TRABAJO FIN DE GRADO

Doble Grado en Ingeniería Informática y Matemáticas

Groebner-Robotics

Subtitulo del Proyecto

Autor
Antonio Solís Izquierdo (alumno)

Director
Pedro A. García-Sánchez (tutor)

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mes de 2016

Groebner-Robotics

Subtítulo del proyecto.

Autor
Antonio Solís Izquierdo (alumno)

Director
Pedro A. García-Sánchez (tutor)

Título del Proyecto: Groebner-Robotics

Antonio Solís Izquierdo (alumno)

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

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 .

D. **Pedro A. García-Sánchez**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Antonio Solís Izquierdo (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

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 sólo 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. La de revolución permite una rotación de un segmento con respecto al 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. La telescópica permite a una parte del brazo moverse a lo largo de un eje.

Si hay varias articulaciones en un robot, vamos a suponer para simplificar, 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 $\mathbf{R}^3$ y articulaciones helicoidales o de "tornillo"que son combinaciones de rotación y traslación a lo largo del eje de rotación en $\mathbf{R}^3$. 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 vamos a estudiar, pero no vamos a entrar en detalle sobre ninguno de ellos.

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 $\mathbf{S}^1$ 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 $\mathbf{S}^1$.) 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

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

donde hay un factor $\mathbf{S}^1$ para cada articulación de revolución, y cada I_j da la configuración de la articulación telescopica 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 $U \subseteq \mathbf{R}^2$. 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 $V = S^1$.

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 determinada, 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 qué se podrían considerar en este punto ya que pueden ser descritos sucintamente en términos de la asignación $f : \mathcal{J} \longrightarrow \mathcal{C}$ descrita 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 $c \epsilon \mathcal{C}$, ¿podemos determinar una o todas las $j \epsilon \mathcal{J}$ tal que $f(j) = c$?

Si estudiásemos el primero veríamos que el problema se resuelve con relativa facilidad. La determinación de 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 corresponden a las posiciones que son no físicamente alcanzables. 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.

Capítulo 2

Cinemática Directa

En esta sección, presentaremos un método estándar para resolver el problema de la cinemática directa para un "brazo". Todos nuestros robots tendrán un primer segmento que está anclado o fijado en posición. En otras palabras, no existe una articulación en el punto inicial del segmento 1. Con esta convención, vamos a colocar el origen de nuestro sistema de coordenadas en la articulación que une los dos primeros segmentos del brazo, cuya posición también 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 revolucióni, introducimos un sistema de coordenadas (x_{i+1}, y_{i+1}) de la siguiente manera. El origen se sitúa en la articulación i. Colocamos la dirección de las x positivas a largo del segmento $i + 1$. Por tanto, para cada $i \geq 2$, las coordenadas de la articulación 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) las coordenadas 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) , osea $q = (a_i, b_i)$, primero hacemos un giro de ángulo θ_i (para alinear los ejesx_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).

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 $3x3$ y vectores de 3 componentes:

$$\begin{pmatrix} a_i \\ b_i \\ 1 \end{pmatrix} = \begin{pmatrix} cos\theta_i & -sin\theta_i & l_i \\ sin\theta_i & cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{i+1} \\ b_{i+2} \\ 1 \end{pmatrix} = A_i \cdot \begin{pmatrix} a_{i+1} \\ b_{i+2} \\ 1 \end{pmatrix} \tag{2.1}$$

Esto nos permite combinar la rotación θ_i con la traslación a lo largo del segmento i en una sola matriz $3x3$ A_i.

