print(configA);
> [1]:
>    gen(1)
> [2]:
>    2*gen(2)
> [3]:
>    3*gen(2)+2*gen(1)
> [4]:
>    4*gen(2)+gen(1)
print(C);
> 1,0,1,-3,
> 0,1,1,-4
```

En SINGULAR los vectores de enteros (`intvec`) se cargan sin delimitadores (`intvec v=1,2`) y los vectores (`vector`) dados por sus coordenadas en los generadores del anillo se definen con corchetes (`vector w=[1,2]`).

Con esto ya estamos en condiciones de llamar a la función `boundpoPolSyst` ingresando en consola:

```
boundpoPolSyst(configA, C);
> // ** loaded /usr/bin/../share/singular/LIB/matrix.lib
   (4.1.2.0,Feb_2019)
> // ** loaded /usr/bin/../share/singular/LIB/nctools.lib
   (4.1.2.0,Feb_2019)
> ...
> // ** redefining i (int i=1;) ::boundposPolSyst:134
> // ** redefining i (  int i=1;) ::boundposPolSyst:186
> // ** redefining j (int j=1;) ::boundposPolSyst:187
> 3
```

El programa devuelve que la cantidad de soluciones positivas al sistema (3.2.1) es a lo sumo 3.

La función comienza cargando las librerías y definiendo las variables necesarias para realizar los cálculos. Define una variable `TheBound` para guardar la cota final del sistema y la inicializa en -1. Tanto para el cálculo de la cota como para decidir si las hipótesis del teorema se cumplen utilizaremos la matriz A asociada a $\mathcal{A}$ que definimos en (2.1.1) y un vector generador del núcleo de A. Lo calcularemos entonces antes de realizar ambas operaciones para evitar duplicar código.

Para definir la matriz A tomamos cada vector de $\mathcal{A}$ y le concatenamos un 1 al inicio, luego concatenamos cada uno de estos vectores. Como la función `concat` concatena columnas y no filas, es necesario transponer los vectores de $\mathcal{A}$ para poder agregar al 1 como primer coordenada y luego volver a transponer. La matriz `A` resultante de esta operación es la matriz buscada.

Luego calcula el vector generador del núcleo de A con la función `syz`, esta función devuelve una lista de vectores generadores del núcleo. Para que el teorema sea aplicable, A debe tener rango máximo y por lo tanto su núcleo dimensión 1. Entonces `syz(A)` es una lista con un solo elemento, definimos `kerA` como el primer elemento de esta lista.

A continuación el programa decide si se cumplen las hipótesis del Teorema 2.18. Si no se cumplen el programa escribirá en pantalla la leyenda `Las hipótesis no se cumplen` y devolverá el valor -1 como cota a la cantidad de soluciones positivas del sistema. Si las hipótesis se cumplen avanzará en el cálculo de la cota. La estructura general del programa es la siguiente:

```
98        }
99      }
100     //En este punto check_hypothesis=1 si y solo si todas las
          hipotesis se cumplen, si no es 1.
101     return(check_hypothesis);
102   }
103
104   //Esta es la funcion principal
105   proc boundposPolSyst(list configA, matrix C){
106   //Cargo librerias
107   LIB "matrix.lib";
108   LIB "rootsur.lib";
109
110   int TheBound=-1;
111   //Defino A la matriz de nx(n+2) con la primer fila de unos
        y tal que la submatriz restante tiene como columnas los
        vectores de configA en orden
112   for(int i=1; i<=size(configA);i++){
113       configA[i]=transpose(concat(matrix(1),transpose(matrix(
      configA [i]))));
```

[desarrollado en la Sección 3.5]

```
210           mu[j]=mu[j]+int(kerA[sigma[i]]);
211           i++;
212         }
```

```
213       }
214     }
```

3.3. La función auxiliar `posLinearComb`

En esta sección desarrollaremos una función que nos permita decidir cuando, dados dos vectores $v,\ w \in \mathbb{R}^n$ existe un vector $z = \alpha v + \beta w$ que tenga todas sus coordenadas positivas, es decir $z \in (\mathbb{R}_{>0})^n$. Si z cumple esto entonces:

$$\begin{aligned} z =& \alpha v + \beta w \\ z =& \beta \left(\frac{\alpha}{\beta} v + w \right) \\ \tilde{z} =& sgn(\beta) \left(\tilde{\alpha} v + w \right). \end{aligned}$$

Luego, el problema se reduce a decidir si existe una combinación lineal de la forma $\overline{\alpha} v + w$ con todas sus coordenadas del mismo signo:

$$(\tilde{\alpha} v + w) \in (\mathbb{R}_{>0})^{n+2} \text{ o } (-\tilde{\alpha} v - w) \in (\mathbb{R}_{>0})^{n+2}.$$

Estudiaremos el caso $(\tilde{\alpha} v + w) \in (\mathbb{R}_{>0})^{n+2}$ y consideraremos que $(-\tilde{\alpha} v - w) \in (\mathbb{R}_{>0})^{n+2}$ se deduce de aplicar el mismo algoritmo a $-v$ y $-w$.

Algoritmo 3.2. Sean v y w dos vectores en $\mathbb{R}^{n+2}$. Queremos decidir si existe $\lambda \in \mathbb{R}$ tal que $z = \lambda v + w \in (\mathbb{R}^{n+2})_{>0}$. Entonces queremos ver si:

$$\lambda v_i + w_i > 0 \text{ para todo } i = 1, \dots n+2.$$

Y esto es equivalente a $\lambda > -\frac{w_i}{v_i}$ para todo $i = 1, \dots n+2$ tal que $v_i > 0$, $\lambda < -\frac{w_i}{v_i}$ para todo $i = 1, \dots n+2$ tal que $v_i < 0$ y $w_i > 0$ para todo $i = 1, \dots n+2$ tal que $v_i = 0$. Por lo tanto

- $\lambda > \text{máx} \left\{ -\frac{w_i}{v_i} :\ v_i > 0,\ i = 1, \dots n+2 \right\}$
- $\lambda < \text{mín} \left\{ -\frac{w_i}{v_i} :\ v_i < 0,\ i = 1, \dots n+2 \right\}$
- Si $v_i = 0$, $w_i > 0$.

Es decir que tal λ existe si y solo si $w_i > 0$ cada vez que $v_i = 0$ y

$$\text{máx} \left\{ -\frac{w_i}{v_i} :\ v_i > 0,\ i = 1, \dots n+2 \right\} \leq \text{mín} \left\{ -\frac{w_i}{v_i} :\ v_i < 0,\ i = 1, \dots n+2 \right\}.$$

El siguiente programa utiliza este algoritmo para decidir cuando, para dos vectores v y w dados, existe λ tal que $\lambda v + w \in (\mathbb{R}_{>0})^{n+2}$.

```
2  proc posLinearComb(vector v, vector w){
3    list Geq; //La lista -w[i]/v[i] cuando v[i]>0
4    list Leq; //La lista -w[i]/v[i] cuando v[i]<0
5    int n=size(v);
6    int verifies=1;
7
```

```
8     //Si z=kv+w entonces kv[i]+w[i]>0 si y solo si
9     //[(k>-w[i]/v[i] y v[i]>0) o (k<-w[i]/v[i] y v[i]<0)]
10    for (int i=1; i<=n; i++){
11      if(v[i]>0){
12        Geq=Geq + list(number(-w[i])/number(v[i]));
13      } else{
14        if(v[i]<0){
15        Leq=Leq+list(number(-w[i])/number(v[i]));
16        } else{
17          if(v[i]==0 & w[i]<=0){
18            verifies=0;
19          }
20        }
21      }
22    }
23
24    //Decidimos si existe k<Geq[i] y k>Leq[i] para todo i
25    //Si cualquiera de las dos listas esta vacia y la
       condicion (v[i]==0 & w[i]<=0) no se satisface, siempre
       existe tal k
26    if(size(Leq) & size(Geq)){
27      def UpperBound=min(Leq[1.. size(Leq)]);
28      def LowerBound=max(Geq[1.. size(Geq)]);
29
30      if( LowerBound < UpperBound){
31        verifies=verifies & 1;
32      } else{
33        verifies=0;
34      }
35    }
36    return(verifies);
37  }
```

3.4. La función auxiliar `CheckHypothesis`

Queremos definir una función que dados $\mathcal{A}$ y C decida si las condiciones para aplicar el teorema se cumplen. Al comienzo de la Sección 2.3 describimos las hipótesis y construcciones necesarias, en particular las primeras se resumen en los ítems (i), (iv), (v) y (viii) al comienzo de la Sección 2.3. Notemos que en general las hipótesis y construcciones sobre $\mathcal{A}$ están en realidad enunciadas sobre A. La construcción de tal matriz A fue descripta anteriormente y la utilizaremos tanto para revisar las hipótesis como para luego calcular la cota. Por lo tanto la función a construir tomará como entrada a la matriz A y no a la configuración $\mathcal{A}$. Lo mismo sucede con `ker(A)` y `ker(C)`, al ser necesarias para ambas secciones se utilizarán como input de la función y se calcularán una sola vez antes de llamarlas.

Por otro lado tengamos en cuenta que en SINGULAR la sentencia `if(A & B){...}` evalúa la condición `A`, la condición `B` y por último evalúa `A & B`. En cambio el encadena-

miento de sentencias de la forma `if(A){if(B){...}}` evalúa la condición `A` y avanza con `B` solo si la primera se cumple. Utilizaremos la cadena de sentencias para escribir una función que evalúe las hipótesis solo si las anteriores se cumplieron y, en caso en que encuentre una hipótesis insatisfecha, salga de la función informando la primera hipótesis no cumplida.

Algoritmo 3.3. Primero la función debe tomar como variables a las matrices A, C y sus núcleos y revisar si se cumplen las hipótesis sobre la dimensión de las matrices enunciadas en (i):

```
41 proc CheckHypothesis(matrix A, matrix C, vector kerA,
       matrix kerC){
42   int check_hypothesis=1;
43   //Las dimensiones deben ser apropiadas para el sistema
44   if((nrows(A)+1)!=ncols(A)){
45     check_hypothesis=0;
46     "Las dimensiones de configA no son correctas";
47   }
48   if(check_hypothesis){
49     if((nrows(C)+2)!=ncols(C)){
50       check_hypothesis=0;
51       "Las dimensiones de C no son correctas";
52     }
53   }
54     //deben coincidir las dimensiones de A y C
55   if(check_hypothesis){
56     if(ncols(A)!=ncols(C)){
57       check_hypothesis=0;
58       "Las dimensiones de A y C no coinciden";
59     }
60   }
```

La variable `check_hypothesis` es un entero que utilizaremos como un booleano: comienza con valor 1 y cambia a 0 solo si encuentra una hipótesis que no se cumple.

El segundo paso es la condición (iv) sobre los rangos: queremos que tanto A como C tengan rango máximo. En ambos casos la máxima dimensión posible coincide con la cantidad de filas, planteamos que:

```
62   if(check_hypothesis){
63     if(nrows(A)!=rank(A)){
64       check_hypothesis=0;
65       "El rango de A no es maximo";
66     }
67   }
68   //El rango de C debe ser maximo
69   if(check_hypothesis){
70     if(nrows(C)!=rank(C)){
71       check_hypothesis=0;
72       "El rango de C no es maximo";
73     }
74   }
```

Como tercer paso evaluaremos la condición (v). El vector `kerA` está definido dentro de la función `boundposPolSyst` como `def kerA=syz(A)[1]` (la primer coordenada de `syz(A)`), lo que nos asegura que este vector pertenecerá al núcleo de A. Sin embargo, si el rango de A no es máximo, el núcleo de A tendrá dimensión mayor a 1 y `kerA` no lo describirá. Como en el paso anterior ya confirmamos que la dimensión de A es máxima, sabemos que el núcleo de A tiene dimensión 1 y que el vector `kerA` que pasamos a la función genera el núcleo de A. Entonces todo vector del núcleo de A es un múltiplo de `kerA` y por lo tanto ver que existe un vector con todas sus coordenadas no nulas es equivalente a ver que el vector `kerA` tiene todas sus coordenadas no nulas:

```
69    if(check_hypothesis){
70      if(nrows(C)!=rank(C)){
71        check_hypothesis=0;
72        "El rango de C no es maximo";
73      }
74    }
```

Por último resta ver la condición (viii). Para esto, como ya sabemos que C tiene rango máximo, definimos dos vectores `v` y `w` tales que $\langle$`v`, `w`$\rangle = ker(C)$. Queremos saber si existen α y β tales que α`v` $+ \beta$`w` tenga todas sus coordenadas positivas. Por lo desarrollado en la sección anterior, basta ver que existe λ tal que λ`v` $+$ `w` $\in (\mathbb{R}_{>0})^{n+2}$ o tal que λ(`-v`) $+$ (`-w`) $\in (\mathbb{R}_{>0})^{n+2}$. Y esto último lo decide la función auxiliar `posLinearComb(v,w)` que definimos en la sección anterior:

```
76    for(int i=1; i<=ncols(A); i++){
77      if(check_hypothesis & (kerA[i]==0)){
78        check_hypothesis=0;
79        "El sistema no esta soportado en un circuito";
80      }
81    }
82    //El origen debe pertenecer al cono positivo generado por
        las columnas de C
83    if(check_hypothesis){
84      def M=module(C);
85      //defino v y w dos generadores del nucleo de C
86      def v=kerC[1];
87      def w=kerC[2];
88      check_hypothesis=0;
89      if( posLinearComb(v,w)){
90        check_hypothesis=1;
91      } else {
92        if( posLinearComb(-v,-w)){
93        check_hypothesis=1;
94        }
95      }
96      if(check_hypothesis==0){
97        "El origen debe pertenecer al cono positivo generado
       por las columnas de C";
98      }
```

