TRABAJO FIN DE GRADO

Doble Grado en Ingeniería Informática y Matemáticas

Gröbner-Robotics

Bases de Gröbner y movimientos de brazos de robot

Autor
Antonio Solís Izquierdo (alumno)

Director
Pedro A. García-Sánchez (tutor)

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mes de 2016

Índice general

Gröbner-Robotics

Bases de Gröbner y movimientos de brazos de robot.

Autor
Antonio Solís Izquierdo (alumno)

Director
Pedro A. García-Sánchez (tutor)

Gröbner-Robotics: Bases de Gröbner y movimientos de brazos de robot

Antonio Solís Izquierdo (alumno)

Palabras clave: Bases, Gröbner, Brazo de Robot, Cinemática, Buchberger

Resumen

En este Trabajo de Fin de Grado se aborda el problema del movimiento de un brazo robótico teniendo en cuenta los tipos de articulaciones y giros que posee dicho brazo. En concreto, se tratará el caso en el que se tendrá un conjunto de segmentos, unidos por articulaciones, de forma que un extremo estará anclado a una posición y en el otro se encontrará la "mano". Dado que los segmentos del brazo son rígidos, los posibles movimientos del brazo se determinan por los de las articulaciones. Todas estas articulaciones estarán en el mismo plano y, por lo tanto, todo movimiento se llevará a cabo en dicho plano.

Dado que se querrán llevar a cabo movimientos en 3 dimensiones, se hará uso de una articulación universal en la base de nuestro robot, lo que hará que aunque se mueva en un plano, dicho plano podrá ser "rotado" y por tanto se tendrá acceso a todas las posiciones del espacio dentro del alcance del brazo. Veremos cómo se puede definir la posición o configuración de una articulación y esto nos dará paso a los dos problemas básicos que vamos a tratar, el de la cinemática directa, el cual nos hará plantearnos si podemos dar una descripción explícita o fórmula para nuestra función a partir de la configuración de conjuntos y las dimensiones de los segmentos del brazo, y el de la cinemática inversa, el cual tratará de, dado un punto que se quiere alcanzar, decidir si ese punto es accesible o no y, en caso de serlo, hallar las posiciones de todas las articulaciones del brazo. Veremos que el primero de los problemas se resuelve con relativa facilidad y por tanto nos será de interés como preliminar a el problema inverso, el cual tendrá una complicación mayor ya que las ecuaciones que se tratarán no son lineales en caso de tener articulaciones de revolución.

En el apartado de cinemática directa se verá como definir de la posición de una articulación con respecto a las demás y como se podrá obtener la posición de la mano, sabiendo los ángulos de las articulaciones de revolución y las longitudes de los segmentos, en caso de haber alguno telescópico. En la cinemática inversa sin embargo, serán estos ángulos los que querremos hallar. Para ello transformaremos los senos y cosenos de las ecuaciones obtenidas en el problema de la cinemática inversa en variables auxiliares sujetas a la condición de que están en la circunferencia unidad. Obtendremos así un sistema de ecuaciones polinomiales, y para resolverlo vamos a hacer uso de las Bases de Gröbner.

Para explicar qué son y cómo usar Bases de Gröbner, se comenzará dando una serie de definiciones y resultados previos que serán necesarios para describir cómo usar Bases de Gröbner para la resolver sistemas de ecuaciones polinomiales. A continuación se explicarán qué son los órdenes monomiales y se definirán algunos de los órdenes más comunes. También veremos algunos conceptos más referentes a los grados de monomios que nos serán útiles para esta ordenación. Una vez visto todo esto pasaremos a ver el Algoritmo de la División en varias variables, el cual nos ayudará a resolver el problema de la pertenencia a los ideales para polinomios de varias variables. En general el residuo que se obtiene a partir del algoritmo de la división depende del orden elegido y del orden en que se da la secuencia de polinomios por los que dividimos, lo cual es una de las motivaciones principales para la definición de bases de Gröbner, pues éstas verifican que para un orden dado, el resto de la división es independiente del orden en que vengan dados los polinomios por los que efectuamos la división.

Nuestra definición de base de Gröbner se basará en el concepto de ideal monomial. Los S-Polinomios nos relacionan esta definición con la idea de no dependencia del resto del orden en los polinomios por los que dividimos. El criterio de Buchberger usa los S-polinomios para detectar si un conjunto de polinomios es una base de Gröbner, y de dicho criterio surge de manera natural el algoritmo de Buchberger. Este Algoritmo nos dará una forma de calcular una Base de Gröbner.

Finalmente, gracias al Teorema de Eliminación, el orden lexicográfico nos da una base de Gröbner en forma triangular, lo cual usaremos para resolver sistemas de ecuaciones polinomiales.

La referencia utilizada para el problema de la cinemática directa e inversa, así como para la resolución de sistemas de ecuaciones polinomiales mediante bases de Gröbner ha sido [1].

Todo este trabajo matemático ha servido como base para el diseño e implementación de un brazo robótico con dos articulaciones. Para ello hemos usado el lenguaje de programación python el cual está siendo uno de los más utilizados en la actualidad por su potencia y simplicidad a la hora de programar (una buena referencia puede ser [2]).

Una de las librerías de cálculo simbólico más conocidas de este lenguaje

es Sympy [4], cuyo propósito es llevar y tratar la matemática simbólica de la manera más sencilla posible de manera que el código sea comprensible y fácilmente ampliable, permitiendo, por ejemplo, trabajar con bases de Gröbner.

Respecto al diseño de la parte gráfica se ha optado por utilizar la librería de python Matplotlib [3], la cual nos permitirá la representación de figuras en tres dimensiones de forma que la aplicación sea interactiva en varias plataformas, haciendo posible la visualización de nuestro brazo interactuando con los widgets que permiten al usuario cambiar las posiciones de la mano del brazo.

Gröbner-Robotics: Bases de Gröbner y movimientos de brazos de robot

Antonio Solís Izquierdo (student)

Keywords: Gröbner, Bases, Robot, Arm, Kinematic, Buchberger

Abstract

In this manuscript we show how the inverse kinematic problem for an arm robot can be solved by means of Gröbner bases. We will present the theory needed to this end, and will give a practical example with a python implementation to better understand how to solve this problem.

Our robot arms will have a set of segments linked by joints, that can be either planar joints or universal joints. The segments can also be telescopic. The arm will be have in an end a hand, and in the other end, it will soldered to the ground (or to a platform, and so this position is fixed). Since the segments of the arms are rigid or telescopic, the set of points accessible by the hand will be determined by the length of the telescopic segments and the angles of the joints. We will assume, in order to simplify, that all of the joints will be on the same plane, and thus, all movements will be produced on this plane. In order to allow three dimensional movements we will place a universal joint in the base of the arm. This allows to "rotate" the plane in which the arm moves, so it can have access to all the positions on space within the reach of the arm.

We will also see how we can define the position or configuration of a joint. Two basic natural problems arise, one the inverse of the other. The first problem is the problem of direct kinematics, in which we want to give an explicit formula for the coordinates of the hand in terms of the angles of the joints and the lenghts of the telescopic segments. The second problem deals with the converse of the first, and it is known as the inverse kinematic problem. We are given the coordintes of a point, and wee need to determine if there exists values for the lengths of the telescopic segments and angles of

the joint that move the hand to this given point. In case such values exist, a formula in terms of the coordinates of the point must also be found.

We will see that our first problem is relatively easy to solve, and it will be of interest state the inverse kinematic problem. The difficulty of the inverse kinematic is due to the fact that we will facing the resolution of a system of nonlinear equations in sinuses and cosines.

Direct kinematic problem can be solve by using matrix compositions that represent rigid movements in the affine space. For each telescopic segment and joint we will have a matrix expression, and the position of the hand will be determined by the composition (multiplication) of these matrices. Thus we will obtain a formula for each coordinate in terms of the lengths of the segments and angles between them. These formulas will be nonlinear combinations of several sinuses and cosines, and the lengths of the segments.

The key idea to solve the inverse kinematic problem will be to transform, for each angle each cosine and sinus to a couple of variables that are linked by the equation of the unit circle. So for each angle, we get two extra variables and an extra equation (that of the unit circle in these two variables). After this transformation, we will obtain a system of polynomial equations, and this is where Gröbner bases show up. They are a perfect tool to deal with systems of polynomial equations.

A chapter is devoted to how to use Gröbner bases to solve systems of polynomial equations in several variables. Gröbner bases can be thought as a generalization of the Division Algorithm in the ring of polynomials in one variable over a field (which makes this ring an Euclidean domain). Actually Division Algorithm solved the membership problem for ideals in Euclidean domains, since these ideals are always principal, and then an element in the ring is in the ideal if and only if the remainder of the division of this element by a generator of the ideal is zero. In several variables this is not the case in general. First, ideals are not principal, and second, we do no longer have a Division Algorithm. But why are we concerned about membership to an ideal of a ring of polynomial in several variables over a field? The reason is precisely the weak version of Hilbert's Nullstellensatz. If our system of polynomial equations is of the form $f_1 = 0, \ldots, f_s = 0$, then Hilbert's Nullstellensatz asserts that the system of equations has no solution (we are considering that our field is algebraically closed; in our setting it will be the field of complex numbers) if and only if 1 is in the ideal spanned by $f_1, \ldots, f_s$.

In order to generalize the Division Algorithm in several variables, one first needs to order the monomials in a given polynomial. This ordering must be compatible with multiplication by monomials. This is why we will introduced admissible orderings in the set of monomials. The idea of dividing remains the same: we first try to cancel the leading term, and then continue until we cannot go further.

The other problem is precisely that ideals will not be principal in general.

So in several variables we have to divide by a set of polynomials, and no by a single one. As a side effect, the residue of a division might depend on which polynomials in this set we choose for canceling the terms of the given polynomial, and in which order we take them. In order to solve this issue, Buchberger added redundancy (S-polynomials) to the original system of equations (the set of generators of the ideal), so that the residue of the division of a polynomial by this new set of polynomials no longer depends on how we arrange these polynomials to apply cancelation of terms.

Our approach to introduce Gröbner bases relies on the concept of monomial ideal. We will say that a set of polynomials of a given ideal is a Gröbner basis with respect to a fixed admissible ordering if the leading coefficients of these polynomials generate the ideal of leading coefficients of all the elements in the ideal. Then we will introduce the concept of S-poynomial and Buchberger's criterium for Gröbner bases: a set of polynomials of an ideal is a Gröbner basis with respect to a monomial ordering if and only if the residue of the division of all possible S-polynomials by this set of polynomials is zero. From this characterization Buchberger's algorithm arises naturally. In order to better understand this procedure we have implemented it in python. We will show an example made by hand and will compare it with the output of our code.

