UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA E INDUSTRIAL

EQUILIBRIO DINÁMICO DE UN ROBOT BÍPEDO UTILIZANDO EL MODELO SIMPLIFICADO CARRO-MESA

T E S I S

PARA OBTENER EL TÍTULO DE:

INGENIERO MECATRÓNICO

P R E S E N T A:

ULISES ALEJANDRO JIMÉNEZ RIOJA

DIRECTOR DE TESIS:

DR. EDMUNDO GABRIEL ROCHA CÓZATL

CIUDAD UNIVERSITARIA, MÉXICO, D.F. 2014

↬ *Dedico este trabajo especialmente a mi madre y a mi padre, por comprenderme y apoyarme incondicionalmente todos estos años.*

Agradecimientos

Le doy sinceramente las gracias a mi familia por apoyarme directa e indirectamente en los momentos difíciles y de enfermedad, por procurar que siempre contara con el tiempo y los recursos necesarios para estudiar la carrera y finalmente hacer posible este trabajo, producto de su esfuerzo también.

Le agradezco a mi mamá por los valores que me ha inculcado y por toda la dedicación y sacrificio que ha empeñado en mí y en mis hermanos, pero sobre todo por haberme ayudado a creer en mí, por enseñarme que con dedicación, paciencia y disciplina se puede lograr lo que uno se proponga.

A mi papá le agradezco por tratar de que fuera una buena persona e inculcarme conocimiento y cultura, así como el formar parte de la máxima casa de estudios en México.

Agradezco a la UNAM, por la educación íntegra que me ha brindado desde que comencé mi bachillerato, por el trato que me ha ofrecido como institución educativa y sobre todo por los excelentes profesores que me han transmitido sus conocimientos.

Al Dr. Edmundo le doy las gracias por la disposición que tuvo para apoyarme durante todo este proceso de titulación, así como por haberme compartido algunos consejos y recomendaciones académicas.

A todos mis amigos y compañeros que me aceptaron y brindaron su apoyo incondicional les agradezco por su amistad y compañerismo.

Le agradezco principalmente a la vida por las lecciones que me ha brindado a lo largo de todos estos años, así como por permitirme estar en contacto con la naturaleza y compartir momentos especiales con seres tan valiosos[†].

∞

Ayúdame a ayudarte.

Índice general

Resumen

En este trabajo se aborda el problema de generar trayectorias dinámicamente estables para mantener el equilibrio dinámico de un robot bípedo durante la marcha.

Actualmente los robots bípedos cuentan con un elevado número de grados de libertad para simular la marcha humana, por lo que generar trayectorias dinámicamente estables considerando la dinámica de cada uno de sus eslabones resulta ser una tarea exhaustiva que consume demasiados recursos computacionales. Por lo tanto, en este trabajo se emplea el modelo carro-mesa para representar de forma simplificada la dinámica del modelo real del robot bípedo Scout, ya que es un modelo robusto que hasta el momento no ha sido explorado en la Facultad de Ingeniería tanto como el modelo simplificado péndulo invertido lineal. En cuanto a la estabilidad, se utiliza el criterio de estabilidad del ZMP (*Zero Moment Point*) el cual ha sido empleado como criterio de estabilidad estándar en investigaciones sobre estabilidad durante la marcha bípeda.

Por otra parte, también se aborda el problema de emplear la menor carga computacional posible en la generación de trayectorias dinámicamente estables sin perder exactitud, por lo que se estudian e implementan los métodos de solución para el modelo carro-mesa que no hacen consideraciones que acarrean errores y que tampoco emplean cálculos matemáticos complejos que consumen demasiados recursos computacionales.

Como resultado de este trabajo se ofrece un panorama más amplio a la hora de seleccionar uno de los dos principales modelos empleados para la representación simplificada de la dinámica real de un robot bípedo, modelo carro-mesa y modelo péndulo invertido lineal. También se determinan los métodos más eficientes para resolver el modelo carro-mesa y se realizan las comparaciones correspondientes. En cuanto a la experimentación con el banco de pruebas, se realizan las pruebas necesarias para determinar la zona de mayor estabilidad en la planta de los pies.

