UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO DE TRAYECTORIAS PARA LA INTERCEPTACIÓN DE OBJETOS MÓVILES SIGUIENDO CURVAS DE BÉZIER E IMPLEMENTACIÓN SOBRE ROBOTS LEGO

AUTORA: MARÍA DEL MAR PLAZA CANO

TUTOR: ENRIQUE JORGE BERNABEU SOLER

COTUTOR: ÁNGEL VALERA FERNÁNDEZ

Curso Académico: 2018-19

AGRADECIMIENTOS

A mis padres, por todo el apoyo y el cariño que me han dado a lo largo de toda la carrera. Sois y seréis siempre mi mayor ejemplo de constancia y lucha.

A mis tíos y a mi abuela, por haber hecho posible que esté donde estoy ahora.

A mis compañeros y amigos, por hacer estos años más llevaderos.

A mi tutor Enrique J. Bernabeu, por toda la ayuda y la amabilidad que he recibido, además de su motivación por hacerme entender los conceptos de los que trata este proyecto.

RESUMEN

En el presente documento se realiza el diseño de trayectorias de dos robots y/u objetos móviles modeladas mediante curvas de Bézier. Mediante este algoritmo matemático se puede elegir la posición, velocidad y aceleración que tiene en cada instante un robot u objeto móvil.

Una vez diseñadas dichas trayectorias, se detectará si existe colisión entre ambos robots o de un robot y un objeto móvil, calculando el instante de tiempo en el que ambos objetos están más cerca. En el caso de que ambos objetos no colisionen, se deforma la trayectoria mínimamente de un robot para que intercepte al otro objeto en dicho instante.

La programación del robot se realizará sobre Matlab y la aplicación será experimentada a través de robots Lego

Palabras Clave: Generación de Trayectorias, Curvas de Evitación, Detección y búsqueda de colisión, Programación de Robots

RESUM

En el present document es realitza el disseny de trajectòries de dos robots i/o objectes mòbils modelats mitjançant corbes de Besiers. Amb aquest algoritme matemàtic es pot triar la posició, velocitat i acceleració que té en cada instant un robot u objecte mòbil.

Una vegada dissenyades les trajectòries, es detectarà si existeix col·lisió entre els dos robots o de un robot i un objecte mòbil, calculant l'instant de temps en el que els dos objectes estan més propers.

En el cas de que no es produïsca la col·lisió, es deforma la trajectòria mínimament d'un robot per a que intercepti al altre objecte en eixe instant.

La programació del robot es realitzarà en Matlab i l'aplicació serà experimentada per mitjà de robots LEGO.

Paraules clau: Generació de Trajectòries, Corbes d'Evitació, Detecció i recerca de col·lisió, Programació de Robots

ABSTRACT

In this document, the trajectories of two robots and / or mobile objects modelled by Bézier curves are designed. Through this mathematical algorithm you can choose the position, speed and acceleration that a robot or mobile object has at each moment.

After trajectories have been designed, it will be detected if there is a collision between both robots or a robot and a mobile object, calculating the instant in which both objects are closer. In the case that both objects do not collide, the path of a robot is minimally distorted to intercept the other object at that moment.

The programming of the robot is done on Matlab and the application is experimented through the LEGO robots.

Keywords: Generation of Trajectories, Avoidance Curves, Detection and Collision Search, Robot Programming

ÍNDICE GENERAL

DOCUMENTOS CONTENIDOS EN EL TFG

I. Memoria

II. Presupuesto

I. MEMORIA

CAPÍTULO 1. INTRODUCCIÓN 1

1.1. Motivación 1

1.2. Objetivos 1

1.3. Estructura del documento 2

CAPÍTULO 2. CURVAS DE BEZIER 5

2.1. Historia 5

2.2. Definición de las curvas de Bézier 5

2.3. Parametrización de las curvas de Bézier 7

2.4. Propiedades de las curvas de Bézier 8

CAPÍTULO 3. ADAPTACIÓN DE LAS CURVAS DE BÉZIER PARA LA DEFINICIÓN DE LAS TRAYECTORIAS DE LOS ROBOTS 13

3.1. Definición de las trayectorias 13

3.2. Definición de las curvas de Velocidad 14

3.3. Definición de las curvas de Aceleración 16

CAPÍTULO 4. BÚSQUEDA DE LA COLISIÓN 19

4.1. Determinación del instante de tiempo de máximo acercamiento 19

4.2. Detección de colisiones 22

4.3. Modificación de la trayectoria 24

CAPÍTULO 5. IMPLEMENTACIÓN DE LAS TRAYECTORIAS EN ROBOT LEGO NXT 29

5.1. Introducción a Lego NXT y RobotC 29
5.2. Control de robots móviles 33
5.2.1. Control cinemático 34
5.2.2. Control dinámico 36
5.3. Implementación de las curvas de Bézier en LEGO MINDSTORMS NXT 36
CAPÍTULO 6. PRUEBAS EXPERIMENTALES 41
6.1. Introducción de datos en Matlab 41
6.2. Seguimiento de la trayectoria implantada en el robot 44
CAPÍTULO 7. CONCLUSIONES 49
CAPÍTULO 8. BIBLIOGRAFÍA 51

II. PRESUPUESTO

1. Necesidad del presupuesto 1
2.Cuadro de precios básicos 1
2.1. Cuadro de mano de obra 1
2.2. Cuadro de maquinaria 2
2.3. Cuadro de licencias de Software 2
2.4. Cuadro de materiales 3
3.Cuadro de precios descompuestos por unidad de obra 4
4.Estado de mediciones de todas las unidades del proyecto 7
5.Presupuesto de inversión total 7

ÍNDICE DE FIGURAS

Figura 2. 1 Proceso de generación de una curva de Bézier mediante interpolación lineal [3]..... 5

Figura 2. 2 Representación bidimensional del polígono de Bézier de orden N=3 para..... 6

Figura 2. 3 Representación bidimensional de la curva de Bézier de orden N=3 a partir de los Control Points P_0= (0 0 0), P_1= (3 3 0), P_2= (4 -1 0), P_3= (7 4 0)..... 7

Figura 2. 4 Curva Bézier bidimensional de orden N=3 localizada en el interior de la envolvente convexa que forman los Control Points P_0= (0 0 0), P_1= (2 3 0), P_2= (5 4 0), P_3= (7 0 0)..... 8

Figura 2. 5 Curva Bézier bidimensional de orden N=3 modificando uno de los *Control Points* P0= (0 0 0), P1= (2 3 0), P2= (5 1 0), P3= (7 0 0)..... 9

Figura 2. 6 Curva Bézier bidimensional de orden N=3 con los *Control Points* P0= (0 0 0),P1= (2 3 0),P2= (5 4 0),P3= (7 0 0)..... 9

Figura 2. 7 Curva Afín con los *Control Points* P0= (0 0 0),P1= (4 6 0),P2= (10 8 0), P3= (14 0 0)..... 10

Figura 2. 8 Transformación de la curva bidimensional en una recta debido a la alineación de los Control Points..... 10

Figura 2. 9 Zoom en el principio de la curva de la Figura 2.4..... 11

Figura 2. 10 Zoom en el final de la curva de la Figura 2.4..... 11

Figura 3. 1 Trayectoria rectilínea de Bézier..... 15

Figura 3. 2 Velocidad respecto de x de un punto que recorre la curva de Bézier respecto al t∈ [0,5]s15

Figura 3. 3 Aceleración respecto de x de un punto que recorre la curva de Bézier respecto a t∈ [0,5]s 17

Figura 4. 1 Resultados obtenidos tras calcular la distancia mínima entre las dos curvas de Bézier 23

Figura 4. 2 Distancia mínima de separación entre las trayectorias de Bézier 23

Figura 4. 3 Nueva trayectoria Bézier modificada para propiciar la colisión en s_{min} 27

Figura 5. 1 Robot LEGO Mindstorms NXT [10]..... 29

Figura 5. 2 "Ladrillo" inteligente NXT y sus componentes [11] 30

Figura 5. 3 Actuador NXT [20] 30
Figura 5. 4 Interior del actuador NXT [9] 31
Figura 5. 5 Sensor de distancia ultrasónico [14] 31
Figura 5. 6 Sensor de luz [9] 31
Figura 5. 7 Sensor de contacto [12] 32
Figura 5. 8 Sensor de sonido [13] 32
Figura 5. 9 "ladrillo" NXT con los sensores y los actuadores conectados [15] 33
Figura 5. 10 Esquema del proceso de control de un robot móvil 33
Figura 5. 11 Configuración diferencial del robot 34
Figura 5. 12 Configuración de oruga 35
Figura 5. 13 Configuración de triciclo 35
Figura 5. 14 Configuración Ackerman y equivalencia con la configuración de triciclo 36
Figura 5. 15 Esquema a seguir para el desarrollo del código en RobotC 37
Figura 5. 16 Punto descentralizado 38

Figura 6. 1 Gráfica de las curvas de Bézier de orden N=3 y M=2 43
Figura 6. 2 Gráfica de las curvas de Bézier de orden N=3, M=2 y Ñ=3 44
Figura 6. 3 Representación de la trayectoria trazada por el robot frente a la referencia con s=0.001 45
Figura 6. 4 Representación de la trayectoria trazada por el robot frente a la referencia con s=0.002 46
Figura 6. 5 Representación de la trayectoria trazada por el robot frente a la referencia con s=0.003 46
Figura 6. 6 Representación de la trayectoria trazada por el robot frente a la referencia con una disminución de la velocidad de referencia 47
Figura 6. 7 Modificación de la distancia de inicio de los robots para una colisión más suave 48
Figura 6. 8 Posición del robot 2 modificada para una colisión suave 48

ÍNDICE DE TABLAS

Tabla 1 Cuadro de precios básicos de la mano de obra..... 1

Tabla 2 Cuadro de precios básicos de la maquinaria..... 2

Tabla 3 Cuadro de precios básicos de las licencias de Software..... 3

Tabla 4 Cuadro de precios básicos de materiales 3

Tabla 5 Cuadro de precios descompuestos por unidad de obra del capítulo 1..... 4

Tabla 6 Cuadro de precios descompuestos por unidad de obra del capítulo 2..... 5

Tabla 7 Cuadro de precios descompuestos por unidad de obra del capítulo 6

Tabla 8 Estado de mediciones de todas las unidades de obra del proyecto..... 7

Tabla 9 Presupuesto de inversión total..... 7

I

MEMORIA

CAPÍTULO 1. INTRODUCCIÓN

1.1. Motivación

Una vez terminado el grado, y con vistas al mundo laboral, me doy cuenta de que la robótica juega un papel muy importante en los procesos industriales. Esta rama de la automática surge de la necesidad de imitar el comportamiento humano en situaciones que generan monotonía, agotamiento, peligro o tal vez un alto margen de error a la hora de realizar una tarea. Se necesita de robots que concentren la tecnología actual en procesos rutinarios o rudimentarios, librando de estos procesos a las personas. De hecho, el crecimiento de la población conlleva un aumento de las necesidades de producción, siendo de vital importancia lograr tiempos y errores mínimos en todo tipo de procesos industriales.

Lo más habitual es que un robot tenga que valerse en un entorno dinámico en el que los objetos que lo rodean no permanecen estáticos. Se requiere entonces conocer el patrón de movimiento de objetos móviles ajenos al objeto de control, para que este pueda actuar en consecuencia trazando una trayectoria determinada en función de la tarea que se desea realizar.

Por este motivo, para el diseño de las trayectorias de dos objetos móviles independientes, se van a emplear trayectorias tridimensionales que quedan expresadas matemáticamente por *curvas de Bézier*. Estas curvas son muy utilizadas en el diseño de las carrocerías de automóviles y en aeronáutica, usualmente conocidas como "*Splines*" en los programas de CAD. No obstante, lo más novedoso es el planteamiento de utilizar estas curvas con el objetivo de generar trayectorias a implementarlas en objetos móviles terrestres, en vez del diseño de cuerpos u objetos como tal.
Con este trabajo pretendía conocer un poco más el mundo del desarrollo y la programación de autómatas, para así obtener una visión más amplia de este sector.

1.2. Objetivos

El objetivo más común en esta área de control de trayectorias es programar un robot que sea independiente y logre sortear los obstáculos que se encuentre a su alrededor a la hora de realizar una tarea. Sin embargo, el objetivo principal de este proyecto es que, en vez de sortear los obstáculos, busque colisionar con ellos.

Esto puede resultar interesante en casos donde se tengan dos objetos móviles, y se quiere producir en uno de ellos un movimiento controlado, cuya orientación y velocidad resultantes estén determinados por la actuación del otro.

Se puede extrapolar esta operación a robots futbolistas, donde uno de ellos busca golpear el balón en movimiento, con el objetivo de meterlo en la portería, trazando un ángulo definido por la colisión.

Por todo ello, se va a estudiar el comportamiento de dos robots móviles LEGO, cada uno con su propia trayectoria, donde uno de ellos modifica la suya para colisionar con el otro en el instante de tiempo en el que ambas trayectorias son más próximas entre sí.

Con este propósito se determinan los siguientes objetivos:

- **Conocer las aplicaciones de las curvas de Bézier**
  Entender la ventaja que suponen en muchos ámbitos de la ingeniería, debido a su aporte de simplicidad a la hora del tratamiento de las curvas y superficies para el diseño asistido por ordenador y, en este caso, la sencillez que aportan para el diseño de trayectorias.

- **Adaptar las curvas de Bézier al diseño de trayectorias**
  Conocer los parámetros más influyentes a la hora de generar estas curvas, así como el planteamiento de las curvas de velocidad y las de aceleración que definirán por completo el movimiento de los robots.

- **Diseñar un algoritmo capaz de calcular nuevas trayectorias para objetos terrestres**
  Con este algoritmo se persigue calcular el instante de tiempo donde la distancia entre ambos robots es mínima antes de su implementación, y una vez conocido, detectar si se va a producir una colisión entre ambos en función del volumen real que ocupan. En caso de que no se produzca una colisión, se fuerza modificando mínimamente la trayectoria de uno de los robots para provocarla en dicho instante de tiempo.

- **Implementación de los algoritmos en robots Lego Mindstorms NXT**
  Para su implementación se va a hacer uso de dos robots LEGO NXT, programados en "*robotC*", donde se requiere un control cinemático y dinámico de los actuadores para lograr que la trayectoria trazada difiera muy poco con la trayectoria de referencia.

1.3. Estructura del documento

El presente proyecto está dividido en dos documentos principales: la memoria y el presupuesto. En cuanto a la memoria del trabajo se ha organizado por capítulos en función de su contenido:

- **Capítulo 1:** en este primer capítulo se pretende contextualizar el contenido del proyecto, introduciendo el motivo que ha llevado a su estudio, así como los objetivos que se persiguen alcanzar en él.

- **Capítulo 2:** en segundo lugar, se introduce el concepto de las curvas de Bézier, tanto su origen, como sus numerosas aplicaciones hoy en día. También se da una definición formal de las fórmulas empleadas para su construcción y se nombran algunas de sus propiedades más características.

- **Capítulo 3:** puesto que uno de los objetivos de este proyecto es implementar en dos robots móviles unas trayectorias diseñadas mediante curvas de Bézier, se explica cómo se pueden adaptar dichas curvas a la generación de trayectorias, así como a la determinación de las velocidades y aceleraciones.

- **Capítulo 4:** se trata del capítulo clave del proyecto, en el cual se va a desarrollar el método empleado para determinar el instante de tiempo de máximo acercamiento entre los dos robots. Una vez conocido este instante, es importante detectar si de manera previa a cualquier modificación, ya existe colisión. En caso negativo, se procede a provocar el desvío de una de las trayectorias para así lograr la colisión, siendo este el objetivo principal del proyecto.

- **Capítulo 5:** en este capítulo se introduce el mundo de los Lego MindStorms NXT, que van a ser los encargados de probar si el algoritmo previamente desarrollado es válido o no. Para su funcionamiento, es importante entender los dos tipos de control que estos robots requieren para el seguimiento de una trayectoria. Por ello, se va a explicar de forma general los dos tipos, para posteriormente mostrar las ecuaciones a introducir en su software.

- **Capítulo 6:** con el fin de demostrar que se cumplen los objetivos antes expuestos en el proyecto, este capítulo trata de exponer mediante ejemplos, y de manera visual, los experimentos realizados en el laboratorio con los robots Lego.

- **Capítulo 7:** por último, se realiza una conclusión para valorar qué grado de cumplimiento de objetivos se ha alcanzado en el proyecto.

- **Capítulo 8:** en este capítulo se adjunta el repertorio bibliográfico empleado a lo largo de todo el proyecto.

Finalmente, se ha realizado un presupuesto general de elaboración del proyecto donde se muestran los costes invertidos en mano de obra, en maquinaria, en materiales, y en licencias de software.

CAPÍTULO 2. CURVAS DE BEZIER

2.1. Historia

El modelado sólido, geométrico y físico ha ido progresando a lo largo de los años desde que el ingeniero francés Pierre Bézier comenzó a trabajar en el desarrollo de un método de generación de curvas en la década de 1950, dando como resultado un conjunto de teorías matemáticas y algoritmos que definen y manipulan representaciones de objetos físicos, sus propiedades y abstracciones asociadas [2].

Dichas curvas fueron desarrolladas de forma paralela también por Paul de Cateljau, quien dotó a estas de un método numéricamente estable para evaluarlas [1]. Ambos llegaron esencialmente al mismo tipo de curvas, aunque con recursos matemáticos distintos [3].