The nullstellensatz combined with Gröbner bases can be used to determine the existence of a solution to a system of polynomial equations. But we actually want to find at least one solution. When we have to deal with linear equations, one can use Gauss-Jordan triangulation procedure to solve a system of equations. This triangularizes the given system and thus we can easily solve the equations with the least possible number of variables and the plug the solution in the next equation. Gröbner bases can do a similar task when we have polynomial equations. Thanks to the Elimination Theorem, the lexicographic order gives us a Gröbner basis in triangular form, which will allow us to proceed as in the linear case, but now solving polynomials in a single variable that might not be linear.

All the contents explained so far can be found in [1], which we used as our main reference for Gröbner basis, solving systems of polynomial equations, and inverse and direct kinematic problems.

As an example to illustrate these techniques, we have solve the inverse kinematic problem for a robotic arm with two joints. One can use many different software tools to solve systems of polynomial equations, and many of them are free software. We have used the programming language python, which is one of the most used nowadays for its power and simplicity when it comes to programming (a good reference could be [2]). In order to deal with Gröbner bases calculations we use Sympy, which is one of the best python libraries for symbolic computation [4]. Sympy provides polynomial classes and is easy to use. It also allows to calculate Gröbner bases with parameters. We will take advantage of this in our example, and so we will calculate a

single Gröbner basis that will allow to determine the set of all points that can be reached with the hand or our robot arm.

Regarding the design of the graphic part we have opted to use another python library called Matplotlib [3], which will allow us to represent three-dimensional figures in a way that will let the application be interactive in different platforms, making possible to visualize our arm interacting with the widgets that allow the user to change the position of the hand of the arm.

The code can be found in the github repository pedritomelenas/tfg-Groebner-Robotics, together with a pdf copy of this document.

Yo, **Antonio Solís Izquierdo**, alumno de la titulación Doble Grado en Ingeniería Informática y Matemáticas de la **Facultad de Ciencias** y la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75571818-M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Antonio Solís Izquierdo

Granada a 20 de febrero de 2016 .

Pedro A. García-Sánchez, Profesor del Área de Álgebra del Departamento Álgebra de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Bases de Gröbner y movimientos de brazos de robot***, ha sido realizado bajo su supervisión por **Antonio Solís Izquierdo (alumno)**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 12 de mes septiembre de 2016.

El director:

Pedro A. García Sánchez

Agradecimientos

Gracias a mi tutor Pedro, por ayudarme en todo lo posible, incluso a horas intempestivas.

Gracias a mi familia, en especial a mis padres y mi hermana por soportar mis cambios de humor y mis nervios.

Gracias a mis compañeros de LPB por soportarme durante estos 5 años, en especial a los minions por ser así de insoportables.

Y a mi traductora, psicóloga y mejor amiga, gracias por todo lo pasado hasta ahora y todo lo que nos queda juntos.

Capítulo 1

Introducción

En este proyecto abordaremos el movimiento de un brazo robótico desde un punto del espacio, hasta alcanzar otro. Resolveremos este problema mediante el uso de bases de Gröbner.

Siempre vamos a considerar brazos construidos a partir de segmentos rígidos, entendiendo por esto que no se doblan, conectados por articulaciones de varios tipos. Para simplificar, vamos a considerar solo los robots en que los segmentos están conectados en serie, como en un miembro humano. Un extremo de nuestro brazo por lo general se fija en su posición y en el otro extremo estará la "mano", que a veces se considera como un segmento final del robot. En robots reales, esta "mano" puede estar provista de mecanismos para agarrar objetos o puede ser una herramienta para realizar alguna tarea. Por lo tanto, uno de los principales objetivos es ser capaz de describir y especificar la posición y orientación de la "mano".

Dado que los segmentos de nuestro brazo son rígidos, los posibles movimientos del brazo se determinan por los de las articulaciones. Muchos robots reales son construidos utilizando articulaciones de revolución y telescópicas. Las articulaciones de revolución permiten la rotación de un segmento con respecto a otro. Nosotros asumiremos que los dos segmentos en cuestión se encuentran en un plano y todo movimiento de la articulación dejara a los dos segmentos en ese plano. Las articulaciones telescópicas permiten a una parte del brazo moverse a lo largo de un eje.

En el caso de haber varias articulaciones en un mismo robot, supondremos que las articulaciones se encuentran todas en el mismo plano, que los ejes de rotación de todos articulaciones angulares son perpendiculares a dicho plano, y, además, que la traducción de los ejes de las articulaciones prismáticas se encuentran todas en el plano de las articulaciones. Por lo tanto, todo movimiento se llevará a cabo en un plano. Por supuesto, esto conduce a una clase muy restringida de robots. Los robots reales deben ser por lo general capaces de moverse en 3 dimensiones. Para lograr esto se utilizan otros tipos y combinaciones de articulaciones. Estos incluyen articulaciones

"bola", también conocidas como universales, que permiten la rotación sobre cualquier eje que pasa por un cierto punto en $\mathbb{R}^3$ y articulaciones helicoidales o de "tornillo" que son combinaciones de rotación y traslación a lo largo del eje de rotación en $\mathbb{R}^3$. También sería posible conectar varios segmentos de un robot con las articulaciones de revolución, pero con ejes no paralelos de rotación. Todas estas posibles configuraciones se pueden tratar por métodos similares a los que se estudiarán a continuación, aunque no entraremos en detalles sobre ninguno de ellos. Sin embargo trataremos un caso en el que la articulación de la base si que es universal para asi permitir movimientos en el espacio.

En general, la posición o configuración de una articulación de giro entre segmentos consecutivos puede ser descrito midiendo el ángulo ϑ (en sentido contrario a las agujas del reloj) formado por dichos segmentos. Debido a esto, la configuración de una articulación de este tipo puede ser parametrizada por un círculo $\mathbb{S}^1$ o por el intervalo $[0, 2\pi]$ con los puntos finales identificados. En algunos casos, una articulación de revolución puede no ser libre para girar a través de un círculo completo, y entonces particularizaríamos las posibles configuraciones mediante un subconjunto de $\mathbb{S}^1$.

Del mismo modo, la configuración de una articulación telescópica se puede especificar dando la distancia entre el extremo de esta articulación y de la anterior). De cualquier manera, la configuración de una articulación telescópica puede ser parametrizada por un intervalo finito de números reales.

Si las configuraciones de las articulaciones de nuestro robot se pueden especificar de forma independiente, entonces las posibles configuraciones de todo el conjunto de articulaciones en un brazo que se mueve en un plano con r articulaciones de revolución y t telescópicas puede ser parametrizado por el producto cartesiano

$$\mathcal{J} = \mathbb{S}^1 \times \cdots \times \mathbb{S}^1 \times I_1 \times \cdots \times I^t,$$

donde hay un factor $\mathbb{S}^1$ para cada articulación de revolución, y cada I_j da la configuración de la articulación telescópica j-ésima. Vamos a llamar a $\mathcal{J}$ el espacio de articulaciones del robot. Podemos describir el espacio de posibles configuraciones de la "mano" de un robot que se mueve en un plano de la siguiente manera. Fijando el sistema de coordenadas cartesianas en el plano, podemos representar las posiciones posibles de la "mano" por los puntos (a, b) de una región $U \subseteq \mathbb{R}^2$. Del mismo modo, podemos representar la orientación de la "mano", dando un vector unitario. Por lo tanto, las posibles orientaciones de mano son parametrizados por los vectores u en $V = \mathbb{S}^1$.

Vamos a llamar a $C = U \times V$ el espacio de configuración o el espacio operativo de la mano del robot. Cada conjunto de parámetros conjuntos posicionará la "mano" en un lugar determinado, con una orientación deter-

minada, de forma única. Debido a esto, tenemos una función

$$f : \mathcal{J} \longrightarrow \mathcal{C}$$

que codifica cómo las diferentes configuraciones de las articulaciones producen diferentes configuraciones de mano.

Hay dos problemas básicos que se podrían considerar en este punto ya que pueden ser descritos sucintamente en términos de la asignación $f : \mathcal{J} \longrightarrow \mathcal{C}$ mencionada anteriormente:

- Problema de la cinemática directa: ¿Podemos dar una descripción explícita o fórmula para f en función de la configuración de conjuntos (nuestros coordenadas en J) y las dimensiones de los segmentos del brazo?

- Problema de la cinemática inversa: Dada $c \in \mathcal{C}$, ¿podemos determinar una o todas las $j \in \mathcal{J}$ tal que $f(j) = c$?

Veremos que el primero de los problemas se resuelve con relativa facilidad. Determinar la posición y orientación de la "mano" a partir de la configuración del "brazo" es sobre todo una cuestión de ser sistemático en la descripción de las posiciones relativas de los segmentos a cada lado de una articulación. Por lo tanto, el problema directo es de interés principalmente como un preliminar para el problema inverso.

El inverso es algo más sutil ya que nuestras fórmulas explícitas no serán lineales si hay articulaciones de revolución presentes. Por lo tanto, tendremos que utilizar los resultados generales de los sistemas de ecuaciones polinómicas para resolver la ecuación $f(j) = c$.

Una característica de los sistemas no lineales de ecuaciones es que puede haber varias soluciones diferentes, incluso cuando todo el conjunto de soluciones es finito. Como cuestión práctica, la potencial no unicidad de las soluciones es a veces muy deseable. Por ejemplo, si nuestro robot en el mundo real va a trabajar en un espacio que contiene obstáculos físicos o barreras al movimiento en ciertas direcciones, puede darse el caso de que algunas de las soluciones correspondan a las posiciones que son no físicamente accesibles. Para determinar si es posible llegar a una posición dada, puede ser que necesitemos calcular todas las soluciones, y luego, ver cuales son factibles debido a las limitaciones del entorno en el que nuestro robot trabaja.

Para comprender todos estos conceptos se hizo uso de [1], en el que lograron plasmar toda esta temática de una forma comprensible y bien estructurada. Facilitando así la aproximación a nuevos conceptos que de otra forma hubiesen resultado más difíciles de comprender.

En base a todo lo anterior se decidió que la mejor forma de plasmar todo esto era realizar el diseño e implementación de un brazo robótico con

dos articulaciones. Para ello hemos usado el lenguaje de programación python, junto con dos librerías, una de cálculo simbólico, Sympy [4], y otra de representación gráfica, Matplotlib [3].

Capítulo 2

Objetivos

Desde el punto de vista matemático, se definieron una serie de objetivos a alcanzar a lo largo del trabajo:

- ✓ Comprender el concepto de cinemática y sus tipos.
- ✓ Saber definir ordenes monomiales.
- ✓ Uso del Algoritmo de la División en varias variables.
- ✓ Entender lo que es un ideal monomial.
- ✓ Ser capaz de construir una Base de Gröbner usando el algoritmo de Buchberger.
- ✓ Aprender la utilidad del Teorema de Eliminación.

Respecto a la parte correspondiente a la ingeniería informática se marcaron los siguiente objetivos principales:

- ✓ Conocer el lenguaje de programación python.
- ✓ Aprender a trabajar con la librería matemática Sympy.
- ✓ Ser capaz de realizar una interfaz gráfica para la aplicación usando matplotlib.