Por otra parte, se presenta una aplicación móvil desarrollada en *Android* para enviar los ángulos a los servomotores del banco de pruebas. Dichos ángulos son obtenidos mediante cinemática inversa y determinan la configuración que debe adoptar el mecanismo del robot para seguir las trayectorias dinámicamente estables, generadas a partir del modelo simplificado carro-mesa y del criterio de estabilidad del ZMP.

Capítulo 1

Introducción

1.1. Estado del Arte

Durante las últimas décadas, el diseño y control de robots humanoides han sido algunos de los principales temas de interés dentro del campo de la robótica. Pero a pesar de esto e independientemente de los problemas relacionados con la autonomía de los robots humanoides, el problema de la locomoción bípeda representa un reto que está aún lejos de ser resuelto. Sí bien es cierto que han sido desarrollados algunos prototipos de robots humanoides demasiado sofisticados, entre los cuales los más destacados son ASIMO y Qrio desarrollados por las compañías Honda y Sony respectivamente, el desempeño de la mayoría de los prototipos desarrollados está aún lejos de igualar la eficiencia y las características de la locomoción humana.

La ventaja de los robots humanoides frente a robots móviles que utilizan otro tipo de tracción para su desplazamiento, comúnmente llantas, es que su desplazamiento no se limita a terrenos planos y continuos, lo que aumenta el número de tareas que pueden ser desarrolladas por éste tipo de robots en diferentes entornos. Más aun, dentro de los robots que utilizan piernas para su desplazamiento, éstos son los únicos que por su naturaleza antropomórfica pueden adaptarse e integrarse con mayor facilidad a un ambiente diseñado para humanos.

Existen dos enfoques diferentes para llevar a cabo la planificación de la locomoción un robot bípedo: el enfoque biológico y el enfoque técnico. Con el enfoque biológico se pretende adoptar conceptos de control encontrados en la naturaleza de la marcha humana con el propósito de simularla fielmente. Con este enfoque las herramientas utilizadas en ingeniería no son demasiado relevantes, más bien se busca en la naturaleza la respuesta al problema planteado para que la solución se adapte de forma más natural al fenómeno estudiado, seleccionando nuevos materiales y sistemas de actuación.

Por otro lado, con el enfoque técnico los métodos de planificación se pueden clasificar en dos categorías. El primer tipo de método utiliza información de los parámetros de la estructura del mecanismo para obtener un modelado cinemático y dinámico de éste, además se utiliza la velocidad y aceleración de los ángulos que se forman entre los eslabones del mecanismo, así como la medición de las fuerzas de interacción entre el pie y el piso. Pero este método implica una carga computacional excesiva, puesto que el procesamiento de la

información resulta demasiado extenso, por lo que han sido propuestos algunos modelos simplificados para representar de forma simplificada el mecanismo del robot bípedo, así como criterios de estabilidad que aseguran la estabilidad dinámica del mecanismo durante la locomoción bípeda.

Por otra parte, el segundo tipo de método utiliza modelos matemáticos y técnicas de inteligencia artificial tales como aprendizaje automático [1], control fuzzy [2], redes neuronales artificiales [3] y algoritmos genéticos [4]. Las ventajas principales del segundo tipo de método con respecto al primero son que no se requiere conocer perfectamente la estructura mecánica del robot y que se toma ventaja de las capacidades de aprender, lo que ayuda a dotar a los robots de mayor autonomía.

Los principales puntos de estudio, en la primera categoría del enfoque técnico, para asegurar la marcha dinámicamente estable son:

1. Criterios de estabilidad para la marcha bípeda.
2. Planeación de trayectorias dinámicamente estables.
3. Control dinámico de la marcha bípeda.