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 A3, A2, A1 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

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} cos(\theta_1+\theta_2+\theta_3) & -sin(\theta_1+\theta_2+\theta_3) & l_3cos(\theta_1+\theta_2)+l_2cos\theta_1 \\ sin(\theta_1+\theta_2+\theta_3) & cos(\theta_1+\theta_2+\theta_3) & l_3sin(\theta_1+\theta_2)+l_2sin\theta_1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_4 \\ y_4 \\ 1 \end{pmatrix}$$

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

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} l_3cos(\theta_1+\theta_2)+l_2cos\theta_1 \\ l_3sin(\theta_1+\theta_2)+l_2sin\theta_1 \\ 1 \end{pmatrix}$$

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

$$f(\theta_1, \theta_2, \theta_3) = \begin{pmatrix} l_3 cos(\theta_1 + \theta_2) + l_2 cos\theta_1 \\ l_3 sin(\theta_1 + \theta_2) + l_2 sin\theta_1 \\ \theta_1 + \theta_2 + \theta_3 \end{pmatrix} \tag{2.2}$$

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

$$g(\theta_1, \theta_2, \theta_3, l_4) = \begin{pmatrix} l_4 cos(\theta_1 + \theta_2 + l_3) + l_3 cos(\theta_1 + \theta_2) + l_2 cos\theta_1 \\ l_4 sin(\theta_1 + \theta_2 + l_3) + l_3 sin(\theta_1 + \theta_2) + l_2 sin\theta_1 \\ \theta_1 + \theta_2 + \theta_3 \end{pmatrix} \tag{2.3}$$

Como antes, l_2 y l_3 son constantes, pero $l_4 \epsilon\, [m_1, m_2]$ 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.

A continuación vamos a discutir cómo fórmulas tales como las anteriores pueden ser convertidos en representaciones de f y g como asignaciones polinómicas o racionales en las variables adecuadas. 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 polinomio es usar la idea de que aunque el coseno y el seno son funciones trascendentales, dan una parametrización

$$x = cos\theta,$$

$$y = sen\theta$$

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 2.2 o, de forma equivalente, las entradas de la matriz $A_1A_2A_3$ en 2.3 como funciones de

$$c_i = cos\theta_i,$$

$$s_i = sen\theta_i$$

sujetas a las restricciones

$$c_i^2 + s_i^2 - 1 = 0 \tag{2.4}$$

para $i = 1, 2, 3$. Tenga en cuenta que la variedad definida por estos tres ecuaciones en $\mathcal{R}^6$ 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 2.2 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:

$$\begin{pmatrix} l_3(c_1c_2 - s_1s_2) + l_2c1 \\ l_3(s_1c_2 + s_2c_1) + l_2s1 \end{pmatrix}. \tag{2.5}$$

Capítulo 3

Cinemática Inversa

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 $(x_1, y_1) = (a, b) \in \mathbf{R}^2$ 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 2.5 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:

$$\begin{aligned} a &= l_3(c_1c_2 - s_1s_2) + l_2c_1, \\ b &= l_3(c_1s_2 - c_2s_1) + l_2s_1, \\ 0 &= c_1^2 + s_1^2 - 1 \\ 0 &= c_2^2 + s_2^2 - 1 \end{aligned} \tag{3.1}$$

para c_1, s_1, c_2, s_2. Para resolver estas ecuaciones vamos a usar bases de Gröbner.

Capítulo 4

Bases de Gröbner.

4.1. Definiciones y Lemas Importantes.

Antes de entrar de lleno en la Bases de Gröbner definamos algunos conceptos necesarios.

Definición 1. Un monomio en $x_1, ..., x_n$ es un producto de la forma

$$x_1^{\alpha_1} \cdot x_2^{\alpha_2} \dots x_n^{\alpha_n},$$

donde todos los exponentes $\alpha_1, ..., \alpha_n$ son enteros no negativos. El grado total de este monomio es la suma $\alpha_1 + ... + \alpha_n$.

Definición 2. Un polinomio f en $x_1, ..., x_n$ con coeficientes en un campo k es una combinación lineal finita(con coeficientes en k) de monomios. Escribiremos un polinomio f de la forma

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

donde la suma es sobre un número finito de n-tuplas $\alpha = (\alpha_1, ..., \alpha_n)$. El conjunto de todos los polinomios en $x_1, ..., x_n$ con coeficientes en k se denota $k[x_1, ..., x_n]$.

Definición 3. Sea $f = \sum_\alpha a_\alpha x^\alpha$ un polinomio en $k[x_1, ..., x_n]$.

(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.