En el presente documento se irá viendo si dichos objetivos son satisfechos o no.

Capítulo 3

Cinemática

3.1. Cinemática Directa

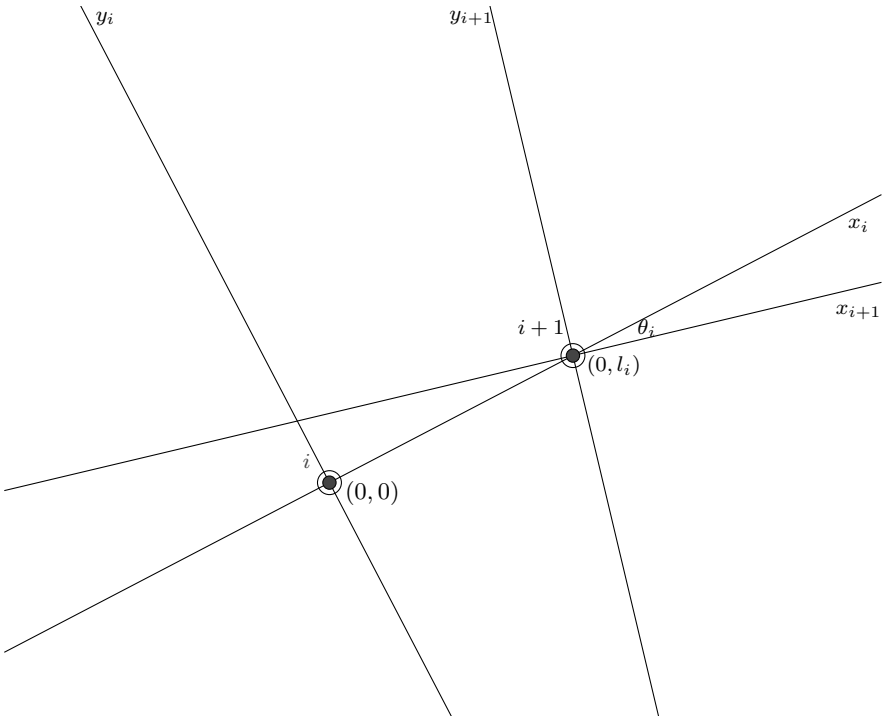
En esta sección, presentaremos un método estándar para resolver el problema de la cinemática directa para un "brazo". Se pretende calcular la posición de la mano una vez conocidos todos los parámetros del brazo (ángulos de las articulaciones y extensiones de los brazos telescópicos).

Vamos a colocar el origen de nuestro sistema de coordenadas en la articulación inicial del primer segmento del brazo, cuya posición es fija. Además del sistema de coordenadas global (x_1, y_1), se introduce un sistema de coordenadas local en cada una de las articulaciones de revolución para describir las posiciones relativas de los segmentos que se unen en esa articulación. Naturalmente, estos sistemas de coordenadas cambiarán a medida que la posición del "brazo"varíe.

En una articulación de revolución $i + 1$, introducimos un sistema de coordenadas (x_{i+1}, y_{i+1}) de la siguiente manera. El origen se sitúa en la articulación $i + 1$ y colocamos la dirección de las x positivas a largo del segmento $i + 1$. Por tanto, para cada i, las coordenadas de la articulación $i+1$ vistas desde el sistema de coordenadas (x_i, y_i) son $(l_i, 0)$, donde l_i es la longitud del segmento i.

Nuestro primer objetivo es relacionar las coordenadas (x_{i+1}, y_{i+1}) de un punto con las coordenadas (x_i, y_i) de dicho punto. Sea θ_i el ángulo en sentido antihorario desde el eje x_i al eje x_{i+1}. Si un punto q tiene las siguientes coordenadas en (x_{i+1}, y_{i+1}), $q = (a_{i+1}, b_{i+1})$, entonces, para obtener sus coordenadas en (x_i, y_i) , o sea $q = (a_i, b_i)$, primero hacemos un giro de ángulo θ_i (para alinear los ejes x_i y x_{i+1}), y luego una traslación mediante el vector $(l_i, 0)$ (para hacer coincidir los orígenes de los sistemas de coordenadas).

Todo esto se ve más fácilmente mirando el siguiente dibujo:



Por tanto, tenemos la siguiente relación entre las coordenadas (x_i, y_i) y (x_{i+1}, y_{i+1}) de q:

$$\begin{pmatrix} a_i \\ b_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix} \cdot \begin{pmatrix} a_{i+1} \\ b_{i+2} \end{pmatrix} + \begin{pmatrix} l_i \\ 0 \end{pmatrix}.$$

Esto también se puede escribir de forma abreviada utilizando una matriz de 3×3 y vectores de 3 componentes:

$$\begin{pmatrix} a_i \\ b_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & l_i \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{i+1} \\ b_{i+2} \\ 1 \end{pmatrix} = A_i \cdot \begin{pmatrix} a_{i+1} \\ b_{i+2} \\ 1 \end{pmatrix}. \tag{3.1}$$

Esto nos permite combinar la rotación θ_i con la traslación a lo largo del segmento i en una sola matriz 3×3 A_i. Por tanto para calcular la posición de la mano bastaría con multiplicar todas las matrices A_i correspondientes a cada articulación.

Veamos ahora como trabajaríamos si considerásemos un "brazo" con 3 articulaciones de revolución que se mueve en el plano. Pensaremos en la mano como el segmento 4, que está unido al 3 por de la articulación 3. Tenemos las matrices A_1, A_2 y A_3 como en la fórmula anterior. La observación clave es que las coordenadas globales de cualquier punto se pueden obtener a partir de sus coordenadas en el sistema de coordenadas (x_4, y_4) y operando hasta llegar al (x_1, y_1) pasando por todas las articulaciones intermedias de una en una. En otras palabras, se multiplica el vector de coordenadas del punto en (x_4, y_4) por A_3, A_2, A_1 en orden:

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = A_1 A_2 A_3 \begin{pmatrix} x_4 \\ y_4 \\ 1 \end{pmatrix}.$$

Usando las fórmulas de adición trigonométricas, esta ecuación puede escribirse como

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & l_3\cos(\theta_1+\theta_2)+l_2\cos\theta_1 \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & l_3\sin(\theta_1+\theta_2)+l_2\sin\theta_1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_4 \\ y_4 \\ 1 \end{pmatrix}$$

Dado que las coordenadas de la mano en (x_4, y_4) son $(0, 0)$ (ya que la mano está unida directamente a la articulación 3), obtenemos las coordenadas en (x_1, y_1) haciendo $x_4 = y_4 = 0$ y haciendo el producto anterior. El resultado es

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} l_3\cos(\theta_1+\theta_2)+l_2\cos\theta_1 \\ l_3\sin(\theta_1+\theta_2)+l_2\sin\theta_1 \\ 1 \end{pmatrix}.$$

La orientación de la mano se determina si conocemos el ángulo entre el eje x_4 y la dirección de cualquier característica particular de interés para nosotros en la mano. Por ejemplo, puede ser que simplemente quieren utilizar la dirección del eje x_4 para especificar esta orientación. De nuestros cálculos, sabemos que el ángulo entre el eje x_1 y el x_4 es simplemente $\theta_1 + \theta_2 + \theta_3$.

Si combinamos este hecho acerca de la orientación de la mano con la fórmula anterior para la posición de la mano, se obtiene una descripción explícita de la correspondencia $f : \mathcal{J} \longrightarrow \mathcal{C}$ introducido en el capitulo anterior. Como una función de los ángulos de las articulación θ_i, la configuración de la mano está dada por

$$f(\theta_1, \theta_2, \theta_3) = \begin{pmatrix} l_3\cos(\theta_1+\theta_2)+l_2\cos\theta_1 \\ l_3\sin(\theta_1+\theta_2)+l_2\sin\theta_1 \\ \theta_1+\theta_2+\theta_3 \end{pmatrix}. \tag{3.2}$$

Las mismas ideas se aplican cuando están presentes cualquier número de articulaciones de revolución.

Veamos ahora que las articulaciones telescópicas también pueden ser tratadas dentro de este marco. Por ejemplo, consideremos un robot cuyos primeros tres segmentos y articulaciones son los mismos que los del robot anterior, pero que tiene una articulación telescópica adicional entre el segmento 4 y la mano. Por lo tanto, el segmento 4 se tiene longitud variable y el segmento 5 será la mano.

Podemos describir el robot tal como sigue. Las tres juntas de revolución nos permiten exactamente la misma libertad en la colocación de junta 3 como en el robot estudiado anteriormente. Sin embargo, la articulación prismática nos permite cambiar la longitud del segmento 4 en cualquier valor entre

$l_4 = m_1$ (cuando está retraído) y $l_4 = m_2$ (cuando está completamente extendido). Siguiendo con el razonamiento dado anteriormente, si se conoce la configuración l_4 de la articulación telescópica, entonces la posición de la mano será dada por la multiplicación de la matriz del producto $A_1A_2A_3$ por las coordenadas del vector de la mano, que serán $(l_4, 0)$. De ello se desprende que la configuración de la mano está dada por

$$g(\theta_1, \theta_2, \theta_3, l_4) = \begin{pmatrix} l_4 \cos(\theta_1 + \theta_2 + l_3) + l_3 \cos(\theta_1 + \theta_2) + l_2 \cos\theta_1 \\ l_4 \sin(\theta_1 + \theta_2 + l_3) + l_3 \sin(\theta_1 + \theta_2) + l_2 \sin\theta_1 \\ \theta_1 + \theta_2 + \theta_3 \end{pmatrix}. \quad (3.3)$$

Como antes, l_2 y l_3 son constantes, pero $l_4 \in [m_1, m_2]$ es ahora otra variable. La orientación de la mano será dada por $\theta_1 + \theta_2 + \theta_3$ como antes, ya que la configuración de la articulación telescópica no afectará a la dirección de la mano.

3.2. Cinemática Inversa

El problema ahora es, dado un punto, determinar qué valores se le dan a las longitudes de brazos telescópicos y ángulos de articulaciones para que el brazo alcance ese punto. Para ello, primero vamos a transformar las ecuaciones de la sección anteriores en ecuaciones polinomiales.