```
99      }
100     //En este punto check_hypothesis=1 si y solo si todas las
          hipotesis se cumplen, si no es 1.
101     return(check_hypothesis);
```

La función termina devolviendo el valor de `check_hypothesis`: 1 si cumplió todas las hipótesis, 0 si no.

3.5. La función principal `boundposPolSyst`

En la Sección 3.2 estudiamos la estructura general de la función `boundPolSyst`, que toma como inputs la configuración de potencias $\mathcal{A}$ como una lista de vectores `configA` y la matriz de coeficientes C. La función carga las librerías que describimos en la Sección 3.1 y define el entero `TheBound` que almacenará la cota que buscamos, la matriz A a partir de `configA` y los núcleos de A y C. Una vez definido esto llama a la función auxiliar `CheckHypothesis`. Si las hipótesis no se cumplen sale de la función `boundposPolSyst` con valor -1. En esta sección desarrollaremos el cálculo de la cota si las hipótesis se cumplen.

3.5.1. Los órdenes σ y $\overline{\sigma}$

Tenemos $C \in \mathbb{R}^{n\times(n+2)}$ una matriz de coeficientes con rango maximal y podemos definir una configuración Gale dual $P_0, \ldots, P_{n+1}$ a partir de las filas de la matriz `kerC = syz(C)` que tiene como columnas a los vectores del núcleo de C. Queremos construir un orden σ y un orden estricto $\overline{\sigma}$ para C, recordemos sus definiciones dadas en 2.7:

Definición 2.7. *Sea C una matriz con rango maximal y $\{P_0, \ldots, P_{n+1}\}$ una elección para su configuración Gale dual que satisfacen (2.1.7). Definimos la relación de equivalencia $\sim$ en $[n+2]$ como:*

$$i \sim j \Leftrightarrow det(P_i, P_j) = 0$$

Entonces podemos definir sobre las clases de equivalencia, de cardinal $k \leq n+1$:

$$[n+2]/\sim = \{K_0, \ldots, K_{k-1}\}$$

un orden canónico, eligiendo una orientación este orden es único. Eligiendo un orden arbitrario dentro de cada clase de equivalencia obtenemos una permutación $\sigma \in \mathbb{S}_{n+2}$ tal que:

$$\varepsilon \cdot \det(P_{\sigma_i}, P_{\sigma_j}) \geq 0, \ \ si\ \sigma_i < \sigma_j$$

donde $\varepsilon \in \{-1, 1\}$ depende de la orientación elegida.
Sea $K \subset [n+2]$ un conjunto de representantes de las clases de equivalencia $K_0, \ldots, K_{k-1}$. Notemos por $\overline{\sigma}$ a la biyección inducida por σ:

$$\overline{\sigma} : [k] \to K$$

Entonces $\varepsilon \cdot \det(P_{\overline{\sigma}_i}, P_{\overline{\sigma}_j}) > 0$ si $\overline{\sigma}_i < \overline{\sigma}_j$.
Decimos que σ es un orden y que $\overline{\sigma}$ es un orden estricto para C.

Tomemos la relación de equivalencia $\sim$ sobre $[n+2]$ de la definición:

$$i \sim j \Leftrightarrow \det(P_i, P_j) = 0$$

y en las clases de equivalencia $[n+2]/\sim = \{K_0, \ldots, K_{k-1}\}$ definamos en $[k]$ el orden canónico $\overline{\sigma}$ dado por el argumento de los vectores $P_0, \ldots P_{n+1}$ en el semiplano al que pertenecen. Entonces para cualquier $i, j \in [k]$ vale que $\overline{\sigma}_j < \overline{\sigma}_i$ o $\overline{\sigma}_i < \overline{\sigma}_j$.

Sea $\alpha : [n+2] \to [k]$ tal que $i \in K_{\alpha(i)}$. Luego $det(P_i, P_j) > 0$ si y solo si $\overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(j)}$. Definamos además los conjuntos:

$$\begin{aligned} A_i &= \{l \in [n+2] / \det(P_i, P_l) > 0\} \\ &= \{l \in [n+2] / \overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(l)}\} \end{aligned} \tag{3.5.1}$$

Proposición 3.4. *Los vectores P_i, P_j son colineales si y solo si $A_i = A_j$.*

Demostración. Si P_i, P_j son colineales entonces $P_i = \lambda P_j$. $k \in A_j$, $D[j,k] = 1$ si y solo si $\det(P_j, P_k) > 0$. Ahora

$$\det(P_i, P_k) = \det(\lambda P_j, P_k) = \lambda \det(P_j, P_k) > 0.$$

Entonces $k \in A_i$.

Si en cambio $A_i = A_j$, tomemos $d = \det(P_i, P_j)$. Si $d > 0$ entonces $j \in A_i$, pero como $\det(P_j, P_j) = 0$ tenemos que $j \notin A_j$, lo que contradice la hipótesis. En cambio si $d < 0$ $i \in A_j$ y tenemos la misma contradicción que antes. Luego debe ser $d = 0$ y por lo tanto P_i y P_j son colineales. □

Proposición 3.5. *Sean $i, j \in [n+2]$ tales que $\alpha(i) \neq \alpha(j)$. Entonces $A_j \subset A_i$ si y solo si $\overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(j)}$.*

Demostración. Como $\alpha(i) \neq \alpha(j)$, entonces i y j pertenecen a clases de equivalencia distintas. Por lo tanto $\overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(j)}$ o $\overline{\sigma}_{\alpha(i)} > \overline{\sigma}_{\alpha(j)}$.

Supongamos primero que $A_j \subset A_i$. Tenemos entonces que para cada l tal que $\det(P_{\alpha_j}, P_{\alpha_l}) > 0$ vale que $\det(P_{\alpha_i}, P_{\alpha_l}) > 0$. Si $\overline{\sigma}_{\alpha(j)} < \overline{\sigma}_{\alpha(i)}$, $\det(P_{\alpha_j}, P_{\alpha_i}) > 0$ y por lo tanto $\det(P_{\alpha_i}, P_{\alpha_i}) > 0$ lo que es claramente un absurdo. Luego $\overline{\sigma}_{\alpha(j)} > \overline{\sigma}_{\alpha(i)}$.

Si ahora $\overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(j)}$, entonces $j \in A_i$. Sea $l \in A_j$, $\overline{\sigma}_{\alpha(j)} < \overline{\sigma}_{\alpha(l)}$ y entonces

$$\overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(j)} < \overline{\sigma}_{\alpha(l)},$$

con lo que $l \in A_i$ como queríamos ver. □

Como corolario a estas dos proposiciones tenemos que:

Corolario 3.6. *Para cualquier $i, j \in [n+2]$,*

$$\overline{\sigma}_{\alpha(i)} < \overline{\sigma}_{\alpha(j)} \Leftrightarrow A_j \subset A_i \Leftrightarrow \#A_j < \#A_i.$$

La última desigualdad vale porque como $\overline{\sigma}$ es un orden en $[k]$ y para cualquier i, j se tiene que $A_i \subset A_j$, $A_j \subset A_i$ o $A_i = A_j$. Luego el orden dado por la inclusión inversa de los conjuntos $\{A_i\}_{i \in [n+2]}$ coincide con el orden σ de la definición.

Con todo esto estamos en condiciones de escribir un algoritmo para calcular el orden.

Algoritmo 3.7. Definimos una matriz auxiliar $D \in \mathbb{R}^{(n+2)\times(n+2)}$ tal que

$$D[i,j] = \begin{cases} 1 & \text{sii } \det(P_i, P_j) > 0 \\ 0 & \text{si no} \end{cases}.$$

Para esto usamos que si $\det(P_i, P_j) < 0$ entonces $\det(P_j, P_i) > 0$.

Recordemos que ya estaba declarada la matriz `kerC` que tiene como columnas una base del núcleo de C y, por lo tanto, como filas a los vectores $P_0, \dots P_{n+2}$.:

```
116 def kerA=syz(A)[1];
117 matrix kerC=syz(C);
118
119 //Si las hipotesis se cumplen vamos a hallar una cota
      superior para la cantidad de soluciones positivas del
      sistema.
120 if( CheckHypothesis(A, C, kerA, kerC) ){
121
122   //Primero necesitamos calcular un orden y un orden
       estricto para C
123   int ColsC=ncols(C);
124   int RowsC=nrows(C);
125   matrix D=diag(0,ColsC);
126   int n=nrows(kerC);
127   int d;
128   intvec VectorGeqThan;
129
130   //Defino una matriz D tal que D[i,j]=1 si y solo si det(
       kerC[i],kerC[j])>0
131   for(int i=1; i< n;i++){
132     for(int j=i+1; j<=n; j++){
133       d=int(kerC[i,1]*kerC[j,2]-kerC[i,2]*kerC[j,1]);
134       if(d>0){
135         D[i,j]=1;
```

Luego $A_i = \{j/\ D[i,j] = 1\}$ y

$$\#A_i = \sum_{j=1}^{n+2} D[i,j].$$

Tomamos `VectorGeqThan` $\in \mathbb{R}^{n+2}$ tal que `VectorGeqThan`$[i] = \#A_i$:

```
137         if(d<0){
138         D[j,i]=1;
139         }
140       }
141     }
142   }
```

Con esto construiremos los ordenes `sigma` y `sigma_barra` que corresponden al orden y al orden estricto sobre $P_0, \dots P_{n+2}$ respectivamente.

El siguiente programa ordena $[n+2]$ en orden creciente según `VectorGeqThan`: Primero busca el mínimo valor del vector que aún no haya sido ordenado, lo guarda en `LookForMin` y su índice en `k`. Define `LastMin` como `LookForMin` para marcar el máximo valor elegido para las próximas iteraciones. Agrega al final del orden `sigma` y del orden estricto `sigma_barra` el elemento `k`, es decir, el último índice elegido se ubica al final del orden. Por último agrega al orden `sigma` todos los índices que son equivalentes a `k`: todos los índices que tienen el mismo `VectorGeqThan`$[i] = \#A_i$ asociado.

```
144   for(int i=1; i<=ColsC; i++){
145     VectorGeqThan[i]=0;
146     for(int j=1; j<=ColsC; j++){
147       VectorGeqThan[i]=VectorGeqThan[i]+int(D[i,j]);
148     }
149   }
150   //Ahora definimos los ordenes
151   intvec sigma;
152   intvec sigma_barra;
153   int LastMin=-1;
154   int MaxVectorGeqThan;
155   int SizeVectorGeqThan;
156   MaxVectorGeqThan=max(VectorGeqThan[1..size(VectorGeqThan)
      ]);
157   SizeVectorGeqThan=size(VectorGeqThan);
158   int l=1; //es el contador de coordenadas de sigma
159   int m=1;//es el contador de coordenadas de sigma_barra
160   int k; //el indice del siguiente vector en el orden
161   int LookForMin;
162   while(LastMin != MaxVectorGeqThan){
163     LookForMin=MaxVectorGeqThan+1;
164     k=0;
165     //Encuentra el minimo vector no ordenado
166     for (int i=1; i<=SizeVectorGeqThan; i++){
167       if(VectorGeqThan[i]<LookForMin & VectorGeqThan[i]>
      LastMin){
168         LookForMin=VectorGeqThan[i];
169         k=i;
170       }
171     }
172     //Pongo el vector al final del orden y del orden
      estricto
173     sigma[l]=k;
174     sigma_barra[m]=k;
175     m++;
176     l++;
177     LastMin=LookForMin;
178     //Pongo todos los vectores que tengan el mismo
      VectorGeqThan[i] a continuacion en el orden
```

Notar que esto describe el orden en la orientación opuesta a la que describimos al prin-

cipio de la sección, pero por la Observación 2.19 la cota que obtendremos será exactamente la misma.

3.5.2. La sucesión μ

Tenemos definido un vector `kerA` generador del núcleo de A, `sigma` y `sigma_barra` un orden y un orden estricto para C. Recordamos la Definición 2.10:

Definición 2.10. *Manteniendo las notaciones de la Definición 2.7, para cada* $l = 0, \dots, k-1$ *tomamos:*

$$L_l = K_{\overline{\sigma}_0} \cup \cdots \cup K_{\overline{\sigma}_l}.$$

Tomemos b *un vector no nulo en el núcleo de* A *y consideremos las sucesiones* $\lambda = \{\lambda_l\}_{l\in[k]}$ *y* $\mu = \{\mu_l\}_{l\in[k]}$ *definidas por:*

$$\lambda_l = \sum_{j\in K_{\overline{\sigma}_l}} b_j \quad y \quad \mu_l = \sum_{j\in L_l} b_j.$$

Notemos que entonces:

$$\mu_0 = \sum_{j\in K_{\overline{\sigma}_0}} \texttt{kerA}[j] \tag{3.5.2}$$

$$\mu_l = \mu_{l-1} + \sum_{j\in K_{\overline{\sigma}_l}} \texttt{kerA}[j]. \tag{3.5.3}$$

Podemos enunciar en este contexto la siguiente Proposición

Proposición 3.8. *Sean* $\sigma :=$ `sigma` *y* $\overline{\sigma} :=$ `sigma_barra` *un orden y un orden estricto para* C *entonces existe* $J \subset [n+2]$ *un subconjunto ordenado tal que:*

- *Para cualquier* $i, j \in J$, *si* $i \neq j$, $i \not\sim j$.
- *Si* $i, j \in [n+2]$, $i \not\sim j$, *entonces existen* $\tilde{i}, \tilde{j} \in J$ *tales que* $i \sim \tilde{i}$ *y* $j \sim \tilde{j}$.
- $\sigma|_J = \overline{\sigma}$.
- *Si* $J = \{j_0, \dots, j_{k-1}\}$, $\sigma_j \in K_{\overline{\sigma}_{j_i}}$ *para todo* $j_i \leq j < j_{i+1}$.

Esta proposición se desprende de tomar J un conjunto de representantes y ordenarlo acorde al Algoritmo 3.7.

Luego (3.5.2) puede reescribirse como

$$\mu_0 = \sum_{j=j_0}^{j_1-1} \texttt{kerA}[j]$$

$$\mu_l = \mu_{l-1} + \sum_{j=j_l}^{j_{l+1}-1} \texttt{kerA}[j] \text{ si } l \neq k-1$$

$$\mu_{k-1} = \mu_{k-2} + \sum_{j=j_{k-1}}^{n+2} \texttt{kerA}[j].$$

Y por lo tanto el siguiente programa calcula la sucesión μ:

```
181           sigma[l]=i;
182           l++;
183         }
184       }
185     }
186
187     //Ahora defino mu una lista con las sumas parciales de
       kerA respecto al orden
188     list mu;
189     int i=1;
190     int total = size(sigma_barra);
191     for(int j=1; j<=total; j++){
192       //j es el indice del orden
193       //i es el indice del orden estricto
194       if(j==1){
195         mu[1]=int(kerA[sigma_barra[1]]);
196       } else{
197         mu[j]=mu[j-1]+int(kerA[sigma_barra[j]]);
198       }
199       i++;
200
201       //Si j no es el ultimo del orden estricto, sumo las
       coordenadas de kerA hasta el siguiente lugar del orden
       estricto
202       if(j!=total){
203         while (sigma_barra[j+1]!=sigma[i]){
204           mu[j]=mu[j]+int(kerA[sigma[i]]);
205           i++;
206         }
207       } else {
```

3.5.3. La cota buscada

Por último solo queda hallar la cota superior a la cantidad de soluciones positivas del sistema de ecuaciones que tiene como configuración de potencias $\mathcal{A} =$ `configA` y coeficientes $C =$ `C` que define el Teorema 2.18.

Recordando que ya teníamos definida la variable `TheBound` para albergar la cota buscada, las últimas líneas del programa serán:

```
209         while (i!=size(sigma)+1){
210           mu[j]=mu[j]+int(kerA[sigma[i]]);
211           i++;
212         }
213       }
214     }
```

En el ejemplo 3.1 mostramos como cargar los datos y usar la función `boundPolSyst`. Como dijimos al principio de este capítulo, el programa completo, sin interrupciones,

con instrucciones para cargarlo en el sistema y un link de descarga pueden encontrarse en el Apéndice A.

4. Recursión del programa: cálculo masivo de cotas

En el Capítulo 2 tomamos una configuración de potencias $\mathcal{A} \subset \mathbb{Z}^n$ y una matriz de coeficientes C y definimos el sistema polinomial de ecuaciones en n variables $x = (x_1, \ldots, x_n)$ como:

$$f_i(x) = \sum_{j=0}^{n+1} c_{ij} \cdot x^{a_j} = 0 \text{ para } i = 1, \ldots n \tag{4.0.1}$$

Llamamos $n_{\mathcal{A}}(C)$ a la cantidad de soluciones de este sistema en $(\mathbb{R}_{>0})^n$ y el Teorema 2.18 nos dió una cota superior para esta cantidad en el caso en el que $\mathcal{A}$ y C cumpliera (2.1.8) y (2.1.5).

Para calcular esta cota, en la Definición 2.7 tomamos los vectores $P_0, \ldots P_{n+1}$ que formen un Gale dual para las columnas de C y definimos una relación de equivalencia $\sim$ en $[n+2]$ tal que $i \sim j \Leftrightarrow \det(P_i, P_j) = 0$. Con esta relación en mente tomamos $J \subset [n+2]$ de cardinal k, un conjunto de representantes para la relación de equivalencia y definimos el par de órdenes $(\sigma, \overline{\sigma})$ con $\sigma \in \mathbb{S}_n$ un orden y $\overline{\sigma} \in \mathbb{S}_k$ un orden estricto para una matriz C que cumple (2.1.8) y (2.1.5) como el par que cumple simultáneamente:

$$\overline{\sigma}_i < \overline{\sigma}_j \Leftrightarrow \det(P_{\overline{\sigma}_i}, P_{\overline{\sigma}_j}) > 0, \tag{4.0.2}$$

$$\sigma_i < \sigma_j \Rightarrow \det(P_{\sigma_i}, P_{\sigma_j}) \geq 0 \text{ y} \tag{4.0.3}$$

$$\sigma|_J = \overline{\sigma} \text{ y } dom(\overline{\sigma}) = J. \tag{4.0.4}$$

Además demostramos en el Lema 2.8 que existe un número real $\delta \neq 0$ fijo tal que:

$$\delta \cdot \det(C(i,j)) = (-1)^{i+j} \det(P_i, P_j) \text{ para cualquier } i < j \in [n+2]. \tag{4.0.5}$$

Con esto podemos reescribir (4.0.2), (4.0.3) y (4.0.4) como:

"Existe $\delta \neq 0$ en $\mathbb{R}$ fijo tal que

$$\overline{\sigma}_i < \overline{\sigma}_j \Leftrightarrow sgn(det(C(\overline{\sigma}_i, \overline{\sigma}_j)) = (-1)^{\overline{\sigma}_i + \overline{\sigma}_j} \cdot sgn(\delta), \tag{4.0.6}$$

$$\sigma_i < \sigma_j \Rightarrow sgn(det(C(\sigma_i, \sigma_j)) = \begin{cases} (-1)^{\sigma_i + \sigma_j} \cdot sgn(\delta) & si \ \det(P_{\sigma_i}, P_{\sigma_j}) \neq 0 \\ 0 & si \ no \end{cases} \tag{4.0.7}$$

$$\sigma|_J = \overline{\sigma} \text{ y } dom(\overline{\sigma}) = J." \tag{4.0.8}$$

Entonces el par de órdenes $(\sigma, \overline{\sigma})$ está determinado por el signo de los menores principales de una matriz C que cumple (2.1.8) y (2.1.5). Inversamente, dos matrices distintas C y D en $\mathbb{R}^{n \times (n+2)}$ que cumplan (2.1.8), (2.1.5) y que los signos de sus menores principales sean iguales definirán el mismo par de órdenes $(\sigma, \overline{\sigma})$.

Esto nos lleva a enunciar

Definición 4.1. *Sean $\sigma \in \mathbb{S}_{n+2}$ y $\overline{\sigma} \in \mathbb{S}_k$ con $2 < k \leq n+2$ decimos que $(\sigma, \overline{\sigma})$ es un par de órdenes compatible si existe $J \subset [n+2]$ de cardinal k tal que $\sigma|_J = \overline{\sigma}$.*

Definición 4.2. *Sea $(\sigma, \overline{\sigma})$ un par de órdenes compatible y J como en la Definición 4.1, definimos el conjunto $\mathcal{U}_{(\sigma,\overline{\sigma})} \in \mathbb{R}^{n\times(n+2)}$ como:*

$$\mathcal{U}_{(\sigma,\overline{\sigma})} = \{C \in \mathbb{R}^{n\times(n+2)} \text{ de rango máximo / } C \text{ cumple (2.1.5) y } \exists\ \delta \in \mathbb{R} \text{ no nulo tq}$$
$$\forall i,j \in [n+2]\ sgn(det(C(\sigma_i,\sigma_j)) = (-1)^{\sigma_i+\sigma_j} \cdot sgn(\delta) \text{ si } i,j \in J, \text{ y es 0 si no.}\}.$$

Observación 4.3. Las condiciones (4.0.7) y (4.0.8) implican la condición (4.0.6). Por lo tanto si $C \in \mathbb{R}^{n\times(n+2)}$ cumple (2.1.8) y (2.1.5) y define el par de órdenes $(\sigma, \overline{\sigma})$ si y solo si $C \in \mathcal{U}_{(\sigma,\overline{\sigma})}$.

Observación 4.4. Como los órdenes compatibles $(\sigma, \overline{\sigma})$ están unívocamente determinados por los signos de los menores principales de la matriz, entonces tenemos que si $(\sigma, \overline{\sigma})$ y $(\theta, \overline{\theta})$ son dos pares de órdenes compatibles distintos entonces $\mathcal{U}_{(\sigma,\overline{\sigma})} \cap \mathcal{U}_{(\theta,\overline{\theta})} = \emptyset$.

Entonces, de la definición de $\mu_{(\sigma,\overline{\sigma})}$ en la Definición 2.10 y de `boundposPolSyst` en la sección 3.5 se desprende:

Proposición 4.5. *Sea $(\sigma, \overline{\sigma})$ un par de órdenes compatibles arbitrarios y sean C y $D \in \mathcal{U}_{(\sigma,\overline{\sigma})}$. Entonces para cualquier $\mathcal{A} \subset \mathbb{Z}^n$ configuración de potencias que cumpla (2.1.8) tenemos que:*

$$\texttt{boundposPolSyst}(\mathcal{A}, C) = \texttt{boundposPolSyst}(\mathcal{A}, D).$$

Observación 4.6. A lo largo de este capítulo nos referiremos a `boundposPolSyst` $(\mathcal{A}, C)$ como la función definida en la Sección 3.5 evaluada en $\mathcal{A}$ y C y como `boundposPolSyst` (sin indicar evaluación) al resultado que devuelve el programa, es decir a $varsigns(\mu)+1$ cuando μ y σ sean claros del contexto.

El objetivo de este capítulo será estudiar ciertas modificaciones al algoritmo principal que permitan calcular `boundposPolSyst` a partir de una configuración de potencias $\mathcal{A}$ dada y para cada uno de los pares $(\sigma, \overline{\sigma})$ de órdenes compatibles que cumplan alguna condición de interés. Para esto definiremos una lista `S` de pares de órdenes compatibles y modificaremos el programa 3.5 para calcular `boundposPolSyst` para una configuración $\mathcal{A}$ fija y cualquier par de órdenes $(\sigma, \overline{\sigma}) \in$ `S`. Luego utilizaremos esta modificación para dar una recursión que permita calcular la cota para múltiples valores distintos de $\mathcal{A}$ en la misma dimensión. Presentaremos resultados para esta recursión en dimensiones 2, 3, 4 y 5. Por último expondremos nuestras conclusiones para los resultados dados.

Los programas definidos en este capítulo se encuentran citados completos con instrucciones para cargarlos en el sistema y utilizar sus funciones en el Apéndice B.

4.1. Modificaciones a `boundposPolSyst`

Primero estudiaremos las modificaciones al programa `boundposPolSyst` del capitulo 3 que nos permitan calcular las cotas para cualquier lista de órdenes que brindemos. Este nuevo programa ya no tendrá entradas $\mathcal{A}$ y C sino, $\mathcal{A}$ y `S` con `S` una lista de pares de órdenes compatibles. Como para cada par de órdenes $(\sigma, \overline{\sigma}) \in$ `S` tenemos que toda función $C \in \mathcal{U}_{(\sigma,\overline{\sigma})}$ cumple (2.1.8) y (2.1.5), entonces las hipótesis a revisar se restringen únicamente a las que caen sobre $\mathcal{A}$.

Ejemplo 4.7. Esta lista fue definida a partir de `list S` para incluir todos los posibles pares de órdenes $(\sigma, \overline{\sigma})$ para tres vectores P_0, P_1, P_2. `S` tiene en el primer lugar al par de órdenes $\sigma = (1, 2, 3)$ y $\overline{\sigma} = (1, 2)$.

```
S;
> [1]:
>    [1]:
>       1,2,3
>    [2]:
>       1,2
> [2]:
>    [1]:
>       1,2,3
>    [2]:
>       1,3
> [3]:
>    [1]:
>       1,2,3
>    [2]:
>       1,2,3
> [4]:
>    [1]:
>       1,3,2
>    [2]:
>       1,3
> [5]:
>    [1]:
>       1,3,2
>    [2]:
>       1,2
> [6]:
>    [1]:
>       1,3,2
>    [2]:
>       1,3,2
> [7]:
>    [1]:
>       3,1,2
>    [2]:
>       3,1
> [8]:
>    [1]:
>       3,1,2
>    [2]:
>       3,2
> [9]:
>    [1]:
>       3,1,2
```

```
>     [2]:
>        3,1,2
```

Escribiremos entonces una función que tome como variables de entrada a una configuración de potencias $\mathcal{A}$ (`configA`) y a una lista de pares de órdenes compatibles S. La función debe calcular primero la matriz A a partir de $\mathcal{A}$ y un vector `kerA` generador del núcleo de A como antes. Sin embargo, a diferencia de la función original, debe revisar únicamente si se cumplen las hipótesis referidas a $\mathcal{A}$: la matriz A tiene rango máximo y `kerA` no tiene coordenadas nulas. Si estas hipótesis se cumplen el programa avanzará en el cálculo de las cotas, si no saldrá con una única cota: -1:

```
2  proc boundsForAllOrders(list configA, list S){
3    LIB "matrix.lib";
4    LIB "rootsur.lib";
5    list AllTheBounds;
6
7    //Defino la matriz A que tiene como columnas un 1 y el
      vector de la configuracion
8    for(int i=1; i<=size(configA);i++){
9      configA[i]=transpose(concat(matrix(1),transpose(matrix(
      configA [i]))));
10   }
11   def A=concat(configA[1.. size(configA)]);
12   def kerA=syz(A)[1];
13
14   //Chequeo las hipotesis para A
15   int check_hipotesis=1;
16   if(nrows(A)!=rank(A)){
17     check_hipotesis=0;
18   }
19   for(int j=1; j<=ncols(A); j++){
20     if(kerA[j]==0){
21       check_hipotesis=0;
22     }
23   }
24
25   if( check_hipotesis ==0){
26     AllTheBounds[1]=-1;
27   } else {
         [...]
72   }
73   return (AllTheBounds);
74 }
```

El programa `boundposPolSyst` luego calcula el orden y el orden estricto definidos por C. En este caso en vez de calcular el orden debemos tomar uno por uno los ordenes `sigma` y `sigma_barra` que vienen listados por S. Con cada par de órdenes reproduciremos el código de la Sección 3.5.2 para calcular la sucesión μ definida por `kerA` y los órdenes `sigma` y

`sigma_barra`. Por último guardaremos la cota obtenida (`TheBound`) junto con los órdenes que la produjeron de la forma $(\texttt{TheBound}, \sigma, \overline{\sigma})$ en la lista `AllTheBounds`.

```
27    } else {
28      intvec sigma_barra;
29      intvec sigma;
30      int TheBound;
31      list vectorBound;
32      def n=size(kerA);
33      int k;
34      int maxIndex;
35
36      for(int i=1; i<=size(S); i++){
37        sigma=S[i][1];
38        sigma_barra=S[i][2];
39
40        list mu;
41        k=1;
42        maxIndex = size(sigma_barra);
43
44        for(int j=1; j<=maxIndex; j++){
45          if(j==1){
46            mu[1]=int(kerA[sigma_barra[1]]);
47          } else{
48            mu[j]=mu[j-1]+int(kerA[sigma_barra[j]]);
49          }
50          k++;
51
52          if(j!=maxIndex){
53            while (sigma_barra[j+1]!=sigma[k]){
54              mu[j]=mu[j]+int(kerA[sigma[k]]);
55              k++;
56            }
57          } else {
58            while (k!=size(sigma)+1){
59              mu[j]=mu[j]+int(kerA[sigma[k]]);
60              k++;
61            }
62          }
63        }
64        TheBound = varsigns(mu)+1;
65
66        vectorBound[1] = TheBound;
67        vectorBound[2] = sigma;
68        vectorBound[3] = sigma_barra;
69
70        AllTheBounds[i] = vectorBound;
71      }
```

```
72      }
```

4.2. Configuraciones derivadas y conteo de cotas

La función descrita calcula, a partir de una configuración de potencias $\mathcal{A}$ y una lista de órdenes `S`, la lista que para cada $(\sigma, \overline{\sigma})$ tiene `boundposPolSyst`$(\mathcal{A}, C)$ con $C \in \mathcal{U}_{(\sigma,\overline{\sigma})}$. El objetivo de esta sección es escribir un programa que nos permita recorrer una gran cantidad de configuraciones de potencias $\mathcal{A}$ de una misma dimensión n y calcular la cota para el sistema que determina cada una de ellas con cada uno de los pares de órdenes de la lista `S`.

Para hacer esto tomaremos una configuración inicial $\mathcal{A} = \{a_0, \dots a_{n+1}\} \subset \mathbb{Z}^n$ y configuraciones derivadas $\mathcal{A}_{i,v}$ para cada $i \in [n+2]$, $v \in \{0, \dots, k\}^n \in \mathbb{Z}^n$ y $k \in \mathbb{N}$ fijo. La configuración $\mathcal{A}_{i,v} := \{a'_0, \dots a'_{n+1}\}$ estará definida por $a'_i = a_i + v$ y $a'_j = a_j$ si $i \neq j$.

Con cada una de estas configuraciones calcularemos `boundposPolSyst` para cada uno de los pares de órdenes de `S`. Como la sucesión μ de sumas parciales que define el teorema pertenece a $\mathbb{R}^{n+2}$ y su última coordenada es 0 entonces $varsigns(\mu) \leq n$ y por lo tanto la máxima cantidad de soluciones positivas que tiene un sistema como los que estamos estudiando es $n+1$. Entonces para cada una de las configuraciones $\mathcal{A}_{i,v}$ definiremos un vector $\Delta_{\mathcal{A}_{i,v}} \in \mathbb{R}^{n+1}$ tal que para cada $m \in [n+2]$:

$$\Delta_{\mathcal{A}_{i,v}}[m] := \#\{(\sigma, \overline{\sigma}) / \texttt{boundposPolSyst}(\mathcal{A}_{i,v}, C) = m \text{ para cualquier } C \in \mathcal{U}_{(\sigma,\overline{\sigma})}\}.$$

Guardaremos luego en un archivo separado por comas la tira i, v, Δ que identifica la cantidad de cotas totales obtenidas con la configuración $\mathcal{A}_{i,v}$ utilizada.

Por otro lado vamos a guardar también la información sobre los sistemas en los que `boundposPolSyst`$= n+1$, es decir que es el máximo posible. Para estos sistemas guardaremos en una misma tira la configuración $\mathcal{A}_{i,v}$, el núcleo de la matriz $A_{i,v}$, el i, el v y el orden σ que genere esta cota. Escribiremos luego en un archivo esta información separada por comas, una línea por cada tira guardada.

Escribir archivos en Singular

Para escribir un archivo en SINGULAR es necesario primero definir un `link` a este archivo. Los links son de la forma

```
link UnLink = ":r  + NombreDelArchivo.txt";
```

Para abrir el archivo en modo lectura.

```
link UnLink = ":w  + NombreDelArchivo.txt";
```

Para sobreescribir el archivo.

```
link UnLink = ":a  + NombreDelArchivo.txt";
```

Para agregar líneas al archivo. Luego para escribir en el archivo usamos el comando `write` indicando el link al archivo y la información que queremos escribir. Por ejemplo:

```
intvec MyVec=1,3,5,7;
link UnLink = ":w ArchivoDePrueba.txt";
write(UnLink, MyVec);
```

generará un archivo `ArchivoDePrueba.txt` (en la carpeta desde la que este corriendo SINGULAR) que contiene en una única línea los números $1, 3, 5, 7$ separados por comas. Es importante resaltar que este funcionamiento del comando `write` es el descrito en el manual de SINGULAR [30], pero solo se comporta así cuando SINGULAR corre sobre el sistema operativo Windows. Corriendo sobre Linux, el comportamiento es distinto y es necesario hacer modificaciones para que la escritura sea la correcta.

La función `CountingBounds`

Primero definiremos la función con sus inputs, cargaremos las librerías necesarias y definiremos los archivos en los que guardaremos la información, sus links y encabezados:

```
77  proc CountingBounds(list config, int dimen, list S, int k){
78    LIB "matrix.lib";
79    LIB "rootsur.lib";
80
81    //defino las variables para guardar los archivos
82    string name="CotasPorDireccion Dim ";
83    string s; //s es el string que usa el script para guardar
        el output
84    string sPrima;
85    link lPrima;
86    link l;
87    string encabezado;
88    encabezado="posicion, vector, #cota=1, #cota=2";
89
90    //genero el encabezado y link para CotasPorDireccion
91    encabezado=string(encabezado+ ", #cota=" , dimen+1) + ",
       sin cota";
92    s=string(name,dimen);
93    l=":w " + s + ".txt";
94    write(l,encabezado);
95    //paso el link de write a append
96    l=":a " + s + ".txt";
97
98    //genero el encabezado y link para OrdenesConCotaMax
99    sPrima= string("OrdenesConCotaMax Dim ", dimen);
100   lPrima=":w " + sPrima + ".txt";
101   write(lPrima,"posicion, vector, orden");
102   //paso el link de write a append
103   lPrima=":a " + sPrima + ".txt";
```

Definimos las variables auxiliares que utilizaremos:

```
105   list configA; //la configuracion que generamos en cada
       paso
106   list alpha; //la lista de cotas que devuelve la funcion
107   intvec Delta;//cuenta la cantidad de apariciones de cada
       cota
```

```
108   vector v;//el vector que define la configuracion a
       utilizar
109   list VectorCotas; //guarda (i,v,Delta)
110   list VectorOrdenes;//guarda (i,v,sigma)
111   matrix A; //La matriz A asociada a la configuracion
112
113   //Las variables auxiliares que necesito para recorrer los
        v
114   int total = k^dimen;
115   int dividendo;
116   intvec vCero;
117   intvec vCreado;
118   vCero[dimen]=0;//genera un vector de ceros de tamano
       dimen
```

Definiremos ahora las configuraciones derivadas $\mathcal{A}_{i,v} \in \mathbb{Z}^n$. Primero necesitamos una forma de recorrer los vectores $v \in \{0, \ldots, k-1\}^n$ que no dependa de k y n. Una forma simple de recorrerlos parecería ser con una cadena de ciclos `for`. Por ejemplo, si $k = 10$ y $n = 2$ podríamos escribir:

```
for(int a=1; a<=k; a++){
  for(int b=1; b<=k; b++){
    v=[a-1,b-1];
  }
}
```

Pero este código no puede generalizarse en más dimensiones: hay n ciclos `for`. Para que el programa funcione en general es necesario plantearlo distinto: como el conjunto $\{0, \ldots, k-1\}^n$ tiene cardinal k^n entonces hay una biyección $\{1, \ldots, k^n\} \to \{0, \ldots, k-1\}^n$. Esta biyección esta dada por escribir los números enteros en base k. Podemos entonces escribir el programa que recorre todos los vectores $v \in \{0, \ldots, k-1\}^n$ como:

```
114   int total = k^dimen;
115   int dividendo;

123     for (int count=1; count<=total; count++){
124       int lugar=1;
125       vCreado=vCero;
126       dividendo=count;
127       while((lugar<= dimen) && (dividendo>0)  ){
128         // toma resto y div es division entera
129         vCreado[lugar]=dividendo%k;
130         dividendo= dividendo div k;
131         lugar=lugar+1;
132       }
133       v=[vCreado[1..dimen]];
```

Donde la línea `v=[vCreado[1..dimen]];` permite cambiar de tipo de variable: `vCreado` era de tipo `intvec`, un vector de enteros y `v` es de tipo `vector`, un vector en los generadores del anillo.

Luego para cada i entre 1 y $n+2=4$ definiremos $\mathcal{A}_{i,v}=\mathcal{A}$. En nuestro programa la configuración $\mathcal{A}$ es `config` y la configuración $\mathcal{A}_{i,v}$ es `configA` para cualquier i, v. Sobreescribiremos el valor de `configA` en cada paso. Generaremos un ciclo que recorra todos los vectores $v \in \{0,1,\dots k-1\}^n$, defina `configA` igual a la configuración inicial y cambie `configA`$[i]$ por `config`$[i]+v$. Por último guardamos en `alpha` la salida de la función `boundsForAllOrders`.