Definición 4. Dado un campo k y un número entero positivo n, se define el espacio afín de dimensión n sobre k como el conjunto

$$k^n = (a_1, ..., a_n) | a_1, ..., a_n \in k.$$

Definición 5. Sea k un campo, y sean $f_1, ..., f_n$ polinomios en $k[x_1, ..., x_n]$. Entonces

$$V(f_1, ..., f_s) = (a_1, ..., a_n) \in k^n | f_i(a_1, ..., a_n) = 0 \forall 1 \leq i \leq s.$$

Llamamos $V(f_1, ..., f_s)$ a la variedad afín definida por $f_1, ..., f_s$.

Definición 6. Un subconjunto $I \subseteq k[x_1, ..., x_n]$ es un ideal si satisface:

(i) $0 \in I$.

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

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

Definición 7. Sean $f_1, ..., f_s$ polinomios en $k[x_1, ..., x_n]$. Entonces

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

El hecho crucial es que $\langle f_1, ..., f_s \rangle$ es un ideal.

Lema 1. Si $f_1, ..., f_s \in k[x_1, ..., x_n]$, entonces $\langle f_1, ..., f_s \rangle$ es un ideal de $k[x_1, ..., x_n]$. Llamaremos a $\langle f_1, ..., f_s \rangle$ el ideal generado por $f_1, ..., f_s$.

Proposición 1. Si $f_1, ..., f_s and g_1, ..., g_s$ son bases del mismo ideal en $k[x_1, ..., x_n]$, tal que $\langle f_1, ..., f_s \rangle = \langle g_1, ..., g_t \rangle$, entonces tenemos que $V(f_1, ..., f_s) = V(g_1, ..., g_t)$.

Definición 8. Sea $V \subseteq k^n$ una variedad afín. Entonces

$$I(V) = f \in k[x_1, ..., x_n] \mid f(a_1, ..., a_n) = 0 \forall (a_1, ..., a_n) \in V.$$

La observación crucial es que $I(V)$ es un ideal.

Lema 2. Si $V \subseteq k^n$ es una variedad afín, entonces $I(V) \subseteq k[x_1, ..., x_n]$ es un ideal. Llamaremos a $I(V)$ el ideal de V.

4.2. Ordenamiento de monomios en $k[x_1, ..., x_n]$

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.

Definición 1. Un orden monomial $>$ en $k[x_1, ..., x_n]$ es una relación $>$ en $\mathbb{Z}^n_{\geq 0}$, o equivalentemente, una relación en el conjunto de monomios $x^\alpha, \alpha \in \mathbb{Z}^n_{\geq 0}$, cumpliendo:

(i) $>$ es un orden total(o lineal) en $\mathbb{Z}^n_{\geq 0}$.

(ii) Si $\alpha > \beta$ y $\gamma \in \mathbb{Z}^n_{\geq 0}$, entonces $\alpha + \gamma > \beta + \gamma$.

(iii) $>$ es un buen orden en $\mathbb{Z}^n_{\geq 0}$. Esto significa que cada subconjunto no vacío de $\mathbb{Z}^n_{\geq 0}$ tiene un elemento más pequeño bajo $>$. En otras palabras, si $A \subseteq \mathbb{Z}^n_{\geq 0}$ es no vacío, entonces existe $\alpha \in A$ tal que $\beta > \alpha$ para cada $\beta \neq \alpha$ en A.

Dado un orden monomial $>$, se dice que $\alpha \geq \beta$ cuando $\alpha > \beta$ o $\alpha = \beta$.

Definición 2. (Orden lexicográfico). Sea $\alpha = (\alpha_1, ..., \alpha_n)$ y $\beta = (\beta_1, ..., \beta_n)$ pertenecientes a $\mathbb{Z}^n_{\geq 0}$. Decimos $\alpha >_{lex} \beta$ si el elemento no nulo más a la izquierda del vector diferencia $\alpha - \beta \in \mathbb{Z}^n$ es positivo. Escribiremos $x^\alpha >_{lex} x^\beta$ si $\alpha >_{lex} \beta$.

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 $(x > y > z)$ se utiliza para definir el orden lexicográfico a menos que digamos lo contrario de forma explícita.