Las variables para articulaciones de revolución y telescópicas se tratan de forma diferente. Para las articulaciones de revolución, la forma más directa de convertir a un conjunto de ecuaciones polinomiales es usar la idea de que aunque el coseno y el seno son funciones trascendentales, dan una parametrización

$$x = \cos\theta, \quad y = \sin\theta,$$

de la variedad algebraica $\mathbf{V}(x^2 + y^2 - 1)$ en el plano. Por lo tanto, podemos escribir los componentes de la parte derecha de (3.2) o, de forma equivalente, las entradas de la matriz $A_1A_2A_3$ en (3.3) como funciones de

$$c_i = \cos\theta_i, \quad s_i = \sin\theta_i,$$

sujetas a las restricciones

$$c_i^2 + s_i^2 - 1 = 0 \quad (3.4)$$

para $i = 1, 2, 3$. Téngase en cuenta que la variedad definida por estos tres ecuaciones en $\mathbb{R}^6$ es una realización concreta del espacio de la articulación $\mathcal{J}$ para este tipo de robot. Geométricamente, esta variedad es sólo un producto cartesiano de tres copias del círculo. Explícitamente, se obtiene a partir de (3.2) una expresión para la posición de la mano como una función de

variables $c_1, s_1, c_2, s_2, c_3, s_3$. Usando las fórmulas trigonométricas de adición, podemos escribir

$$\cos(\theta_1 + \theta_2) = \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2 = c_1c_2 - s_1s_2.$$

De manera similar,

$$\sin(\theta_1 + \theta_2) = \sin\theta_1 \cos\theta_2 - \sin\theta_1 \cos\theta_2 = s_1c_2 - s_2c_1.$$

De este modo, las coordenadas de la posición de la mano en (x_1, y_1) son:

$$\begin{pmatrix} l_3(c_1c_2 - s_1s_2) + l_2c1 \\ l_3(s_1c_2 + s_2c_1) + l_2s1 \end{pmatrix}. \tag{3.5}$$

Para empezar, vamos a considerar el problema de la cinemática inversa para un brazo robot que se mueve en un plano con tres articulaciones de revolución. Dado un punto $(x_1, y_1) = (a, b) \in \mathbb{R}^2$ y una orientación, que desean determinar si es posible colocar la mano del robot en ese punto con esa orientación. Si es posible, deseamos encontrar todas las combinaciones de configuración de conjuntos que lograr esto.

Es bastante fácil ver geométricamente que si $l_3 = l_2 = l$, la mano de nuestro robot se puede colocar en cualquier punto del disco cerrado de $2l$ de radio con centro en la articulación 1, que a su vez está en el origen de coordenadas. Por otro lado, si $l_3 \neq l_2$, entonces las posiciones de las manos rellenan un anillo cerrado centrada en la articulación 1.

Para un robot de este tipo, también es fácil controlar la orientación de la mano. Puesto que la configuración de la articulación 3 es independiente de las de la 1 y 2, se observa que, para cualquiera θ_1 y θ_2, es posible alcanzar cualquier orientación deseada $\alpha = \theta_1 + \theta_2 + \theta_3$ configurando $\theta_3 = \alpha - (\theta_1 + \theta_2)$ de acuerdo a ella.

Para simplificar nuestra resolución del problema de cinemática inversa, utilizaremos la observación anterior para ignorar la orientación mano. Por lo tanto, nos concentraremos en la posición de la mano, que es una función que depende solo de θ_1 y θ_2. A partir de la ecuación 3.5 del apartado anterior, vemos que las posibles maneras de colocar la mano en un punto dado $(x_1, y_1) = (a, b)$ se describen mediante el siguiente sistema de ecuaciones polinómicas:

$$\begin{aligned} a &= l_3(c_1c_2 - s_1s_2) + l_2c_1, \\ b &= l_3(c_1s_2 - c_2s_1) + l_2s_1, \\ 0 &= c_1^2 + s_1^2 - 1, \\ 0 &= c_2^2 + s_2^2 - 1, \end{aligned} \tag{3.6}$$

con c_1, s_1, c_2, s_2 variables.

Todo lo que hemos visto en este ejemplo es escalable a k articulaciones, para ellos necesitaríamos substituir $\sin\theta_i$ por s_i y $\cos\theta_i$ por c_i, y añadir una

ecuación del tipo $c_i^2 + s_i^2 - 1 = 0$. Pasamos así a un sistema de ecuaciones polinomiales.

Para resolver estas ecuaciones vamos a usar bases de Gröbner.

Capítulo 4

Bases de Gröbner

Dado un cuerpo k, el anillo de polinomios $k[x]$ es un dominio Euclídeo, y en particular es un anillo de ideales principales. Por tanto, la pertenencia a un ideal queda resuelta por el algoritmo de la división en $k[x]$. En el caso de más de una variable, el anillo de polinomios deja de ser un dominio de Euclídeo, pero se puede seguir construyendo un algoritmo de división. Para ello primero hay dar un orden en los monomios para poder hablar de conceptos como término líder y coeficiente líder. Además no todos los sistemas de generadores del ideal dado tienen por qué ser buenos para detectar la pertenencia al ideal usando el algoritmo de la división.

4.1. Definiciones y resultados previos

Sean $x_1, \ldots, x_n$ variables. Recordemos que un monomio en estas variables es un producto finito de ellas (algunas pueden aparecer repetidas). Para simplificar, dado $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ (con $\mathbb{N}$ el conjunto de enteros no negativos), denotamos por

$$x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n},$$

y $|\alpha| = \alpha_1 + \cdots + \alpha_n$ es el grado total del monomio x^α.

Un polinomio $f \in k[x_1, \ldots, x_n]$ se puede expresar de la forma

$$f = \sum_{\alpha \in A} a_\alpha x^\alpha,$$

con A un subconjunto finito de $\mathbb{N}^n$ y $a_\alpha \in k$ para todo $\alpha \in A$.

Definición 4.1 *Sea $f = \sum_{\alpha \in A} a_\alpha x^\alpha$ un polinomio en $k[x_1, \ldots, x_n]$.*

(i) Llamamos a_α al coeficiente del monomio x^α.

(ii) Si $a_\alpha \neq 0$, entonces decimos que $a_\alpha x^\alpha$ es un término de f.

(iii) El grado total de $f \neq 0$, denotado $\deg(f)$*, es el máximo $|\alpha|$ tal que el coeficiente a_α es distinto de cero. El grado total del polinomio cero es indefinido.*

Recordemos que estamos interesados en resolver el problema de la cinemática inversa, que hemos traducido a resolver un sistema de ecuaciones polinomiales. Esto es, estamos interesados en encontrar los ceros comunes de un conjunto finito de polinomios. Este concepto viene recogido en la definición de variedad.

Definición 4.2 *Sea k un cuerpo, y sean $f_1, \ldots, f_n$ polinomios en el anillo $k[x_1, \ldots, x_n]$. Definimos la variedad afín definida por $f_1, \ldots, f_s$ como*

$$V(f_1, \ldots, f_s) = \{a \in k^n \mid f_i(a) = 0 \text{ para todo } i \in \{1, \ldots, s\}\}.$$

Recordemos que un subconjunto $I \subseteq k[x_1, \ldots, x_n]$ es un ideal si satisface las siguientes propiedades.

(i) Si $f, g \in I$, entonces $f + g \in I$.

(ii) Si $f \in I$ y $h \in k[x_1, \ldots, x_n]$, entonces $hf \in I$.

Además, si $f_1, \ldots, f_s$ polinomios en $k[x_1, \ldots, x_n]$, entonces

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \;\middle|\; h_1, \ldots, h_s \in k[x_1, \ldots, x_n] \right\}$$

es el menor ideal que contiene a $f_1, \ldots, f_s$. El conjunto de ceros de los polinomios $f_1, \ldots, f_s$ no depende de los polinomios en sí, si no más bien del ideal que generan. Esto se debe a que si un punto se anula en estos polinomios, lo hará también en cualquier combinación lineal de ellos.

Proposición 4.1 *Si $\{f_1, \ldots, f_s\}$ y $\{g_1, \ldots, g_s\}$ generan el mismo ideal en el anillo $k[x_1, \ldots, x_n]$, entonces $V(f_1, \ldots, f_s) = V(g_1, \ldots, g_t)$.*

Definición 4.3 *Sea $V \subseteq k^n$ una variedad afín (los ceros comunes de un conjunto de polinomios). Definimos*

$$I(V) = \{f \in k[x_1, \ldots, x_n] \mid f(a) = 0 \text{ para todo } a \in V\}.$$

Lema 4.1 *Si $V \subseteq k^n$ es una variedad afín, entonces $I(V) \subseteq k[x_1, \ldots, x_n]$ es un ideal. Llamaremos a $I(V)$ el ideal de V.*

Se puede determinar si un sistema de ecuaciones polinomiales tiene solución sobre un cuerpo algebraicamente cerrado usando el teorema (débil) de los ceros de Hilbert.

Teorema 4.1 (Teorema de los ceros de Hilbert (débil)) *Dado un cuerpo algebraicamente cerrado k y $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$. La variedad $V(f_1, \ldots, f_s)$ es vacía si y sólo si $1 \in \langle f_1, \ldots, f_s \rangle$.*

Ahora bien, dados $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$, en principio no es trivial decidir si $1 \in \langle f_1, \ldots, f_s \rangle$. Esto lo resolveremos utilizando bases de Gröbner.

4.2. Órdenes monomiales

Si examinamos el algoritmo de la división en $k[x]$ y el algoritmo de reducción por filas (eliminación de Gauss-Jordan) para los sistemas de ecuaciones lineales (o matrices) en detalle, vemos que la noción de orden de los términos en los polinomios es un ingrediente clave de ambos.

En esta sección, discutiremos las propiedades deseables que un ordenamiento debe tener y veremos unos ejemplos que satisfacen dichas propiedades. Utilizaremos en todo momento la correspodencia entre monomios en $k[x_1, \ldots, x_n]$ is sus (multi)exponenetes en $\mathbb{N}^n$.

Definición 4.4 *Un orden monomial $\preceq$ en $k[x_1, \ldots, x_n]$ es una relación $\preceq$ en $\mathbb{N}^n$, o equivalentemente, una relación en el conjunto de monomios x^α, $\alpha \in \mathbb{N}^n$, cumpliendo:*

(i) *$\preceq$ es un orden total (o lineal) en $\mathbb{N}^n$.*

(ii) *Si $\alpha \preceq \beta$ y $\gamma \in \mathbb{N}^n$, entonces $\alpha + \gamma \preceq \beta + \gamma$.*

(iii) *$\preceq$ es un buen orden en $\mathbb{N}^n$. Esto significa que cada subconjunto no vacío de $\mathbb{N}^n$ mínimo respecto de $\preceq$.*

Dado un orden monomial $\preceq$, denotamos por $\alpha \prec \beta$ cuando $\alpha \preceq \beta$ y $\alpha \neq \beta$. También escribiremos $\alpha \succeq \beta$ y $\alpha \succ \beta$ cuando $\beta \preceq \alpha$ y $\beta \prec \alpha$, respectivamente.

Definición 4.5 (Orden lexicográfico) *Sea $\alpha, \beta \in \mathbb{N}^n$. Escribimos $\alpha \preceq_{lex} \beta$ si el elemento no nulo más a la izquierda del vector diferencia $\beta - \alpha \in \mathbb{Z}^n$ es positivo. Escribiremos $x^\alpha \preceq_{lex} x^\beta$ si $\alpha \preceq_{lex} \beta$.*

Proposición 4.2 *El orden lexicográfico en $\mathbb{N}^n$ es un orden monomial.*

En la práctica, cuando trabajamos con polinomios en dos o tres variables, llamaremos a las variables x, y, z en lugar de x_1, x_2, x_3. También vamos a suponer que el orden alfabético de las variables ($x \succ y \succ z$) se utiliza para definir el orden lexicográfico a menos que digamos lo contrario de forma explícita.