Los criterios de estabilidad son criterios utilizados para llevar a cabo una marcha estable manteniendo el equilibrio del mecanismo durante el ciclo de marcha. Actualmente existen diferentes criterios de estabilidad como el criterio *Foot Rotation Indicator* [5] y el criterio *Ground Projection of the Center of Mass* [6], pero el criterio de estabilidad estándar y más ampliamente utilizado en investigaciones sobre robótica bípeda es el criterio del ZMP por sus siglas en inglés *Zero Moment Point* [7], en el capítulo 2 se explicará a detalle en que consiste éste último.

Para la planeación de trayectorias dinámicamente estables existen diferentes métodos:

- **Planificación mediante generación de trayectorias en el espacio cartesiano o articular.** En el espacio cartesiano se toman en consideración la posición y orientación de los efectores finales, mientras que en el espacio articular se toman en consideración la posición, velocidad y aceleración de las articulaciones. Aunque la acción de control se lleva a cabo en el espacio articular, es más conveniente realizar la planificación en el espacio cartesiano puesto que se describe de forma más natural el movimiento del mecanismo y sus efectores finales, además de que en el espacio cartesiano se pueden ubicar con mayor facilidad determinados puntos que pueden ser considerados restricciones. Para generar las trayectorias se definen posiciones de la cadera y del tobillo, después se procede a obtener las trayectorias entre estos puntos mediante interpolación polinómica. Para cada posición de la cadera y del tobillo se obtiene, por geometría, la posición correspondiente de la rodilla.
- **Planificación mediante consecución de metas.** Consiste en generar posturas que definen estados o metas, las cuales son adoptadas por el mecanismo de manera jerárquica para definir la caminata. Para determinar los valores articulares del mecanismo entre dos estados se utiliza interpolación lineal entre las posiciones angulares referidas a las posturas de cada uno de los estados. Mientras se defina un mayor número de posturas la caminata será más continua.

- **Planificación con base en el criterio de estabilidad del ZMP.** Consiste en determinar las trayectorias de los pies de acuerdo a los parámetros de la marcha delimitados por la estructura mecánica, y con base en el criterio de estabilidad del ZMP y en un modelo dinámico simplificado, determinar la trayectoria del centro de masa. Una vez que se generan las trayectorias de los efectores finales se procede a obtener por medio de cinemática inversa los ángulos de los eslabones que determinan la configuración que debe adoptar el mecanismo conforme van evolucionando las trayectorias para generar el ciclo de marcha.

En cuanto a los modelos empleados para la planeación y control de trayectorias dinámicamente estables, han sido propuestos diferentes modelos simplificados como el Modelo de Fuerzas Virtuales propuesto por Jerry Pratt y sus compañeros de trabajo [8], el cual consiste en el control intuitivo de la marcha bípeda tomando en consideración la fuerza de reacción de la superficie de contacto y controlando el par en las articulaciones del robot.

Existen otros modelos empleados para la generación de la marcha bípeda como el Modelo de Masa Distribuida [9], en el que se considera la inercia y la masa total o parcial del mecanismo del robot. La ventaja de este modelo es que al considerar gran parte del mecanismo se obtiene una buena precisión, aunque la generación de trayectorias no puede llevarse a cabo en tiempo real puesto que los recursos computacionales consumidos son demasiado elevados. Los modelos de masa distribuida *Two Masses Inverted Pendulum Mode (TMIPM)* y *Multiple Masses Inverted Pendulum Mode (MMIPM)* propuestos por A. Albert [10], modelan la dinámica del robot mediante péndulos invertidos, caracterizando a la pierna oscilante con una o múltiples masas respectivamente y al torso del robot con una sola masa que representa la masa del mecanismo restante. Aunque los recursos computacionales empleados para resolver los sistemas de ecuaciones que modelan el comportamiento de estos modelos son menores con respecto al modelo de masa distribuida, la generación de trayectorias no puede llevarse a cabo en tiempo real.