```
120    for(int i=1; i<=dimen+2; i++){
121
136        configA=config;
137        configA[i]=config[i]+v;
138        alpha =boundsForAllOrders(configA,S);
```

Cuando las hipótesis sobre $\mathcal{A}$ no se cumplen, `alpha` solo contiene el valor -1 y en este caso no tiene sentido el análisis de los datos. Si las hipótesis se cumplen, la lista de vectores `alpha` es de la forma `(Bound, sigma, sigma_barra)`. Entonces para cada valor de `Bound` sumaremos uno a Δ[`Bound`] para llevar la cuenta de las cantidad de veces que devolvió la misma cota. Además, cada vez que la función arroje la cota máxima para un orden escribiremos en el archivo `lPrima` la tira (i, v, σ). Por último escribimos en otro archivo la tira (i, v, Δ).

```
140        if(size(alpha[1])!=1){
141          Delta=0,0,0,0;
142          VectorCotas[1]=i;
143          VectorCotas[2]=v;
144          for (int j=1; j<=size(alpha); j++){
145            Delta[alpha[j][1]]=Delta[alpha[j][1]]+1;
146            if(alpha[j][1]==dimen+1){
147              VectorOrdenes[1]=i;
148              VectorOrdenes[2]=v;
149              VectorOrdenes[3]=alpha[j][2];
150              VectorOrdenes[4]=alpha[j][3];
151              write(lPrima,VectorOrdenes);
152            }
153          }
154          VectorCotas[3]=Delta;
155          write(l,VectorCotas);
```

La repetición de este ciclo sobre todos los i escribirá dos archivos separados por comas con la información recabada. Estos archivos pueden luego analizarse en Excel (si los resultados son pocos) o con scripts en R [43].

4.3. Funciones generadoras de órdenes

En este punto es razonable empezar a preguntarse como generar listas exhaustivas de pares de órdenes compatibles para una dimensión n. Quisiéramos dar funciones que, para una dimensión dada, generen una lista con todos los posibles pares de órdenes compatibles que cumplan cierta característica. En esta sección presentaremos dos de estas funciones, una que calcule los pares de órdenes compatibles generados por las matrices $C \in \mathbb{R}^{n\times(n+2)}$

que tienen todos sus menores maximales no nulos y otra función que calcule todos los pares de órdenes compatibles en una dimensión arbitraria. Como enunciamos en la Observación 2.19, el valor de `boundposPolSyst` es el mismo para un par de órdenes y para el par con orientación inversa. Para evitar cálculos duplicados computaremos solo una de las orientaciones.

Para definir estas listas de pares de órdenes utilizaremos la librería de SINGULAR `qmatrix.lib`. En particular necesitamos la función `SymGroup(int n)` que, para cada n, devuelve una matriz que en cada fila tiene una permutación de los elementos $\{1, \dots n\}$.

Ejemplo 4.8. La presentación de $\mathbb{S}_3$ en SINGULAR es:

```
SymGroup (3);
> 1,2,3,
> 1,3,2,
> 3,1,2,
> 2,1,3,
> 2,3,1,
> 3,2,1
```

Órdenes uniformes

Sea $C \in \mathbb{R}^{n\times(n+2)}$ que cumple (2.1.8), (2.1.5) y tal que todos sus menores maximales son no nulos, entonces $\det(C(i,j)) \neq 0$ para todo $i,j \in [n+2]$ y por lo tanto $\sigma = \overline{\sigma}$. Considerando el par de órdenes compatible $(\sigma, \overline{\sigma})$ tales que $\sigma = \overline{\sigma} \in \mathbb{S}_{n+2}$ y el orden con orientación inversa $(\sigma^{op}, \overline{\sigma}^{op})$, por la Observación 2.19 sabemos que `boundposPolSyst` es exactamente igual considerando cualquiera de los dos pares de órdenes. Luego, para definir `S` consideraremos solo una de las orientaciones posibles y por lo tanto hay $\frac{(n+2)!}{2}$ pares de órdenes disponibles. Definiremos la función generadora de órdenes `UnparalleledOrders` que calcule esta lista, para esto debemos tomar uno a uno los lugares de `SymGroup(int n)`, revisar que el orden con la orientación inversa no se haya agregado antes a la lista y si no está aún agregarlo a la lista final con `sigma=sigma_barra`. La siguiente función hace exactamente esto:

```
164 proc UnparalleledOrders(int n){
165   LIB "qmatrix.lib";
166   def S=SymGroup (n);
167   list oneOrder;//oneOrder es un orden posible
168   list Orders;//es la lista de todos los oneOrder posibles,
        salvo orientacion inversa

171   intvec v;
172   intvec w;
173   int repeat=0;
174
175   for(int i=1; i<= nrows(S); i++){
176     repeat=0;
177     v=intvec(S[i,1.. ncols(S)]);
178
179     for(int j=1; j<=size(v); j++){
```

```
180         w[j]=v[size(v)+1-j];
181       }
182
183       for(int k=1; k<=size(oneOrder); k++){
184         if(oneOrder[k]==w){
185           repeat=1;
186         }
187       }
188
189       if(repeat==0){
190         oneOrder[i] = v;
191         Orders[i]= list(oneOrder[i])+list(oneOrder[i]);
192       }
193     }
194
195     return(Orders);
196   }
```

La función `UnparalleledOrders` devolverá una lista con todos los posibles pares de órdenes en los que $\sigma = \overline{\sigma}$:

```
UnparalleledOrders (3);
> [1]:
>    [1]:
>      1,2,3
>    [2]:
>      1,2,3
> [2]:
>    [1]:
>      1,3,2
>    [2]:
>      1,3,2
> [3]:
>    [1]:
>      3,1,2
>    [2]:
>      3,1,2
```

Observación 4.9. El par (σ, σ) siempre es un par compatible trivialmente.

Todos los órdenes

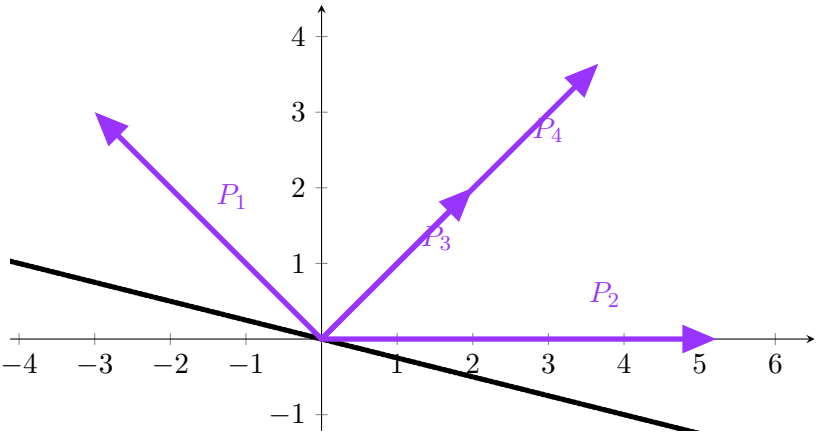
Definiremos ahora la función `AllOrders` que calcule, para un n dado todos los posibles pares de órdenes compatibles. Para esto primero haremos una lista con todos los posibles órdenes σ para n elementos reutilizando el algoritmo anterior:

```
199 proc AllOrders(int n){
200   LIB "qmatrix.lib";
201   def S=SymGroup (n);
202   list OrdersSn;
```

```
206   intvec v;
207   intvec w;
208   int repeat=0;
209
210   for(int i=1; i<= nrows(S); i++){
211     repeat=0;
212     v=intvec(S[i,1.. ncols(S)]);
213
214     for(int j=1; j<=size(v); j++){
215       w[j]=v[size(v)+1-j];
216     }
217
218     for(int k=1; k<=size(OrdersSn); k++){
219       if(OrdersSn[k]==w){
220         repeat=1;
221       }
222     }
223
224     if(repeat==0){
225       OrdersSn[i] = v;
226     }
227   }
```

Ahora queremos asignar a cada σ todos los posibles ordenes estrictos $\overline{\sigma}$ asociados. Para esto notemos que dar un par $(\sigma, \overline{\sigma})$ es equivalente a dar un orden σ y el conjunto J en el que coinciden. Escribiremos $J \in [n+2]$ a partir de una tira de unos y ceros que indiquen en cada lugar 1 si el elemento pertenece a J y 0 si no.

Ejemplo 4.10. Si tomo $\sigma = (2,3,4,1)$ un orden sobre los vectores $\{P_1, P_2, P_3, P_4\}$ y el vector $v = (1,1,0,1)$, entonces el par (σ, v) indica que el orden sobre el semiplano es P_2, P_3, P_4, P_1 y que los vectores P_3 y P_4 son colineales. Luego el orden estricto será $\overline{\sigma} = (2,3,1)$ y el par de órdenes que corresponde a $((2,3,4,1),(1,1,0,1))$ será $((2,3,4,1),(2,3,1))$.



Construiremos primero una lista con todos los vectores de $\{0,1\}^n$ no nulos. Para hacer esto utilizaremos la biyección entre $\{1,\ldots,2^{n-1}\} \to \{0,1\}^n$ dada por escribir los números naturales menores a 2^{n-1} en sistema binario.

```
234   list ZerosAndOnes;
235   intvec v;
236   int k;
237   int y;
238   v[n]=0;
239
240   for(int x=1; x< 2^(n-1) ; x++){
241     k=2;
242     v[1]=1;
243     y=x;
244     while (y>0){
245       v[k]=y%(2);
246       y=y div 2;
247       k++;
248     }
249     ZerosAndOnes [x]= v;
250   }
```

Por último, para cada σ en la lista `OrdersSn` y cada vector v en la lista `ZerosAndOnes` generaremos el orden estricto $\overline{\sigma}$ asociado como el orden que coincide con σ en los lugares en los que v es 1.

```
254   list FinalOrders;
255   intvec StrictOrder;
256   int k=1;
257   int m=1;
258
259   for(int i=1; i<= size(OrdersSn); i++){
260     for (int j=1; j<= size(ZerosAndOnes); j++){
261       for(int l=1; l<= n;l++){
262         if(ZerosAndOnes[j][l]){
263           StrictOrder[m]=OrdersSn[i][l];
264           m++;
265         }
266       }
267       FinalOrders[k]= list(OrdersSn[i])+list(StrictOrder);
268       k++;
269       StrictOrder=0;
270       m=1;
271     }
272   }
```

Ambos programas se encuentran completos en el Apéndice B

4.4. Aplicaciones a los casos $n = 2, 3, 4$ y 5

Utilizamos los programas desarrollados en este capítulo para calcular y contar las cotas a múltiples sistemas en forma masiva. Definimos una configuración inicial $\mathcal{A} \subset \mathbb{Z}^n$ para $n = 2, 3, 4$ y 5. Para cada una de estas configuraciones aplicamos la función

`CountingBounds` definida en la Sección 4.2 con $k = 10$ y `S` la lista de órdenes definida por `UnparalleledOrders`.

Las configuraciones iniciales que usamos para cada dimensión fueron:

- Para $n = 2$, $\mathcal{A} = \{(1,0),(0,1),(2,3),(1,2)\}$.
- Para $n = 3$, $\mathcal{A} = \{(1,0,0),(0,1,0),(0,0,1),(2,3,1),(1,4,5)\}$.
- Para $n = 4$, $\mathcal{A} = \{(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1),(2,3,1,2),(1,4,5,2)\}$.
- Para $n = 5$, $\mathcal{A} = \{(1,0,0,0,0),\ (0,1,0,0,0),\ (0,0,0,1,0),\ (0,0,0,1,0),\ (0,0,0,0,1),\ (2,3,1,2,1),\ (1,4,5,1,2)\}$.

Con estas configuraciones generamos un script que, para cada dimensión defina la lista de órdenes, la configuración a utilizar y aplique la función `CountingBounds` a cada una de ellas:

```
ring r=0,(x,y),dp;
>"Sing OptimalDescartesRuleOfSigns - No Order.sing";

int k=10;

int dimen=2;
def S=UnparalleledOrders(dimen+2);
config= list([1,0])+list([0,1])+list([2,3])+list([1,2]);
CountingBounds(config, dimen, S, k);

dimen=3;
S=UnparalleledOrders(dimen+2);
config= list([1,0,0])+list([0,1,0])+list([0,0,1])+list
   ([2,3,1])+list([1,4,5]);
CountingBounds(config, dimen, S, k);

dimen=4;
S=UnparalleledOrders(dimen+2);
config= list([1,0,0,0])+list([0,1,0,0])+list([0,0,1,0])+
   list([0,0,0,1])+list([2,3,1,2])+list([1,4,5,2]);
CountingBounds(config, dimen, S, k);

dimen=5;
S=UnparalleledOrders(dimen+2);
config= list([1,0,0,0,0])+ list([0,1,0,0,0])+ list
   ([0,0,0,1,0]) +list([0,0,0,1,0]) +list([0,0,0,0,1]) +
   list([2,3,1,2,1]) +list([1,4,5,1,2]);
CountingBounds(config, dimen, S, k);

exit;
```

Este script generó los archivos `CotasPorDireccion Dim N.txt` y `OrdenesConCotaMax Dim N.txt` para cada $N = 2,3,4$ y 5. Reproducimos a conti-

nuación las primeras líneas de los dos archivos para $N = 2$, los archivos completos se encuentran disponibles en [15].

SINGULAR utiliza `gen(i)` para referirse a los vectores canónicos del espacio. En este caso, con $N = 2$, `gen(1)`$= (1, 0)$ y `gen(2)`$= (0, 1)$.

Listing 4.1: CotasPorDireccion Dim 2.txt

```
posicion, vector, #cota=1, #cota=2, #cota=3,sin cota
2,gen(2),6,6,0,0
2,2*gen(2),6,6,0,0
2,3*gen(2),6,6,0,0
2,4*gen(2),6,6,0,0
2,5*gen(2),6,6,0,0
2,6*gen(2),6,6,0,0
2,7*gen(2),6,6,0,0
2,8*gen(2),6,6,0,0
2,9*gen(2),6,6,0,0
```

Listing 4.2: OrdenesConCotaMax Dim 2.txt