Estas representaciones tienen su aplicación en el diseño, simulación, fabricación y análisis de procesos comunes en el automovilismo, la aeronáutica, la biomédica y las artes gráficas. [2]

2.2. Definición de las curvas de Bézier

Una curva de Bézier es una representación matemática para la cual es fácil construir una interfaz que le permita al usuario diseñar y controlar la forma de curvas y superficies complejas. El enfoque general es que el usuario ingresa una secuencia de puntos y se construye una curva cuya forma sigue de cerca esta secuencia [4]. Este modelado se realiza mediante la interpolación de los puntos mediante combinaciones lineales de algunas funciones, y es a la vez sencillo y flexible [5]. Los puntos se llaman puntos de control o “Control Points”.

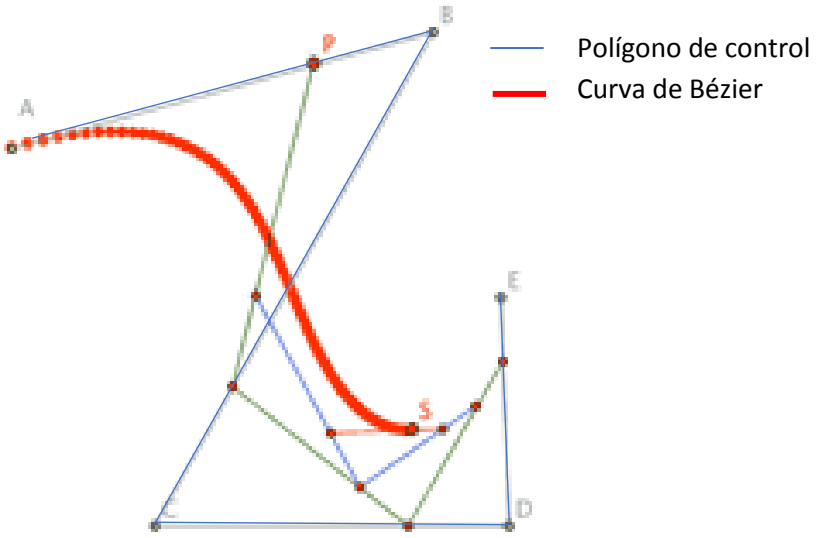

Figura 2. 1 Proceso de generación de una curva de Bézier mediante interpolación lineal [3]

Para el diseño de una curva de Bézier de orden N son necesarios N+1 *Control Points*, cada uno de ellos con sus correspondientes coordenadas espaciales: $P_i = (P_{ix}, P_{iy}, P_{iz})$

La secuencia de los puntos de control $P_0, P_1, P_2, \dots, P_{N-1}, P_N \in \mathbb{R}^3$ en el espacio utilizados para manipular la forma de un objeto, comenzando por P_0 y terminando en P_N, se denomina *polígono de Bézier* (o *polígono de control*) [24] .

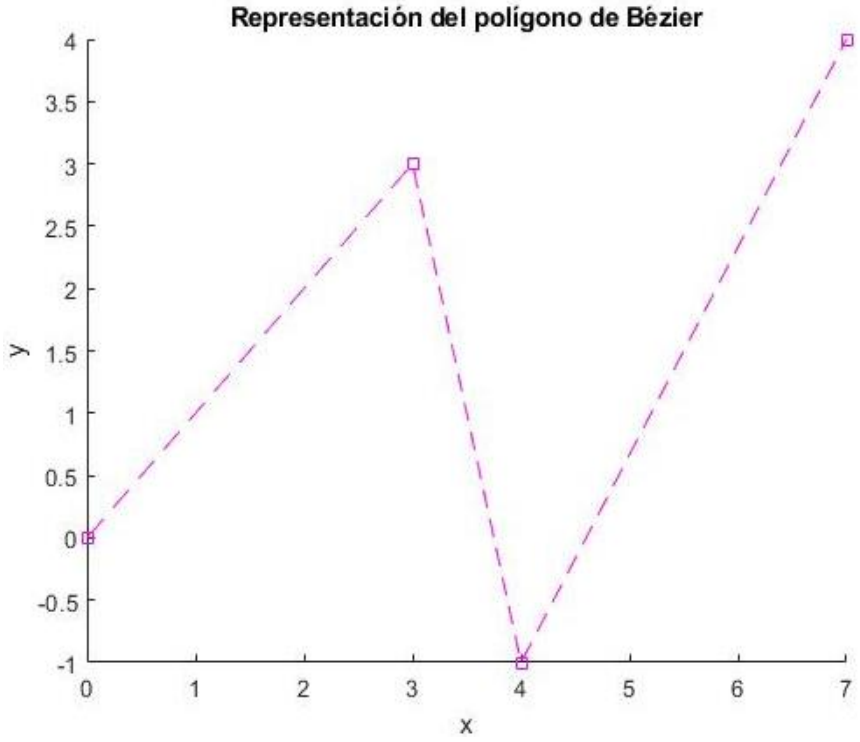

Figura 2. 2 Representación bidimensional del polígono de Bézier de orden N=3 para

P_0= (0 0 0), P_1= (3 3 0), P_2= (4 -1 0),P_3= (7 4 0)

A partir de los *Control Points* es posible trazar la curva de Bézier siguiendo la ecuación de de Casteljau:

$$B(s) = \sum_{i=0}^{N} \binom{N}{i} P_i \ (1-s)^{N-i} s^i \ = P_0(1-s)^N + \binom{N}{1} P_1 (1-s)^{N-1} s + \dots + P_N s^N \,, s \in [0,1] \qquad (2.1)$$

Donde el coeficiente binomial viene dado por la fórmula:

$$\binom{N}{i} = \frac{N!}{i!(N-i)!} \qquad (2.2)$$

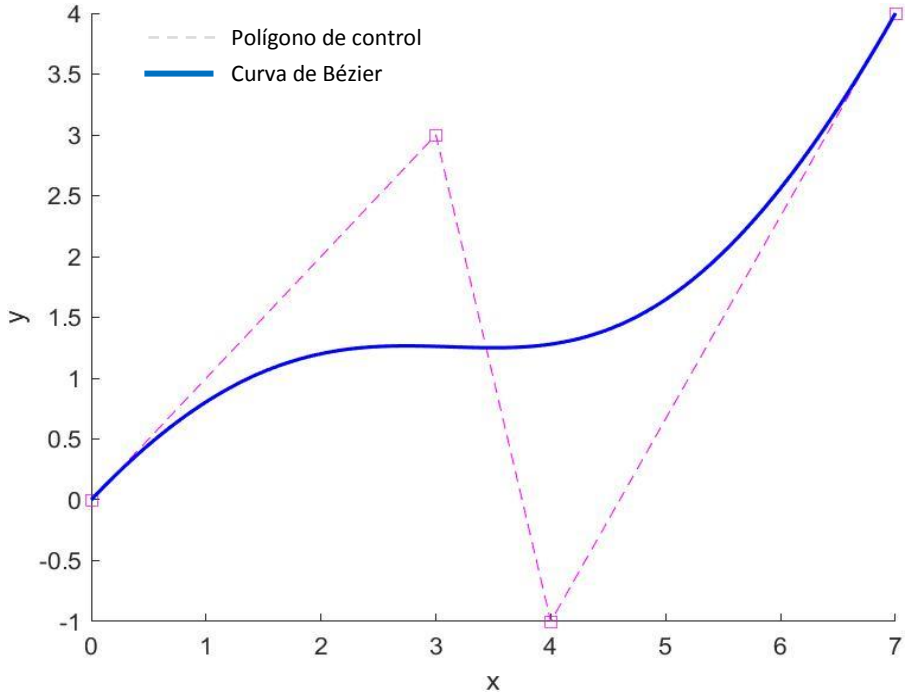

Figura 2. 3 Representación bidimensional de la curva de Bézier de orden N=3 a partir de los Control Points P_0= (0 0 0), P_1= (3 3 0), P_2= (4 -1 0), P_3= (7 4 0)

Se observa cómo los *Control Points* proporcionan direccionalidad a la curva, partiendo desde el punto de control P_0 y finalizando la curva en el punto de control P_3, sin ser estrictamente necesario que dicha curva pase por los puntos P_2 y P_3 [1].

2.3. Parametrización de las curvas de Bézier

Para el diseño asistido por ordenador es conveniente emplear representaciones sencillas de curvas y superficies, que involucren operaciones elementales, como sumas y multiplicaciones. Es por esto por lo que lo más sencillo es parametrizar estas curvas. [6]

De esta manera se pueden representar las curvas de Bézier de grado N como:

$$B(s) = \sum_{j=0}^{N} s^j C_j = C_0 + C_1 s + C_2 s^2 + \ldots + C_{N-1} s^{N-1} + C_N s^N, s \in [0,1]. \quad (2.3)$$

donde cada coeficiente C_j es un punto tridimensional $C_j = (C_{jx}, C_{jy}, C_{jz})$, denominado *Coefficient Point.*

Los *Coefficient Points* (C_j) se obtienen a partir de los *Control Points* (P_i), siguiendo la fórmula:

$$C_j = \frac{N!}{(N-j)!j!}\left(\sum_{i=0}^{j}\frac{j!}{i!(j-i)!}(-1)^i P_{j-i}\right) \quad (2.4)$$

Nótese que:

$$P_0 = C_0 \quad (2.5)$$

$$P_N = \sum_{j=0}^{N} C_j \quad (2.6)$$

Ambas curvas son idénticas (2.1) (2.3), simplemente son dos formas distintas de desarrollo de la ecuación.

2.4. Propiedades de las curvas de Bézier

Partiendo de la referencia [1], se procede a enumerar las propiedades más destacables de las curvas de Bézier:

- La curva se localiza en el interior de la envolvente convexa de los *Control Points* [1].

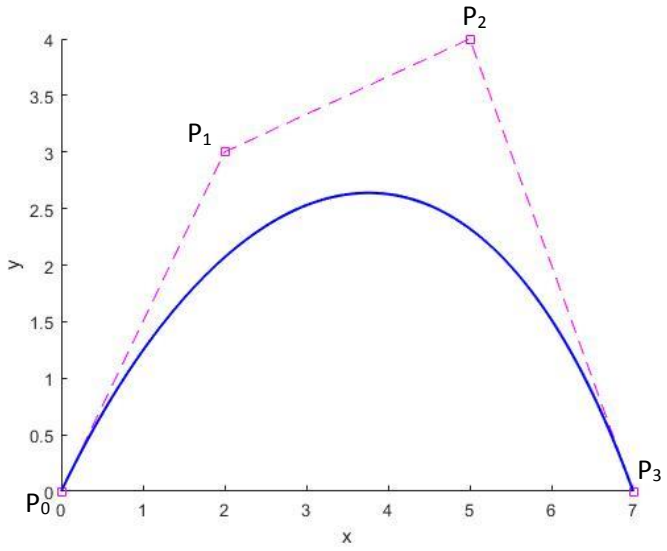

Figura 2. 4 Curva Bézier bidimensional de orden N=3 localizada en el interior de la envolvente convexa que forman los Control Points P_0= (0 0 0), P_1= (2 3 0), P_2= (5 4 0), P_3= (7 0 0)

- Las curvas tienen un control global. Esto supone que la modificación de al menos uno de los *Control Points* modifique por completo la curva de Bézier [1]. Para su visualización se va a cambiar únicamente uno de los puntos de control, en concreto P_2.

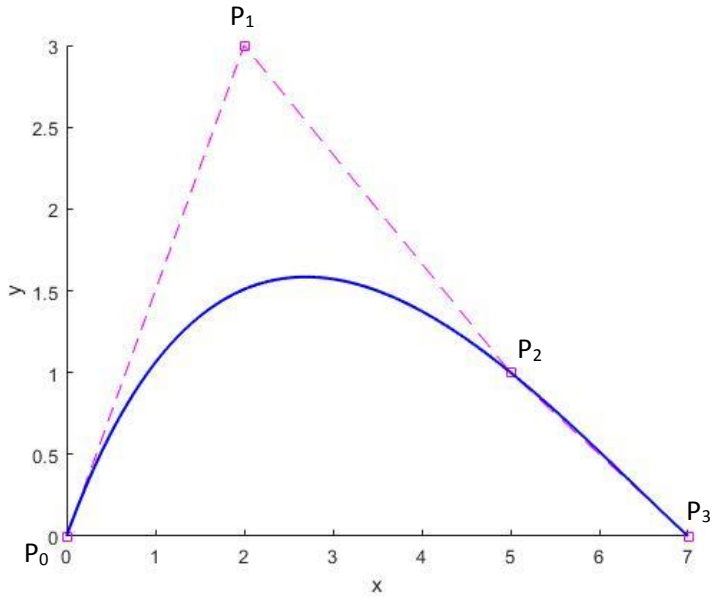

Figura 2. 5 Curva Bézier bidimensional de orden N=3 modificando uno de los *Control Points* P_0= (0 0 0), P_1= (2 3 0), P_2= (5 1 0), P_3= (7 0 0)

- Para lograr una transformación afín de la curva, hay que realizar dicha transformación en todos y cada uno de los *Control Points* que la generan. Se puede observar en las figuras de abajo que únicamente se ha dado un cambio en las proporciones de la curva.

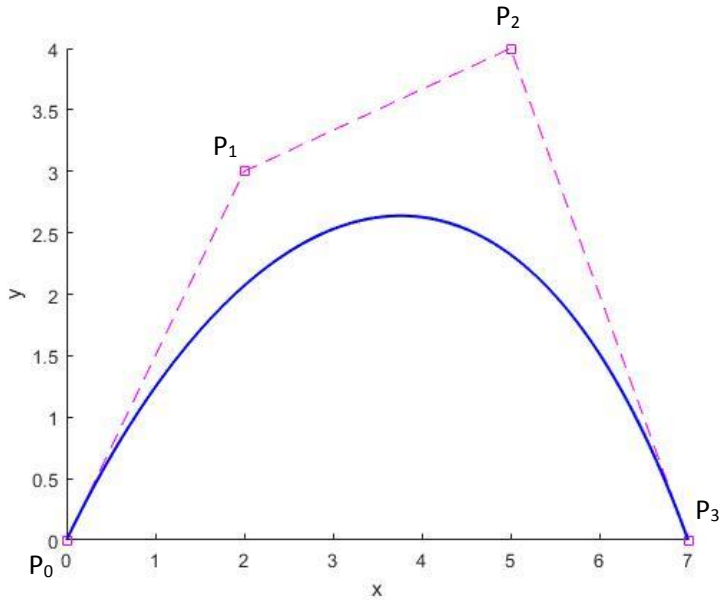

Figura 2. 6 Curva Bézier bidimensional de orden N=3 con los *Control Points* P_0= (0 0 0),P_1= (2 3 0),P_2= (5 4 0),P_3= (7 0 0)

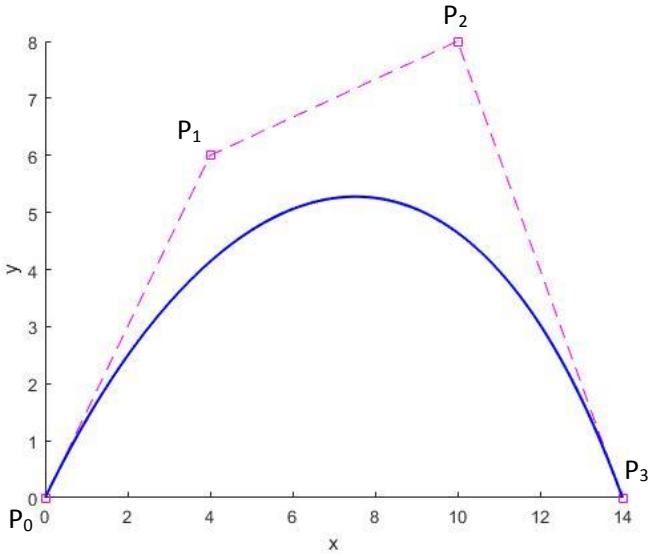

Figura 2. 7 Curva Afín con los *Control Points* P_0= (0 0 0),P_1= (4 6 0),P_2= (10 8 0), P_3= (14 0 0)

- La curva pasa a ser una recta sólo si todos los *Control Points* están alineados [1].