Los modelos de masa concentrada Carro-Mesa [11] y *3D Linear Inverted Pendulum Mode* [12], [13], también conocido este último como modelo péndulo invertido, son modelos que permiten la generación de trayectorias dinámicamente estables en tiempo real y que modelan únicamente la dinámica del centro de masa (COM), el cual es considerado siempre a nivel de la cintura del robot, tal y como se muestra en la figura 1.1.

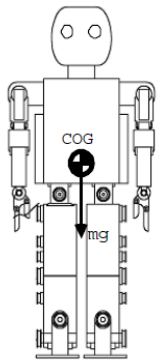

Figura 1.1: Centro de masa del mecanismo. Imagen tomada de [14]

El modelo péndulo invertido [12] consta de una masa fija a una pierna telescópica que rota en torno al ZMP [7]. El movimiento de la masa es restringido a un plano horizontal a una altura constante Zh. Por otro lado, el modelo carro-mesa modela la dinámica del robot con un carrito (COM) que se desplaza sobre una superficie representada por medio de una mesa, cuya altura no tiene que ser necesariamente constante [15]. En [14] se propone el modelo *Rolling Sphere* el cual consta de una esfera que se desplaza girando sobre una superficie a nivel constante, dicho modelo corresponde al modelo carro-mesa, sólo que la masa que se desplaza es considerada como una esfera en vez de un carrito.

1.2. Motivación

En la actualidad los robots humanoides son diseñados para simular el cuerpo humano y sus funciones por lo que su estudio es una herramienta que ayuda a comprender mejor los mecanismos de locomoción humana, lo que resulta ser de gran utilidad para el desarrollo de prótesis más sofisticadas para el miembro inferior, por ejemplo. Por otra parte, los robots humanoides están también siendo diseñados con el propósito de desarrollar tareas humanas como atención a individuos que por su situación requieren mayor atención, así como desarrollo de tareas humanas en ambientes peligrosos. Pero debido al elevado número de grados de libertad con los que cuentan las piernas de los robots bípedos para simular la caminata humana, la dificultad que implica mantener el equilibrio de un robot bípedo durante el ciclo de marcha es muy elevada, por lo que considerar la dinámica del modelo completo de un robot bípedo para generar trayectorias dinámicamente estables resulta ser una tarea exhaustiva que consume demasiados recursos computaciones y que hace imposible la generación de patrones de caminata en tiempo real.

Por otro lado, los modelos simplificados que distribuyen la masa del mecanismo de un robot bípedo en puntos estratégicos, con el propósito de obtener un modelo simplificado de éste y de facilitar la generación trayectorias dinámicamente estables, han adquirido gran relevancia debido a su desempeño mostrado.

De los dos modelos simplificados de masa concentrada que han sido propuestos, el modelo péndulo invertido [12], [13] es el que hasta el momento se ha explorado más en la Facultad de Ingeniería para el análisis del equilibrio dinámico durante la marcha bípeda: [16], [17]. Por este motivo es importante estudiar, entender y aplicar no solo el modelo péndulo invertido, sino también el modelo carro-mesa con el propósito de ampliar las opciones en investigaciones futuras que puedan llevarse a cabo en la Facultad de Ingeniería. Por otro lado, los métodos empleados para resolver el modelo simplificado de masa concentrada carro-mesa [11] son contados y en ocasiones inexactos debido a consideraciones que se tienen que hacer para su solución.

1.3. Planteamiento del Problema

En la figura 1.2 se muestra la simplificación del modelo de un robot bípedo utilizando el modelo carro-mesa, donde M representa la masa total del modelo real, Z_h la altura del centro de gravedad, g la aceleración de la gravedad, $\ddot{x}$ la aceleración del centro de masa y P_{zmp} el punto de equilibro donde los momentos angulares debidos al peso y a la aceleración del carro (centro de masa) se compensan uno con otro, es decir se anulan. Puesto que el pie

de la mesa es demasiado angosto como para permitir que ésta se mantenga en equilibrio cuando el carro se encuentra en un extremo de la mesa, se tiene que calcular la aceleración apropiada del carro sobre la mesa de forma que ésta se mantenga derecha y en equilibrio, al mismo tiempo que el punto de equilibrio P_{zmp} se encuentre dentro del área de soporte (pie de la mesa).