```
posicion, vector, orden
2,gen(2)+2*gen(1),1,2,3,4
2,3*gen(2)+2*gen(1),1,3,2,4
2,5*gen(2)+2*gen(1),3,1,4,2
2,6*gen(2)+2*gen(1),3,1,4,2
2,7*gen(2)+2*gen(1),3,1,4,2
2,8*gen(2)+2*gen(1),3,1,4,2
2,9*gen(2)+2*gen(1),3,1,4,2
2,3*gen(1),1,4,3,2
2,3*gen(1),3,4,1,2
```

Notemos que este script analizó en dimensión 2 una lista de $\frac{4!}{2} = 12$ órdenes y $4 \cdot 10^2$ configuraciones, es decir 4800 sistemas, y sobre ellos sólo guardó la información sobre los que cumplían las hipótesis del Teorema, poco más de 250. Sin embargo la cantidad de sistemas crece exponencialmente con la dimensión. En dimensión 5 el programa trabajó con una lista de $\frac{5!}{2} = 60$ órdenes y $5 \cdot 10^5$ configuraciones, la cantidad total de sistemas es de $3 \cdot 10^7$ y la cantidad de sistemas que cumplen las hipótesis es mayor a 190.000. Notemos además que esta cantidad es chica si consideramos que en la elección de la configuración inicial $\mathcal{A}$ en dimensión 5 hay dos vectores de la configuración inicial iguales y por lo tanto las configuraciones derivadas $\mathcal{A}_{i,v}$ con $i \neq 3, 4$ no pueden tener dimensión máxima para ningún v. Así en este caso solo quedaron $1,2 \cdot 10^7$ sistemas potencialmente válidos.

Estos datos los analizamos luego en R [43]: escribimos los archivos separados por comas en formato de `data.frame` y aislamos los datos más relevantes. Agregamos el cálculo del núcleo de la matriz A para cada $\mathcal{A}_{iv}$ que devuelva, para algún orden, la cota máxima. Para poder visualizar estos datos escribimos una aplicación de Shiny [17] que nos permitió dar un formato de planilla con filtros y a la vez condensar la gran cantidad de datos. Esta aplicación de Shiny se encuentra disponible en [16].

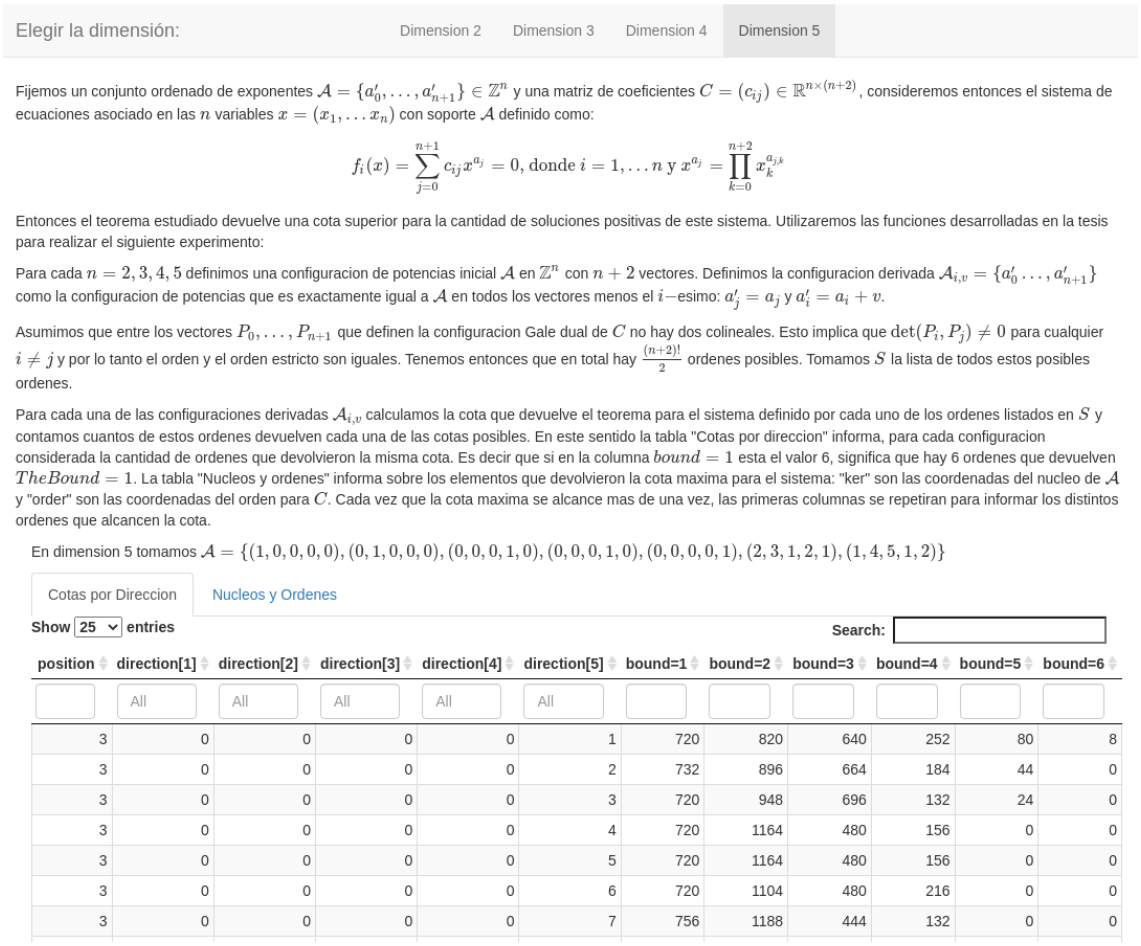


Figura 4.1.: Aplicación Web: OptimalDescartesRuleOfSigns

4.5. Nuestros resultados

A lo largo de esta Sección asumiremos que $\mathcal{A} \subset \mathbb{Z}^n$ y $C \in \mathbb{R}^{n\times(n+2)}$ cumplen (2.1.8) y (2.1.5) y además la matriz C es uniforme. Es decir, nos restringiremos los sistemas polinomiales definidos sobre matrices de coeficientes $C \in \mathbb{R}^{n\times(n+2)}$ que no tienen menores maximales nulos, esto es equivalente a asumir que los pares de órdenes compatibles $(\sigma, \overline{\sigma})$ son tales que $\sigma = \overline{\sigma}$ o que entre el conjunto de vectores Gale dual a las columnas de C, $\{P_0, \dots P_{n+1}\}$ no hay dos colineales. En este contexto notaremos $\mathcal{U}_{(\sigma,\overline{\sigma})} = \mathcal{U}_\sigma$. Nuestro objetivo es estudiar las condiciones sobre el sistema (4.0.1) que hacen que `boundposPolSyst` sea máxima.

Definición 4.11. *Sea $\mathcal{A} \subset \mathbb{Z}^n$ un conjunto de cardinal $n+2$ que verifica (2.1.8), definimos el número $\vartheta(\mathcal{A}) \in \mathbb{N}_0$ como:*

$$\vartheta(\mathcal{A}) = \#\{\sigma / \texttt{boundposPolSyst}(\mathcal{A}, C) = n + 1 \text{ para todo } C \in \mathcal{U}_\sigma\}.$$

Observación 4.12. $\vartheta(\mathcal{A})$ es siempre un número par, pues si $\sigma \in \{\sigma / \texttt{boundposPolSyst}(\mathcal{A}, C) = n + 1 \text{ para todo } C \in \mathcal{U}_\sigma\}$ entonces σ^{op} también, con σ^{op} el orden con la orientación contraria a σ por la Observación 2.19.

Revisaremos primero las condiciones sobre C y $\mathcal{A}$ para los que $\vartheta(\mathcal{A}) > 0$ para luego enunciar y demostrar los Teoremas 4.17 y 4.19 que dan cotas superiores para el valor de $\vartheta(\mathcal{A})$.

4.5.1. Sobre C y sus órdenes

Sea $C \in \mathbb{R}^{n\times(n+2)}$ una matriz uniforme y tomemos el par de órdenes definido por C, $(\sigma, \overline{\sigma})$. Usando el Lema 2.8 podemos afirmar que existe $\delta \in \mathbb{R} \setminus \{0\}$ tal que:

$$sgn(\det(C(\sigma_i, \sigma_j)) = (-1)^{\sigma_i+\sigma_j} \cdot sgn(\delta),$$

si $\sigma_i < \sigma_j$. Luego el orden σ define los signos de los menores maximales de todas las matrices $C \in \mathcal{U}_\sigma$. Podemos entonces reescribir $\mathcal{U}_\sigma$ como:

$$\mathcal{U}_\sigma = \{C \in \mathbb{R}^{n\times(n+2)} / \ C \text{ cumple (2.1.5) y } \exists\ \delta \in \mathbb{R} \text{ no nulo tal que } \ \forall \sigma_i < \sigma_j, \\ sgn(\det(C(\sigma_i, \sigma_j)) = (-1)^{\sigma_i+\sigma_j} \cdot sgn(\delta)\}. \quad (4.5.1)$$

Por otro lado, tomemos $\sigma \in \mathbb{S}_{n+2}$ arbitrario y un conjunto de $n+2$ vectores $\mathcal{P} = \{P_0, \dots, P_{n+1}\} \subset \mathbb{R}^2$ contenidos en un semiplano y entre los que no hay dos colineales. Sea $\alpha \in \mathbb{S}_{n+2}$ tal que

$$\alpha_i < \alpha_j \Leftrightarrow \det(P_{\alpha_i}, P_{\alpha_j}) > 0.$$

Podemos renumerar los vectores de $\mathcal{P}$ para que coincidan con el orden σ definiendo $\hat{P}_i = P_{(\sigma^{-1}\alpha)_i}$. Así consideramos $\hat{\mathcal{P}} = \{\hat{P}_0, \dots, \hat{P}_n\}$ y para todo i, j tales que $\sigma_i < \sigma_j$ tenemos que:

$$\det(\hat{P}_{\sigma_i}, \hat{P}_{\sigma_j}) = \det(P_{(\sigma\sigma^{-1}\alpha)_i}, P_{(\sigma\sigma^{-1}\alpha)_j}) = \det(P_{\alpha_i}, P_{\alpha_j}) > 0,$$

por lo que σ es un orden para $\hat{\mathcal{P}}$. Luego, si definimos la matriz $M_{\hat{\mathcal{P}}}$ que tiene como filas los vectores de $\hat{\mathcal{P}}$:

$$M_{\hat{\mathcal{P}}} = \begin{pmatrix} \hat{P}_0 \\ \vdots \\ \hat{P}_{n+1} \end{pmatrix}.$$

Tomamos C la matriz que tiene como filas vectores que forman una base del núcleo de $M_{\hat{\mathcal{P}}}$, $C = (\ker(M_{\hat{\mathcal{P}}}^t)^t$. Como en $\hat{\mathcal{P}}$ no hay dos vectores colineales entonces $rk(M_{\hat{\mathcal{P}}}) = 2$ y por lo tanto $\dim(\ker(M_{\hat{\mathcal{P}}})) = n$, así $rk(C) = n$. Luego C es una matriz uniforme de coeficientes que cumple (2.1.8) y (2.1.5) y cuyo par de órdenes asociado es (σ, σ), por lo tanto $C \in \mathcal{U}_\sigma$. Concluimos entonces que $\mathcal{U}_\sigma \neq \emptyset$ para todo $\sigma \in \mathbb{S}_{n+2}$.

4.5.2. Sobre $\mathcal{A}$ y $varsigns(\mu)$

Sea $\mathcal{A} = \{a_0, \dots, a_{n+1}\} \subset \mathbb{Z}^n$ un circuito y supongamos que $\sigma = id$, si no basta tomar $\mathcal{A}_\sigma = \{a_{\sigma_0}, \dots, a_{\sigma_{n+1}}\}$. Recordemos que en (2.1.1) consideramos la matriz $A \in \mathbb{R}^{(n+1)\times(n+2)}$ asociada a $\mathcal{A}$ como:

$$A = \begin{pmatrix} 1 & \dots & 1 \\ a_0 & \dots & a_{n+1} \end{pmatrix}.$$

Como $\mathcal{A}$ cumple (2.1.8), $rk(A) = n+1$ y $b \in \mathbb{R}^{n+2}$ definido como $b_j = (-1)^j \cdot \det(A(j))$ es un generador del núcleo de A que tiene todas sus coordenadas no nulas como vimos en (2.3). Luego las sucesiones λ y μ dadas en la Definición 2.10 serán para cada $j \in [n+1]$:

$$\lambda_j = b_j = (-1)^j \det(A(j)), \quad (4.5.2)$$

$$\mu_j = \mu_{j-1} + b_j \text{ y } \mu_0 = b_0. \tag{4.5.3}$$

Aplicando el Teorema 2.18 y la Proposición 2.21 tenemos:

$$\texttt{boundposPolSyst} = 1 + varsigns(\mu) \leq varsigns(\lambda) \tag{4.5.4}$$

Entonces para que exista C con $\texttt{boundposPolSyst}(\mathcal{A}, C) = n+1$ máxima debe suceder que $varsigns(\lambda) = n+1$, por lo tanto los signos de $\lambda_j = (-1)^j \cdot \det(A(j))$ deben ser alternados y $\det(A(j))$ debe tener el mismo signo para todo $j \in [n+2]$, es decir, *todos los menores maximales de A deben tener el mismo signo* y podemos suponer que son todos positivos.

Proposición 4.13. *Sea $A \in \mathbb{R}^{(n+1)\times(n+2)}$ de rango máximo y tal que $\det(A(j)) > 0$ para todo $j = 0, \ldots, n+1$. Entonces si $b \in \mathbb{R}^{n+2}$ es un generador del núcleo, vale que $varsigns(b_0, \ldots, b_{n+1}) = n+1$.*

Lo mismo vale si $\det(A(j)) < 0$ para todo $j = 0, \ldots, n+1$.

Queremos ahora estudiar cuándo $1 + varsigns(\mu) = n+1$. Para esto veamos primero el siguiente ejemplo:

Ejemplo 4.14. Sea $\mathcal{A} \subset \mathbb{Z}^3$, tal que $\mathcal{A}$ define la matriz asociada $A \in \mathbb{R}^{4\times 5}$ con todos sus menores positivos y $b \in \mathbb{R}^5$ generador del núcleo de A cumple que $varsigns(b) = 4$. Luego $b_j = (-1)^j \det(A(j))$ y $\mu_j = \mu_{j-1} + b_j$, para que $varsigns(\mu) = 3$ máximo debe suceder que $\mu_0 > 0,\ \mu_1 < 0,\ \mu_2 > 0,\ \mu_3 < 0,\ \mu_4 > 0$ y $\mu_5 = 0$:

$$\begin{aligned}
&\det(A(0)) > 0\\
&\det(A(0)) - \det(A(1)) < 0\\
&\det(A(0)) - \det(A(1)) + \det(A(2)) > 0\\
&\det(A(0)) - \det(A(1)) + \det(A(2)) - \det(A(3)) < 0\\
&\det(A(0)) - \det(A(1)) + \det(A(2)) - \det(A(3)) + \det(A(4)) > 0.
\end{aligned}$$

Despejando tenemos:

$$\begin{aligned}
\det(A(0)) &> 0\\
\det(A(0)) &< \det(A(1))\\
\det(A(0)) + \det(A(2)) &> \det(A(1))\\
\det(A(0)) + \det(A(2)) &< \det(A(1)) + \det(A(3))\\
\det(A(0)) + \det(A(2)) + \det(A(4)) &> \det(A(1)) + \det(A(3)).
\end{aligned}$$

De lo que se deduce

$$\begin{aligned}
0 < \det(A(0)) < \det(A(1)) < \det(A(0)) + \det(A(2)) < \det(A(1)) + \det(A(3))\\
< \det(A(0)) + \det(A(2)) + \det(A(4)).
\end{aligned}$$

Podemos generalizar esta idea para $n \in \mathbb{N}$ arbitrario.

Proposición 4.15. *Sea* $A \in \mathbb{R}^{(n+1)\times(n+2)}$ *de rango máximo y tal que* $\det(A(j)) > 0$ *para todo* $j = 0, \ldots, n+1$. *Sean* $\lambda, \mu \in \mathbb{R}^{n+2}$ *tales que* $\lambda_j = (-1)^j \cdot det(A(j))$, $\mu_0 = \lambda_0$ *y* $\mu_j = \mu_{j-1} + \lambda_j$. *Entonces* $varsigns(\mu) = n$ *si y solo si para cada* $k = 1, \ldots n$ *par vale:*

$$\sum_{j=0}^{\frac{k}{2}-1} \lambda_{2j} < -\sum_{j=0}^{\frac{k}{2}-1} \lambda_{2j+1} < \sum_{j=0}^{\frac{k}{2}} \lambda_{2j} \tag{4.5.5}$$

La Proposición 4.15 da una condición necesaria y suficiente para que `boundposPolSyst` $varsigns(\mu)+1 = n+1$, sin embargo El Hilany prueba en [25, Th. 1.1] que estas condiciones son suficientes para que exista una matriz C de coeficientes tal que $n_{\mathcal{A}}(C) = n+1$.

Demostración. Por la definición de μ tenemos que $varsigns(\mu) = n$ si y solo si $\sum_{j=0}^{k} \lambda_j < 0$ si k es impar y $\sum_{j=0}^{k} \lambda_j > 0$ si k es par, $k < n+1$. Entonces si k es impar:

$$0 > \sum_{j=0}^{k} (-1)^j \lambda_j = \sum_{j=0}^{\frac{k-1}{2}} \lambda_{2j} + \sum_{j=0}^{\frac{k-1}{2}} \lambda_{2j+1},$$

es decir,

$$-\sum_{j=0}^{\frac{k-1}{2}} \lambda_{2j+1} > \sum_{j=0}^{\frac{k-1}{2}} \lambda_{2j} \tag{4.5.6}$$

Si k es par:

$$0 < \sum_{j=0}^{k} (-1)^j \lambda_j = \sum_{j=0}^{\frac{k}{2}} \lambda_{2j} + \sum_{j=0}^{\frac{k}{2}-1} \lambda_{2j+1}$$

$$-\sum_{j=0}^{\frac{k-2}{2}} \lambda_{2j+1} < \sum_{j=0}^{\frac{k}{2}} \lambda_{2j}. \tag{4.5.7}$$

Tomemos $s < n+1$, s par. Evaluando (4.5.6) en $s-1$:

$$-\sum_{j=0}^{\frac{s-2}{2}} \lambda_{2j+1} > \sum_{j=0}^{\frac{s-2}{2}} \lambda_{2j}, \tag{4.5.8}$$

y evaluando (4.5.7) en s:

$$-\sum_{j=0}^{\frac{s-2}{2}} \lambda_{2j+1} < \sum_{j=0}^{\frac{s}{2}} \lambda_{2j}. \tag{4.5.9}$$

Unificando (4.5.8) y (4.5.9) conseguimos la desigualdad buscada:

$$\sum_{j=0}^{\frac{s-2}{2}} \lambda_{2j} < -\sum_{j=0}^{\frac{s-2}{2}} \lambda_{2j+1} < \sum_{j=0}^{\frac{s}{2}} \lambda_{2j}. \tag{4.5.10}$$

□

Observación 4.16. Notar que la desigualdad (4.5.5) es equivalente a:

$$0 < \sum_{j=0}^{\frac{k}{2}-1} \det(A(2j)) < \sum_{j=0}^{\frac{k}{2}-1} \det(A(2j+1)) < \sum_{j=0}^{\frac{k}{2}} \det(A(2j)). \tag{4.5.11}$$

Para $k = 1, \dots n$ par, usando que $\lambda_j = (-1)^j \det(A(j))$ para $j = 0, \dots, n+1$.

4.5.3. Sobre la cantidad de órdenes

Por último enunciaremos dos Teoremas que nos permiten acotar la cantidad $\vartheta(\mathcal{A})$. Seguiremos asumiendo como hasta ahora que $\mathcal{A} \subset \mathbb{Z}^n$ circuito y $C \in \mathbb{R}^{n\times(n+2)}$ cumplen (2.1.8). C cumple (2.1.5) y es uniforme. $\mathcal{A}$ define una matriz $A \in \mathbb{R}^{(n+1)\times(n+2)}$ de rango máximo y cuyos menores maximales son todos del mismo signo.

Teorema 4.17. *Sea $\mathcal{A} \subset \mathbb{Z}^n$ una configuración de potencias de $n+2$ elementos que forman un circuito y que cumple (2.1.8). Si n es impar obtenemos la siguiente cota como producto de dos factoriales:*

$$\vartheta(\mathcal{A}) \leq \left(\frac{n+3}{2}\right)! \cdot \left(\frac{n+1}{2}\right)! \tag{4.5.12}$$

y existe $\mathcal{A}$ tal que la desigualdad se alcanza.