Para el orden léxico con $x \succ y \succ z$, tenemos $x \succ_{lex} y^5$. Para algunos propósitos, es posible que también deseemos tener los grados totales de los monomios en cuenta y poner los monomios de mayor grado primero. Una forma de hacer esto es con el orden lexicográfico graduado (u orden grlex).

Definición 4.6 (Orden lexicográfico graduado) *Sean $\alpha, \beta \in \mathbb{N}^n$. Escribimos $\alpha \preceq_{grlex} \beta$ si*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i < |\beta| = \sum_{i=1}^{n} \beta_i, \ \ o \ |\alpha| = |\beta| \ \ y \ \alpha \prec_{lex} \beta.$$

Otro orden es el orden lexicográfico graduado inverso (u orden grevlex).

Definición 4.7 (Orden lexicográfico graduado inverso) *Sean* $\alpha, \beta \in \mathbb{N}^n$. *Escribimos* $\alpha \preceq_{grevlex} \beta$ *si*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i < |\beta| = \sum_{i=1}^{n} \beta_i, \text{ o } |\alpha| = |\beta| \text{ y el elemento no negativo más a la derecha de } \beta - \alpha \in \mathbb{N}^n \text{ es negativo.}$$

Hay muchos otros órdenes de monomios, además de los considerados aquí. La mayoría de los sistemas de álgebra computacional tienen implementado el orden lexicográfico, y la mayoría también permiten otros órdenes, como grlex y grevlex.

Veamos ahora un ejemplo para entender mejor cada uno, consideremos el polinomio $f = 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2 \in k[x, y, z]$. Entonces:

- Con orden lex $f = -5x^3 + 7x^2z^2 + 4xy^2z + 4z^2$
- Con orden grlex $f = 7x^2z^2 + 4xy^2z - 5x^3 + 4z^2$
- Con orden grvlex $f = 4xy^2z + 7x^2z^2 - 5x^3 + 4z^2$

Dado un orden monomial $\preceq$ y un conjunto finito de monomios X, por ser $\preceq$ un orden total, existe siempre $\max_{\preceq} X$, el máximo con respecto a $\preceq$ de X.

Definición 4.8 *Sea* $f = \sum_\alpha a_\alpha x^\alpha \in k[x_1, \ldots, x_n]$, $f \neq 0$, *y sea* $\preceq$ *un orden monomial.*

(i) El multigrado de f *es*

$$\text{multideg}(f) = \max_{\preceq}\{\alpha \in \mathbb{N}^n \mid a_\alpha \neq 0\}.$$

(ii) El coeficiente líder de f *es*

$$\text{LC}(f) = a_{\text{multideg}(f)} \in k.$$

(ii) El monomio líder de f *es*

$$\text{LM}(f) = x^{\text{multideg}(f)}.$$

(ii) El término líder de f *es*

$$\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f).$$

Para ilustrar esto usemos el polinomio del ejemplo anterior, $f = 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2$, tomando $\preceq$ como el orden lexicográfico. Entonces

$$\begin{aligned} \text{multideg}(f) &= (3,0,0), \\ \text{LC}(f) &= -5, \\ \text{LM}(f) &= x^3, \\ \text{LT}(f) &= -5x^3. \end{aligned}$$

El multigrado tiene las siguientes propiedades útiles.

Lema 4.2 *Sea $f, g \in k[x_1, \ldots, x_n]$ un polinomio distinto de cero. Entonces:*

(i) $\text{multideg}(fg) = \text{multideg}(f) + \text{multideg}(g)$,

(ii) si $f + g \neq 0$, entonces

$$\text{multideg}(f+g) \leq \text{máx}\{\text{multideg}(f), \text{multideg}(g)\}.$$

Si, además, $\text{multideg}(f) \neq \text{multideg}(g)$*, entonces se produce la igualdad.*

4.3. Algoritmo de División en varias variables

Sabemos que el algoritmo de la división se puede utilizar para resolver el problema de la pertenencia a los ideales para polinomios de una variable. Para estudiar este mismo problema cuando hay más variables, vamos a describir un algoritmo de la división de polinomios en $k[x_1, \ldots, x_n]$ que extiende al algoritmo para $k[x]$. La idea básica del algoritmo es la misma que en el caso de una variable: queremos cancelar el término líder de f (con respecto a un orden monomial fijo) usando los polinomios por los que estamos dividiendo.

Teorema 4.2 (Algoritmo de la División) *Sea $\preceq$ un orden monomial en $\mathbb{N}^n$, y sea $F = (f_1, \ldots, f_s)$ una s-tupla ordenada de polinomios en $k[x_1, \ldots, x_n]$. Entonces, cada $f \in k[x_1, \ldots, x_n]$ se puede escribir como*

$$f = q_1 f_1 + \cdots + q_s f_s + r,$$

donde $q_i, r \in k[x_1, \ldots, x_n]$, y, o bien $r = 0$, o r es una combinación lineal, con coeficientes en k, de monomios, ninguno de los cuales es divisible por cualquiera de $\text{LT}(f_1), \ldots, \text{LT}(f_s)$. Llamamos a r el resto de f en una división por F. Por otra parte, si $q_i f_i \neq 0$, entonces

$$\text{multideg}(f) \geq \text{multideg}(q_i f_i).$$

Ejemplo 4.1 *Consideramos la siguiente división con orden lex. Sea*

$$f = x^3y^2 + xy + x$$

y nuestros divisores

$$g_1 = y^2 + 1,$$

$$g_2 = xy + 1.$$

Vemos que $\mathrm{LT}(g_1) = y^2$ *divide a* $\mathrm{LT}(f) = x^3y^2$*. Por lo que actualizamos el cociente, añadiendo* x^3*. Ahora actualizamos el dividendo:*

$$(x^3y^2 + xy + x) - x^3(y^2 + 1) = -x^3 + xy + x.$$

Como $\mathrm{LT}(g_1) = y^2$ *no divide a* $-x^3$*, nos fijamos en el siguiente polinomio en la lista de divisores y vemos que* $\mathrm{LT}(g_2) = xy$ *tampoco lo divide. Por lo tanto continuamos con el siguiente término en* f*, que es* xy*. Como* g_1 *es lo primero en la lista de divisores, comprobamos si* $\mathrm{LT}(g_1) = y^2$ *divide a* xy*. Ya que no es así, pasamos a* $\mathrm{LT}(g_2) = xy$ *y vemos que sí lo hace. Así que actualizamos el cociente correspondiente a* g_2 *sumándole uno. De nuevo actualizamos el dividendo:*

$$(-x^3 + xy + x) - 1 \cdot (xy + 1) = -x^3 + x - 1.$$

Obsérvese que ninguno de los términos lideres de los divisores divide a los términos del dividendo. Así, la división se termina y el dividendo se convierte en el resto.

Poniéndolo todo en conjunto tenemos:

$$x^3y^2 + xy + x = 0 \cdot (y^2 + 1) + (x^2y - x + 1) \cdot (xy + 1) + (2x - 1).$$

La necesidad de un resto bien definido en la división, que no dependa del orden en que vengan dados los elementos de F, es una de las motivaciones para la definición de las bases de Gröbner.

4.4. Ideales monomiales

En general el residuo que se obtiene a partir del algoritmo de la división depende del orden elegido y del orden en que se da la secuencia de polinomios por los que dividimos. Precisamente la idea de base de Gröbner viene a mitigar este problema: fijado el orden, no importa en qué orden usemos los polinomios de la base para realizar el algoritmo de la división; el resto siempre será el mismo. Tendremos además que el resto de dividir un polinomio por una base de Gröbner será cero si y solo si el polinomio está en el ideal generado por dicha base. Se tendrá por tanto resuelto el problema de la pertenencia y a la vez, por el Teorema de los ceros de Hilbert, el problema de la resolubilidad de un sistema de ecuaciones polinomiales.

La idea que hay debajo de la construcción de una base de Gröbner para un orden dado es, precisamente, añadir al conjunto de polinomios inicial las diferencias de los restos que surgen de cambiar el orden de la secuencia por la que dividimos.

La definición que vamos a dar se basa en la idea de ideal monomial.

Definición 4.9 *Un ideal monomial es un ideal generado por un conjunto de monomios.*

Por tanto, I es un ideal monomial si existe un subconjunto $A \subseteq \mathbb{N}^n$ de forma I consta de todos los polinomios que son sumas finitas de la forma $\sum_{\alpha \in A} h_\alpha x^\alpha$, donde $h_\alpha \in k[x_1, \ldots, x_n]$. Este hecho lo denotaremos como $I = \langle x^\alpha \mid \alpha \in A \rangle$. Por ejemplo, $I = \langle x^5y^2z, x^2yz^2, xy^3z^2 \rangle \subset k[x, y, z]$ es un ideal monomial.

Para todo ideal monomial se verifica que si x^β está en I, entonces x^β es divisible por algún x^α con $\alpha \in A$. Por otra parte, para cada polinomio f en un ideal monomial I, podemos decir que cada término de f está en I y que f es una combinación k-lineal de los monomios de I.

Sea $I \in k[x_1, \ldots, x_n]$ un ideal no nulo.

1. Sea $\mathrm{LT}(I)$ el conjunto de términos lideres de los elementos de I:

$$\mathrm{LT}(I) = \{cx^\alpha \mid \text{ tales que existe } f \in I \text{ con } \mathrm{LT}(f) = cx^\alpha\}.$$

2. Denotaremos con $\langle \mathrm{LT}(I) \rangle$ al ideal generado por los elementos de $\mathrm{LT}(I)$.

Así, por ejemplo, $\langle \mathrm{LT}(I) \rangle$ es un ideal monomial. Como veremos a continuación, si $I = \langle g_1, \ldots, g_s \rangle$, los ideales $\langle \mathrm{LT}(I) \rangle$ y $\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_s) \rangle$ no son siempre iguales. Aunque siempre tendremos $\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_s) \rangle \subset \langle \mathrm{LT}(I) \rangle$, hay casos en que la inclusión contraria no se cumple.

Ejemplo 4.2 *Sea $I = \langle f_1, f_2 \rangle$ donde $f_1 = x^3 - 2xy$ y $f_2 = x^3 - 2y^2 + x$ y usando orden lex. Entonces tomando*

$$f_3 := y \cdot (x^3 - 2xy) - (x^3y - 2y^2 + x) = -2xy^2 + 2y^2 - x$$

tenemos que $f_3 \in I$ y $\mathrm{LT}(f_3) = -2xy^2 \in \langle \mathrm{LT}(I) \rangle$. Sin embargo $-2xy^2 \notin \langle \mathrm{LT}(f_1), \mathrm{LT}(f_2) \rangle$ ya que no es divisible por el término líder de f_1 ni de f_2.