Proposición 1. El orden léxico en $\mathbb{Z}^n_{\geq 0}$ es un orden monomial.

Para el orden léxico con $x > y > z$, tenemos $x >_{lex} y^5$. 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 el orden lexicográfico graduado(u orden grlex).

Definición 3. (Orden lexicográfico graduado). Sean $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. Decimos $\alpha >_{grlex} \beta$ si

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \quad \text{or} \quad |\alpha| = |\beta| \, and \, \alpha >_{lex} \beta.$$

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

Definición 4. (Orden lexicográfico graduado inverso). Sean $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. Decimos $\alpha >_{grevlex} \beta$ si

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ y el elemento no negativo mas a la derecha de } \alpha - \beta \in \mathbb{Z}^n_{\geq 0} \text{ es negativo.}$$

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 $f = 4xy^2z + 7x^2z^2 - 5x^3 + 4z^2$

Definición 5. Sea $f = \sum_\alpha a_\alpha x^\alpha$ un polinomio distinto de cero en $k[x_1, ..., x_n]$ y sea $>$ un orden monomial.

(i) El multigrado de f es

$$\text{multideg}(f) = \max(\alpha \in \mathbb{Z}^n_{\geq 0} | a_\alpha \neq 0)$$

(El máximo se toma con respecto a $>$).

(ii) El coeficiente principal de f es

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

(ii) El monomio principal de f es

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

(Con coeficiente 1).

(ii) El término líder de f es

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

Para ver esto usemos el ejemplo anterior, sea $f = 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2$ y denotamos por $>$ el orden lexicográfico. Entonces

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

El multigrado tiene las siguientes propiedades útiles.

Lema 1. Sea $f, g \in k[x_1, ..., x_n]$ un polinomio distinto de cero. Entonces:

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

(ii) Si $f+g \neq 0$, entonces $\text{multideg}(f+g) \leq \max(\text{multideg}(f), \text{multideg}(g))$. Si, además, $\text{multideg}(f) \neq \text{multideg}(g)$, entonces se produce la igualdad.

4.3. Un Algoritmo de División en $k[x_1, ..., x_n]$

Sabemos que el algoritmo de la división se podría utilizar para resolver el problema de la pertenencia a los ideales para polinomios de una variable. Para estudiar este problema cuando hay más variables, vamos a formular un algoritmo de la división de polinomios en $k[x_1, ..., x_n]$ 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).

Teorema 1. (Algoritmo de la División en $k[x_1, ..., x_n]$). Sea $>$ un orden monomial en $\mathbb{Z}^n_{\geq 0}$, y sea $F = (f_1, ..., f_s)$ una s-tupla ordenada de polinomios en $k[x_1, ..., x_n]$. Entonces, cada $f \in k[x_1, ..., x_n]$ se puede escribir como

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

donde $q_i, r \in k[x_1, ..., x_n]$, 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 LT(f_1),...,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

$$multideg(f) \geq multideg(q_i f_i).$$

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

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

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

Poniendolo 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 es una de las motivaciones para la definición de las bases de Gröbner.

4.4. Ideales monomiales

En general no se obtiene un residuo determinar claramente a partir del algoritmo de la división. Sin embargo, la definición posterior de una base de Gröbner tendrá la cualidad de que la división de f por G da el mismo resto r sin importar como estén clasificados los elementos de G en la división. Puesto que vamos a demostrar que todo ideal tiene una base de Gröbner, somos capaces de resolver el problema de pertenencia de ideales con una condición necesaria y suficiente para que un polinomio f sea un miembro de un ideal I, que la división de f por la base de Gröbner de I devuelva resto 0.

Definición 1. Un ideal monomial es un ideal generado por un conjunto de monomios.