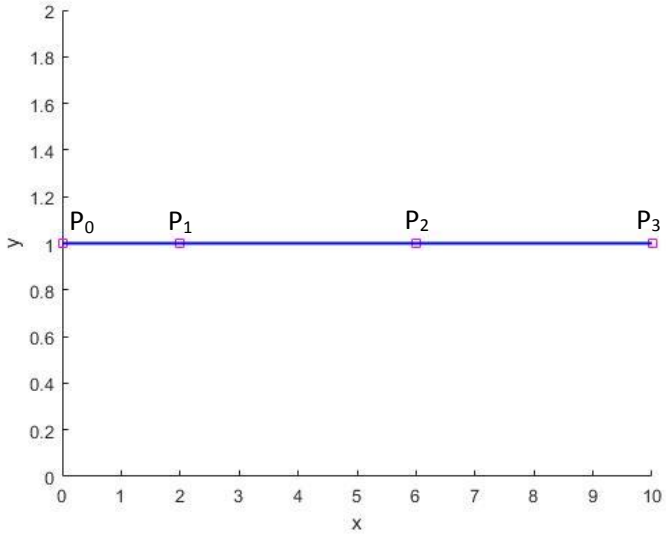

Figura 2. 8 Transformación de la curva bidimensional en una recta debido a la alineación de los Control Points

- Tanto el principio como el final de la curva son tangentes a sus correspondientes secciones del polígono de Bézier [1]. Ampliando la Figura 2.3 se puede observar cómo se cumple esta propiedad:

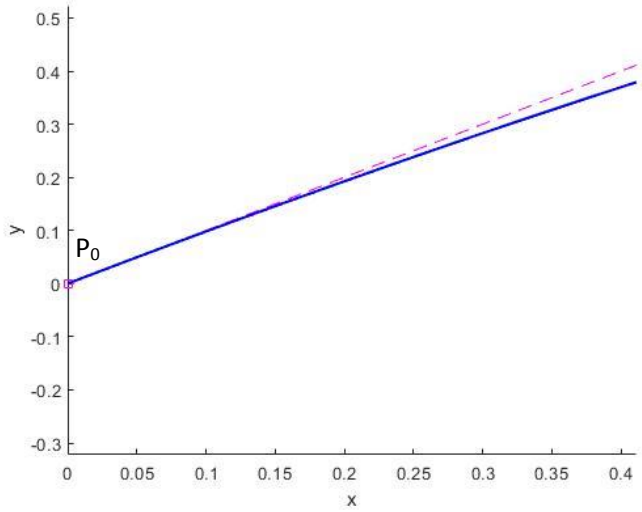

Figura 2. 9 Zoom en el principio de la curva de la Figura 2.4

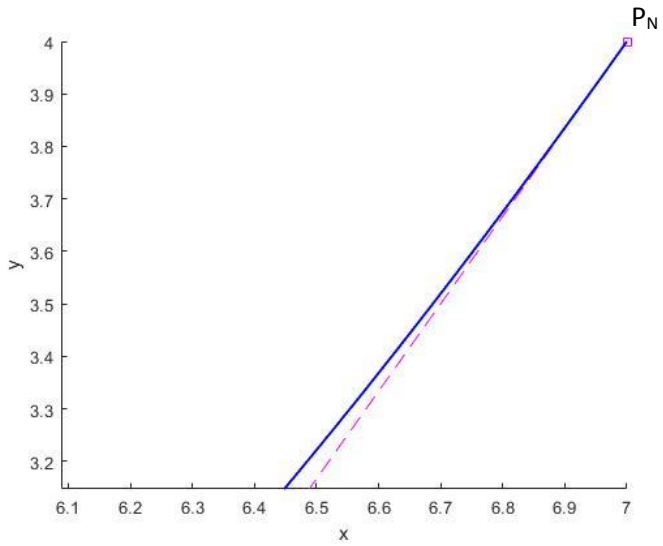

Figura 2. 10 Zoom en el final de la curva de la Figura 2.4

CAPÍTULO 3. ADAPTACIÓN DE LAS CURVAS DE BÉZIER PARA LA DEFINICIÓN DE LAS TRAYECTORIAS DE LOS ROBOTS

Las curvas de Bézier se pueden usar como método que describe el movimiento de un punto o de una esfera en el espacio. Dado que en este proyecto se persigue la generación de trayectorias para su posterior implementación en robots móviles terrestres, supone restringir uno de los grados de libertad de los que gozan estas curvas, para su aplicación en superficies planas de entorno conocido. Para ello basta con anular la coordenada Z del sistema de coordenadas global.

Mediante estas curvas se va a definir la posición, la velocidad y la aceleración del robot u objeto móvil a lo largo del tiempo.

3.1. Definición de las trayectorias

Cabe plantearse que el objetivo de dicho proyecto es que dos robots colisionen entre sí, cada uno siguiendo su propia trayectoria independientemente antes de dicha colisión. Para adaptar las dos trayectorias de Bézier a las trayectorias que han de seguir los robots, se ha utilizado Matlab, debido a la sencillez que proporciona en cuanto al cálculo vectorial y matricial.

El código está pensado para dos robots, con dos trayectorias de Bézier independientes, siendo la trayectoria del primer robot de orden N, y la del segundo robot de orden M.

Curva Bézier de orden N $\longrightarrow$ N+1 *Control Points* $P_0, P_1, P_2, \dots, P_N \in \mathbb{R}^3$

N+1 *Coefficient Points* $C_0, C_1, C_2, \dots, C_N \in \mathbb{R}^3$

Curva Bézier de orden M $\longrightarrow$ M+1 *Control Points* $P_0, P_1, P_2, \dots, P_M \in \mathbb{R}^3$

M+1 *Coefficient Points* $C_0, C_1, C_2, \dots, C_M \in \mathbb{R}^3$

Ambas trayectorias seguirán la ecuación (2.3) de las curvas paramétricas de Bézier antes citada:

$$B(s) = \sum_{j=0}^{N} s^j C_j = C_0 + C_1 s + C_2 s^2 + \dots + C_{N-1} s^{N-1} + C_N s^N, s \in [0,1]. \qquad (2.3)$$

Dicha ecuación muestra para cada valor de $s \in [0,1]$ la posición en la que se encuentra el robot.

Se observa que:

1. Para el instante inicial (s=0) el punto que recorre la curva se encuentra en la posición $B(0) = C_0$, que coincide con el punto de origen de la trayectoria según la ecuación (2.5) $C_0 = P_0$.

2. Para el instante el instante final (s=1) el punto que recorre la curva se encuentra en la posición $B(1) = \sum_{j=0}^{N} C_j$, punto final de dicha trayectoria según la ecuación (2.6) $P_N = \sum_{j=0}^{N} C_j$.

Por tanto, se deduce que el tiempo que tarda un objeto móvil en recorrer la trayectoria desde su principio hasta el fin es función de s, quedando:

$$\begin{cases} s \in [0\,,1] \\ t \in [t_0, t_f] \end{cases} \longrightarrow \; t = t_0 + s\,(t_f - t_0) \qquad (3.1)$$

Donde:

to= tiempo de inicio de la trayectoria

tf= tiempo de finalización de la trayectoria

Ya es posible conocer la posición del robot a lo largo del tiempo.

3.2. Definición de las curvas de Velocidad

Una vez se tiene la ecuación de la trayectoria, es posible obtener la ecuación de la velocidad con la que es recorrida la curva desde que comienza en P_0 hasta que finaliza en P_N . Dicha ecuación se obtiene derivando la ecuación de la trayectoria respecto de s:

$$B'(s) = v(s) = \sum_{j=1}^{N} j\, s^{j-1} C_j = \qquad (3.2)$$

$$= C_1 + 2C_2 s + 3C_3 s^2 + \ldots + (N-1)\, C_{N-1} s^{N-2} + N\, C_N s^{N-1}\,, s \in [0,1]$$

Para definir la curva de velocidad respecto al tiempo se ha de tener en cuenta que:

$$t = t_0 + s\,(t_f - t_0) \quad (3.1) \longrightarrow s = \frac{t - to}{tf - to} \qquad (3.3)$$

Quedando la curva definida como:

$$B'(t) = v(t) = \sum_{j=1}^{N} j \left(\frac{t - to}{tf - to}\right)^{j-1} C_j = \qquad (3.4)$$

$$= C_1 + 2C_2 \left(\frac{t-to}{tf-to}\right) + 3C_3 \left(\frac{t-to}{tf-to}\right)^2 + \ldots + (N-1)\, C_{N-1} \left(\frac{t-to}{tf-to}\right)^{N-2} + N\, C_N \left(\frac{t-to}{tf-to}\right)^{N-1}, t \in [t_0, t_f]$$

Además, cabe añadir que:

$$v(t) = \frac{\text{Distancia}}{t} \tag{3.5}$$

Por tanto, si se aumenta la distancia a recorrer por un punto, pero se mantiene el tiempo total del recorrido, el punto se moverá a mayor velocidad entre puntos separados a mucha distancia que entre puntos cercanos entre sí.

Esto se puede observar en las siguientes figuras:

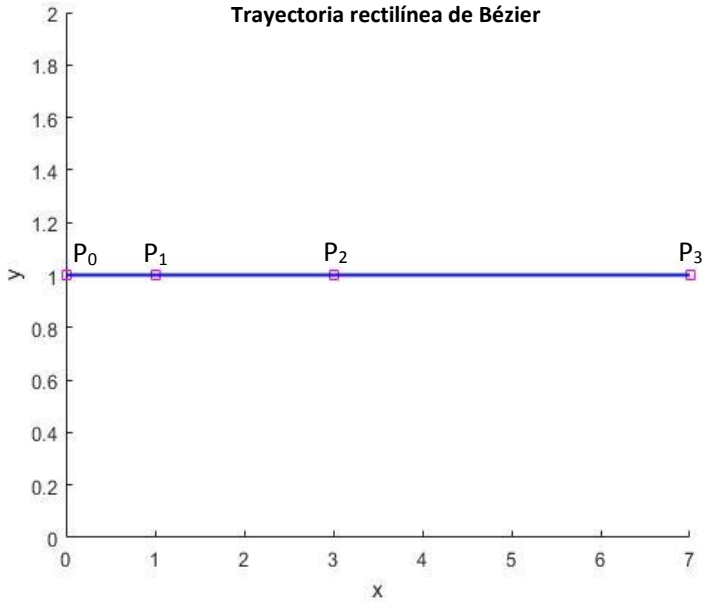

Figura 3. 1 Trayectoria rectilínea de Bézier

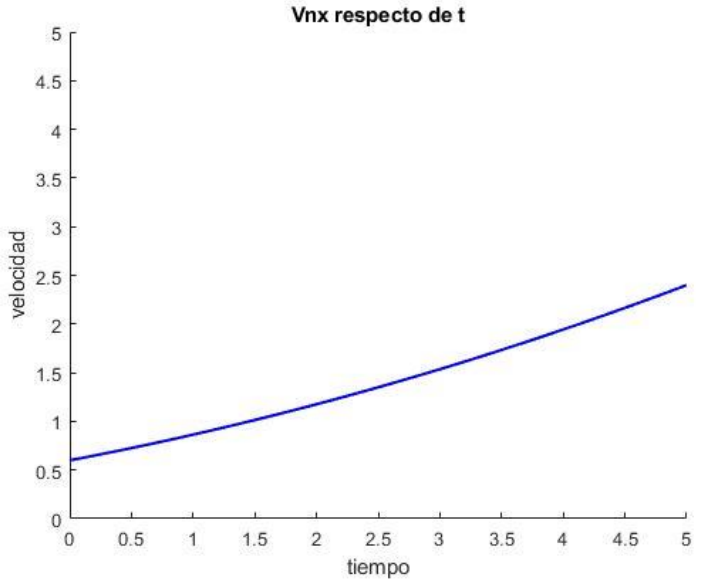

Figura 3. 2 Velocidad respecto de x de un punto que recorre la curva de Bézier respecto al t∈ [0,5]s

Nótese que hay un claro aumento en la velocidad de la coordenada x (Figura 3.2) debido a la creciente separación entre los *Control Points* de la Figura 3.1.

3.3. Definición de las curvas de Aceleración

La aceleración se obtiene de derivar la velocidad, o lo que es lo mismo, de hacer la segunda derivada de la trayectoria de Bézier (2.3). Se obtiene, por tanto:

$$B''(s) = a(s) = \sum_{j=2}^{N} j(j-1)s^{j-2}C_j = \tag{3.6}$$

$$= 2C_2 + 6C_3 s^2 + \ldots + (N-2)(N-1)\,C_{N-1}s^{N-3} + (N-1)N\,C_N s^{N-2}\ , s \in [0,1]$$

Para definirla en función del tiempo:

$$B''(t) = a(t) = \sum_{j=2}^{N} j(j-1)\left(\frac{t-to}{tf-to}\right)^{j-2} C_j\ = \tag{3.7}$$

$$= 2C_2 + 6C_3\left(\frac{t-to}{tf-to}\right)^2 + \ldots + (N-2)(N-1)\,C_{N-1}\left(\frac{t-to}{tf-to}\right)^{N-3} + (N-1)N\,C_N\left(\frac{t-to}{tf-to}\right)^{N-2}, t \in\ [t_0, t_f]$$

Siguiendo el ejemplo de la Figura 3.1, se aprecia cómo un aumento de la velocidad produce un aumento en la aceleración, pues:

$$a(t) = \frac{v_f - v_0}{t} \tag{3.8}$$

Esto se puede observar en la siguiente figura:

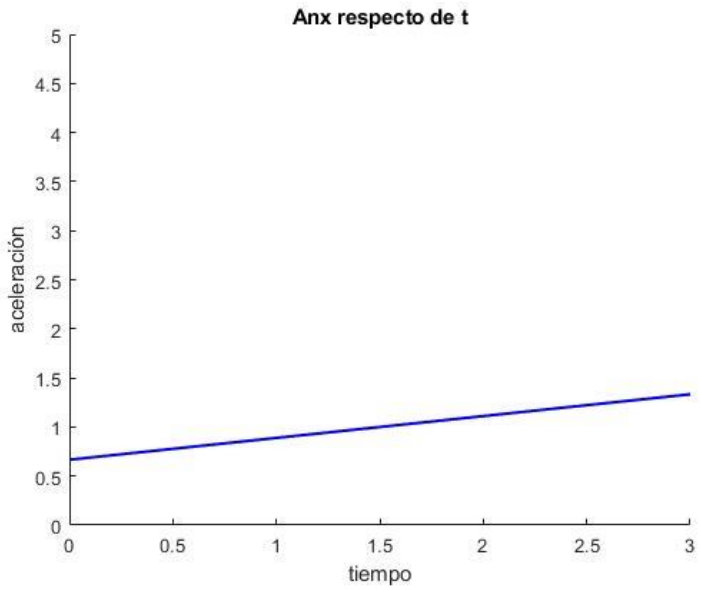

Figura 3. 3 Aceleración respecto de x de un punto que recorre la curva de Bézier respecto a t∈ [0,5]s

CAPÍTULO 4. BÚSQUEDA DE LA COLISIÓN

En este capítulo se desarrolla el algoritmo que realiza el desvío de una de las trayectorias de los robots para buscar la colisión e interceptar con el otro robot, siempre y cuando dicha colisión no se presente con las trayectorias de origen [23].

4.1. Determinación del instante de tiempo de máximo acercamiento

Con el fin de acotar el campo de posibilidades en las que dos trayectorias pueden cruzarse o estar muy cerca una de la otra, se pretende estudiar el instante de tiempo en el que estas son más próximas entre sí, para después centrar la atención en ese instante en el que debería darse la colisión.

En primer lugar, se tienen las dos trayectorias de Bézier antes citadas en función de los *Coefficient Points*:

Bézier de orden N: $\mathrm{p(s)} = \mathrm{p_0} + \mathrm{p_1 s} + \mathrm{p_2 s^2} + \ldots + \mathrm{p_N s^N} \quad , \mathrm{s} \in [0,1]$

Bézier de orden M: $\mathrm{q(s)} = \mathrm{q_0} + \mathrm{q_1 s} + \mathrm{q_2 s^2} + \ldots + \mathrm{q_M s^M} \quad , \mathrm{s} \in [0,1]$

Con $\mathrm{p_0}, \mathrm{p_1}, \mathrm{p_2}, \ldots, \mathrm{p_N} \in \mathbb{R}^3$

$\mathrm{q_0}, \mathrm{q_1}, \mathrm{q_2}, \ldots, \mathrm{q_N} \in \mathbb{R}^3$

Se pretende encontrar el instante s en el cual la distancia de separación entre ambas trayectorias es mínima:

$$min\ (\ \|p(s) - q(s)\|\ ),\ \mathrm{s} \in [0,1]\ \Rightarrow\ \mathrm{s_{min}} \tag{4.1}$$

En el cálculo de p(s)-q(s) surgen tres posibilidades en función del grado de ambas trayectorias:

1. Si orden M>orden N:

$$(p_0 - q_0) + (p_1 - q_1)s + \cdots + (p_N - q_N)s^N - q_{N+1}s^{N+1} - q_{N+2}s^{N+2} - \ldots - q_M s^M \tag{4.2}$$

2. Si orden M<orden N:

$$(p_0 - q_0) + (p_1 - q_1)s + \cdots + (p_M - q_M)s^M + p_{M+1}s^{M+1} + p_{M+2}s^{M+2} + \ldots + p_N s^N \tag{4.3}$$

3. Si orden M=orden N:

$$(p_0 - q_0) + (p_1 - q_1)s + \cdots + (p_N - q_N)s^N \tag{4.4}$$

Dado que los *Coefficient Points* son puntos tridimensionales $P_i = (P_{ix}, P_{iy}, P_{iz})$, se va a construir con los coeficientes de las ecuaciones (4.2), (4.3) o (4.4) la siguiente "matriz de diferencias" auxiliar para simplificar su desarrollo en el código de Matlab y sea más sencillo abordar los cálculos que vienen a continuación:

$$M = \begin{pmatrix} D_{0x} & D_{0y} & D_{0z} \\ D_{1x} & D_{1y} & D_{1z} \\ \vdots & \vdots & \vdots \\ D_{wx} & D_{wy} & D_{wz} \end{pmatrix} \tag{4.5}$$

Donde:

$$w = max(N, M)$$

$$D_{ix} = p_{ix} - q_{ix}$$

$$D_{iy} = p_{iy} - q_{iy}$$

$$D_{iz} = p_{iz} - q_{iz}$$

Por ejemplo, si se tienen dos trayectorias de Bézier con orden N=3 y orden M=2:

N>M: $p(s) - q(s) = (p_0 - q_0) + (p_1 - q_1)s + (p_2 - q_2)s^2 + p_3 s^3$

La matriz de diferencias tendría la siguiente forma:

$$M = \begin{pmatrix} D_{0x} & D_{0y} & D_{0z} \\ D_{1x} & D_{1y} & D_{1z} \\ D_{2x} & D_{2y} & D_{2z} \\ D_{3x} & D_{3y} & D_{3z} \end{pmatrix}$$