Precisamente, cuando se de la igualdad $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_s) \rangle$ diremos que $\{g_1, \ldots, g_s\}$ es una base de Gröbner para I (no tiene por qué ser única).

Definición 4.10 *Fijado un orden monomial en $k[x_1, \ldots, x_n]$. Un subconjunto finito $G = \{g_1, \ldots, g_t\}$ es una base de Gröbner si*

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_s) \rangle = \langle \mathrm{LT}(I) \rangle.$$

Como corolario del Teorema de la base de Hilbert aplicado a $\langle \mathrm{LT}(I) \rangle$ tenemos que cualquier ideal tiene una base de Gröbner.

Corolario 4.1 *Sea I un ideal polinomial distinto de cero, entonces I tiene una base de Gröbner.*

Si bien este corolario nos permite demostrar la existencia de una base de Gröbner, la prueba no es constructiva y nos ofrece poca información sobre como obtener una. Nos gustaría obtener un conjunto de generadores de forma que todos los términos lideres de los polinomios en el conjunto, generasen los términos lideres del ideal I. Esto falla cuando hay una cancelación de los términos lideres del tipo del ejemplo anterior. Para determinar mejor cuándo se produce esta cancelación construimos un polinomio especial que produce nuevos términos lideres.

Definición 4.11 *Sean $f, g \in k[x_1, \ldots, x_n]$ polinomios distintios de 0.*

1. *Si* $\mathrm{multideg}(f) = \alpha$ *y* $\mathrm{multideg}(g) = \beta$, *entonces sea* $\gamma = (\gamma_1, \ldots, \gamma_n)$, *donde* $\gamma_i = \max(\alpha_i, \beta_i)$ *para todo* i. *Llamaremos* x^γ *al mínimo común múltiplo de* $\mathrm{LT}(f)$ *y* $\mathrm{LT}(g)$, *y lo escribiremos* $x^\gamma = \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))$.

2. *El S-Polinomio de f y g es la combinación:*

$$S(f,g) = \frac{x^\gamma}{\mathrm{LT}(f)} \cdot f - \frac{x^\gamma}{\mathrm{LT}(g)} \cdot g.$$

Ejemplo 4.3 *Sean $f = x^4yz + x^2y^3z + xz$ y $g = 2x^2y^2z + xy^2 + xz^3$ en $\mathbb{Q}[x,y,z]$ con el orden lexicográfico en el monomios. Entonces $\gamma = (4,2,1)$ y tenemos:*

$$\begin{aligned} S(f,g) &= \frac{x^4y^2z}{x^4yz} \cdot f - \frac{x^4y^2z}{2x^2y^2z} \cdot g \\ &= y \cdot f - \frac{1}{2}x^2 \cdot g \\ &= -\frac{1}{2}x^3y^2 - \frac{1}{2}x^3z^3 + x^2y^4z + xyz. \end{aligned}$$

Nótese la cancelación de los términos lideres ocurrida por la construcción del S-polinomio. Una vez que una base contiene todos los posibles S-polinomios de los polinomios del conjunto de generación de ideales, no hay polinomios extra en $\langle \mathrm{LT}(I) \rangle$ que no estén en $\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_s) \rangle$. Esta es precisamente la idea del criterio de Buchberger para caracterizar bases de Gröbner.

Teorema 4.3 (Criterio de Buchberger) *Sea I un ideal de $k[x_1, \ldots, x_n]$. Entonces un sistema de generadores $G = \{g_1 \ldots, g_s\}$ de I es una base de Gröbner de I si y solo si para todos los pares $i \neq j$, el resto de la división de $S(g_i, g_j)$ entre G es cero.*

4.5. Algoritmo de Buchberger

Basándose en su criterio Buchberger dio un algoritmo para calcular una base de Gröbner, que se basa en ir añadiendo al sistema de generadores (y por tanto adjuntamos generadores redundantes) los restos de S-polinomios que no se anulen. El proceso, aunque es de complejidad doblemente exponencial en el número de variables, termina en un número finito de pasos.

Para una mayor claridad en nuestra discusión, introducimos la siguiente notación.

Escribimos $\overline{f}^G$ para el resto de la división de f por la lista de polinomios $G = \{g_1, \ldots, g_s\}$.

Por ejemplo si $G = \left(x^3y^2 - y^2z, xy^2 - yz\right)$ usando orden lex, tenemos:

$$\overline{x^5y^3}^G = yz^3$$

ya que tenemos por el algoritmo de la división

$$x^5y^3 = (x^2y) \cdot (x^3y^2 - y^2z) + (xyz + z^2) \cdot (xy^2 - yz) + yz^3.$$

Algoritmo de Buchberger. Fijado un orden monomial, la entrada es $F = \{f_1, \ldots, f_s\}$, y la salida es una base de Gröbner para $\langle F \rangle$.

```
def buchberger(F):
  G = list(F)
  pairs = []
  for i, f1 in enumerate(F):
    for f2 in F[i + 1:]:
      pairs.append((f1, f2))
  while pairs:
    f1, f2 = pairs.pop(0)
    s = s_polynomial(f1, f2)
    h = reduced(s, G)[1]

    if h != 0:
      for g in G:
        pairs.append((g, h))
      G.append(h)
  return G
```

Como hemos comentado arriba, Buchberger demuestra que este algoritmo termina en un número finito de pasos.

Ejemplo 4.4 *Consideramos el anillo* $k[x, y, z]$ *con orden lex y sean* $I = \langle -2xy + x, x^3y - 2x^2 + y\rangle$ *y* $F = (-2xy + x, x^3y - 2x^2 + y)$.

Como $S(f_1, f_2) = \frac{1}{2}x^3 - 2x^2 + y$ *y* $\overline{S(f_1, f_2)}^F = \frac{1}{2}x^3 - 2x^2 + y \neq 0$, *añadimos* $\overline{S(f_1, f_2)}^F$ *a* F *como nuevo generador* $f_3 = \frac{1}{2}x^3 - 2x^2 + y$. *Ahora*

comenzamos de nuevo, pero con $F = (f_1, f_2, f_3)$*:*

$$\begin{aligned} S(f_1, f_2) &= \frac{1}{2}x^3 - 2x^2 + y \\ \overline{S(f_1, f_2)}^F &= 0 \\ S(f_1, f_3) &= \frac{1}{2}x^3 - 4x^2y + 2y^2 \\ \overline{S(f_1, f_3)}^F &= 2y^2 - y \end{aligned}$$

Por tanto debemos añadir $f_4 = 2y^2 - y$ *a nuestro conjunto de generadores. Trabajando ahora con* $F = (f_1, f_2, f_3, f_4)$*:*

$$\begin{aligned} \overline{S(f_1, f_2)}^F &= \overline{S(f_1, f_3)}^F = 0 \\ S(f_2, f_3) &= -4x^2y + 2x^2 + 2y^2 - y = 2x \cdot (-2xy + x) + 1 \cdot (2y^2 - y) \end{aligned}$$

$$\begin{aligned} \overline{S(f_2, f_3)}^F &= 0 \\ S(f_2, f_4) &= -\tfrac{1}{2}x^3y + 2x^2y - y^2 \\ &= \left(\tfrac{1}{4}x^2 - x\right) \cdot (-2xy + x) - \tfrac{1}{2} \cdot \left(\tfrac{1}{2}x^3 - 2x^2 + y\right) \\ &\quad -\tfrac{1}{2} \cdot (2y^2 - y) \end{aligned}$$

$$\begin{aligned} \overline{S(f_2, f_4)}^F &= 0 \\ S(f_3, f_4) &= -\tfrac{1}{2}x^2y^2 - 2y^3 \\ &= \left(\tfrac{1}{4}x^2 - 2xy - x\right) \cdot (-2xy + x) \\ &\quad -\tfrac{1}{2} \cdot \left(\tfrac{1}{2}x^3 - 2x^2 + y\right) + \left(-y - \tfrac{1}{2}\right) \cdot (2y^2 - y) \end{aligned}$$

$$\overline{S(f_3, f_4)}^F = 0$$

Como $S(f_1, f_4) = 0$*, tenemos* $\overline{S(f_i, f_j)}^F = 0$ *para todo* i*,* j*, con* $1 \leq i \leq j \leq 4$*. Por el criterio de Buchberger obtenemos que*

$$F = (-2xy + x, x^3y - 2x^2 + y, \frac{1}{2}x^3 - 2x^2 + y, 2y^2 - y)$$

es una base de Gröbner para I*.*

Vamos a ver ahora que el algoritmo que se ha programado funciona correctamente para ello se ejecutará una versión de dicho algoritmo, que incluye salidas por pantalla, con datos iguales a los del ejemplo anterior para así poder comparar ambos. Esta es la salida que hemos obtenido:

Para los polinomios

$$f1 = -2xy + xf2 = x^3y - 2x^2 + y$$

Su S-Polinomio es:

$$-\frac{x^3}{2} + 2x^2 - y$$

Reducimos el S-Polinomio con respecto a G:

$$-\frac{x^3}{2} + 2x^2 - y$$

El nuevo sistema de generadores es:

$$\left[-2xy + x, \quad x^3y - 2x^2 + y, \quad -\frac{x^3}{2} + 2x^2 - y\right]$$

Para los polinomios

$$f1 = -2xy + x f2 = -\frac{x^3}{2} + 2x^2 - y$$

Su S-Polinomio es:

$$-\frac{x^3}{2} + 4x^2y - 2y^2$$

Reducimos el S-Polinomio con respecto a G:

$$-2y^2 + y$$

El nuevo sistema de generadores es:

$$\left[-2xy + x, \quad x^3y - 2x^2 + y, \quad -\frac{x^3}{2} + 2x^2 - y, \quad -2y^2 + y\right]$$

Para los polinomios

$$f1 = x^3y - 2x^2 + y f2 = -\frac{x^3}{2} + 2x^2 - y$$

Su S-Polinomio es:

$$4x^2y - 2x^2 - 2y^2 + y$$

Reducimos el S-Polinomio con respecto a G:

$$0$$

No hace falta añadirlo al sistema de generadores
Para los polinomios

$$f1 = -2xy + x f2 = -2y^2 + y$$

Su S-Polinomio es:

$$0$$

Reducimos el S-Polinomio con respecto a G:

$$0$$

No hace falta añadirlo al sistema de generadores
Para los polinomios

$$f1 = x^3y - 2x^2 + yf2 = -2y^2 + y$$

Su S-Polinomio es:

$$\frac{x^3y}{2} - 2x^2y + y^2$$

Reducimos el S-Polinomio con respecto a G:

$$0$$

No hace falta añadirlo al sistema de generadores
Para los polinomios

$$f1 = -\frac{x^3}{2} + 2x^2 - yf2 = -2y^2 + y$$

Su S-Polinomio es:

$$\frac{x^3y}{2} - 4x^2y^2 + 2y^3$$

Reducimos el S-Polinomio con respecto a G:

$$0$$

No hace falta añadirlo al sistema de generadores
La Base de Gröbner es:

$$\left[-2xy + x, \quad x^3y - 2x^2 + y, \quad -\frac{x^3}{2} + 2x^2 - y, \quad -2y^2 + y\right]$$

4.6. Teorema de Eliminación

El siguiente teorema nos permite triangular un sistema de ecuaciones polinomiales para poder resolverlo.