Es decir, I es un ideal monomial si hay un subconjunto $A \subset Z^n_{n\geq 0}$ tal que I consta de todos los polinomios que son sumas finitas de la forma $\sum_{\alpha \in A} h_\alpha x^\alpha$, donde $h_\alpha \in k[x_1, \ldots, x_n]$. Nosotros notaremos $I = \langle x^\alpha | \alpha \in A \rangle$. Por ejemplo, $I = \langle x^5y^2z, x^2yz^2, xy^3z^2 \rangle \subset k[x, y, z]$ es un ideal monomial. Para todos los ideales monomiales tenemos el hecho de que si x^β está en I, entonces x^β es divisible por x^α para algún $\alpha \in A$. 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.

Definición 2. Sea $I \in k[x_1, \ldots, x_n]$ un ideal no nulo.

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

$$\mathrm{LT}(I) = \{cx^\alpha | \text{ entonces existe } f \in I \text{ con } \mathrm{LT}(f) = cx^\alpha\}$$

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

Así, por ejemplo, $\mathrm{LT}(I)$ es un ideal monomial. Como veremos a continuación 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. Por ejemplo consideremos el siguiente ejemplo:

Ejemplo. 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:

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

así que $f_3 \in I$ y LT(f_3)=$-2xy^2 \in \langle \text{LT}(I)\rangle$,pero no en $\langle \text{LT}(f_1),\text{LT}(f_2)\rangle$ ya que no es divisible por los terminos lideres de f_1 o f_2.

Dado que nuestro objetivo es la obtención de ideales que cumplan la propiedad de que $\langle \text{LT}(I)\rangle = \langle \text{LT}(g_1),\ldots,\text{LT}(g_s)\rangle \subset$, se quieren eliminar los casos como el de arriba, asegurándose de que nuestra base genera $\langle \text{LT}(I)\rangle$. Esto motiva la siguiente definición:

Definición 3. 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

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

Como corolario del Teorema de la base de Hilbert aplicado a $\langle \text{LT}(I)\rangle$ tenemos:

Corolario 1. Sea I un ideal polinomial distinto de cero, entonces I tiene una base de Gröbner.

Si bien este corolario nos permite comenzar a 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 cuando esta cancelación se produce construimos un polinomio especial que produce nuevos términos lideres.

Definición 4. Sean $f, g \in k[x_1,\ldots,x_n]$ polinomios distintios de 0.

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

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

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

Ejemplo. 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$\langle \mathrm{LT}(g_1),\ldots,\mathrm{LT}(g_s)\rangle = \langle \mathrm{LT}(I)\rangle$. Esto lleva a:

Teorema 1.(Criterio de Buchberger) Sea I un ideal polinomial. Entonces una base $G = \{g_1 \ldots, g_s\}$ de I es una base de Gröbner de I si y solo si para todos los pares $i \neq j$, el resto de la division de $S(g_i, g_j)$ entre G es cero.

4.5. Bases de Gröbner y Algoritmo de Buchberger

A continuación vamos a dar la formulación del algoritmo de Buchberger y un ejemplo de su uso para calcular una base de Gröbner. Para una mayor claridad en nuestra discusión, introducimos la siguiente notación:

Definición. 1 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

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

Teorema 1.(Algoritmo de Buchberger) Sea $I = \langle f_1, \ldots, f_s\rangle \neq (0)$ sea un ideal polinomial. Entonces una base de Gröbner para I se puede construir en un numero de pasos finito.

El algoritmo funciona de la siguiente manera: Sean $F = (f_1, \ldots, f_s)$ una lista de los polinomios que definen I. Para cada par de polinomios f_i, f_j en F se calcula su S-polinomio, S, y se divide por los polinomios $f_1, \ldots, f_s$ en F obteniendo $\overline{S}^F$. Si $\overline{S}^F \neq 0$, añadimos $\overline{S}^F$ a F y empezar de nuevo con $F = F \cup \{\overline{S}^F\}$. Se repite el proceso hasta que todos los S-polinomios de polinomios en F tengan resto 0 después de dividir por F.