Con:

$$(p_0 - q_0) \rightarrow \begin{cases} D_{0x} = p_{0x} - q_{0x} \\ D_{0y} = p_{0y} - q_{0y} \\ D_{0z} = p_{0z} - q_{0z} \end{cases}$$

$$(p_1 - q_1) \rightarrow \begin{cases} D_{1x} = p_{1x} - q_{1x} \\ D_{1y} = p_{1y} - q_{1y} \\ D_{1z} = p_{1z} - q_{1z} \end{cases}$$

$$(p_2 - q_2) \rightarrow \begin{cases} D_{2x} = p_{2x} - q_{2x} \\ D_{2y} = p_{2y} - q_{2y} \\ D_{2z} = p_{2z} - q_{2z} \end{cases}$$

$$(p_3 - q_3) \rightarrow \begin{cases} D_{3x} = p_{3x} \\ D_{3y} = p_{3y} \\ D_{3z} = p_{3z} \end{cases}$$

Véase el procedimiento general de construcción de la matriz de diferencias en Matlab:

```
n=ordenN+1; %Dado que en Matlab los índices de las matrices empiezan por 1
m=ordenM+1;

    if ordenN > ordenM
        for i = 1:m
            for j = 1:3
                M(i,j)= Cn(i,j)-Cm(i,j);
            end
        end
        for i = (m+1):n
            for j=1:3
                M(i,j)= Cn(i,j);
            end
        end

    elseif ordenM > ordenN
        for i = 1:n
            for j= 1:3
                M(i,j)= Cn(i,j)-Cm(i,j);
            end
        end
        for i = (n+1):m
            for j=1:3
                M(i,j)= -Cm(i,j);
            end
        end

    else ordenN == ordenM
        for i= 1:n
            for j= 1:3
               M(i,j)= Cn(i,j)-Cm(i,j);
            end
        end

    end
```

Una vez calculada la matriz de diferencias M, para obtener s_{min}:

$$\frac{d(\|p(s)-q(s)\|)}{ds} = 0 \Rightarrow s_{min} \tag{4.6}$$

Según la referencia [22], un procedimiento sencillo a seguir para la resolución de la ecuación (4.6) y que simplificará su implementación en Matlab es la siguiente ecuación:

$$\sum_{k=0}^{2w-1} s^k \Big(\sum_{\substack{i\geq 0, j\geq 1 \\ i+j=k+1}}^{w} jD_{ix}D_{jx} + jD_{iy}D_{jy} + jD_{iz}D_{jz} \Big) = 0 \qquad (4.7)$$

Cuyo desarrollo en Matlab sería el siguiente:

```
w=max(ordenN,ordenM);
Z=0;

    for k=1:(2*w)
        for j=1:w
            for i=0:w
                if i+j==k

t(k)=t(k)+(j*M(i+1,1)*M(j+1,1)+j*M(i+1,2)*M(j+1,2)+j*M(i+1,3)*M(j+1,3));
                end
            end
        end
        Z=Z+t(k)*s^(k-1);
    end
```

Tras la resolución de la ecuación (4.7) se obtiene s_{min}, la menor de las soluciones reales y pertenecientes al intervalo $[0, 1]$. Conocida s_{min} ya es posible calcular el instante de tiempo en el que se produce el máximo acercamiento entre las dos trayectorias mediante la ecuación (3.1):

$$t_{min} = t_0 + s_{min}\,(t_f - t_0)$$

4.2. Detección de colisiones

En segundo lugar, se ha de saber si una vez diseñadas las trayectorias de los dos objetos móviles mediante las curvas de Bézier existe colisión entre ambos en el instante de máximo acercamiento. En caso de que se produjera una colisión, carecería de sentido modificar una de las dos trayectorias para propiciarla.

Ambos robots se van a simplificar virtualmente como dos esferas móviles, cuyos radios serán $r_N \geq 0$ y $r_M \geq 0$, por tanto, la distancia que separa ambas esferas en el instante de mínima separación entre las trayectorias es:

$$MTD = \|p(s_{min}) - q(s_{min})\| - (r_N + r_M) \qquad (4.8)$$

Por ejemplo,

Sean las trayectorias:

Curva de Bézier N=3 : *Control Points* $P_0 = (0\ 1\ 0)$, $P_1 = (2\ 1\ 0)$, $P_2 = (5\ 4\ 0)$, P_3= (7 2 0)

Curva de Bézier M=2 : *Control Points* $P_0 = (1\ 3\ 0)$, $P_1 = (3\ 2\ 0)$, $P_2 = (4\ 5\ 0)$

Con dos esferas móviles de radios $r_N = 0.2\ m$ y $r_M = 0.5\ m$

En el intervalo de tiempo t ∈ $[0, 5]$s

Se obtienen los resultados mostrados en la figura 4.1, con las unidades de longitud en metros y el tiempo en segundos:

```
smin =

0.376895459960164903582800626795 45


tmin =

1.884477299800824517914003133977 2


MTD =

0.417625192953067795190941594247 82
```

Figura 4. 1 Resultados obtenidos tras calcular la distancia mínima entre las dos curvas de Bézier

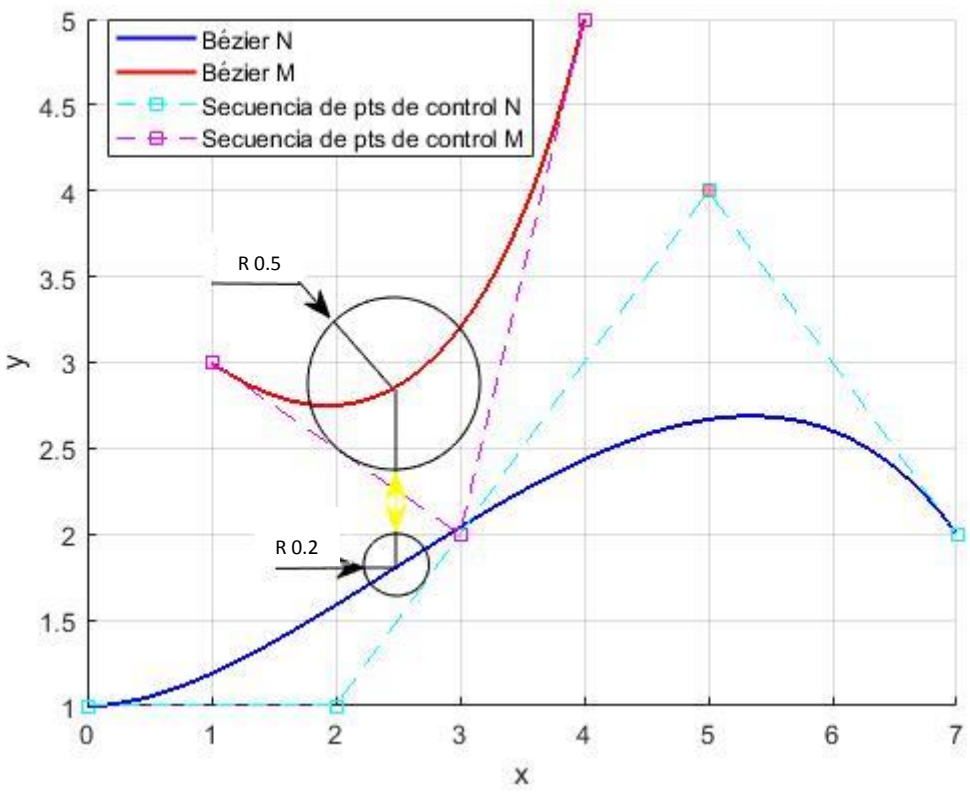

Figura 4. 2 Distancia mínima de separación entre las trayectorias de Bézier

Se observa cómo la mínima distancia entre ambas trayectorias es MTD=0.4176 m, teniendo en cuenta el volumen ocupado por las esferas. Al ser la distancia de separación positiva, se sabe que no existe colisión a partir de las trayectorias primitivas de ambos robots, por lo tanto, es posible trabajar en el cálculo de la modificación de una de las curvas para que uno de los robots intercepte al otro en ese instante donde la distancia entre ambos es mínima.

4.3. Modificación de la trayectoria

Una vez se ha determinado que no existe colisión entre las trayectorias (MTD≥0) y se conoce el instante en el que se produce el máximo acercamiento entre las curvas, se procede a modificar la trayectoria de orden N para que intercepte a la trayectoria de orden M mediante el planteamiento de los mínimos cuadrados con restricciones.

Este planteamiento es una técnica de análisis numérico enmarcada dentro de la optimización matemática [7], mediante la cual se pretende minimizar la discrepancia entre la trayectoria que se va a modificar y la trayectoria primitiva, con la imposición de determinadas restricciones que obligarán a la nueva curva a colisionar en el instante de tiempo deseado y con igual comienzo y final que la curva primitiva, de acuerdo con el criterio de *mínimo error cuadrático*.

Por tanto, partiendo de las dos trayectorias primitivas:

Bézier de orden N: $\mathrm{p}(\mathrm{s}) = \mathrm{p}_0 + \mathrm{p}_1\mathrm{s} + \mathrm{p}_2\mathrm{s}^2 + \ldots + \mathrm{p}_\mathrm{N}\mathrm{s}^\mathrm{N} \quad , \mathrm{s} \in [0,1]$

Con $\mathrm{p}_0, \mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_\mathrm{N} \in \mathbb{R}^3$ *Coefficient Points*

Bézier de orden M: $\mathrm{q}(\mathrm{s}) = \mathrm{q}_0 + \mathrm{q}_1\mathrm{s} + \mathrm{q}_2\mathrm{s}^2 + \ldots + \mathrm{q}_\mathrm{M}\mathrm{s}^\mathrm{M} \quad , \mathrm{s} \in [0,1]$

Con $\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_\mathrm{M} \in \mathbb{R}^3$ *Coefficient Points*

Se desea modificar la curva de Bézier de orden N para obtener una nueva trayectoria Bézier manteniendo el mismo orden N:

Nueva Bézier N: $\bar{\mathrm{p}}(\mathrm{s}) = \overline{\mathrm{p}_0} + \overline{\mathrm{p}_1}\mathrm{s} + \overline{\mathrm{p}_2}\mathrm{s}^2 + \ldots + \overline{\mathrm{p}_\mathrm{N}}\mathrm{s}^\mathrm{N} \quad , \mathrm{s} \in [0,1]$

Siendo $\overline{\mathrm{p}_0}, \overline{\mathrm{p}_1}, \overline{\mathrm{p}_2}, \ldots, \overline{\mathrm{p}_\mathrm{N}} \in \mathbb{R}^3$ los *Coefficient Points*

Que cumpla las siguientes restricciones:

1. La curva de Bézier N primitiva y la nueva Bézier tengan la misma posición inicial:

$$\overline{\mathrm{p}_0} = \mathrm{p}_0 \tag{4.9}$$

2. La curva de Bézier N primitiva y la nueva Bézier tengan la misma posición final:

$$\overline{p_0} + \overline{p_1} + \cdots + \overline{p_N} = p_0 + p_1 + \cdots + p_N \quad (4.10)$$

Dado que se trata de puntos tridimensionales $p_i = (p_{ix}, p_{iy}, p_{iz})$ las ecuaciones quedarían de la siguiente manera:

$$\overline{p_{0x}} + \overline{p_{1x}} + \cdots + \overline{p_{Nx}} = p_{0x} + p_{1x} + \cdots + p_{Nx} \quad (4.11)$$
$$\overline{p_{0y}} + \overline{p_{1y}} + \cdots + \overline{p_{Ny}} = p_{0y} + p_{1y} + \cdots + p_{Ny} \quad (4.12)$$
$$\overline{p_{0z}} + \overline{p_{1z}} + \cdots + \overline{p_{Nz}} = p_{0z} + p_{1z} + \cdots + p_{Nz} \quad (4.13)$$

3. La nueva Bézier ha de pasar por $q(s_{min})$ en el instante de tiempo $t_{min} = t_0 + s_{min}\,(t_f - t_0)$
Esto supone que:

$$\overline{p_0} + \overline{p_1}s_{min} + \overline{p_2}{s_{min}}^2 + \ldots + \overline{p_N}{s_{min}}^N = q_0 + q_1 s_{min} + q_2{s_{min}}^2 + \ldots + q_M{s_{min}}^M \quad (4.14)$$

De otra manera, sabiendo que $\overline{p_0} = p_0$ (restricción 1):

$$\overline{p_1}s_{min} + \overline{p_2}{s_{min}}^2 + \ldots + \overline{p_N}{s_{min}}^N = (q_0 - p_0) + q_1 s_{min} + q_2{s_{min}}^2 + \ldots + q_M{s_{min}}^M \quad (4.15)$$

Separando por coordenadas espaciales:

$$\overline{p_{1x}}s_{min} + \overline{p_{2x}}{s_{min}}^2 + \cdots + \overline{p_{Nx}}{s_{min}}^N = \quad (4.16)$$
$$= (q_{0x} - p_{0x}) + q_{1x}s_{min} + q_{2x}{s_{min}}^2 + \ldots + q_{Mx}{s_{min}}^M$$

$$\overline{p_{1y}}s_{min} + \overline{p_{2y}}{s_{min}}^2 + \cdots + \overline{p_{Ny}}{s_{min}}^N = \quad (4.17)$$
$$= (q_{0y} - p_{0y}) + q_{1y}s_{min} + q_{2y}{s_{min}}^2 + \ldots + q_{My}{s_{min}}^M$$

$$\overline{p_{1z}}s_{min} + \overline{p_{2z}}{s_{min}}^2 + \cdots + \overline{p_{Nz}}{s_{min}}^N = \quad (4.18)$$
$$= (q_{0z} - p_{0z}) + q_{1z}s_{min} + q_{2z}{s_{min}}^2 + \ldots + q_{Mz}{s_{min}}^M$$

Todos los conceptos que se van a utilizar a continuación son parte de la referencia [22], de la que se cogen los recursos vectoriales y matriciales para proceder al cálculo. El planteamiento para obtener la nueva curva siguiendo el método de los mínimos cuadrados con restricciones es el siguiente:

Encontrar "x" minimizando $\|Ax - B\|^2$ sujeto a Cx=d

Donde:

- Ax es la nueva curva de Bézier modificada, con :

$$x = \begin{bmatrix} \overline{p_{1x}} \\ \overline{p_{1y}} \\ \overline{p_{1z}} \\ \overline{p_{2x}} \\ \overline{p_{2y}} \\ \overline{p_{2z}} \\ \vdots \\ \overline{p_{Nx}} \\ \overline{p_{Ny}} \\ \overline{p_{Nz}} \end{bmatrix} \tag{4.19}$$

$$A = \begin{pmatrix} s_k & 0 & 0 & {s_k}^2 & 0 & 0 & \dots & {s_k}^N & 0 & 0 \\ 0 & s_k & 0 & 0 & {s_k}^2 & 0 & \dots & 0 & {s_k}^N & 0 \\ 0 & 0 & s_k & 0 & 0 & {s_k}^2 & \dots & 0 & 0 & {s_k}^N \\ 1 & 0 & 0 & 2s_k & 0 & 0 & \dots & N{s_k}^{N-1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 2s_k & 0 & \dots & 0 & N{s_k}^{N-1} & 0 \\ 0 & 0 & 1 & 0 & 0 & 2s_k & \dots & 0 & 0 & N{s_k}^{N-1} \end{pmatrix} \tag{4.20}$$

Con $s_k \in (0,1) - \{s_{min}\}$ ya que en s_{min} es donde se va a aplicar una de las restricciones.

-B es la curva de Bézier primitiva:

$$B = \begin{pmatrix} p_{1x}s_k + p_{2x}{s_k}^2 + \dots + p_{Nx}{s_k}^N \\ p_{1y}s_k + p_{2y}{s_k}^2 + \dots + p_{Ny}{s_k}^N \\ p_{1z}s_k + p_{2z}{s_k}^2 + \dots + p_{Nz}{s_k}^N \\ p_{1x} + 2p_{2x}s_k + \dots + Np_{Nx}{s_k}^{N-1} \\ p_{1y} + 2p_{2y}s_k + \dots + Np_{Ny}{s_k}^{N-1} \\ p_{1z} + 2p_{2z}s_k + \dots + Np_{Nz}{s_k}^{N-1} \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} \text{Ecs. De trayectoria} \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} \text{Ecs. De velocidad} \end{matrix} \tag{4.21}$$

-Restricciones:

$$C = \left(\begin{array}{ccccccccccc} 1 & 0 & 0 & & 1 & 0 & 0 & \dots & & 1 & 0 & 0 \\ 0 & 1 & 0 & & 0 & 1 & 0 & \dots & & 0 & 1 & 0 \\ 0 & 0 & 1 & & 0 & 0 & 1 & \dots & & 0 & 0 & 1 \\ \hline {s_{min}} & 0 & 0 & {s_{min}}^2 & 0 & 0 & \dots & {s_{min}}^N & 0 & 0 \\ 0 & {s_{min}} & 0 & 0 & {s_{min}}^2 & 0 & \dots & 0 & {s_{min}}^N & 0 \\ 0 & 0 & {s_{min}} & 0 & 0 & {s_{min}}^2 & \dots & 0 & 0 & {s_{min}}^N \end{array}\right) \tag{4.22}$$

$$d = \begin{pmatrix} p_{1x} + p_{2x} + \dots + p_{Nx} \\ p_{1y} + p_{2y} + \dots + p_{Ny} \\ p_{1z} + p_{2z} + \dots + p_{Nz} \\ q_{0x} - p_{0x} + q_{1x}s_{min} + \dots + q_{Mx}{s_{min}}^M \\ q_{0y} - p_{0y} + q_{1y}s_{min} + \dots + q_{My}{s_{min}}^M \\ q_{0z} - p_{0z} + q_{1z}s_{min} + \dots + q_{Mz}{s_{min}}^M \end{pmatrix} \tag{4.23}$$

Cabe destacar que para construir la nueva Bézier sólo se busca que tenga una trayectoria y una velocidad parecidas, lo que conlleva de manera implícita que también cumpla con la aceleración.