Teorema 4.4 (Teorema de Eliminación) *Sea $I \subseteq K[x_1, \ldots, x_n]$ un ideal y sea G una base de Gröbner de I con respecto al orden lexicográfico, donde $x_1 \succ x_2 \succ \cdots \succ x_n$. Entonces para todo entero l cumpliendo $1 \leq l \leq n$, el conjunto*

$$G_l = G \cap k\left[x_l, \ldots, x_n\right]$$

es una base de Gröbner de $I \cap k\left[x_l, \ldots, x_n\right]$.

Al calcular una base de Gröbner como en el teorema anterior, obtendríamos n bases de Gröbner. La última de ellas, G_n, que contendría polinomios en x_n, mientras que G_{n-1} tendría polinomios en x_n y x_{n-1}, y así sucesivamente. La idea es resolver los polinomios de G_n, que tienen una sola variable, ver que tienen soluciones comunes (o sólo hay un polinomio en esa variable), y substituir dichas soluciones en los polinomios de G_{n-1}. Volvemos a repetir el proceso con los nuevos polinomios en la variable x_{n-1}, y así sucesivamente hasta que no nos queden variables. Este proceso sirve para cuando hay un número finito de soluciones, pero también nos permite parametrizar soluciones a partir de un conjunto de variables.

Recordemos que si queremos ver si el sistema de ecuaciones polinomiales $f_1 = 0, \ldots, f_s = 0$ tiene solución, entonces basta con determinar si $1 \in \langle f_1, \ldots, f_s \rangle$. Esto lo podemos hacer determinando una base de Gröbner G para $\langle f_1, \ldots, f_s \rangle$ y comprobar si $\overline{1}^G = 0$, lo que equivale a que $1 \in G$.

Capítulo 5

Desarrollo de un ejemplo práctico

Para ver el uso de lo desarrollado a lo largo de esta memoria, se ha realizado un pequeño programa usando matplotlib que dibujará un brazo de dos articulaciones en un entorno 3D. Se podrá elegir mediante widgets tanto la posición final del brazo como si queremos que el brazo se mueva en el plano o en el espacio, es decir, si la articulación de la base es planar o universal. Se ha elegido usar matplotlib por la comodidad que proporciona en el uso de widgets y la buena representación 3D que realiza.

A continuación pasamos a explicar con detalle el código que hemos implementado.

Las primeras líneas de código importan la librerías que vamos a utilizar. Vamos a utilizar una única clase, Draw_Robot. El constructor (__init__) de la clase inicializa las variables que vamos a necesitar. A las longitudes (l) les hemos dado valor 100 y la posición inicial (tx,ty,tz) será (0,100,100).

```
1  from matplotlib.widgets import Slider, RadioButtons
2  from mpl_toolkits.mplot3d import Axes3D
3  import matplotlib.pyplot as plt
4  import numpy as np
5  from sympy import *
6
7  class Draw_Robot():
8      '''CONSTRUCTOR'''
9      def __init__(self):
10         '''VARIABLES'''
11         ARTICULACIONES = int(3)
12         self.l = np.array([0, 100, 100])
13         self.x = np.array([0]*ARTICULACIONES, dtype=float)
14         self.y = np.array([0]*ARTICULACIONES, dtype=float)
15         self.z = np.array([0]*ARTICULACIONES, dtype=float)
16
17         self.tx=0
```

```
18          self.ty=100
19          self.tz=100
20          self.estado_posible = True
21          self.modo = 0
```

Además precalculamos la base de Gröbner que nos servirá para resolver el sistema de ecuaciones para nuestro problema de cinemática inversa.

$$a - l[2] * (c1 * c2 - s1 * s2) - l[1] * c1$$
$$b - l[2] * (c1 * s2 + c2 * s1) - l[1] * s1$$
$$c1^2 + s1^2 - 1$$
$$c2^2 + s2^2 - 1$$

Usamos Gröbner para estas ecuaciones, indicando que el orden va a ser lex, que el sistema resultante va a depender de s1, c1, s2, c2 y que vamos a tomar a y b como parámetros, ya que sympy nos lo permite. De esta forma solo calcularemos una base de Gröbner para nuestro problema, luego solo tendremos que ir sustituyendo a y b, en cada caso.

Si quisiésemos añadir un segmento telescópico, tendríamos que poner su longitud "l" correspondiente como variable y añadirlo al campo R. Si hiciésemos esto, el tiempo de ejecución aumentaría.

```
1           '''CALCULOS PREVIOS GROEBNER'''
2           a,b,c1,c2,s1,s2= var("a,b,c1,c2,s1,s2")
3           eq= [a-self.l[2]*(c1*c2-s1*s2)-self.l[1]*c1, b-self.l
                [2]*(c1*s2+c2*s1)-self.l[1]*s1, c1**2+s1**2-1,c2**2+
                s2**2-1]
4           R = QQ.frac_field(a,b)
5           G = groebner(eq,c2,s2,c1,s1,order='lex',domain=R)
```

Este sería el resultado:

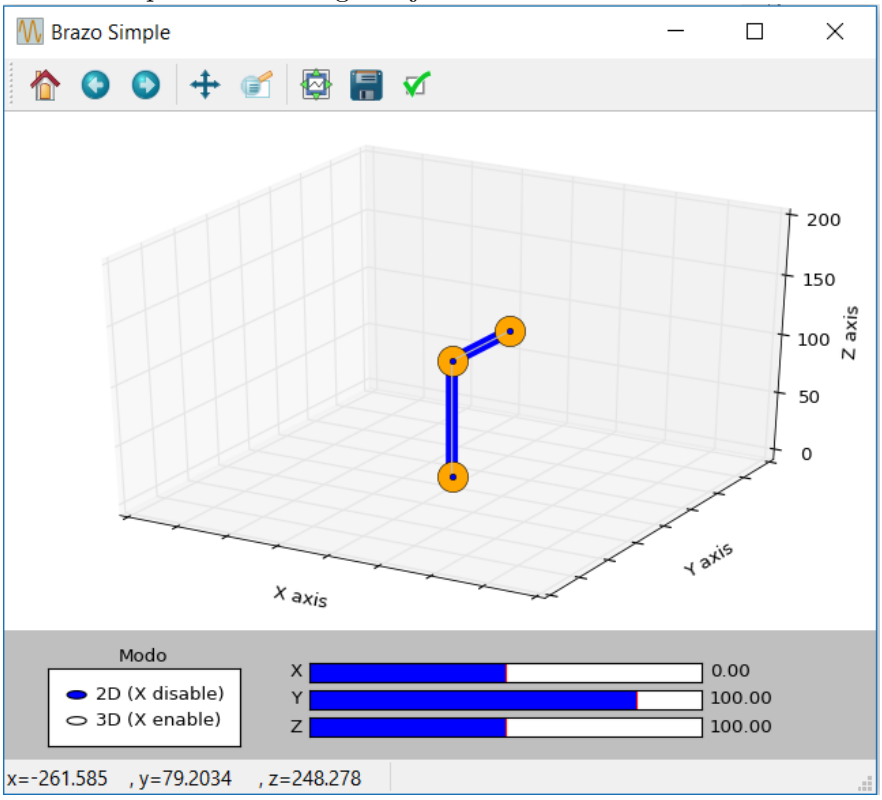
```
1   GroebnerBasis([
2   -a**2/20000 - b**2/20000 + c2 + 1,
3   s2 + s1*(a**2 + b**2)/(100*a) + (-a**2*b - b**3)/(20000*a),
4   c1 + b*s1/a + (-a**2 - b**2)/(200*a),
5   -b*s1/100 + s1**2 + (a**4 + 2*a**2*b**2 - 40000*a**2 + b**4)
        /(40000*a**2 + 40000*b**2)],
6   c2, s2, c1, s1, domain='QQ(a,b)', order='lex')
```

Como vemos hay 4 ecuaciones que, teniendo en cuenta que a,b,l2 y l3 son conocidos, forman un sistema de ecuaciones triangular en el que la primera solo depende de c2, la cuarta de c2 y s1, la segunda de c2, s1 y s2 y la tercera de c2, s1, s2 y c1.

A continuación vamos resolviendo el sistema y obteniendo c2, s1, s2 y c1; quedando estas en funcion de a, b, l2 y l3. Una vez tengamos esto, será solamente necesaria una simple sustitución para tener los datos que necesitamos para dibujar el brazo.

```
        self.c2_var = solve(G[0],c2)
        self.s1_var = solve(G[3].subs([(c2,self.c2_var)]),s1)
        self.s2_var = []
        self.s2_var.append(solve(G[1].subs([(c2,self.c2_var),(s1
            ,self.s1_var[0])]),s2)[0])
        self.s2_var.append(solve(G[1].subs([(c2,self.c2_var),(s1
            ,self.s1_var[1])]),s2)[0])
        self.c1_var = []
        self.c1_var.append(solve(G[2].subs([(c2,self.c2_var),(s1
            ,self.s1_var[0]),(s2,self.s2_var[0])]),c1)[0])
        self.c1_var.append(solve(G[2].subs([(c2,self.c2_var),(s1
            ,self.s1_var[1]),(s2,self.s2_var[1])]),c1)[0])
```

Pasamos a ver el diseño de la ventana principal del programa, que nos ayudara a comprender el código mejor:



Empezaremos creando la ventana(fig) con un nombre, y dentro de ella la ventana de dibujado(ax) y el panel de error(axerror).

```
        '''DIBUJADO'''
        self.fig = plt.figure("Brazo Simple")
        self.ax = plt.axes([0, 0.2, 1, .8], projection='3d')
        self.axerror = plt.axes([0.40, 0.92, 0.001, .001])
```

Ahora crearemos los widgets que vamos a usar que, como se ve en la

imagen, son tres Sliders, uno por coordenada y un Radio Button para cambiar de Modo, dejaremos el Slider de la X desactivado ya que inicialmente el Modo será 2D y por tanto se moverá en el plano X=0.

```
        '''WIDGETS'''
        #Slider de X
        axxval = plt.axes([0.35, 0.12, 0.45, 0.03])
        self.x_val = Slider(axxval, 'X', -150, 150, valinit=self
            .tx)
        self.x_val.set_active(False)

        #Slider de Y
        axyval = plt.axes([0.35, 0.0775, 0.45, 0.03])
        y_val = Slider(axyval, 'Y', -150, 150, valinit=self.ty)

        #Slider de Z
        axzval = plt.axes([0.35, 0.035, 0.45, 0.03])
        z_val = Slider(axzval, 'Z', 0, 200, valinit=self.tz)

        #Radio buttons de modo
        rax = plt.axes([0.05, 0.02, 0.22, 0.12])
        rax.set_title('Modo', fontsize=12)
        set_modo = RadioButtons(rax, ('2D (X disable)', '3D (X
            enable)'), active=0)
```