Demostración. Tomamos $A \in \mathbb{R}^{(n+1)\times(n+2)}$ de rango máximo la matriz definida por $\mathcal{A}$ y $b \in \mathbb{R}^{n+2}$ generador del núcleo de A, $b_j \neq 0$ para $j = 0, \dots, n+1$. Sea $(\sigma, \overline{\sigma})$ un par de órdenes compatibles tales que $\sigma = \overline{\sigma}$. Definamos λ_σ y $\mu_\sigma \in \mathbb{R}^{n+2}$ como $(\lambda_\sigma)_j = b_{\sigma_j}$, $(\mu_\sigma)_0 = (\lambda_\sigma)_0$ y $(\mu_\sigma)_j = (\mu_\sigma)_{j-1} + (\lambda_\sigma)_j$. Como vimos en (4.5.4) para conseguir `boundposPolSyst` máximo es necesario que $varsigns(\lambda)$ sea máximo: $\lambda_\sigma \in \mathbb{R}^{n+2}$ con $varsigns(\lambda_\sigma) = n+1$ y λ_σ es una permutación de las coordenadas de b. Luego, como n es impar, b debe tener $\lfloor \frac{n+2}{2} \rfloor$ coordenadas positivas y $\lceil \frac{n+2}{2} \rceil$ negativas o viceversa.

Si λ_σ es una permutación de las coordenadas de b tal que $varsigns(\lambda_\sigma) = n+1$, entonces λ_σ debe alternar sus signos. Si hay más coordenadas negativas que positivas, $(\lambda_\sigma)_j$ debe ser negativo si j es impar y $(\lambda_\sigma)_j$ positivo si j es par, si hay más positivas es exactamente al revés. Entonces σ debe permutar las coordenadas pares y las impares por separado. Hay $\left(\frac{n+3}{2}\right)!$ formas de permutar las $\lceil \frac{n+2}{2} \rceil = \frac{n+3}{2}$ coordenadas impares y $\left(\frac{n+1}{2}\right)!$ formas de permutar las $\lfloor \frac{n+2}{2} \rfloor = \frac{n+1}{2}$ coordenadas pares. Luego:

$$\vartheta(\mathcal{A}) \leq \left(\frac{n+3}{2}\right)! \cdot \left(\frac{n+1}{2}\right)!.$$

Para ver que la desigualdad se alcanza notemos primero que para dar la configuración $\mathcal{A}$ basta dar el vector $b \in \mathbb{R}^{n+2}$ tal que $b_j \neq 0$ para ningún j y $\sum_{j=0}^{n+1} b_j = 0$. Tomamos M_b la matriz que tiene como filas generadores del núcleo de b^t: $M_b = (\ker(b^t))^t$, entonces $M_b \in \mathbb{R}^{(n+1)\times(n+2)}$ y $rk(M_b) = \dim(\ker(b)) = n+1$. Como b genera el núcleo de M_b entonces $m \in \mathbb{R}^{n+2}$ definido por $m_j = (-1)^j \det(M_b(j))$ es múltiplo de b y como las coordenadas de b son todas no nulas tenemos que todos los menores maximales de M_b son no nulos. Además, como $\sum_{j=0}^{n+2} b_j = 0$, el vector $v = (1, \dots, 1) \in \mathbb{R}^{n+2}$ pertenece al núcleo de b^t. Luego multiplicando M_b por una matriz inversible podemos conseguir A tal que: tiene una primer fila de unos, su rango es máximo y todos sus menores maximales son no nulos. Obtener $\mathcal{A}$ a partir de A es claro.

Ahora, sea $a \in \mathbb{R} \setminus \{0\}$ y n impar. Notemos que $\frac{n+1}{2}$ y $\frac{n+3}{2}$ son números consecutivos y definamos $k = \frac{n+1}{2}$. Definamos $b \in \mathbb{R}^{n+2}$ como $b_j = \frac{a}{k}$ si j es impar y $b_j = -\frac{a}{k+1}$ si j es par. Sea $\sigma \in \mathbb{S}_{n+2}$ arbitraria tal que σ_j es impar si y solo si j es impar. Sea $\mathcal{A} \subset \mathbb{Z}^n$ la configuración asociada a b y $C \in \mathcal{U}_\sigma$. Entonces afirmo que $\texttt{boundposPolSyst}(\mathcal{A}, C) = 1 + varsigns(\mu_\sigma) = n + 1$, con $(\mu_\sigma)_0 = b_{\sigma_0}$ y $(\mu_\sigma)_j = (\mu_\sigma)_{j-1} + b_{\sigma_j}$. En efecto, $b_{\sigma_j} = \frac{a}{k}$ si j es impar y $b_{\sigma_j} = -\frac{a}{k+1}$ si j es par por lo tanto:

$$(\mu_\sigma)_0 = b_{\sigma_0} = -\frac{a}{k+1} < 0.$$

Si $j < n+1$ es impar

$$(\mu_\sigma)_j = \sum_{i=0}^{j} b_{\sigma_i} = \frac{j+1}{2} \cdot \frac{a}{k} - \frac{j+1}{2} \cdot \frac{a}{k+1} = \frac{j+1}{2}\left(\frac{a}{k} - \frac{a}{k+1}\right) > 0.$$

Si $j < n+1$ es par

$$(\mu_\sigma)_j = \sum_{i=0}^{j} b_{\sigma_i} = \frac{j}{2} \cdot \frac{a}{k} - \frac{j+2}{2} \cdot \frac{a}{k+1} = \frac{1}{2} \cdot \frac{(j-2k)\cdot a}{k(k+1)} = \frac{1}{2} \cdot \frac{(j-n-1)\cdot a}{k(k+1)} < 0.$$

Y

$$(\mu_\sigma)_{n+1} = \sum_{i=0}^{n+1} b_{\sigma_i} = 0.$$

Luego $varsigns(\mu_\sigma) = n$ como queríamos ver. Como esto sucede para cualquier $C \in \mathcal{U}_\sigma$ tal que $\sigma \in \mathbb{S}_{n+2}$ cumple σ_j es impar si y solo si j es impar, entonces

$$\vartheta(\mathcal{A}) \geq \#\{\sigma \in \mathbb{S}_{n+2} / \ \sigma_j \text{ es impar } \Leftrightarrow j \text{ es impar}\} = \left(\frac{n+3}{2}\right)! \cdot \left(\frac{n+1}{2}\right)!,$$

por lo que la igualdad en (4.5.12) se alcanza. □

Ejemplo 4.18. En la tabla [16], en la columna `bound=n+1`, se puede observar el valor $\vartheta(\mathcal{A})$ para cada una de las configuraciones de potencias consideradas y la lista de pares de órdenes `S` definida en la Sección 4.3. Es importante tener en cuenta que en esta la lista de pares de órdenes están todos los órdenes $\sigma \in \mathbb{S}_{n+2}$ tales que $\sigma^{op} \notin$ `S`. Esto implica que el experimento se realizó en exactamente la mitad de los pares de órdenes compatibles $(\sigma, \overline{\sigma})$ tales que $\sigma = \overline{\sigma}$.

- Si $n = 3$ el Teorema provee la cota $\vartheta(\mathcal{A}) \leq 3! \cdot 2! = 12$. Si tomamos $\mathcal{A} = \{(2,1,0), (0,1,0), (0,0,1), (2,3,1), (1,4,5)\}$ que corresponde a la configuración derivada $\mathcal{A}_{i,v}$ con $i = 1$ y $v = (1,1,0)$ en la tabla [16], podemos ver que hay exactamente 6 pares de órdenes posibles en `S` que devuelven la cota máxima. Como estos son exactamente la mitad, tenemos exactamente 12 pares de órdenes compatibles en $(\sigma, \overline{\sigma})$ con $\sigma = \overline{\sigma}$. Es decir que para esta configuración $\mathcal{A}$, $\vartheta(\mathcal{A})$ alcanza el valor máximo.
- Si $n = 5$ la cota que provee el Teorema es $\vartheta(\mathcal{A}) \leq 4! \cdot 3! = 144$ Si buscamos la configuración derivada $\mathcal{A}_{i,v} = \{(1,0,0,0,0), (0,1,0,0,0), (1,2,6,1,1), (0,0,0,1,0), (0,0,0,0,1), (2,3,1,2,1), (1,4,5,1,2)\}$ que corresponde a $i = 3$ y $v = (1,2,6,0,1)$ en la tabla [16], hay 72 pares de órdenes que devuelven la cota máxima. De nuevo, como estos son exactamente la mitad, para esta configuración $\mathcal{A}$ hay 144 pares de órdenes compatibles $(\sigma, \overline{\sigma})$ tales que $\sigma = \overline{\sigma}$.

Para el caso n par podemos enunciar una cota con más restricciones.

Teorema 4.19. *Sea $\mathcal{A} \subset \mathbb{Z}^n$ una configuración de potencias de $n+2$ elementos que forman un circuito y que cumple (2.1.8). Si n es par:*

$$\vartheta(\mathcal{A}) \leq 2 \cdot \left(\left(\frac{n+2}{2}\right)!\right)^2 - \left(\frac{n+2}{2}\right)! \cdot \left(\frac{n}{2}\right)! \tag{4.5.13}$$

Demostración. Tomamos $A \in \mathbb{R}^{(n+1)\times(n+2)}$ de rango máximo la matriz definida por $\mathcal{A}$ y $b \in \mathbb{R}^{n+2}$ generador del núcleo de A, $b_j \neq 0$ para $j = 0, \ldots, n+1$. Sea $(\sigma, \overline{\sigma})$ un par de órdenes compatibles tales que $\sigma = \overline{\sigma}$. Definamos λ_σ y $\mu_\sigma \in \mathbb{R}^{n+2}$ como $(\lambda_\sigma)_j = b_{\sigma_j}$, $(\mu_\sigma)_0 = (\lambda_\sigma)_0$ y $(\mu_\sigma)_j = (\mu_\sigma)_{j-1} + (\lambda_\sigma)_j$. Como vimos en (4.5.4) para conseguir `boundposPolSyst` máximo es necesario que $varsigns(\lambda)$ sea máximo: $\lambda_\sigma \in \mathbb{R}^{n+2}$ con $varsigns(\lambda_\sigma) = n+1$ y λ_σ es una permutación de las coordenadas de b. Luego, como n es par, b debe tener $\frac{n+2}{2}$ coordenadas positivas y $\frac{n+2}{2}$ negativas.

Si λ_σ es una permutación de las coordenadas de b tal que $varsigns(\lambda_\sigma) = n+1$, entonces λ_σ debe alternar sus signos. Entonces debe suceder que las coordenadas pares son positivas y las impares negativas o viceversa, pero una vez elegido el signo de $(\lambda_\sigma)_0$ quedan definidos los signos de todas las coordenadas. Entonces σ debe permutar las coordenadas positivas y las negativas por separado o cambiar el lugar de todas las positivas con todas las negativas. Hay $\left(\frac{n+2}{2}\right)!$ formas de permutar las $\frac{n+2}{2}$ coordenadas positivas y $\left(\frac{n+2}{2}\right)!$ formas de permutar las $\frac{n+2}{2}$ coordenadas negativas. Además hay dos formas de ordenar esto: empezando con positivos o con negativos, entonces:

$$\begin{aligned}\vartheta(\mathcal{A}) &\leq \#\{\sigma \in \mathbb{S}_{n+2} / \ \sigma_j \text{ es impar } \Leftrightarrow j \text{ es impar o } \sigma_j \text{ es par } \Leftrightarrow j \text{ es impar}\} \\ &\leq 2 \cdot \left(\left(\frac{n+2}{2}\right)!\right)^2. \end{aligned} \tag{4.5.14}$$

Por otro lado, σ debe ser tal que $varsigns(\mu_\sigma) = n+1$. Sea $(\sigma, \overline{\sigma})$ par de órdenes compatibles tal que $\sigma = \overline{\sigma}$ y $varsigns(\lambda_\sigma) = n+1$. Sea k tal que $|b_k| \geq |b_j|$ para todo $j \in [n+2]$ y supongamos que b_k es positivo, si no el argumento que sigue es análogo. Si $\sigma_0 = k$ entonces $b_{\sigma_0} > 0$, $b_{\sigma_1} < 0$ y $b_{\sigma_2} > 0$:

$$\begin{aligned}&(\mu_\sigma)_0 = (\lambda_{sigma})_0 = b_k > 0, \\ &(\mu_\sigma)_1 = b_k + b_{\sigma_1} > 0 \text{ porque } |b_k| \geq |b_j| \text{ para todo } j = 0, \ldots, n+1, \\ &(\mu_\sigma)_2 = b_k + b_{\sigma_1} + b_{\sigma_2} > 0 \text{ porque } b_{\sigma_2} > 0.\end{aligned}$$

Luego $varsigns(\mu_\sigma) \leq n-1$.

Entonces no alcanza con que σ intercale coordenadas positivas y negativas. Es necesario también que no comience con el elemento de mayor módulo. Hay en total $\left(\frac{n+2}{2}\right)! \cdot \left(\frac{n}{2}\right)!$ órdenes que comienzan con b_k y por lo tanto:

$$\vartheta(\mathcal{A}) \leq 2 \cdot \left(\left(\frac{n+2}{2}\right)!\right)^2 - \left(\frac{n+2}{2}\right)! \cdot \left(\frac{n}{2}\right)!.$$

□

Ejemplo 4.20. Observamos de nuevo la tabla [16], teniendo en cuenta que la cantidad de órdenes considerados en ella son exactamente la mitad de los que considera el Teorema 4.19. Mantenemos la misma notación que en el ejemplo 4.18.

- Si $n = 2$, la cota del Teorema 4.19 es $\vartheta(\mathcal{A}) \leq 2 \cdot (2!)^2 - 2! \cdot 1! = 6$. Tomemos la configuración de potencias $\mathcal{A} = \{(1,0),(3,1),(2,3),(1,2)\}$ que corresponde a la configuración derivada con $i = 2$ y $v = (3,0)$ en la Tabla. La cantidad de pares de órdenes $(\sigma, \overline{\sigma}) \in \texttt{S}$ para los que $\texttt{boundposPolSyst} = n+1$ es 2, entonces la cantidad de pares $(\sigma, \overline{\sigma})$ tales que $\sigma = \overline{\sigma}$ que devuelven cota máxima es $4 < 6$.

- Para $n = 4$, tenemos que $\vartheta(\mathcal{A}) \leq 2 \cdot (3!)^2 - 3! \cdot 2! = 60$. Tomemos la configuración derivada $\mathcal{A}_{i,v} = \{(1,2,1,0),\ (0,1,0,0),\ (0,0,1,0),\ (0,0,0,1),\ (2,3,1,2),\ (1,4,5,2)\}$ que corresponde a $i = 1$ y $v = (0,2,1,0)$. En la columna `bound=5` se observan 12 pares de órdenes en `S`, lo que corresponde a 24 pares de órdenes con $\sigma = \overline{\sigma}$. Entonces $\vartheta(\mathcal{A}) = 24 < 60$.

En la tabla [16] se puede observar que, para $n = 2$ el máximo valor que alcanza $\vartheta(\mathcal{A})$ es 4 y para $n = 4$ el máximo es 24. Ambos valores están muy por debajo de la cota que provee el Teorema 4.19. Además, la construcción del sistema $\mathcal{A}$ para el que $\vartheta(\mathcal{A})$ es máximo para n impar depende fuertemente del hecho de que la cantidad de coordenadas pares e impares son distintas y consecutivas (en particular coprimas). Esta construcción no puede extenderse para n par, al menos no naturalmente. Luego para n par no poseemos una construcción que devuelva la igualdad de la cota del Teorema y en los ejemplos considerados no hay ninguno para el que el valor de $\vartheta(\mathcal{A})$ alcance la cota.

Completamos esta tesis planteando el siguiente Problema:

Problema Sea n par. ¿Es posible dar un circuito $\mathcal{A} \subset \mathbb{Z}^n$ tal que $\vartheta(\mathcal{A})$ alcance la cota que da el Teorema 4.19? O, en cambio ¿Es posible mejorar la cota para $\vartheta(\mathcal{A})$ que brinda este Teorema?

A. El programa `OptimalDescartesRuleOfSigns`

El programa contiene las tres funciones necesarias para el cálculo de la cota: `CheckHypothesis(matrix A, matrix C, vector kerA)` y `posLinearComb(vector v, vector w)` funciones auxiliares y `boundpoPolSyst(list configA, matrix C)` la función principal de esta tesis.

Para poder utilizar cualquiera de estas funciones es necesario primero cargar el programa desde una consola. El código tal cual se presenta en este apéndice puede encontrarse disponible para descargar en [15]. El archivo `OptimalDescartesRuleOfSigns.sing` debe guardarse en el directorio de trabajo, por ejemplo `usr/workpath/OptimalDescartesRuleOfSigns.sing`, iniciar SINGULAR y cargar el archivo con el comando ">":