Para realizar el cálculo matricial y obtener "x" es necesario plantear en Matlab la siguiente ecuación que proporciona la referencia [22] para resolver el método de optimización de los Mínimos Cuadrados con Restricciones:

$$\text{Solución } \hat{x} \longrightarrow \begin{pmatrix} \hat{x} \\ \hat{z} \end{pmatrix} = \left[\begin{array}{c|c} 2A^TA & C^T \\ \hline C & 0 \end{array}\right]^{-1} * \begin{pmatrix} 2A^TB \\ \hline d \end{pmatrix} \tag{4.24}$$

El vector $\hat{x}$ se constituye de los primeros 3*N componentes del vector resultado, que contiene los *Coefficient Points* de la nueva curva Bézier.

Su implementación en Matlab queda:

```
%x solución
    X=[2*A.'*A,C.';C,zeros(6)]^(-1)*[2*A.'*B.';d.'];
    for i=1:(ordenN*3)
        x(i)=X(i);
    end
```

puesto que las matrices B y d se generan como vectores horizontales en el programa.

Continuando con el ejemplo mostrado en las figuras 4.1 y 4.2, se observa cómo se modifica la trayectoria de la curva de Bézier N para colisionar con la curva de orden M en el instante de mínima separación entre ambas trayectorias:

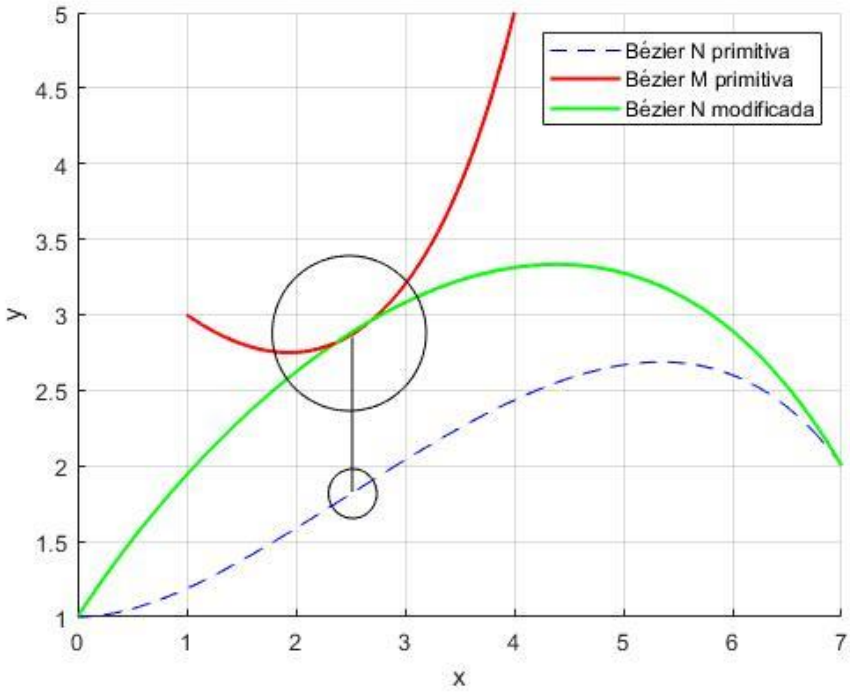

Figura 4. 3 Nueva trayectoria Bézier modificada para propiciar la colisión en s_{min}

CAPÍTULO 5. IMPLEMENTACIÓN DE LAS TRAYECTORIAS EN ROBOT LEGO NXT

5.1. Introducción a Lego NXT y RobotC

Los LEGO MINDSTORMS han añadido una nueva dimensión al universo de LEGO. Es una gama de productos orientada a la mecatrónica y a la automática. Estos dispositivos ahora disponen de microcontroladores, actuadores y sensores. Debido a estos mecanismos, los LEGO pueden sentir y responder a su entorno, y además pueden ser programados para llevar a cabo casi cualquier función [8].

Figura 5. 1 Robot LEGO Mindstorms NXT [10]

Estos robots son actualmente una herramienta muy común en la enseñanza, dado que despiertan un gran interés por su simplicidad y alta funcionalidad. Además, su coste medianamente elevado lo distingue de un juguete infantil, y lo convierte en material de laboratorio, siendo en este ámbito barato en comparación con otros dispositivos automáticos que tengan funcionalidades parecidas.

Este kit contiene un ladrillo inteligente NXT y variedad de sensores y actuadores, así como multitud de piezas LEGO para posibilitar infinidad de montajes distintos en función del uso que se le quiera dar, además del cableado eléctrico.

La versión empleada en este proyecto es "LEGO Mindstorm NXT education", cuyo hardware está compuesto por los siguientes elementos [11]:

1. **El "ladrillo" inteligente NXT**: es el elemento principal de los robots, cuyas características electrónicas se describen a continuación:

    - El procesador principal: Atmel 32-bit ARM *processor*, AT91SAM7S256
        -256 KB FLASH
        -64 KB RAM
        -48 MHz
    - El coprocesador: Atmel 8-bit AVR *processor*, ATmega48
        -4 KB FLASH
        -512 Byte RAM
        -8 MHz.

- Cuatro puertos de entrada analógicos y digitales en su parte inferior
- Cuatro botones para la introducción de instrucciones
- Nivel de tensión de las baterías
- Temporizadores internos
- Comunicaciones USB y Bluetooth
- Tres puertos de salida en la parte superior que soportan la lectura de los *encoders*
- Pantalla LDC monocromática de 100 x 64 *pixels* y dimensiones 26 x 40.6 mm
- Altavoz de resolución de 8 bits y periodo de muestreo 2-16 KHz

Figura 5. 2 "Ladrillo" inteligente NXT y sus componentes [11]

2. **Actuadores NXT**: Estos componentes dotan al robot móvil del movimiento y la autonomía para llevar a cabo las funciones programadas.
   El set contiene tres unidades idénticas que tienen incorporado un motor de corriente continua, un conjunto de reductores de velocidad y codificadores rotatorios ópticos, conocidos como *encoders*, que perciben las rotaciones (360 cuentas/rotación). Los *encoders* son incrementales y dependen del sentido de movimiento del motor, incrementándose en un sentido y decrementándose al girar en sentido contrario. Estos motores tienen una velocidad máxima de 200 rpm aproximadamente.

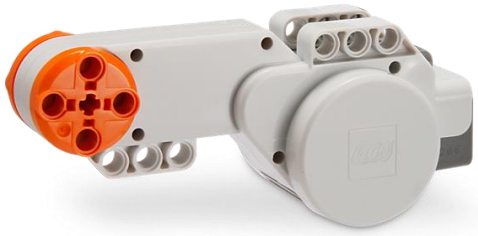

Figura 5. 3 Actuador NXT [20]

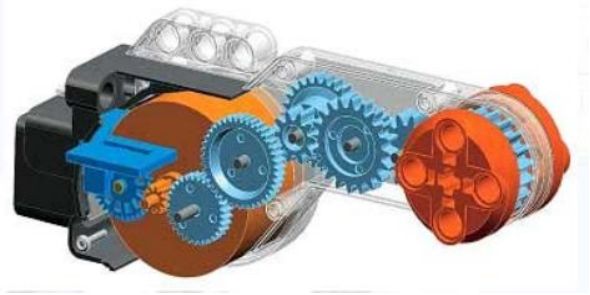

Figura 5. 4 Interior del actuador NXT [9]

3. **Sensores NXT:** se describen a continuación los sensores que van incluidos en este kit:

   -Sensor de distancia de precisión entre 3cm y 100cm, mediante el uso de ultrasonidos.

Figura 5. 5 Sensor de distancia ultrasónico [14]

   -Sensor de luz que detecta el nivel de luz en una sola dirección, también incluye un LED para iluminar un objeto. La escala de luz es de 0 al 100, siendo 100 muy claro y 0 muy oscuro.

Figura 5. 6 Sensor de luz [9]

   -Sensor de contacto binario, se le asigna un valor de 0 cuando no está presionado y un valor de 1 si fue o está presionado

Figura 5. 7 Sensor de contacto [12]

-Sensor de sonido (DB y DBA). La escala es de 0 al 100, siendo el 100 un volumen alto, y el 0 silencio absoluto.

Figura 5. 8 Sensor de sonido [13]

Sin embargo, hay una infinidad de sensores que suministran otros fabricantes que también son aptos para su uso en este dispositivo NXT [16][17][18]. Algunos como:

-Brújulas

-Acelerómetros
-Giróscopos
-Buscadores IR
-Sensores de color
-Sensores de presión neumáticos
-Sensores de temperatura
-Cámaras de visión artificial
-Multiplexores de motores
-Comunicaciones de radio
-GPS

Figura 5. 9 "ladrillo" NXT con los sensores y los actuadores conectados [15]

En cuanto al software, se emplea para su programación el programa ROBOTC a pesar de haber multitud de posibilidades de programas distintos para su programación. Este programa emplea un sistema de programación de la universidad Carnegie Mellon basado en el lenguaje C, que permite escribir y depurar programas para robots. Dentro de la aplicación se pueden encontrar numerosos ejemplos escritos en dicho lenguaje para probarlos directamente con el robot para comprobar su funcionamiento. Además, soporta varias plataformas de robots, algunas como VEX, RCX, y NXT. [8]

5.2. Control de robots móviles

El trazado de trayectorias en automatismos requiere de un seguimiento continuo para que se produzca la menor discordancia posible entre la trayectoria que se desea implementar y la que el robot móvil va a realizar. Con este fin, se debe realizar tanto un control cinemático como dinámico para que el robot logre seguir la trayectoria de Bézier establecida con la máxima precisión posible. El control cinemático consiste en determinar la posición del robot en todo momento, así como la determinación de las acciones de control que han de intervenir para aproximar dicha posición a la de referencia. Mientras que el control dinámico trata de que las articulaciones del robot ajusten su movimiento al de las referencias propuestas por el control cinemático. El esquema de control es el siguiente:

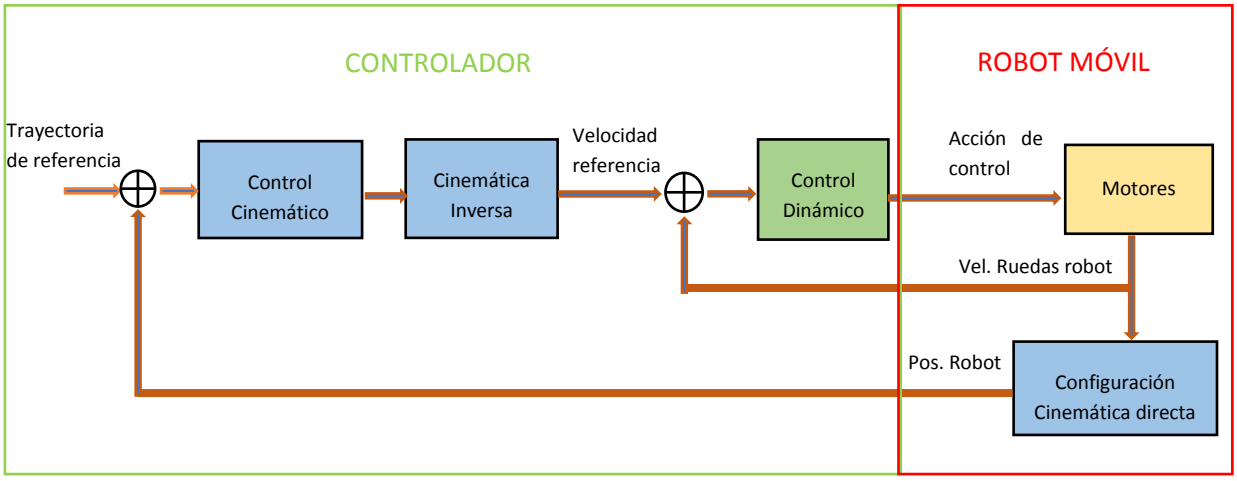

Figura 5. 10 Esquema del proceso de control de un robot móvil

5.2.1. Control cinemático

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia determinado. [19]

Siguiendo la referencia [21], el control cinemático de robots móviles consiste en determinar las acciones del robot necesarias para llevarlo desde una posición inicial hasta una posición final deseada, considerando la velocidad y la orientación de este, es decir, siguiendo una curva temporal. Este tipo de control tiene dos vertientes fundamentales a estudiar:

1. El modelo cinemático directo: permite obtener la posición y la orientación del robot en función de las velocidades y posiciones de las ruedas, conociendo los parámetros geométricos del robot.

2. El modelo cinemático inverso: permite obtener las velocidades y posiciones que se tienen que aplicar a las ruedas para que la posición y orientación del robot sea la deseada.

Para llevar a cabo este tipo de control, es necesario conocer en primer lugar el modelo cinemático por el cual se rige el robot móvil de estudio. Para ello es necesario conocer las diferentes disposiciones en las que este se puede montar, puesto que dependiendo de la estrategia de montaje se manejan unas variables u otras. Hay cuatro tipos de configuraciones cinemáticas distintas:

1. **Configuración Diferencial:** el robot presenta dos ruedas traseras de tracción, y una o dos ruedas locas de apoyo delantero. Con esta disposición el eje de rotación está situado en la mitad del eje de tracción. Conlleva una mayor dificultad a la hora de su control, pues es difícil conseguir trazar trayectorias totalmente rectas y se pueden producir deslizamientos, pero su diseño es muy sencillo.

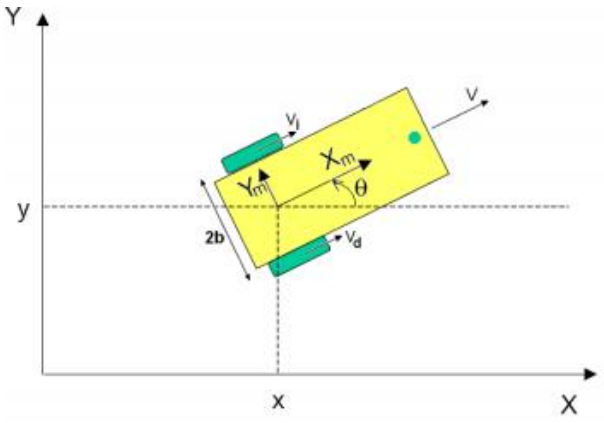

Figura 5. 11 Configuración diferencial del robot

Con este modelo cinemático se utilizan las siguientes variables para el control cinemático directo:

$$v = \frac{v_i + v_d}{2} \quad (5.1) \qquad \omega = \dot{\theta} = \frac{v_d - v_i}{2b} \quad (5.2)$$

2. **Configuración Oruga:** el robot tiene dos cadenas en lugar de ruedas en los laterales. Se pueden modelar como dos ruedas equivalentes en el centro de las cadenas. El diseño es parecido al de la configuración diferencial, pero añade un mejor sistema de tracción y previene los deslizamientos. Utiliza las mismas variables.

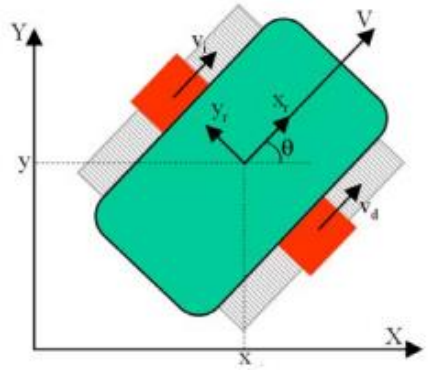

Figura 5. 12 Configuración de oruga

3. **Configuración de Triciclo:** el robot tiene dos ruedas traseras de tracción y otra rueda orientable, ya sea delantera o trasera. Normalmente se actúa sólo con dos variables, en este caso la orientación de la rueda delantera y la velocidad de avance, puesto que en otras configuraciones para determinar la orientación se actúa sobre cada una de las ruedas. También añade mayor fiabilidad frente a los deslizamientos.

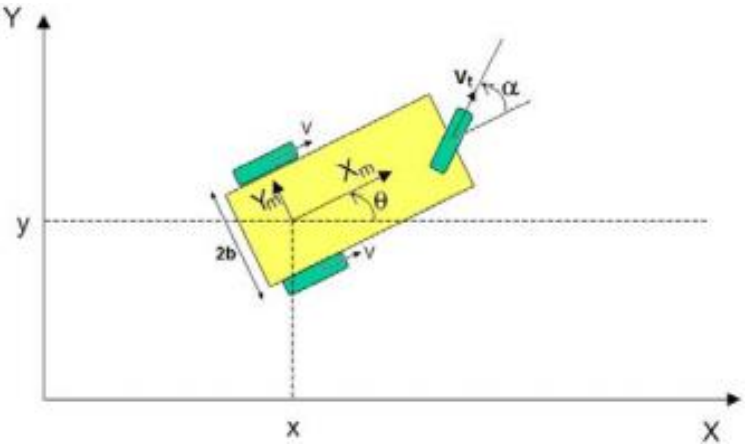

Figura 5. 13 Configuración de triciclo

Con este modelo cinemático se emplean las siguientes variables:

$$v = cos(\alpha)V_t \quad (5.3) \qquad \omega = \dot{\theta} = \frac{-sen(\alpha)}{l}V_t \quad (5.4)$$

4. **Configuración Ackerman:** el robot cuenta con cuatro ruedas, sin embargo, desde el punto de vista cinemático, esta configuración se puede considerar con una rueda equivalente a las dos ruedas delanteras. Esto supone modelar esta configuración de la misma manera que la configuración de triciclo.