Después añadimos los manejadores de eventos de los 4 widgets. Cuando el valor de alguno de los Sliders se actualizará el valor de las variables de clase tx, ty o tz, según de cual se haya modificado el valor, y se llamará a la función draw_robot, la cuál veremos más adelante. En el caso del manejador del Radio Button del Modo, este activará o desactivará el Slider X y cambiará la variable de clase modo.

```
        '''MANEJADORES'''
        #X slider event
        def actualiza_val_x(val):
            self.tx = val
            self.draw_robot()
        self.x_val.on_changed(actualiza_val_x)

        #Y slider event
        def actualiza_val_y(val):
            self.ty= val
            self.draw_robot()
        y_val.on_changed(actualiza_val_y)

        #Z slider event
        def actualiza_val_z(val):
            self.tz = val
            self.draw_robot()
        z_val.on_changed(actualiza_val_z)

```

```
20          #Radio Button Modo
21          def seleccion_modo(label):
22              if label==('2D (X disable)'):
23                  self.modo = 0
24                  self.x_val.set_active(False)
25                  self.x_val.reset()
26              if label==('3D (X enable)'):
27                  self.modo = 1
28                  self.x_val.set_active(True)
29          set_modo.on_clicked(seleccion_modo)
```

Ya solo nos queda llamar a display_error, a drow_robot y hacemos show para dibujarlo todo.

```
1
2           self.display_error()
3           self.draw_robot()
4
5           plt.show()
```

Y con esto habríamos terminado el constructor. Ahora escribimos algunas funciones de la clase, empezando por preparar una señal de error que nos indicara cuando se le a solicitado al brazo acceder a una posición fuera de su alcance.

```
1       def display_error(self):
2           self.axerror.set_visible(False)
3           self.axerror.set_yticks([])
4           self.axerror.set_xticks([])
5           self.axerror.set_navigate(False)
6           self.axerror.text(0, 0, 'El brazo no alcanza esa
                posicion!', style='oblique',
7                             bbox={'facecolor':'red', 'alpha':0.5, 'pad
                                  ':10}, size=20, va = 'baseline')
```

La función siguiente es la encargada de, usando todo lo preparado en el constructor y los valores introducidos por los Sliders, obtener los valores de c1,s1,c2,s2 y una vez comprobado que son números reales, si son imaginarios activamos el display_error, actualizar los valores de y[1] y z[1]. Para hacer esto usaremos, de los dos valores posibles de s1, el positivo para así evitar problemas al cambiar de cuadrante. También situaremos y[2] y z[2] en la posición elegida en los Sliders.

Tanto x[1], como x[2] serán puestas a 0 ya que esta función es para el modo 2D.

```
1       def calcula_groebner2D(self):
2           s1,s2,c1= [],[],[]
3
```

```
        s1.append(self.s1_var[0].subs([(a, self.ty), (b, self.tz
            )]))
        s1.append(self.s1_var[1].subs([(a, self.ty), (b, self.tz
            )]))
        s2.append(self.s2_var[0].subs([(a, self.ty), (b, self.tz
            )]))
        s2.append(self.s2_var[1].subs([(a, self.ty), (b, self.tz
            )]))
        c1.append(self.c1_var[0].subs([(a, self.ty), (b, self.tz
            )]))
        c1.append(self.c1_var[1].subs([(a, self.ty), (b, self.tz
            )]))

        if(im(c1[0])==0 and im(s1[0])==0 and im(s2[0])==0):
            self.estado_posible = True

            self.x[1] = 0
            if (s1[0]*self.l[1]>0):
                self.y[1] = c1[0]*self.l[1]
                self.z[1] = s1[0]*self.l[1]
            else :
                self.y[1] = c1[1]*self.l[1]
                self.z[1] = s1[1]*self.l[1]

            self.x[2] = 0
            self.y[2] = self.ty
            self.z[2] = self.tz

        else :
            self.estado_posible = False
```

Para el caso en 3D vamos a usar un cambio a coordenadas esféricas, esto es:

$$\begin{cases} x = r \cdot sin(\theta) \cdot cos(\varphi) \\ y = r \cdot sin(\theta) \cdot sin(\varphi) \\ z = r \cdot (\varphi) \end{cases} , \; siendo \; \varphi = \begin{cases} arctan\left(\frac{y}{x}\right) & x > 0 \; y \; y > 0 \\ 2\pi + arctan\left(\frac{y}{x}\right) & x > 0 \; y \; y < 0 \\ \frac{\pi}{2} \cdot sgn(y) & x = 0 \\ \pi + arctan\left(\frac{y}{x}\right) & x < 0 \end{cases}$$

```
    def calcula_groebner3D(self):
        dist = sqrt(self.tx**2+self.ty**2)
        s1,s2,c1= [],[],[]

        s1.append(self.s1_var[0].subs([(a, dist), (b,self.tz)]))
        s1.append(self.s1_var[1].subs([(a, dist), (b,self.tz)]))
        s2.append(self.s2_var[0].subs([(a, dist), (b,self.tz)]))
        s2.append(self.s2_var[1].subs([(a, dist), (b,self.tz)]))
        c1.append(self.c1_var[0].subs([(a, dist), (b,self.tz)]))
        c1.append(self.c1_var[1].subs([(a, dist), (b,self.tz)]))

        if(im(c1[0])==0 and im(s1[0])==0 and im(s2[0])==0):
            self.estado_posible = True
```

```
14
15              if(self.tx>0 and self.ty>0):
16                  fi = atan(self.ty/self.tx)
17              elif(self.tx>0 and self.ty<0):
18                  fi = atan(self.ty/self.tx) + 2*pi
19              elif(self.tx<0):
20                  fi = atan(self.ty/self.tx) + pi
21              elif(self.tx==0 and self.ty>0):
22                  fi = pi/2
23              else:
24                  fi = -pi/2
25
26              if ( s1[0]*self.l[1] > 0 ):
27                  self.x[1] = self.l[1]*c1[0]*cos(fi)
28                  self.y[1] = self.l[1]*c1[0]*sin(fi)
29                  self.z[1] = self.l[1]*s1[0]
30              else:
31                  self.x[1] = self.l[1]*c1[1]*cos(fi)
32                  self.y[1] = self.l[1]*c1[1]*sin(fi)
33                  self.z[1] = self.l[1]*s1[1]
34
35              self.x[2] = self.tx
36              self.y[2] = self.ty
37              self.z[2] = self.tz
38
39          else :
40              self.estado_posible = False
```

A continuación vemos las funciones set_positions y set_ax. La primera será la que dibuje el robot, y la segunda la que dibuje el sistema de coordenadas.

```
1
2       def set_positions(self):
3           xs = np.array(self.x).tolist()
4           ys = np.array(self.y).tolist()
5           zs = np.array(self.z).tolist()
6           self.ax.cla()
7           self.ax.plot(xs, ys, zs, 'o-', markersize=20,
8                        markerfacecolor="orange", linewidth = 8,
                             color="blue")
9           self.ax.plot(xs, ys, zs, 'o-', markersize=4,
10                       markerfacecolor="blue", linewidth = 1,
                             color="silver")
11
12      def set_ax(self):
13          self.ax.set_xlim3d(-200, 200)
14          self.ax.set_ylim3d(-200, 200)
15          self.ax.set_zlim3d(-5, 200)
16          self.ax.set_xlabel('X axis')
17          self.ax.set_ylabel('Y axis')
18          self.ax.set_zlabel('Z axis')
19          for j in self.ax.get_xticklabels() + self.ax.
```

```
            get_yticklabels():
20              j.set_visible(False)
21          self.ax.set_axisbelow(True)
```

Por último tenemos la función draw_robot la cual tendrá en cuenta el modo para calcular Gröbner en 2D o en 3D y, tras estos cálculos, ver si estamos ante un estado posible o no. Si el estado es posible no dibujaremos la señal de error y dibujaremos tanto los ejes como el brazo y en caso contrario, solo dibujaremos el error. Por ultimo hacemos draw para dibujar todo.

```
1
2       def draw_robot(self):
3           if self.modo==0:
4               self.calcula_groebner2D()
5           else:
6               self.calcula_groebner3D()
7
8           if self.estado_posible:
9               self.axerror.set_visible(False)
10              self.set_positions()
11              self.set_ax()
12          else:
13              self.axerror.set_visible(True)
14          plt.draw()
```

Y ya fuera de la clase, tenemos el main que lo único que hace es crear un objeto Draw_robot.

```
1   def main():
2       myRobot=Draw_Robot()
3
4   if __name__ == '__main__':
5       main()
```

Capítulo 6

Conclusiones

A lo largo del documento hemos podido comprobar como se han cumplido todos los objetivos iniciales tanto los matemáticos como los informáticos.

Los primeros objetivos en completarse fueron los matemáticos, ya que sin la base matemática no podía comenzar con el programa, aun así, mientras intentaba comprender todos los conceptos matemáticos, fui aprendiendo a usar Python. Por tanto, una vez terminada la parte matemática, la realización de la aplicación no resulto complicada, siendo lo más nuevo el uso de matplotlib.

Desde mi punto de vista, el trabajo ha resultado interesante, ya que he conseguido aunar en un mismo proyecto matemáticas e informática.

En cuanto a la parte matemática, el aprender un tema como las Bases de Gröbner por mi mismo, aunque siempre con la ayuda de mi tutor, me ha hecho ampliar, además de mi conocimiento, mi capacidad de comprensión de las matemáticas. El trabajar con temas con los que no he tratado a lo largo de la carrera me ha servido como motivación para el desarrollo de este trabajo.

En cuanto a la parte informática, el aprender un nuevo lenguaje es siempre bueno, más aun si es un lenguaje que se está usando en la actualidad en el mundo empresarial. He podido realizar, gracias a poder trabajar con este lenguaje, algunos otros proyectos que tenía en mente. Aprendí a usar la librería Sympy, o al menos la parte que usaré de ella, ya que es muy extensa, ver lo bien que se integra el calculo simbólico en un lenguaje en principio no matemático me pareció muy interesante.

Bibliografía

[1] David A. Cox, John Little y Donal O'Shea, *Ideals, Varieties, and Algorithms, An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Undergraduate Texts in Mathematics, Springer, 2015.

[2] Allen B. Downey, *Think Python*, Green Tea Press, versión 2.0.17, `http://greenteapress.com/wp/think-python`.

[3] J. D. Hunter, Matplotlib: A 2D graphics environment, Computing In Science & Engineering, **9**, (2007) 90–95.

[4] SymPy Development Team, SymPy: Python library for symbolic mathematics, `http://www.sympy.org`, 2016.