```
> "usr/workpath/OptimalDescartesRuleOfSigns.sing"
```

A continuación el programa completo:

```
1  //Esta funcion decide si existe z de la forma kv+w que
       tenga todas sus coordenadas positivas para algun k, con
       v y w dados.
2  proc posLinearComb(vector v, vector w){
3    list Geq; //La lista -w[i]/v[i] cuando v[i]>0
4    list Leq; //La lista -w[i]/v[i] cuando v[i]<0
5    int n=size(v);
6    int verifies=1;
7
8    //Si z=kv+w entonces kv[i]+w[i]>0 si y solo si
9    //[(k>-w[i]/v[i] y v[i]>0) o (k<-w[i]/v[i] y v[i]<0)]
10   for (int i=1; i<=n; i++){
11     if(v[i]>0){
12       Geq=Geq + list(number(-w[i])/number(v[i]));
13     } else{
14       if(v[i]<0){
15       Leq=Leq+list(number(-w[i])/number(v[i]));
16       } else{
17         if(v[i]==0 & w[i]<=0){
18           verifies=0;
19         }
20       }
21     }
22   }
23
```

```
24    //Decidimos si existe k<Geq[i] y k>Leq[i] para todo i
25    //Si cualquiera de las dos listas esta vacia y la
       condicion (v[i]==0 & w[i]<=0) no se satisface, siempre
       existe tal k
26    if(size(Leq) & size(Geq)){
27      def UpperBound=min(Leq[1.. size(Leq)]);
28      def LowerBound=max(Geq[1.. size(Geq)]);
29
30      if( LowerBound < UpperBound){
31        verifies=verifies & 1;
32      } else{
33        verifies=0;
34      }
35    }
36    return(verifies);
37  }
38
39
40  //Esta funcion decide si las hipotesis del teorema se
       cumplen
41  proc CheckHypothesis(matrix A, matrix C, vector kerA,
       matrix kerC){
42    int check_hypothesis=1;
43    //Las dimensiones deben ser apropiadas para el sistema
44    if((nrows(A)+1)!=ncols(A)){
45      check_hypothesis=0;
46      "Las dimensiones de configA no son correctas";
47    }
48    if(check_hypothesis){
49      if((nrows(C)+2)!=ncols(C)){
50        check_hypothesis=0;
51        "Las dimensiones de C no son correctas";
52      }
53    }
54      //deben coincidir las dimensiones de A y C
55    if(check_hypothesis){
56      if(ncols(A)!=ncols(C)){
57        check_hypothesis=0;
58        "Las dimensiones de A y C no coinciden";
59      }
60    }
61    //El rango de A debe ser maximo
62    if(check_hypothesis){
63      if(nrows(A)!=rank(A)){
64        check_hypothesis=0;
65        "El rango de A no es maximo";
66      }
```

```
67    }
68    //El rango de C debe ser maximo
69    if(check_hypothesis){
70      if(nrows(C)!=rank(C)){
71        check_hypothesis=0;
72        "El rango de C no es maximo";
73      }
74    }
75    //configA debe ser un circuito: todas las coordenadas de
       ker(A) deben ser distintas de 0
76    for(int i=1; i<=ncols(A); i++){
77      if(check_hypothesis & (kerA[i]==0)){
78        check_hypothesis=0;
79        "El sistema no esta soportado en un circuito";
80      }
81    }
82    //El origen debe pertenecer al cono positivo generado por
        las columnas de C
83    if(check_hypothesis){
84      def M=module(C);
85      //defino v y w dos generadores del nucleo de C
86      def v=kerC[1];
87      def w=kerC[2];
88      check_hypothesis=0;
89      if( posLinearComb(v,w)){
90        check_hypothesis=1;
91      } else {
92        if( posLinearComb(-v,-w)){
93        check_hypothesis=1;
94        }
95      }
96      if(check_hypothesis==0){
97        "El origen debe pertenecer al cono positivo generado
       por las columnas de C";
98      }
99    }
100   //En este punto check_hypothesis=1 si y solo si todas las
        hipotesis se cumplen, si no es 1.
101   return(check_hypothesis);
102 }
103
104 //Esta es la funcion principal
105 proc boundposPolSyst(list configA, matrix C){
106 //Cargo librerias
107 LIB "matrix.lib";
108 LIB "rootsur.lib";
109
```

```
110 int TheBound=-1;
111 //Defino A la matriz de nx(n+2) con la primer fila de unos
       y tal que la submatriz restante tiene como columnas los
       vectores de configA en orden
112 for(int i=1; i<=size(configA);i++){
113     configA[i]=transpose(concat(matrix(1),transpose(matrix(
       configA [i]))));
114 }
115 def A=concat(configA[1.. size(configA)]);
116 def kerA=syz(A)[1];
117 matrix kerC=syz(C);
118
119 //Si las hipotesis se cumplen vamos a hallar una cota
       superior para la cantidad de soluciones positivas del
       sistema.
120 if( CheckHypothesis(A, C, kerA, kerC) ){
121
122   //Primero necesitamos calcular un orden y un orden
       estricto para C
123   int ColsC=ncols(C);
124   int RowsC=nrows(C);
125   matrix D=diag(0,ColsC);
126   int n=nrows(kerC);
127   int d;
128   intvec VectorGeqThan;
129
130   //Defino una matriz D tal que D[i,j]=1 si y solo si det(
       kerC[i],kerC[j])>0
131   for(int i=1; i< n;i++){
132     for(int j=i+1; j<=n; j++){
133       d=int(kerC[i,1]*kerC[j,2]-kerC[i,2]*kerC[j,1]);
134       if(d>0){
135         D[i,j]=1;
136       }else{
137         if(d<0){
138         D[j,i]=1;
139         }
140       }
141     }
142   }
143   //Defino VectorGeqThan[i]=k si hay exactamente k columnas
       del ker(C) con det(kerC[i],kerC[j])>0
144   for(int i=1; i<=ColsC; i++){
145     VectorGeqThan[i]=0;
146     for(int j=1; j<=ColsC; j++){
147       VectorGeqThan[i]=VectorGeqThan[i]+int(D[i,j]);
148     }
```

```
149   }
150   //Ahora definimos los ordenes
151   intvec sigma;
152   intvec sigma_barra;
153   int LastMin=-1;
154   int MaxVectorGeqThan;
155   int SizeVectorGeqThan;
156   MaxVectorGeqThan=max(VectorGeqThan[1..size(VectorGeqThan)
      ]);
157   SizeVectorGeqThan=size(VectorGeqThan);
158   int l=1; //es el contador de coordenadas de sigma
159   int m=1;//es el contador de coordenadas de sigma_barra
160   int k; //el indice del siguiente vector en el orden
161   int LookForMin;
162   while(LastMin != MaxVectorGeqThan){
163     LookForMin=MaxVectorGeqThan+1;
164     k=0;
165     //Encuentra el minimo vector no ordenado
166     for (int i=1; i<=SizeVectorGeqThan; i++){
167       if(VectorGeqThan[i]<LookForMin & VectorGeqThan[i]>
      LastMin){
168         LookForMin=VectorGeqThan[i];
169         k=i;
170       }
171     }
172     //Pongo el vector al final del orden y del orden
      estricto
173     sigma[l]=k;
174     sigma_barra[m]=k;
175     m++;
176     l++;
177     LastMin=LookForMin;
178     //Pongo todos los vectores que tengan el mismo
      VectorGeqThan[i] a continuacion en el orden
179     for (int i=1; i<=size(VectorGeqThan); i++){
180       if(i!=k & VectorGeqThan[i]==LookForMin){
181         sigma[l]=i;
182         l++;
183       }
184     }
185   }
186
187   //Ahora defino mu una lista con las sumas parciales de
      kerA respecto al orden
188   list mu;
189   int i=1;
190   int total = size(sigma_barra);
```

```
191     for(int j=1; j<=total; j++){
192       //j es el indice del orden
193       //i es el indice del orden estricto
194       if(j==1){
195         mu[1]=int(kerA[sigma_barra[1]]);
196       } else{
197         mu[j]=mu[j-1]+int(kerA[sigma_barra[j]]);
198       }
199       i++;
200
201       //Si j no es el ultimo del orden estricto, sumo las
      coordenadas de kerA hasta el siguiente lugar del orden
      estricto
202       if(j!=total){
203         while (sigma_barra[j+1]!=sigma[i]){
204           mu[j]=mu[j]+int(kerA[sigma[i]]);
205           i++;
206         }
207       } else {
208         //si j es el ultimo del orden estricto sumo hasta el
      ultimo lugar del orden
209         while (i!=size(sigma)+1){
210           mu[j]=mu[j]+int(kerA[sigma[i]]);
211           i++;
212         }
213       }
214     }
215     //La cota buscada es la variacion de signos de mu +1
216     TheBound=varsigns(mu)+1;
217   } else{
218       "Las hipotesis no se cumplen";
219     }
220     return (TheBound);
221   }
```

B. Las extensiones al programa boundposPolSyst

Para poder utilizar cualquiera de estas funciones es necesario primero cargar el programa desde una consola. El código tal cual se presenta en este apéndice puede encontrarse disponible para descargar en [15]. El archivo `OptimalDescartesRuleOfSigns - No Order.sing` debe guardarse en el directorio de trabajo, por ejemplo `usr/workpath/OptimalDescartesRuleOfSigns - No Order.sing`, iniciar SINGULAR y cargar el archivo con el comando ">":

```
> "usr/workpath/OptimalDescartesRuleOfSigns - No Order.sing"
```

```
1  //Esta funcion hace una lista con todas las cotas posibles
       para la misma configuracion de potencias
2  proc boundsForAllOrders(list configA, list S){
3    LIB "matrix.lib";
4    LIB "rootsur.lib";
5    list AllTheBounds;
6
7    //Defino la matriz A que tiene como columnas un 1 y el
       vector de la configuracion
8    for(int i=1; i<=size(configA);i++){
9      configA[i]=transpose(concat(matrix(1),transpose(matrix(
       configA [i]))));
10   }
11   def A=concat(configA[1.. size(configA)]);
12   def kerA=syz(A)[1];
13
14   //Chequeo las hipotesis para A
15   int check_hipotesis=1;
16   if(nrows(A)!=rank(A)){
17     check_hipotesis=0;
18   }
19   for(int j=1; j<=ncols(A); j++){
20     if(kerA[j]==0){
21       check_hipotesis=0;
22     }
23   }
24
25   if( check_hipotesis ==0){
26     AllTheBounds[1]=-1;
27   } else {
```

```
28      intvec sigma_barra;
29      intvec sigma;
30      int TheBound;
31      list vectorBound;
32      def n=size(kerA);
33      int k;
34      int maxIndex;
35
36      for(int i=1; i<=size(S); i++){
37        sigma=S[i][1];
38        sigma_barra=S[i][2];
39
40        list mu;
41        k=1;
42        maxIndex = size(sigma_barra);
43
44        for(int j=1; j<=maxIndex; j++){
45          if(j==1){
46            mu[1]=int(kerA[sigma_barra[1]]);
47          } else{
48            mu[j]=mu[j-1]+int(kerA[sigma_barra[j]]);
49          }
50          k++;
51
52          if(j!=maxIndex){
53            while (sigma_barra[j+1]!=sigma[k]){
54              mu[j]=mu[j]+int(kerA[sigma[k]]);
55              k++;
56            }
57          } else {
58            while (k!=size(sigma)+1){
59              mu[j]=mu[j]+int(kerA[sigma[k]]);
60              k++;
61            }
62          }
63        }
64        TheBound = varsigns(mu)+1;
65
66        vectorBound[1] = TheBound;
67        vectorBound[2] = sigma;
68        vectorBound[3] = sigma_barra;
69
70        AllTheBounds[i] = vectorBound;
71      }
72    }
73    return (AllTheBounds);
74  }
```

```
75
76 //Esta funcion calcula y guarda en archivos txt la cantidad
       de veces que el programa devuelve cada cota para ciclos
       sobre una configuracion dada y una lista de ordenes
77 proc CountingBounds(list config, int dimen, list S, int k){
78   LIB "matrix.lib";
79   LIB "rootsur.lib";
80
81   //defino las variables para guardar los archivos
82   string name="CotasPorDireccion Dim ";
83   string s; //s es el string que usa el script para guardar
       el output
84   string sPrima;
85   link lPrima;
86   link l;
87   string encabezado;
88   encabezado="posicion, vector, #cota=1, #cota=2";
89
90   //genero el encabezado y link para CotasPorDireccion
91   encabezado=string(encabezado+ ", #cota=" , dimen+1) + ",
      sin cota";
92   s=string(name,dimen);
93   l=":w " + s + ".txt";
94   write(l,encabezado);
95   //paso el link de write a append
96   l=":a " + s + ".txt";
97
98   //genero el encabezado y link para OrdenesConCotaMax
99   sPrima= string("OrdenesConCotaMax Dim ", dimen);
100  lPrima=":w " + sPrima + ".txt";
101  write(lPrima,"posicion, vector, orden");
102  //paso el link de write a append
103  lPrima=":a " + sPrima + ".txt";
104
105  list configA; //la configuracion que generamos en cada
      paso
106  list alpha; //la lista de cotas que devuelve la funcion
107  intvec Delta;//cuenta la cantidad de apariciones de cada
      cota
108  vector v;//el vector que define la configuracion a
      utilizar
109  list VectorCotas; //guarda (i,v,Delta)
110  list VectorOrdenes;//guarda (i,v,sigma)
111  matrix A; //La matriz A asociada a la configuracion
112
113  //Las variables auxiliares que necesito para recorrer los
       v
```

```
114   int total = k^dimen;
115   int dividendo;
116   intvec vCero;
117   intvec vCreado;
118   vCero[dimen]=0;//genera un vector de ceros de tamano
       dimen
119
120   for(int i=1; i<=dimen+2; i++){
121
122     //uso los ciclos for y while que siguen para definir v
123     for (int count=1; count<=total; count++){
124       int lugar=1;
125       vCreado=vCero;
126       dividendo=count;
127       while((lugar<= dimen) && (dividendo>0)  ){
128         // toma resto y div es division entera
129         vCreado[lugar]=dividendo%k;
130         dividendo= dividendo div k;
131         lugar=lugar+1;
132       }
133       v=[vCreado[1..dimen]];
134
135       //desde aca mueve el ultimo vector de la config en la
      direccion v
136       configA=config;
137       configA[i]=config[i]+v;
138       alpha =boundsForAllOrders(configA,S);
139
140       if(size(alpha[1])!=1){
141         Delta=0,0,0,0;
142         VectorCotas[1]=i;
143         VectorCotas[2]=v;
144         for (int j=1; j<=size(alpha); j++){
145           Delta[alpha[j][1]]=Delta[alpha[j][1]]+1;
146           if(alpha[j][1]==dimen+1){
147             VectorOrdenes[1]=i;
148             VectorOrdenes[2]=v;
149             VectorOrdenes[3]=alpha[j][2];
150             VectorOrdenes[4]=alpha[j][3];
151             write(lPrima,VectorOrdenes);
152           }
153         }
154         VectorCotas[3]=Delta;
155         write(l,VectorCotas);
156       }
157     }
158   }
```