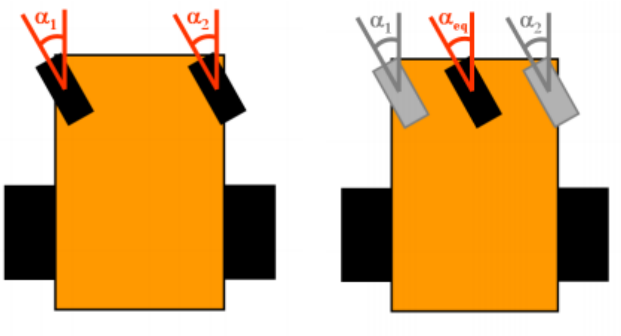

Figura 5. 14 Configuración Ackerman y equivalencia con la configuración de triciclo

En el caso práctico de este proyecto se utilizan dos robots de configuración diferencial, debido a la sencillez que proporciona en cuanto al montaje. Una vez conocido el modelo cinemático quedan definidas las variables que se van a utilizar en el control cinemático de los robots.

5.2.2. Control dinámico

Procura que la respuesta de las articulaciones del robot sean lo más parecidas posibles a las referencias propuestas por el control cinemático. Dado que en este proyecto se aborda el caso práctico de dos robots con configuración cinemática diferencial, se establece el control de velocidad de las ruedas izquierda (vi) y derecha (vd) considerando las referencias generadas por el control cinemático.

5.3. Implementación de las curvas de Bézier en LEGO MINDSTORMS NXT

Para entender cómo se ha logrado que el robot siga la trayectoria de Bézier diseñada se va a seguir el siguiente esquema en el que se muestra el proceso de control del movimiento del robot, siendo la trayectoria de referencia la diseñada anteriormente:

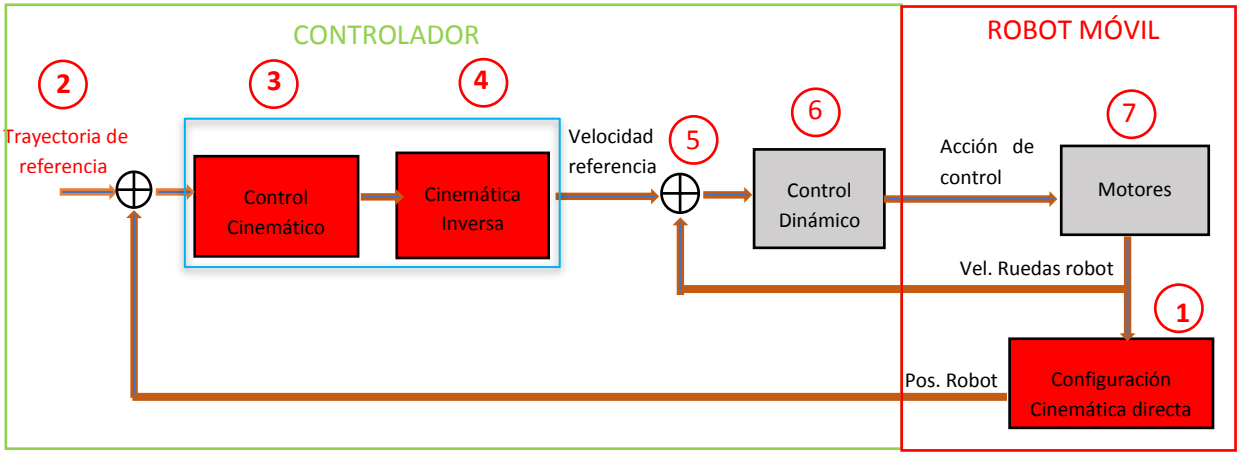

Figura 5. 15 Esquema a seguir para el desarrollo del código en RobotC

1. **Cinemática directa: determinación de la posición del robot.**
   Dado que se trata de dos robots de configuración diferencial, la posición se determina a partir de la velocidad de cada una de las ruedas, mediante los *encoders*. Además, se discretiza el tiempo en instantes k debido a que el periodo de muestreo (T_S) se va incrementando con un valor muy pequeño:

$$\omega_{d_k} = \frac{2\pi}{360}\frac{encoder_{d_k} - encoder_{d_{k-1}}}{T_S} \tag{5.5}$$

$$\omega_{i_k} = \frac{2\pi}{360}\frac{encoder_{i_k} - encoder_{i_{k-1}}}{T_S} \tag{5.6}$$

$$v_{d_k} = \omega_{d_k} * radioRueda \tag{5.7}$$

$$v_{i_k} = \omega_{i_k} * radioRueda \tag{5.8}$$

   Una vez conocida la velocidad de cada rueda se puede calcular mediante las ecuaciones (5.1) y (5.2) del control cinemático en modelos cinemáticos diferenciales, la velocidad lineal y la velocidad angular del robot:

$$v_k = \frac{v_{i_k} + v_{d_k}}{2} \tag{5.1}$$

$$\omega_k = \dot{\theta}_k = \frac{v_{d_k} - v_{i_k}}{2b} \tag{5.2}$$

Por tanto, la posición y la orientación del robot:

$$x_{k+1} = x_k + v_k T_S cos(\theta_k) \tag{5.9}$$

$$y_{k+1} = y_k + v_k T_S sen(\theta_k) \tag{5.10}$$

2. Cálculo de las referencias de posición y velocidad

Se trata de las trayectorias de Bézier y de las curvas de velocidad definidas en el plano, pues son las curvas que debería trazar el robot en condiciones ideales:

Referencias de posición:

$$x_{ref} = \mathrm{C_{0x}} + \mathrm{C_{1x}}s + \mathrm{C_{2x}}\mathrm{s}^2 + \cdots + \mathrm{C_{Nx}}\mathrm{s}^{\mathrm{N}}, s \in [0,1] \tag{5.12}$$

$$y_{ref} = \mathrm{C_{0y}} + \mathrm{C_{1y}}\mathrm{s} + \mathrm{C_{2y}}\mathrm{s}^2 + \ldots + \mathrm{C_{Ny}}\mathrm{s}^{\mathrm{N}}, \mathrm{s} \in [0,1] \tag{5.13}$$

Referencias de velocidad:

$$v_{xref} = \mathrm{C_{1x}} + 2\mathrm{C_{2x}}s + 3\mathrm{C_{3x}}\mathrm{s}^2 + \cdots + \mathrm{NC_{Nx}}\mathrm{s}^{\mathrm{N}-1}, s \in [0,1] \tag{5.14}$$

$$v_{yref} = \mathrm{C_{1y}} + 2\mathrm{C_{2y}}s + 3\mathrm{C_{3y}}\mathrm{s}^2 + \cdots + \mathrm{NC_{Ny}}\mathrm{s}^{\mathrm{N}-1}, s \in [0,1] \tag{5.15}$$

3. Cálculo de la expresión del control cinemático del robot

Se realiza un control de punto descentralizado, puesto que funciona muy bien para el control de trayectorias. Este control se realiza a en función de la posición y la velocidad de un punto que está a una distancia “g” del robot.

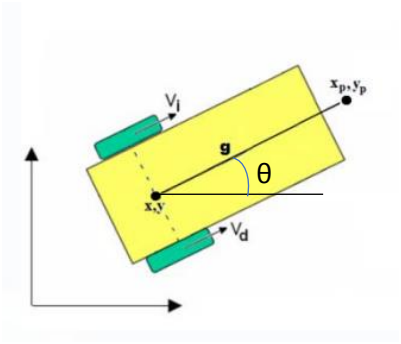

Figura 5. 16 Punto descentralizado

Posición del punto descentralizado:

$$x_p = x + gcos(\theta) \tag{5.16}$$

$$y_p = y + gsen(\theta) \tag{5.17}$$

Velocidad del punto descentralizado:

$$\dot{x}_p = \dot{x} - gsen(\theta)\dot{\theta} \tag{5.18}$$

$$\dot{y}_p = \dot{y} + gcos(\theta)\dot{\theta} \tag{5.19}$$

Para este control se propone un control proporcional con prealimentación de la velocidad [21]:

$$\dot{x}_p = k_{r_v} v_{xref} + k_{r_p}(x_{ref} - (x + gcos(\theta)) \tag{5.19}$$

$$\dot{y}_p = k_{r_v} v_{yref} + k_{r_p}(y_{ref} - (y + gsen(\theta)) \tag{5.20}$$

Donde la ganancia proporcional es k_{r_p}=4.0 y la ganancia derivativa es k_{r_v}=0.5

4. Cinemática inversa: determinación de las velocidades de referencia del robot

Una vez se ha aplicado el control cinemático, se introduce en las siguientes ecuaciones ((5.21) y (5.22)), resultantes de sustituir en las velocidades del punto descentralizado (5.16) y (5.17) las ecuaciones del control cinemático directo (5.1) y (5.2):

$$\dot{x}_p = \dot{x} - gsen(\theta)\dot{\theta} = vcos(\theta) - gsen(\theta)\omega = \frac{v_i + v_d}{2} cos(\theta) - gsen(\theta)\frac{v_d - v_i}{2b} \tag{5.21}$$

$$\dot{y}_p = \dot{y} + gcos(\theta)\dot{\theta} = vsen(\theta) + gcos(\theta)\omega = \frac{v_i + v_d}{2} sen(\theta) + gcos(\theta)\frac{v_d - v_i}{2b} \tag{5.22}$$

Despejando v_d y v_i:

$$v_{iref} = \frac{1}{g}((gcos(\theta) + bsen(\theta))\dot{x}_p + (gsen(\theta) - bcos(\theta))\dot{y}_p) \tag{5.23}$$

$$v_{dref} = \frac{1}{g}((gcos(\theta) - bsen(\theta))\dot{x}_p + (gsen(\theta) + bcos(\theta))\dot{y}_p) \tag{5.24}$$

$$\omega_{iref} = \frac{v_{iref}}{radioRueda} \tag{5.25}$$

$$\omega_{dref} = \frac{v_{dref}}{radioRueda} \tag{5.26}$$

5. **Cálculo de los errores de la velocidad angular de las ruedas**
   Se calcula la diferencia entre la velocidad angular que tiene la rueda del robot y la que debería tener según la referencia, para así poder realizar el control dinámico:

$$\text{Error\_wd} = \omega_{dref} - \omega_d \tag{5.27}$$

$$\text{Error\_wi} = \omega_{iref} - \omega_i \tag{5.28}$$

6. **Control dinámico: Cálculo de las acciones de control a aplicar a cada rueda**
   Para el control dinámico se va a emplear un control proporcional con Kmp=9.4276:

$$acontrolA = Error\_wd * Kmp \tag{5.29}$$

$$acontrolB = Error\text{\_wi} * Kmp \tag{5.30}$$

7. **Aplicación de las acciones de control a los motores**
   Se aplica a cada motor la acción de control resultante, que producirá una velocidad en las ruedas muy similar a la velocidad que se busca implementar en el robot según las curvas de Bézier, cerrando así el ciclo

CAPÍTULO 6. PRUEBAS EXPERIMENTALES

En este capítulo se va a exponer un caso real, en el cual se van a implementar dos trayectorias de Bézier en dos robots LEGO Mindstorms NXT. Al tratarse de un proceso real, se va a poder observar el error producido por el robot en comparación con la trayectoria de referencia. Asimismo, se adjunta el vídeo del experimento para su comprobación.

6.1. Introducción de datos en Matlab

En primer lugar, se introducen las matrices que contienen los *Control Points* de la curva de Bézier de orden N y de la curva de Bézier de orden M en Matlab:

>> Pn= [0 0 0;0.5714 1.14286 0;1.42857 0.28571 0;2 0 0]

Pn =

0	**0**	**0**
0.5714	**1.1429**	**0**
1.4286	**0.2857**	**0**
2.0000	**0**	**0**

>> Pm= [0 0.8571 0;0.85714 0.85714 0;2 0.8571 0]

Pm =

0	**0.8571**	**0**
0.8571	**0.8571**	**0**
2.0000	**0.8571**	**0**

A continuación, se llama a la función que ha sido programada para que, al introducir dichas matrices junto con el orden de las curvas, el radio de las esferas y el tiempo inicial y final en el que se deseen definir las curvas devuelva el instante de máximo acercamiento entre las curvas s_{min}, y t_{min} . Además, también devuelve el valor de la distancia entre las curvas en dicho instante, así como los *Coefficient Points* de las tres curvas: la curva N, la curva M, y la curva N modificada en busca de la colisión en el instante de máximo acercamiento entre ellas.

En este caso orden N= 3, orden M=2, $t_0 = 0, t_f = 3,\ radio\ N = 0.07, radio\ M = 0.07$

Trabajando Matlab con las unidades de longitud en metros, y el tiempo en segundos.

Se ha escogido un radio de 0.07 m para ambas esferas ya que es una buena aproximación del radio que tendría cada uno de los robots si se consideraran dentro de un volumen esférico.

Por tanto:

>> [smin,tmin,MTD,Cn,Cm,Cnb] = trayectorias_Bezier(3,2,Pn,Pm,0,3,0.07,0.07)

smin =

0.37179797974738525766567956028461

tmin =

1.1153939392421557729970386808538

MTD =

0.14399318518908292855951181911 3

Cn =

0	0	0
1.7142	3.4286	0
0.8573	-6.0000	0
-0.5715	2.5715	0

Cm =

0	0.8571	0
1.7143	0.0001	0
0.2857	-0.0001	0

Cnb =

[0, 0, 0]
[1.435649625286992439047980269 9802, 4.998611095375323254233508924 6984, 0]
[1.313763738664908507217372703 6438, -8.572801851563873289437882815785, 0]
[-0.749413363951903574909837126 56589, 3.574190756188548014840129695 8239, 0]

Dichos datos se representan en la siguiente gráfica:

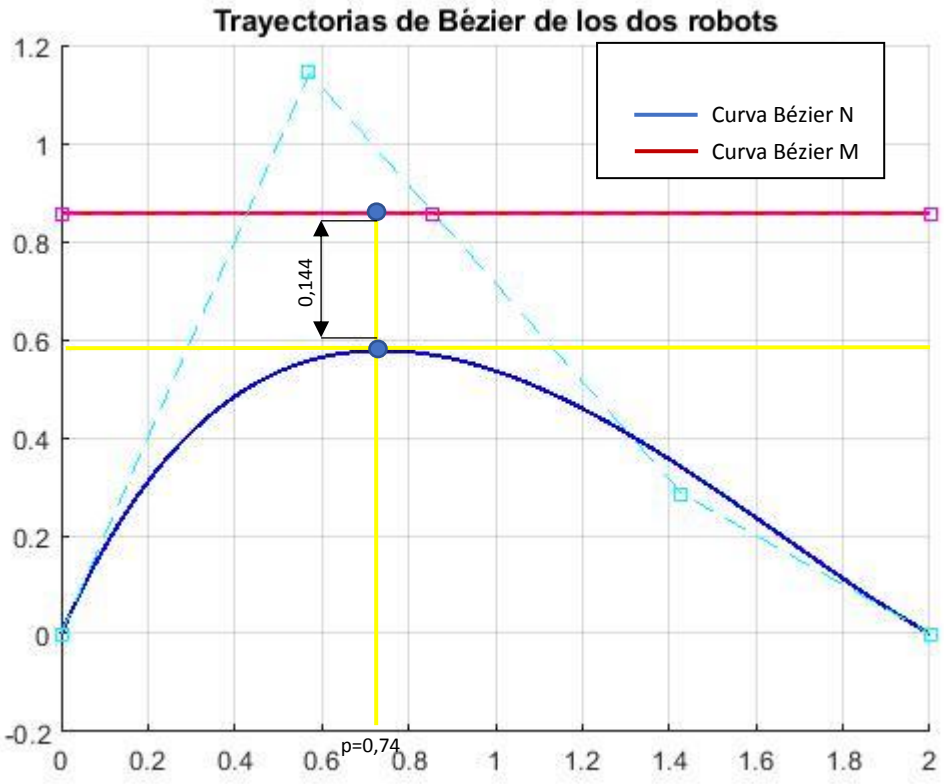

Figura 6. 1 Gráfica de las curvas de Bézier de orden N=3 y M=2

Las trayectorias no se han representado en función del tiempo, se han representado en función de su polígono de control correspondiente. No obstante, se puede comprobar que se cumple la distancia mínima entre las curvas en el instante $t_{min} = 1{,}1154$ de la siguiente forma:

$$\begin{cases} \mathrm{p} \in [0\,,2] \\ \mathrm{t} \in [0{,}3] \end{cases} \longrightarrow \frac{(\mathrm{p}-p_0)}{(p_f-p_0)} = \frac{(\mathrm{t}-t_0)}{(t_f-t_0)} \xrightarrow{\boldsymbol{t=1{,}1154}} \mathrm{p} = 0{,}7426 \qquad (6.1)$$

Donde: p es el intervalo en el cual queda definida la gráfica.