Ejemplo. 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 añadiendolo 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. Trabando ahora con $F = (f_1, f_2, f_3, f_4)$:

$$
\begin{aligned}
\overline{S(f_1, f_2)}^F &= \overline{S(f_1, f_3)}^F = 0 \\
S(f_2, f_3) &= -4x^2y + 2x^2 + 2y^2 - y = 2x \cdot (-2xy + x) + 1 \cdot (2y^2 - y) \\
\overline{S(f_2, f_3)}^F &= 0 \\
S(f_2, f_4) &= -\frac{1}{2}x^3y + 2x^2y - y^2 = \left(\frac{1}{4}x^2 - x\right) \cdot (-2xy + x) - \frac{1}{2} \cdot \left(\frac{1}{2}x^3 - 2x^2 + y\right) - \frac{1}{2} \cdot (2y^2 - y) \\
\overline{S(f_2, f_4)}^F &= 0 \\
S(f_3, f_4) &= -\frac{1}{2}x^2y^2 - 2y^3 = \left(\frac{1}{4}x^2 - 2xy - x\right) \cdot (-2xy + x) - \frac{1}{2} \cdot \left(\frac{1}{2}x^3 - 2x^2 + y\right) + \left(-y - \frac{1}{2}\right) \cdot (2 \\
\overline{S(f_3, f_4)}^F &= 0
\end{aligned}
$$

Como $S(f_1, f_4) = 0$, tenemos $\overline{S(f_i, f_j)}^F \forall 1 \leq i \leq j \leq 4$. Por el criterio de Buchberger obtenemos que $F = (f_1, f_2, f_3, f_4) = (-2xy + x, x^3y - 2x^2 + y, \frac{1}{2}x^3 - 2x^2 + y, 2y^2 - y)$ es una base de Gröbner para I.

Capítulo 5

Desarrollo.

5.1. Algoritmo de Buchberger.

Para un mejor entendimiento del Algoritmo de Buchberger pensé que lo mejor sería programarlo para asi ver paso a paso su funcionamiento, con esta meta y una vez programado hice una versión que me diera una salida por pantalla que me permitiese hacer un seguimiento en cada iteración del Algoritmo. A continuación incluyo las dos versiones que he realizado asi como la salida por pantalla de un ejemplo para ver funcionar el algoritmo.

5.1.1. Versión para importar y usar.

```
from sympy import *

def s_polynomial(f, g):
  return expand(lcm(LM(f),LM(g))*(1/LT(f)*f-1 /LT(g)*g))

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
```

5.1.2. Versión explicada.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from sympy import *


def s_polynomial(f, g):
    return expand(lcm(LM(f), LM(g)) * (1 / LT(f) * f - 1 / LT(g)
        * g))

def buchberger(F):
    G = list(F) #hacemos una copia de la lista F
    pairs = [] #lista de pares

    for i, f1 in enumerate(F): #i es el indice y f1 el polinomio
        correspondiente
        for f2 in F[i + 1:]:
            pairs.append((f1, f2)) #metemos pares en la lista
                pairs

    while pairs: #recorremos la lista de pares hasta que se
        vacie
        f1, f2 = pairs.pop(0) #almacenamos en f1 y f2 la ultima
            pareja y la quitamos
        print "\nPara los polinomios"
        print "\tf1 = "+ str(f1)
        print "\tf2 = "+ str(f2)

        s = s_polynomial(f1, f2) #calculamos su s-polinomio
        print "\nSu S-Polinomio es: "
        print "\t"+str(s)

        h = reduced(s, G)[1] #reducimos s con respecto a G
        print "\nReducimos el S-Polinomio con respecto a G: "
        print "\t"+str(h)

        if h != 0: #si su resto es distinto de 0
            for g in G:
                pairs.append((g, h)) #Se aniaden los pares
                    nuevos a comprobar

            G.append(h)#Se aniade h a la lista de generadores
            print "\nEl nuevo sistema de generadores es: "
            print "\t"+str(G)
        else:
            print "\nNo hace falta aniadirlo al sistema de
                generadores"
        print "\n
            ##########################################################
            "

```