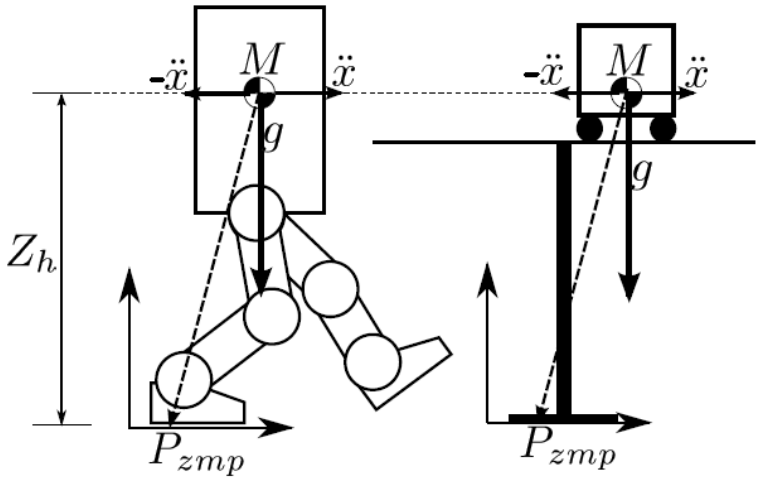

Figura 1.2: Modelo simplificado carro-mesa. Imagen tomada de [18]

1.4. Objetivo General

El objetivo general de esta tesis es mantener el equilibrio dinámico de un robot bípedo durante la locomoción, generando trayectorias dinámicamente estables a partir del criterio de estabilidad del punto de momento cero (ZMP) y del análisis de la representación simplificada de un robot bípedo mediante el modelo carro-mesa. Puesto que se pretende que en un futuro se pueda llevar a cabo la marcha dinámicamente estable en tiempo real, en este trabajo también se pretende que el cómputo empleado para la generación de las trayectorias dinámicamente estables sea el mínimo posible.

1.5. Objetivos Particulares

Los objetivos particulares de este trabajo son los siguientes:

- Implementar al menos dos de las soluciones que han sido propuestas para el modelo carro-mesa.
- Comparar y determinar la solución más eficiente para el modelo carro-mesa.
- Verificar experimentalmente los resultados obtenidos en simulación.
- Desarrollar una interfaz de usuario, para controlar al banco de pruebas, que sea amigable con el usuario y que pueda comunicarse inalámbricamente con el banco de pruebas.

1.6. Trabajo Previo

Esta tesis forma parte de un proyecto llevado a cabo en la Facultad de Ingeniería conformado por investigadores, alumnos de posgrado y de licenciatura que han hecho aportaciones diferentes para que éste tenga los alcances que hasta el momento se han logrado.

Previo a este trabajo se desarrollaron los modelos de cinemática y dinámica espacial para el robot bípedo de 12 GDL internos *Scout*® [19]. Para deducir la cinemática inversa se consideró cada una de las piernas del robot como manipuladores. También, se desarrolló un programa en *Mathematica*® para el cálculo de la cinemática inversa y la simulación del ciclo de marcha del robot, utilizando un diagrama simplificado unifilar.

En [16] se llevó a cabo la planificación del ciclo de marcha y se optimizó mediante un algoritmo genético que, con base en el criterio de estabilidad del ZMP y en el modelo péndulo invertido, encuentra la trayectoria óptima de la cadera, con lo que se aumenta la posibilidad de encontrar una trayectoria para la cadera que asegure la estabilidad dinámica. También se realizó una interfaz gráfica de usuario que permite implementar ciclos de marcha tanto en el prototipo físico como en un prototipo virtual.