Una vez determinado el instante de mínima separación entre las curvas, se calcula la curva de Bézier N modificada para que en el punto p colisione con la curva de orden M. Esta curva se representa mediante los *Coefficient Points* de la nueva Bézier obtenidos en Matlab "Cnb", y se la va a denominar curva de Bézier Ñ:

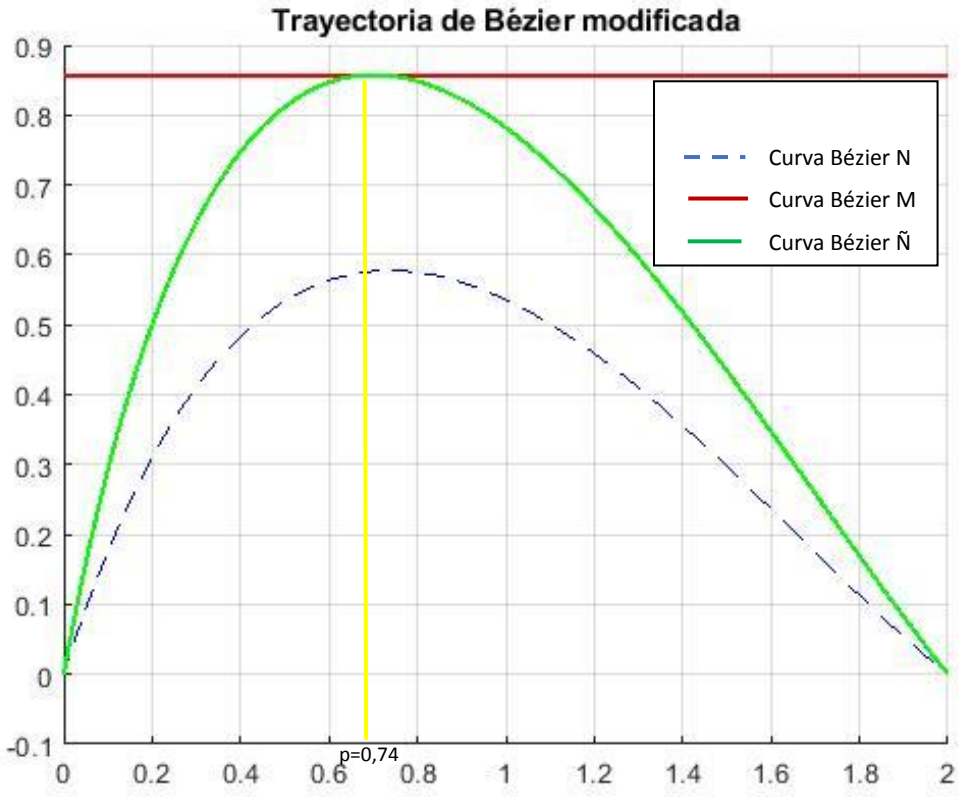

Figura 6. 2 Gráfica de las curvas de Bézier de orden N=3, M=2 y Ñ=3

6.2. Seguimiento de la trayectoria implantada en el robot

A la hora de la implementación de estas curvas en los robots Lego, se dan tres casos experimentales:

1. Implementación de las trayectorias y curvas de velocidad con un periodo de generación de puntos dentro de la curva de s=0.001:

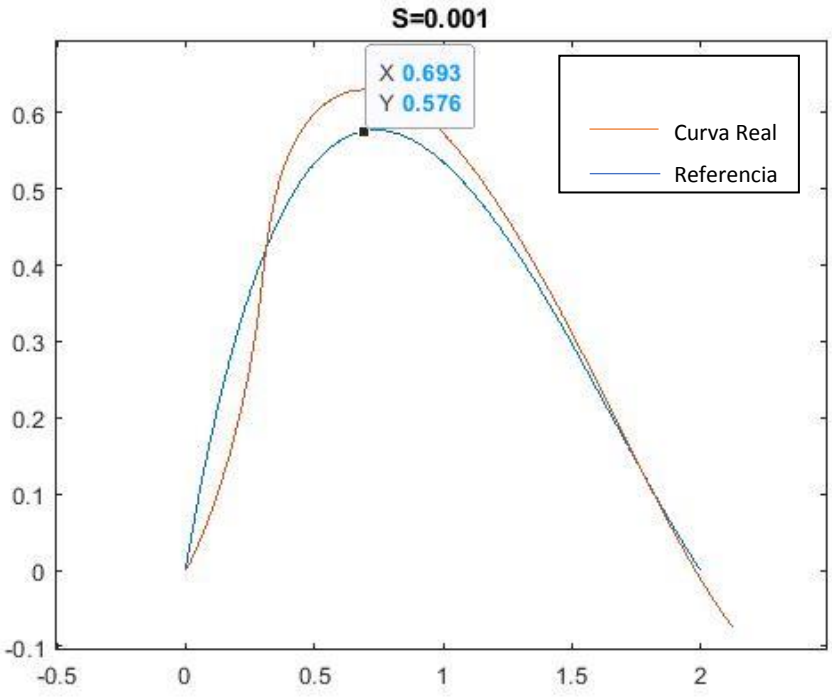

Figura 6. 3 Representación de la trayectoria trazada por el robot frente a la referencia con s=0.001

2. Implementación de las trayectorias y curvas de velocidad con un periodo de generación de puntos dentro de la curva de s=0.002:

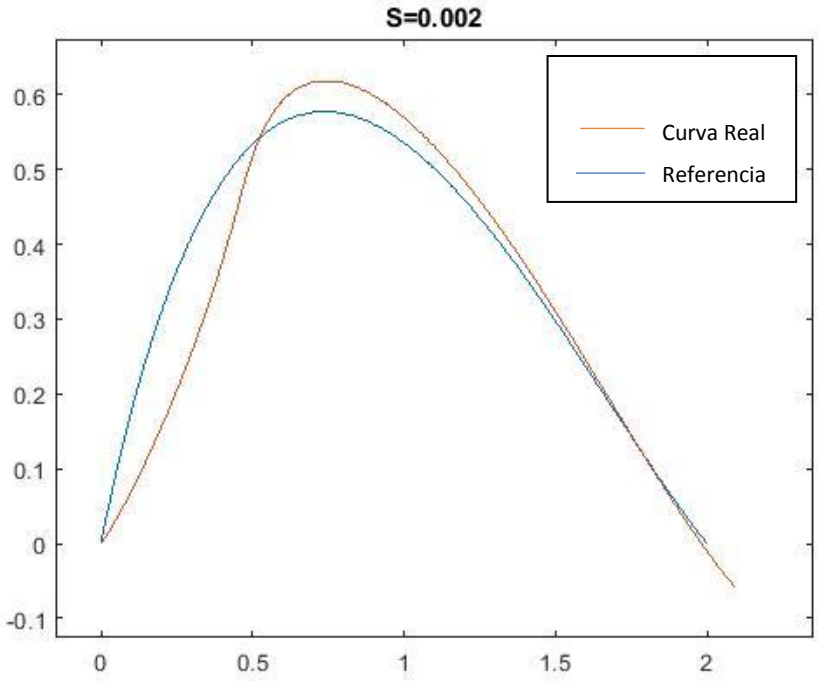

Figura 6. 4 Representación de la trayectoria trazada por el robot frente a la referencia con s=0.002

3. Implementación de las trayectorias y curvas de velocidad con un periodo de generación de puntos dentro de la curva de s=0.003:

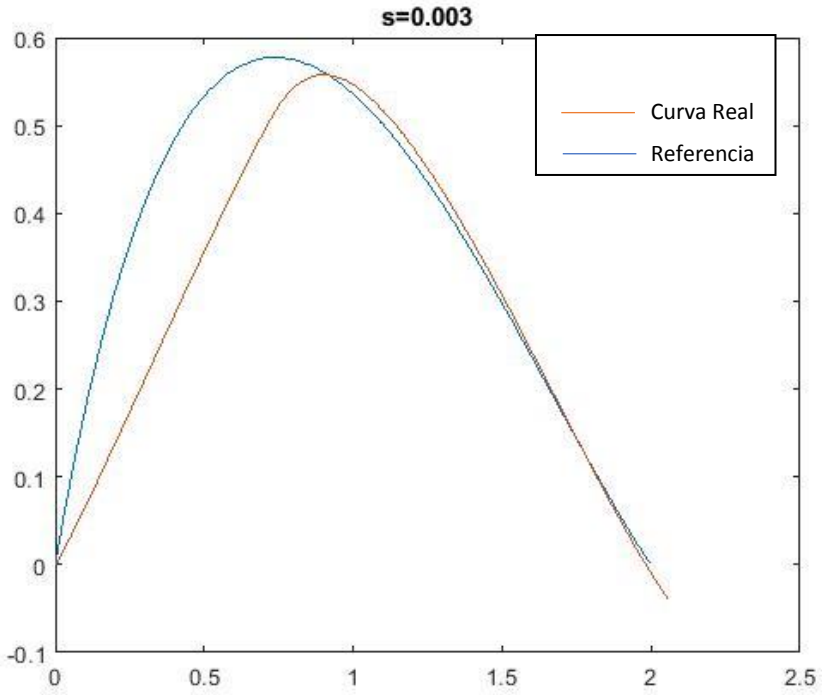

Figura 6. 5 Representación de la trayectoria trazada por el robot frente a la referencia con s=0.003

Se observa cómo al aumentar el periodo de generación de puntos en la trayectoria aumenta el error que se produce entre ambas curvas. Con el fin de minimizarlo lo máximo posible, se opta por un periodo de s=0.001

Además de estos casos, se observa en el laboratorio que, a mayor velocidad de trazado de la curva por el robot, mayor es el error. Debido a esto, a pesar de introducir como referencia de velocidad proveniente de derivar la trayectoria de Bézier, se ha probado a aplicar a dicha ecuación un coeficiente de decremento de (1/3), reduciendo en cada instante la velocidad del robot a un tercio de la de referencia.

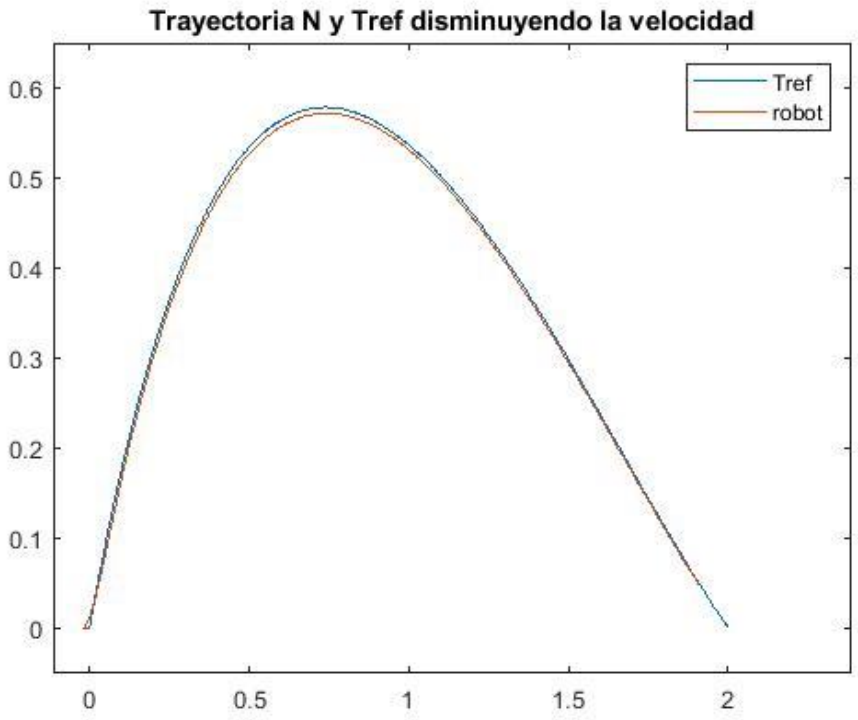

Figura 6. 6 Representación de la trayectoria trazada por el robot frente a la referencia con una disminución de la velocidad de referencia

Es claro que el error se hace prácticamente nulo, con la condición de una pérdida de velocidad considerable en el robot. A todos los efectos, y al tratarse de robots LEGO de baja resistencia a colisiones, se persigue que exista una colisión, pero sin que estos modifiquen su posterior trazado de la trayectoria, tal y como muestran las gráficas. Por esta razón no se considera una gran pérdida la disminución de la velocidad para el perfeccionamiento en el seguimiento de la trayectoria en el robot.

De la misma manera, también se ha modificado ligeramente el punto donde se produce la colisión, modificando la distancia de separación que tienen las trayectorias de referencia en el eje y, para que la colisión no modifique las trayectorias de los robots ni peligre la integridad estructural de ambos. La separación mínima necesaria para que se siga produciendo la colisión entre los robots, pero haciendo que ésta sea los más suave posible es “d”, se puede apreciar en la figura 6.7.

Suponiendo que el radio de la esfera que ocupa cada robot es de 0.07m, la posición en el eje ordenadas de uno de los robots debería alejarse una distancia de d=0.14m de la posición de referencia, que equivale a dos veces el radio de las esferas. Es decir, observando la figura 6.8, habría que situar al robot 2 a una distancia de 0.857+0.14= 1m en el eje de ordenadas respecto del origen de coordenadas.

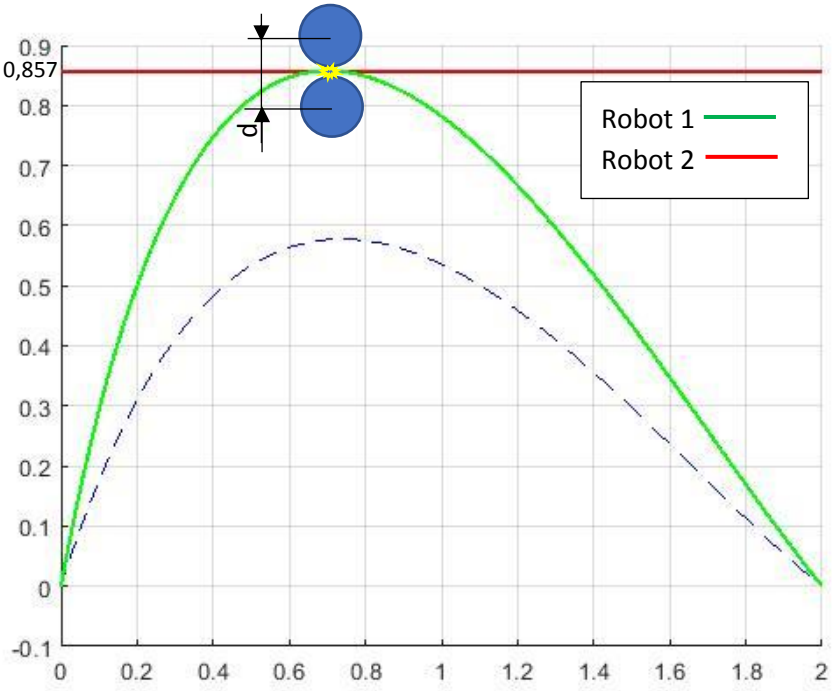

Figura 6. 7 Modificación de la distancia de inicio de los robots para una colisión más suave

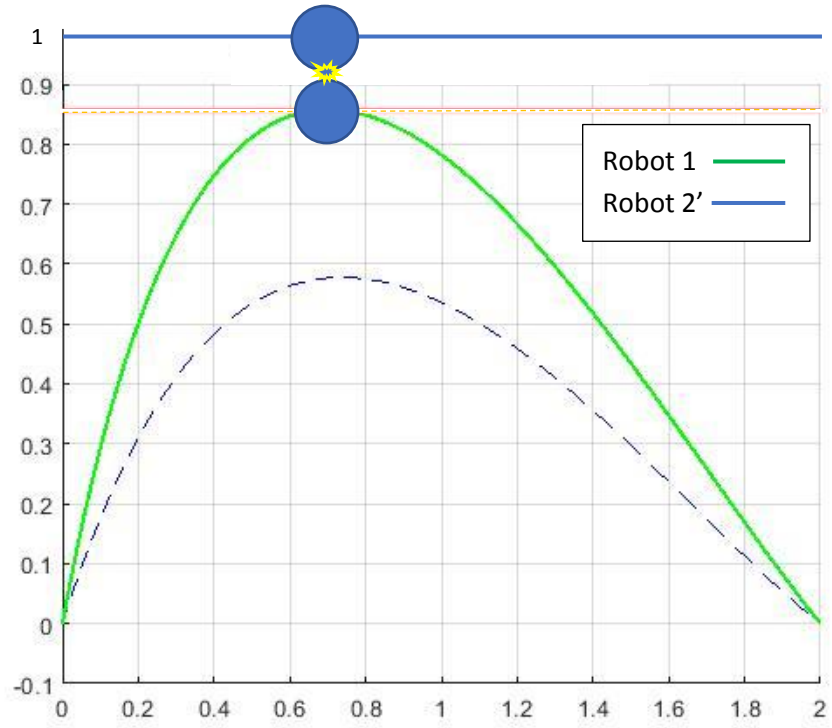

Figura 6. 8 Posición del robot 2 modificada para una colisión suave

CAPÍTULO 7. CONCLUSIONES

El objetivo principal de este proyecto era provocar una colisión entre dos robots móviles Lego, cuyos resultados han sido favorables, teniendo en cuenta sus limitaciones a nivel estructural. Esto supone que se ha alcanzado el poder definir las trayectorias siguiendo curvas de Bézier, y su definición gráfica. A su vez, estas se han podido adaptar adecuadamente a las trayectorias de referencia de los robots, a pesar de tener que adaptar el código de Matlab a RobotC, pues son lenguajes de programación distintos.

Gracias a esto, y al control realizado a los robots, se ha logrado que tracen dichas trayectorias a la velocidad con la que éstas han de ser trazadas con un error muy bajo. Sin embargo, se ha considerado la posibilidad de asumir una pérdida en la velocidad de recorrido en el robot, para lograr una mejora en el error que se produce entre la trayectoria diseñada de referencia y la trayectoria real.