```
43      print "\nLa Base de Grobner es: "
44      print "\t"+str(G)
45
46      return G
47
48
49  #Ejemplo de uso
50  x = Symbol('x')
51  y = Symbol('y')
52  z = Symbol('z')
53
54  F = [-2*x*y+x, (x**3)*y-2*x**2+y]
55  buchberger(F)
```

5.2. Brazo.

Para ver el uso de lo desarrollado a lo largo de esta memoria he realizado un pequeño programa en matplotlib el cual 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. He elegido usar matplotlib por la comodidad que me proporciona el uso de widgets y la buena representación 3D que realiza. Ahora pasaré a explicar el código y los detalles sobre él:

En principio importamos los paquetes necesarios y comenzamos a definir la única clase que vamos a tener, Draw_Robot, dentro de la cual llevaremos todo acabo. Empezamos con el constructor y vemos la definición de variables, a las longitudes (l) les he dado valor 100 y en principio 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
```

Seguimos dentro del constructor y ahora lo que vamos a hacer es preparar todo lo concerniente al calculo de nuestra base de Gröbner. Preparamos nuestras ecuaciones particulares para este caso:

$$a - l3 * (c1 * c2 - s1 * s2) - l2 * c1$$
$$b - l3 * (c1 * s2 + c2 * s1) - l2 * s1$$
$$c1^2 + s1^2 - 1$$
$$c2^2 + s2^2 - 1$$

Usamos groebner para estas ecuaciones, indicando que el orden va a ser lex y que el sistema resultante va a depender de s1,c1,s2,c2.

```
1           '''CALCULOS PREVIOS GROEBNER'''
2           a,b,c1,c2,s1,s2,l2,l3 = var("a,b,c1,c2,s1,s2,l2,l3")
3           eq= [a-l3*(c1*c2-s1*s2)-l2*c1, b-l3*(c1*s2+c2*s1)-l2*s1,
                c1**2+s1**2-1,c2**2+s2**2-1]
4           R = QQ.frac_field(a,b,l2,l3)
5           G = groebner(eq,c2,s2,c1,s1,order='lex',domain=R)
```

Este sería el resultado:

```
1   GroebnerBasis([
2   c2 + (-a**2 - b**2 + l2**2 + l3**2)/(2*l2*l3),
3   s2 + s1*(a**2 + b**2)/(a*l3) + (-a**2*b - b**3 - b*l2**2 + b*
        l3**2)/(2*a*l2*l3),
4   c1 + b*s1/a + (-a**2 - b**2 - l2**2 + l3**2)/(2*a*l2),
5   s1**2 + s1*(-a**2*b - b**3 - b*l2**2 + b*l3**2)/(a**2*l2 + b
        **2*l2) + (a**4 + 2*a**2*b**2 - 2*a**2*l2**2 - 2*a**2*l3**2
         + b**4 + 2*b**2*l2**2 - 2*b**2*l3**2 + l2**4 - 2*l2**2*l3
        **2 + l3**4)/(4*a**2*l2**2 + 4*b**2*l2**2)],
6     c2, s2, c1, s1, domain='QQ(a,b,l2,l3)', order='lex')
```