```
159   close(l);
160   close(lPrima);
161 }
162
163 //Funcion para calcular todos los pares de ordenes en los
      que el orden y el orden estricto coinciden para una
      dimension n dada
164 proc UnparalleledOrders(int n){
165   LIB "qmatrix.lib";
166   def S=SymGroup (n);
167   list oneOrder;//oneOrder es un orden posible
168   list Orders;//es la lista de todos los oneOrder posibles,
        salvo orientacion inversa
169   //cada lugar de la lista Orders tiene dos elementos: el
      orden y el orden estricto
170
171   intvec v;
172   intvec w;
173   int repeat=0;
174
175   for(int i=1; i<= nrows(S); i++){
176     repeat=0;
177     v=intvec(S[i,1.. ncols(S)]);
178
179     for(int j=1; j<=size(v); j++){
180       w[j]=v[size(v)+1-j];
181     }
182
183     for(int k=1; k<=size(oneOrder); k++){
184       if(oneOrder[k]==w){
185         repeat=1;
186       }
187     }
188
189     if(repeat==0){
190       oneOrder[i] = v;
191       Orders[i]= list(oneOrder[i])+list(oneOrder[i]);
192     }
193   }
194
195   return(Orders);
196 }
197
198 //Funcion que calcula todos los pares de ordenes posibles
      para una dimension dada.
199 proc AllOrders(int n){
200   LIB "qmatrix.lib";
```

```
201   def S=SymGroup (n);
202   list OrdersSn;
203   //OrdersSn calcula los posibles ordenes up to complete
       reversal
204
205   //auxiliares para OrdersSn
206   intvec v;
207   intvec w;
208   int repeat=0;
209
210   for(int i=1; i<= nrows(S); i++){
211     repeat=0;
212     v=intvec(S[i,1.. ncols(S)]);
213
214     for(int j=1; j<=size(v); j++){
215       w[j]=v[size(v)+1-j];
216     }
217
218     for(int k=1; k<=size(OrdersSn); k++){
219       if(OrdersSn[k]==w){
220         repeat=1;
221       }
222     }
223
224     if(repeat==0){
225       OrdersSn[i] = v;
226     }
227   }
228   //Elimino las variables auxiliares para poder
       reutilizarlas
229   kill v,w,repeat,i,j,k;
230
231
232   //ahora quiero definir los ordenes en el cociente
233   //voy a hacer una tira de 0 y 1: 1 si es distinto del
       lugar anterior, 0 si es igual
234   list ZerosAndOnes;
235   intvec v;
236   int k;
237   int y;
238   v[n]=0;
239
240   for(int x=1; x< 2^(n-1) ; x++){
241     k=2;
242     v[1]=1;
243     y=x;
244     while (y>0){
```

```
245         v[k]=y%(2);
246         y=y div 2;
247         k++;
248       }
249       ZerosAndOnes [x]= v;
250     }
251     kill k,v,y,x;
252
253     //Ahora para cada vector del OrdersSn defino todas las
        asignaciones del orden cociente
254     list FinalOrders;
255     intvec StrictOrder;
256     int k=1;
257     int m=1;
258
259     for(int i=1; i<= size(OrdersSn); i++){
260       for (int j=1; j<= size(ZerosAndOnes); j++){
261         for(int l=1; l<= n;l++){
262           if(ZerosAndOnes[j][l]){
263             StrictOrder[m]=OrdersSn[i][l];
264             m++;
265           }
266         }
267         FinalOrders[k]= list(OrdersSn[i])+list(StrictOrder);
268         k++;
269         StrictOrder=0;
270         m=1;
271       }
272     }
273
274     return(FinalOrders);
275 }
```

Bibliografía

[1] A. A. Albert, «An inductive proof of Descartes' rule of signs,» English, *Am. Math. Mon.*, vol. 50, págs. 178-180, 1943, ISSN: 0002-9890. DOI: 10.2307/2302399.

[2] A. Albouy e Y. Fu, «Some remarks about Descartes' rule of signs,» English, *Elem. Math.*, vol. 69, n.º 4, págs. 186-194, 2014, ISSN: 0013-6018. DOI: 10.4171/EM/262.

[3] B. Anderson, J. Jackson y M. Sitharam, «Descartes' rule of signs revisited,» English, *Am. Math. Mon.*, vol. 105, n.º 5, págs. 447-451, 1998, ISSN: 0002-9890. DOI: 10.2307/3109807.

[4] M. Baker y O. Lorscheid, «Descartes' rule of signs, Newton polygons, and polynomials over hyperfields,» English, *J. Algebra*, vol. 569, págs. 416-441, 2021, ISSN: 0021-8693. DOI: 10.1016/j.jalgebra.2020.10.024.

[5] S. Barone y S. Basu, «Refined bounds on the number of connected components of sign conditions on a variety,» English, *Discrete Comput. Geom.*, vol. 47, n.º 3, págs. 577-597, 2012, ISSN: 0179-5376. DOI: 10.1007/s00454-011-9391-3.

[6] M. Bartolozzi y R. Franci, «The sign rule from the enunciation by R. Descartes (1637) to the demonstration by C. F. Gauss (1828),» Italian, *Arch. Hist. Exact Sci.*, vol. 45, n.º 4, págs. 335-374, 1993, ISSN: 0003-9519. DOI: 10.1007/BF01886631.

[7] S. Basu, R. Pollack y M.-F. Roy, *Algorithms in real algebraic geometry*, English. Berlin: Springer, 2006, vol. 10, págs. x + 662, ISBN: 3-540-33098-4. DOI: 10.1007/3-540-33099-2.

[8] ——, «On the number of cells defined by a family of polynomials on a variety,» English, *Mathematika*, vol. 43, n.º 1, págs. 120-126, 1996, ISSN: 0025-5793. DOI: 10.1112/S0025579300011621.

[9] M. Bensimhoun, «Historical account and ultra-simple proofs of Descartes's rule of signs, De Gua, Fourier, and Budan's rule,» 2016. arXiv: 1309.6664 [math.HO].

[10] F. Bihan y A. Dickenstein, «Descartes' rule of signs for polynomial systems supported on circuits,» *Int. Math. Res. Not. IMRN*, n.º 22, págs. 6867-6893, 2017, ISSN: 1073-7928. DOI: 10.1093/imrn/rnw199. URL: https://doi.org/10.1093/imrn/rnw199.

[11] F. Bihan, A. Dickenstein y J. Forsgård, «Optimal Descartes' rule of signs for systems supported on circuits,» *Mathematische Annalen*, vol. 381, págs. 1283-1307, 2021.

[12] F. Bihan, F. Santos y P.-J. Spaenlehauer, «A Polyhedral Method for Sparse Systems with Many Positive Solutions,» *SIAM Journal on Applied Algebra and Geometry*, vol. 2, abr. de 2018. DOI: 10.1137/18M1181912.

[13] F. Bihan y F. Sottile, «New fewnomial upper bounds from Gale dual polynomial systems,» English, *Mosc. Math. J.*, vol. 7, n.º 3, págs. 387-407, 2007, ISSN: 1609-3321.

[14] E. Britannica, *Equations*, Edinburgh, 1824.

[15] A. Busca Canosa, https://drive.google.com/drive/folders/190uEG783Mx6kHqRincl9dou3SF0XUqMe?usp=sharing.

[16] ——, URL: https://anamilla-busca-canosa.shinyapps.io/ConteoCasosPorDimension/.

[17] W. Chang, J. Cheng, J. Allaire y col., *shiny: Web Application Framework for R*, R package version 1.7.0, 2021. URL: https://CRAN.R-project.org/package=shiny.

[18] H. Cheriha, Y. Gati y V. P. Kostov, «Degree 5 polynomials and Descartes' rule of signs,» English, *Acta Univ. M. Belii, Ser. Math.*, vol. 28, págs. 3-21, 2020, ISSN: 1338-712X.

[19] ——, «Descartes' rule of signs, Rolle's theorem and sequences of compatible pairs,» English, *Stud. Sci. Math. Hung.*, vol. 57, n.º 2, págs. 165-186, 2020, ISSN: 0081-6906. DOI: 10.1556/012.2020.57.2.1463.

[20] ——, «On Descartes' rule for polynomials with two variations of signs,» English, *Lith. Math. J.*, vol. 60, n.º 4, págs. 456-469, 2020, ISSN: 0363-1672. DOI: 10.1007/s10986-020-09491-9.

[21] G. Craciun, L. D. García-Puente y F. Sottile, «Some geometrical aspects of control points for toric patches,» English, en *Mathematical methods for curves and surfaces. 7th international conference, MMCS 2008, Tønsberg, Norway, June 26–July 1, 2008. Revised selected papers*, Berlin: Springer, 2010, págs. 111-135, ISBN: 978-3-642-11619-3. DOI: 10.1007/978-3-642-11620-9_9.

[22] D. R. Curtiss, «Recent Extentions of Descartes' Rule of Signs,» *Annals of Mathematics*, vol. 19, n.º 4, págs. 251-278, 1918, ISSN: 0003486X. URL: http://www.jstor.org/stable/1967494.

[23] *Sur le nombre des racines Réelles ou Imaginaires, Réelles positives ou Réelles négatives, qui se trouvent dans les équations de tous les degrés.* sec. mémoires, Paris: Imprimerie Royale, 1741, págs. 72-95.

[24] W. Decker, G.-M. Greuel, G. Pfister y H. Schönemann, *SINGULAR 4-2-1 — A computer algebra system for polynomial computations*, http://www.singular.uni-kl.de, 2021.

[25] B. El Hilany, «Characterization of circuits supporting polynomial systems with the maximal number of positive solutions,» English, *Discrete Comput. Geom.*, vol. 58, n.º 2, págs. 355-370, 2017, ISSN: 0179-5376. DOI: 10.1007/s00454-017-9897-4.

[26] G. Enestr̈om, *Bibliotheca Mathematica*, German. Druck Und Verlag Von B. G Teubner, Leipzig, 1906, pág. 307. URL: https://archive.org/details/bibliothecamath01enesgoog/page/n25/mode/2up.

[27] E. Feliu y M. L. Telek, *On generalizing Descartes' rule of signs to hypersurfaces*, 2021. arXiv: 2107.10002 [math.AG].

[28] C. F. Gauß, «Beweis eines algebraischen Lehrsatzes.,» German, *J. Reine Angew. Math.*, vol. 3, págs. 1-4, 1828, ISSN: 0075-4102. DOI: 10.1515/crll.1828.3.1.

[29] D. J. Grabiner, «Descartes' rule of signs: Another construction,» English, *Am. Math. Mon.*, vol. 106, n.º 9, págs. 854-856, 1999, ISSN: 0002-9890. DOI: 10.2307/2589619.

[30] G.-M. Greuel, G. Pfister y H. Schönemann, *Singular - A Computer Algebra System for Polynomial Computations - Manual*. University of Kaiserslautern, Department of Mathematics y Centre for Computer Algebra, 2010.

[31] I. Itenberg y M.-F. Roy, «Multivariate Descartes' rule,» English, *Beitr. Algebra Geom.*, vol. 37, n.º 2, págs. 337-346, 1996, ISSN: 0138-4821.

[32] V. P. Kostov y B. Z. Shapiro, «Polynomials, sign patterns and Descartes' rule,» English, *Acta Univ. M. Belii, Ser. Math.*, vol. 27, págs. 51-61, 2019, ISSN: 1338-712X.

[33] V. P. Kostov, «Descartes' rule of signs and moduli of roots,» English, *Publ. Math.*, vol. 96, n.º 1-2, págs. 161-184, 2020, ISSN: 0033-3883. DOI: 10.5486/PMD.2020.8640.

[34] ——, «Moduli of roots of hyperbolic polynomials and Descartes' rule of signs,» English, en *Constructive theory of functions. Proceedings of the 13th international conference, Sozopol, Bulgaria, June 2–8, 2019. Dedicated to the memory of Blagovest Sendov*, Sofia: Prof. Marin Drinov Academic Publishing House, 2020, págs. 131-146, ISBN: 978-619-245-060-1.

[35] ——, «Polynomials, sign patterns and Descartes' rule of signs.,» English, *Math. Bohem.*, vol. 144, n.º 1, págs. 39-67, 2019, ISSN: 0862-7959. DOI: 10.21136/MB.2018.0091-17.

[36] P. V. Krishnaiah, «A Simple Proof of Descartes' Rule of Signs,» *Mathematics Magazine*, vol. 36, n.º 3, págs. 190-190, 1963, ISSN: 0025570X, 19300980. URL: http://www.jstor.org/stable/2688458.

[37] J. C. Lagarias y T. J. Richardson, «Multivariate descartes rule of signs and sturmfels's challenge problem,» English, *The Mathematical Intelligencer*, vol. 19, n.º 3, págs. 9-15, 1997. DOI: 10.1007/BF03025343.

[38] E. Laguerre, *Oeuvres de Laguerre publées sous les auspices de l'Académie des Sciences par MM. Ch. Hermite, H. Poincaré et E. Rouché. Tome I. Algèbre. Calcul intégral.* French, Paris: Gauthier-Villars et Fils. XV + 471 S. gr. 8°. [Darboux Bull. (2) 22, 304-310, Anzeige von Borel.] (1898). 1898.

[39] T. Y. Li y X. Wang, «On multivariate Descartes' rule – a counterexample,» English, *Beitr. Algebra Geom.*, vol. 39, n.º 1, págs. 1-5, 1998, ISSN: 0138-4821.

[40] S. Müller, E. Feliu, G. Regensburger, C. Conradi, A. Shiu y A. Dickenstein, «Sign conditions for injectivity of generalized polynomial maps with applications to chemical reaction networks and real algebraic geometry,» English, *Found. Comput. Math.*, vol. 16, n.º 1, págs. 69-97, 2016, ISSN: 1615-3375. DOI: 10.1007/s10208-014-9239-3.

[41] P. Pedersen, M.-F. Roy y A. Szpirglas, «Counting real zeros in the multivariate case,» English, en *Computational algebraic geometry. Papers from a conference, held in Nice, France, April 21–25, 1992*, Boston: Birkhäuser, 1993, págs. 203-224, ISBN: 0-8176-3678-1.

[42] G. Polya y G. Szegö, *Problems and theorems in analysis. Vol. II: Theory of functions, zeros, polynomials, determinants, number theory, geometry. Translation by C. E. Billigheimer*, English. Springer, Cham, 1976, vol. 216.

[43] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020. URL: https://www.R-project.org/.

[44] *Démonstration de la règle de Descartes, pour connoitre le nombre des racines affirmatives et négatives qui peuvent se trouver dans les équations*, Berlin, 1756, págs. 292-299.

[45] J. A. Segner, *Dissertatio epistolica, qua regulam harriotti...* University of Jena, 1728.

[46] D. E. Smith y M. L. Latham, *The geometry of René Descartes*, English, New York: Dover Publications XII, 244 p. (1954). 1954.

[47] R. P. Stanley, «Log-concave and unimodal sequences in algebra, combinatorics, and geometry,» English, en *Graph theory and its applications: East and West. Proceedings of the first China-USA international conference, held in Jinan, China, June 9–20, 1986.* New York: New York Academy of Sciences, 1989, págs. 500-535, ISBN: 0-89766-579-1.

[48] D. J. Struik, *A Source Book in Mathematics, 1200-1800*. Princeton University Press, 2014, ISBN: 9781400858002. DOI: doi:10.1515/9781400858002. URL: https://doi.org/10.1515/9781400858002.

[49] B. Szénássy, *History of mathematics in Hungary until the 20th century. Transl. from the Hungarian by Judit Pokoly. English text revised by János Bognár*, English. Berlin: Springer-Verlag, 1992, pág. 369, ISBN: 3-540-55497-1.

[50] E. A. Tobis, «Libraries for Counting Real Roots,» English, *Reports on computer Algebra*, n.º 34, 2005.

[51] X. Wang, «A simple proof of Descartes' rule of sign,» English, *Am. Math. Mon.*, vol. 111, n.º 6, págs. 525-526, 2004, ISSN: 0002-9890. DOI: 10.2307/4145072.