Esto ha resultado cumplirse, pues al disminuir la velocidad del robot, este puede actualizar con más precisión el valor que miden los *encoders* de las ruedas a lo largo de todo el recorrido, mejorando los datos de la posición que va trazando.

Además, se ha modificado ligeramente el punto de interceptación de los robots para que no se vea afectada su integridad estructural en el momento de la colisión, y para que no se produzcan cambios en la trayectoria que siguen los dos después de que dicho encuentro.

No obstante, se han encontrado diversas dificultades a la hora de la programación en Matlab, debido a que las matrices introducidas eran de tamaños variables y por lo general, de gran tamaño, necesitando de varios bucles anidados para su definición. Así como la implementación en el robot Lego, en el que se han tenido que realizar muchas pruebas experimentales para poder averiguar por qué en un principio no seguía la trayectoria implementada en su totalidad.

En conclusión, el trabajo contenido en este proyecto ha resultado ser viable según la parte experimental.

CAPÍTULO 8. BIBLIOGRAFÍA

[1] Página Web. Curvas de Bézier Wikipedia, La enciclopedia libre [última fecha de consulta: 15 de mayo de 2019]. Disponible en <https://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier>.

[2] Página Web. Solid Modeling Association, [última fecha de consulta: 20 de mayo de 2019]. Disponible en <http://solidmodeling.org/awards/bezier-award/>

[3] Página Web. Geometría dinámica [última fecha de consulta: 3 de junio de 2019] Disponible en <http://www.geometriadinamica.cl/2010/12/curvas-de-bezier/>

[4] Página Web. Spline Curves [última fecha de consulta: 3 de junio de 2019]. Disponible en <https://people.cs.clemson.edu/~dhouse/courses/405/.../splines.pdf>

[5] Página Web. Interpolation and Spline Modeling [última fecha de consulta: 3 de junio de 2019]. Disponible en <https://www.cs.csustan.edu/~rsc/CS3600F02/Splines.pdf>

[6] Página Web. Tema 2. Curvas de Bézier [última fecha de consulta: 5 de junio de 2019]. Disponible en <https://dcain.etsin.upm.es/~leonardo/tema2.pdf>

[7] Página Web. Mínimos cuadrados, Wikipedia, La enciclopedia libre [última fecha de consulta: 13 de junio de 2019]. Disponible en <https://es.wikipedia.org/wiki/M%C3%ADnimos_cuadrados>

[8] Página Web. LEGO MindStorms NXT, Wikipedia, La enciclopedia libre [última fecha de consulta: 28 de mayo de 2019]. Disponible en <https://es.wikipedia.org/wiki/Lego_MindStorm_NXT>.

[9] Página Web. LEGO Mindstorms en español [última fecha de consulta: 28 de mayo de 2019]. Disponible en <http://rbtnxt.blogspot.com/2009/06/los-motores-del-lego-mindstorms-nxt.html>

[10] Página Web. The Next Step [última fecha de consulta: 28 de mayo de 2019]. Disponible en <http://www.thenxtstep.com/2006/02/lego-mindstorms-nxt-user-panel-no-news.html>

[11] Página Web. EsMindstorms [última fecha de consulta: 28 de mayo de 2019]. Disponible en <https://www.esmindstorms.com/ladrillo-inteligente-mindstorms-nxt/>.

[12] Página Web. Amazon [última fecha de consulta: 16 de junio de 2019]. Disponible en <https://www.amazon.com/LEGO-4297096-MINDSTORMS>.

[13] Página Web. Lego.com [última fecha de consulta: 16 de junio de 2019]. Disponible en <https://shop.lego.com/de-CH/>

[14] Página Web. Mercado libre [última fecha de consulta: 16 de junio de 2019]. Disponible en <https://articulo.mercadolibre.com.co/MCO-472438361-sensor-ultrasonico-lego-mindstorms-nxt-9846-_JM?quantity=1>

[15] Página Web. Research Gate [última fecha de consulta: 18 de junio de 2019]. Disponible en <https://www.researchgate.net/figure/Figura-5-Bloque-NXT-sensores-y-actuadores-El-controlador-bloque-NXT-o-brick-contiene_fig7_49599136>

[16] Página Web. Modern Robotics inc [última fecha de consulta: 18 de junio de 2019]. Disponible en <https://modernroboticsinc.com/product-category/hitechnic/>

[17] Página Web. Mind Sensors [última fecha de consulta: 18 de junio de 2019]. Disponible en <http://www.mindsensors.com/>

[18] Página Web. Lego Manía [última fecha de consulta: 24 de junio de 2019]. Disponible en <http://members.tripod.com/agiecco/>

[19] Página Web. Prezi.com [última fecha de consulta: 25 de junio de 2019]. Disponible en <https://prezi.com/9jalchlvy3-s/cinematica-directa-e-inversa-para-robots/>

[20] Página Web. XYZ sobre plastilina [última fecha de consulta: 25 de junio de 2019]. Disponible en <https://sites.google.com/site/xyzsobreplastilina/home/10-dispositivos-de-salida>

[21] Página Web. PoliformaT Laboratorio de Automatización [última fecha de consulta: 25 de junio de 2019]. Disponible en <https://poliformat.upv.es>

[22] S. Boyd and L. Vandenberghe, Introduction to Applied Linear Algebra, Cambridge University Press, 2018

[23] Enrique J. Bernabeu, Á. Valera, G. Kubiliute, "Deformation of Bézier Curves for Collision Avoidance and Trajectory Generation in a n-dimensional Space", remitido a publicación

[24] Página Web. Solid Works Help. [última fecha de consulta: 30 de junio de 2019] Disponible en: <http://help.solidworks.com/2010/spanish/solidworks/sldworks/legacyhelp/sldworks/SW_Sketch/Control_Polygons.htm>

II
PRESUPUESTO

PRESUPUESTO

1.Necesidad del presupuesto

Todo proyecto requiere una valoración económica del trabajo que necesita para su realización, pues en determinados casos es imprescindible para sopesar si es rentable su elaboración. En este proyecto se han tenido en cuenta todos los recursos utilizados para su elaboración, siendo estos la mano de obra, la maquinaria a emplear, los materiales y las licencias de software.

2.Cuadro de precios básicos

2.1. Cuadro de mano de obra

Si se considera la jornada laboral de 5 días a la semana y de 8 horas al día, se obtiene un total de:

$\frac{8\ horas}{día}\ \frac{5\ días}{semana}\ \frac{4\ semanas}{mes} = 160\ \frac{horas}{mes}$ de trabajo.

Si un ingeniero en tecnologías industriales recién titulado cobra 1500 $\frac{€}{mes}$, entonces:

$1500\ \frac{€}{mes} * \frac{1}{160}\frac{mes}{horas} = 9.375\ \frac{€}{hora}$

Si se trata de un profesor titular de la Universidad de salario aproximadamente 3600 $\frac{€}{mes}$:

$3600\ \frac{€}{mes} * \frac{1}{160}\frac{mes}{horas} = 22.5\ \frac{€}{hora}$

MANO DE OBRA				
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (€/Mes)**	**PRECIO (€/h)**
MO-ING	h	Ingeniero en Tecnologías Industriales	1500	9.38
MO-PRO	h	Profesor Titular de la Universidad	3600	22.5

Tabla 1 Cuadro de precios básicos de la mano de obra

2.2. Cuadro de maquinaria

Se considera que la vida útil de los objetos tecnológicos empleados en este proyecto es aproximadamente 5 años, debido al desgaste de las piezas y/o la obsolescencia. Este tiempo se corresponde con su periodo de amortización. Se tiene en cuenta que en este año 2019 hay 8 días festivos a nivel nacional, y 104 días libres en concepto de fines de semana, resultan 253 días laborales. Si la jornada laboral es de 8 horas al día, entonces el coste de amortización es:

Para el kit de LEGO Mindstorms NXT de coste 470,71 $\frac{€}{Ud}$:

470,71 $\frac{€}{Ud}\ \frac{1\ Ud}{5\ años}\ \frac{1\ año}{253\ días}\ \frac{1\ día}{8\ horas}$ = 0,0465 $\frac{€}{hora}$

Para el ordenador portátil de la marca Toshiba de coste 500 $\frac{€}{Ud}$:

500 $\frac{€}{Ud}\ \frac{1\ Ud}{5\ años}\ \frac{1\ año}{253\ días}\ \frac{1\ día}{8\ horas}$ = 0,05 $\frac{€}{hora}$

Para la cámara Web Logitech c170 de coste 263 $\frac{€}{Ud}$:

263 $\frac{€}{Ud}\ \frac{1\ Ud}{5\ años}\ \frac{1\ año}{253\ días}\ \frac{1\ día}{8\ horas}$ = 0,026 $\frac{€}{hora}$

MAQUINARIA			
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (€/h)**
MQ-LEG	h	LEGO Mindstorms NXT	0,05
MQ-ORD	h	Ordenador portátil	0,05
MQ-CAM	h	Cámara Web Logitech c170	0,03

Tabla 2 Cuadro de precios básicos de la maquinaria

2.3. Cuadro de licencias de Software

De nuevo se considera la jornada laboral de 8 horas al día y 253 días laborales al año, con lo que se obtiene que:

Para la licencia de Matlab:

500 $\frac{€}{año}\ \frac{1\ año}{253\ días}\ \frac{1\ día}{8\ horas}\ =\ 0{,}25\ \frac{€}{hora}$

Para la licencia de Microsoft Office

$$69\,\frac{€}{año}\,\frac{1\,año}{253\,días}\,\frac{1\,día}{8\,horas} = 0{,}0341\,\frac{€}{hora}$$

Para la licencia de RobotC:

$$53\,\frac{€}{año}\,\frac{1\,año}{253\,días}\,\frac{1\,día}{8\,horas} = 0{,}0262\,\frac{€}{hora}$$

LICENCIAS DE SOFTWARE				
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (€/Año)**	**PRECIO (€/h)**
LI-MAT	h	Licencia Matlab	500	0,25
LI-OFF	h	Licencia Microsoft Office	69	0,03
LI-ROB	h	Licencia RobotC	53	0,03

Tabla 3 Cuadro de precios básicos de las licencias de Software

2.4. Cuadro de materiales

MATERIALES			
CÓDIGO	**UNIDAD**	**DESCRIPCIÓN**	**PRECIO (€/Ud.)**
MF-FOT	UD	Fotocopias en color	0,05
MF-ENC	UD	Encuadernación	2
MF-MEM	UD	Memoria USB	15

Tabla 4 Cuadro de precios básicos de materiales

3. Cuadro de precios descompuestos por unidad de obra

En este apartado del presupuesto se exponen por capítulos las unidades de obra de las que consta el proyecto, así como la descomposición de los precios que contiene cada una de ellas.
Asimismo, se ha tenido en cuenta un valor del 2% en concepto de costes directos complementarios, como puede ser el gasto en electricidad para cargar el ordenador, la luz del habitáculo donde se trabaja, etc.

CÓDIGO	UNIDAD	DESCRIPCIÓN	RDTO.	PRECIO (€)	IMPORTE (€)
CAPÍTULO 1. TRABAJO PREVIO					
UD1.0	h	Búsqueda de la información		**48,35**	
	Recopilación de información e instalación de programas				
MO-ING	h	Ingeniero en Tecnologías Industriales	5	9,38	46,9
MQ-ORD	h	Ordenador portátil	5	0,05	0,25
LI-MAT	h	Licencia Matlab	1	0,25	0,25
	%	Costes Directos Complementarios	0,02	47,40	0,95
			Coste Total		**48,35**

Tabla 5 Cuadro de precios descompuestos por unidad de obra del capítulo 1

CÓDIGO	UNIDAD	DESCRIPCIÓN	RDTO.	PRECIO (€)	IMPORTE (€)
CAPÍTULO 2. REALIZACIÓN DEL PROYECTO					
UD2.0	h	Tutorías		**32,52**	
	Reunión con el profesor para la explicación de la parte teórica				
MO-ING	h	Ingeniero en Tecnologías Industriales	1	9,38	9,38
MO-PRO	h	Profesor Titular de la Universidad	1	22,5	22,5
	%	Costes Directos Complementarios	0,02	31,88	0,64
			Coste Total		**32,52**
UD2.1	h	Programación y simulación en Matlab		**9,87**	
	Programación de las trayectorias de Bézier, de las curvas de velocidad y de las curvas de aceleración, así como el algoritmo que modifica la trayectoria para la interceptación de un objeto móvil				
MO-ING	h	Ingeniero en Tecnologías Industriales	1	9,38	9,38
MQ-ORD	h	Ordenador portátil	1	0,05	0,05
LI-MAT	h	Licencia Matlab	1	0,25	0,25
	%	Costes Directos Complementarios	0,02	9,68	0,19
			Coste Total		**9,87**
UD2.2	h	Implementación de las trayectorias en RobotC		**9,90**	
	Adaptación del código de Matlab a RobotC para su posterior implementación en el LEGO Mindstorms NXT				
MO-ING	h	Ingeniero en Tecnologías Industriales	1	9,38	9,38
MQ-ORD	h	Ordenador portátil	1	0,05	0,05
LI-ROB	h	Licencia RobotC	1	0,03	0,03
LI-MAT	h	Licencia Matlab	1	0,25	0,25
	%	Costes Directos Complementarios	0,02	9,71	0,19
			Coste Total		**9,90**
UD2.3	h	Pruebas experimentales		**9,99**	
	Experimentación del funcionamiento real del robot, así como de la grabación de los resultados				
MO-ING	h	Ingeniero en Tecnologías Industriales	1	9,38	9,38
MQ-ORD	h	Ordenador portátil	1	0,05	0,05
MQ-LEG	h	LEGO Mindstorms NXT	1	0,05	0,05
MQ-CAM	h	Cámara Web Logitech c170	1	0,03	0,03
LI-ROB	h	Licencia RobotC	1	0,03	0,03
LI-MAT	h	Licencia Matlab	1	0,25	0,25
	%	Costes Directos Complementarios	0,02	9,79	0,20
			Coste Total		**9,99**

Tabla 6 Cuadro de precios descompuestos por unidad de obra del capítulo 2

CÓDIGO	UNIDAD	DESCRIPCIÓN	RDTO.	PRECIO (€)	IMPORTE (€)
CAPÍTULO 3. FORMALIZACIÓN DE LA MEMORIA					
UD3.0	h	Redacción de la memoria		**9,65**	
	Lectura de las normas de maquetación de la memoria y redacción del contenido del proyecto				
MO-ING	h	Ingeniero en Tecnologías Industriales	1	9,38	9,38
MQ-ORD	h	Ordenador portátil	1	0,05	0,05
LI-OFF	h	Licencia Microsoft Office	1	0,03	0,03
	%	Costes Directos Complementarios	0,02	9,46	0,19
			Coste Total		**9,65**
UD3.1	h	Revisión de la memoria y evaluación		**22,95**	
	Evaluación del profesor acerca del trabajo realizado y corrección de la memoria				
MO-PRO	h	Profesor Titular de la Universidad	1	22,5	22,5
	%	Costes Directos Complementarios	0,02	22,5	0,45
			Coste Total		**22,95**
UD3.2	UD	Impresión del documento		30,99	
	Impresión del documento del proyecto				
MO-ING	h	Ingeniero en Tecnologías Industriales	1	9,38	9,38
MF-FOT	UD	Fotocopias en color	80	0,05	4
MF-ENC	UD	Encuadernación	1	2	2
MF-MEM	UD	Memoria USB	1	15	15
	%	Costes Directos Complementarios	0,02	30,38	0,61
			Coste Total		**30,99**

Tabla 7 Cuadro de precios descompuestos por unidad de obra del capítulo

4. Estado de mediciones de todas las unidades del proyecto

CÓDIGO	UNIDAD	DESCRIPCIÓN	CANTIDAD	PRECIO (€)	IMPORTE (€)
CAPÍTULO 1. TRABAJO PREVIO					
UD1.0	h	Búsqueda de la información	25	48,35	1208,75
			Coste Total		1208,75
CAPÍTULO 2. REALIZACIÓN DEL PROYECTO					
UD2.0	h	Tutorías	5	32,5176	162,59
UD2.1	h	Programación y simulación en Matlab	160	9,8736	1579,78
UD2.2	h	Implementación de las trayectorias en RobotC	80	9,9042	792,34
UD2.3	h	Pruebas experimentales	10	9,9858	99,86
			Coste Total		2634,56
CAPÍTULO 3. FORMALIZACIÓN DE LA MEMORIA					
UD3.0	h	Redacción de la memoria	50	9,6492	482,46
UD3.1	h	Revisión de la memoria y evaluación	2	22,95	45,9
UD3.2	UD	Impresión del documento	1	30,9876	30,99
			Coste Total		559,35

Tabla 8 Estado de mediciones de todas las unidades de obra del proyecto

5. Presupuesto de inversión total

PRESUPUESTO DE INVERSIÓN		
CÓDIGO	**DESCRIPCIÓN**	**IMPORTE (€)**
CAP.1	TRABAJO PREVIO	1208,75
CAP.2	REALIZACIÓN DEL PROYECTO	2634,56
CAP.3	FORMALIZACIÓN DE LA MEMORIA	559,35
Presupuesto Ejecución Material (PEM)		4402,66
Gastos Generales (0,13 x PEM)		572,35
Beneficio Industrial (0,06 x BI)		264,16
Subtotal (PEC)		5239,16
IVA (0,21 x PEC)		1100,22
Presupuesto de Inversión Total		**6339,38**

Tabla 9 Presupuesto de inversión total

El presupuesto asciende a:

SEIS MIL TRESCIENTOS TREINTA Y NUEVE EUROS CON TREINTA Y OCHO CÉNTIMOS