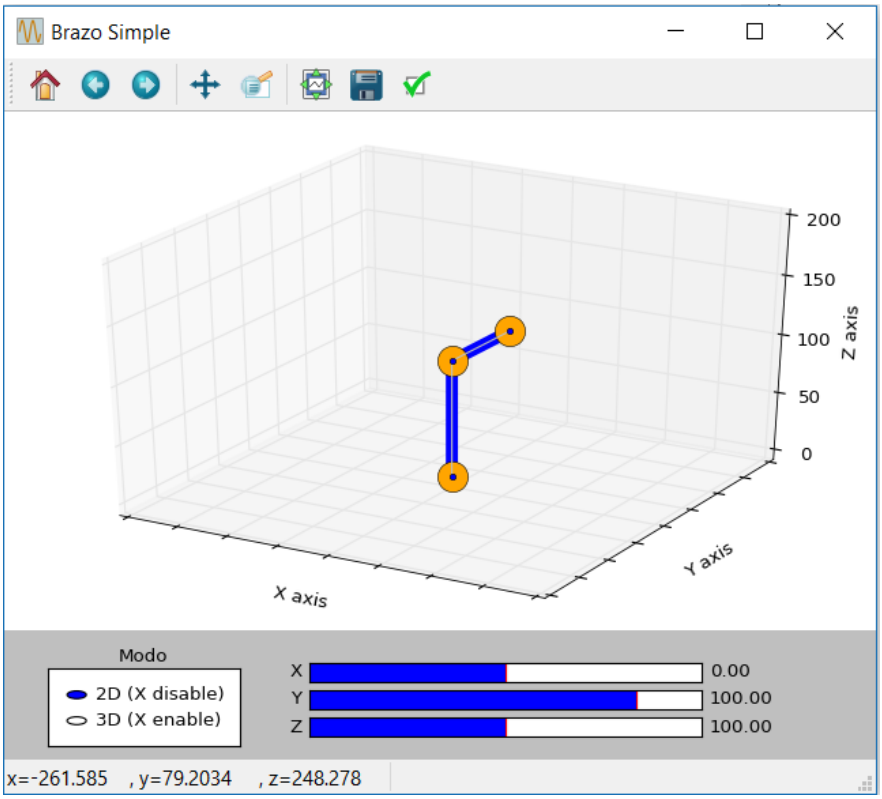
Como vemos hay 4 ecuaciones que, teniendo en cuenta que a,b,l2 y l3 son conocidos, forman un sistema de ecuaciones triangular en el que la primera solo depende de c2, la cuarta de c2 y s1, la segunda de c2, s1 y s2 y la tercera de c2, s1, s2 y c1.

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.

```
1           self.c2_var = solve(G[0].subs([(l2,self.l[1]),(l3,self.l
                [2])]),c2)
```

```
        self.s1_var = solve(G[3].subs([(l2,self.l[1]),(l3,self.l
            [2]),(c2,self.c2_var)]),s1)
        self.s2_var = []
        self.s2_var.append(solve(G[1].subs([(l2,self.l[1]),(l3,
            self.l[2]),(c2,self.c2_var),(s1,self.s1_var[0])]),s2)
            [0])
        self.s2_var.append(solve(G[1].subs([(l2,self.l[1]),(l3,
            self.l[2]),(c2,self.c2_var),(s1,self.s1_var[1])]),s2)
            [0])
        self.c1_var = []
        self.c1_var.append(solve(G[2].subs([(l2,self.l[1]),(l3,
            self.l[2]),(c2,self.c2_var),(s1,self.s1_var[0]),(s2,
            self.s2_var[0])]),c1)[0])
        self.c1_var.append(solve(G[2].subs([(l2,self.l[1]),(l3,
            self.l[2]),(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')
```

```
4         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.

```
1         '''WIDGETS'''
2         #Slider de X
3         axxval = plt.axes([0.35, 0.12, 0.45, 0.03])
4         self.x_val = Slider(axxval, 'X', -150, 150, valinit=self
              .tx)
5         self.x_val.set_active(False)
6
7         #Slider de Y
8         axyval = plt.axes([0.35, 0.0775, 0.45, 0.03])
9         y_val = Slider(axyval, 'Y', -150, 150, valinit=self.ty)
10
11        #Slider de Z
12        axzval = plt.axes([0.35, 0.035, 0.45, 0.03])
13        z_val = Slider(axzval, 'Z', 0, 200, valinit=self.tz)
14
15        #Radio buttons de modo
16        rax = plt.axes([0.05, 0.02, 0.22, 0.12])
17        rax.set_title('Modo', fontsize=12)
18        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.

```
1         '''MANEJADORES'''
2         #X slider event
3         def actualiza_val_x(val):0
4             self.tx = val
5             self.draw_robot()
6         self.x_val.on_changed(actualiza_val_x)
7
8         #Y slider event
9         def actualiza_val_y(val):
10            self.ty= val
11            self.draw_robot()
12        y_val.on_changed(actualiza_val_y)
13
14        #Z slider event
15        def actualiza_val_z(val):
16            self.tz = val
```

```
17              self.draw_robot()
18          z_val.on_changed(actualiza_val_z)
19
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):
```

```
        s1,s2,c1= [],[],[]

        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)]))

```

```
12          if(im(c1[0])==0 and im(s1[0])==0 and im(s2[0])==0):
13              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()
```