En [20] se instrumentó el banco de pruebas mediante sensores de posición, presión e inercia. Los sensores de posición sirvieron para obtener información del ángulo generado en cada una de las juntas del banco de pruebas y analizar el comportamiento real de éste. Por otra parte, los sensores de presión ayudaron a calcular el ZMP de forma experimental, mientras que el sensor inercial se utilizó para calcular los grados de inclinación del banco de pruebas, únicamente en la posición inicial, modificando la inclinación de la base del banco de pruebas.

1.7. Metodología

La metodología para alcanzar los objetivos se presenta a continuación:

1. Estudiar y analizar el modelo simplificado carro-mesa y su relación con el criterio de estabilidad del ZMP (*Zero Moment Point*).
2. Estudiar a detalle los diferentes métodos para la solución del modelo carro-mesa y seleccionar los más exactos.
3. Simular en *Mathematica* los métodos seleccionados.
4. Realizar las comparaciones correspondientes a través del costo computacional empleado en cada método.
5. Incluir las simulaciones realizadas en el programa de *Mathematica*® que resuelve la cinemática inversa [19].
6. Poner en marcha el banco de pruebas e implementar una comunicación inalámbrica vía bluetooth.

1.8. Estructura de la Tesis

Este trabajo se encuentra estructurado de la siguiente manera:

En el capítulo 2 se presenta una introducción al tema de la locomoción humana, se define la marcha y se describen sus periodos, así como el sistema de referencia que se utiliza para la descripción del ciclo de marcha. También se incluye la explicación de algunos temas y definiciones empleadas en este trabajo.

El capítulo 3 describe la arquitectura del banco de pruebas con el propósito de que el lector comprenda la nomenclatura que se empleará en las simulaciones, así como el porqué de los parámetros que se seleccionarán para la planificación de la marcha bípeda. También se presentan los problemas que se tuvieron para poner en marcha el banco de pruebas y como se resolvieron.

En el capítulo 4 se realiza la planificación de la marcha bípeda con base en el criterio de estabilidad del ZMP y en los parámetros delimitados por la estructura del mecanismo. Se emplean splines de segundo grado para generar las trayectorias de los pies y se implementan las soluciones, del modelo carro-mesa, seleccionadas para generar las trayectorias de la cadera. Por último se muestran los resultados obtenidos en simulación y se realizan las comparaciones correspondientes.

En el capítulo 5 se presenta la interfaz de usuario desarrollada, las simulaciones y los experimentos realizados con el prototipo físico y la comparación entre los resultados obtenidos con el modelo simplificado carro-mesa y los obtenidos considerando la dinámica total del robot. Finalmente, en el capítulo 6, se presentan las conclusiones y se propone el trabajo a futuro.

Capítulo 2

Preliminares

2.1. Locomoción Humana

2.1.1. Definición

La locomoción humana es un modo de marcha bípeda y una de las actividades más importantes realizadas por los seres humanos. La marcha bípeda es llevada a cabo mediante la alternación de los miembros inferiores para producir un desplazamiento. De acuerdo con [21] la marcha bípeda se describe formalmente como:

> Serie de movimientos alternantes y rítmicos de las extremidades y del tronco, que determinan un desplazamiento hacia delante del centro de gravedad con un mínimo gasto de energía.

Por otra parte, la marcha bípeda también puede ser descrita como:

> Sucesión de impulsos y frenados, en los que el impulso se sitúa a nivel del miembro inferior posterior y el frenado en el anterior [22].

Para proporcionar una mejor descripción de la locomoción humana es necesario dividirla en periodos y emplear un sistema de referencia para la inspección visual de cada región anatómica.

2.1.2. Sistema de referencia

Para describir la marcha humana es necesario emplear un sistema de referencia espacial que describa el desplazamiento del cuerpo humano con respecto al piso, por lo que comúnmente se emplea el sistema de referencia mostrado en la figura 2.1. Este sistema de referencia consta de tres planos perpendiculares entre sí, dos verticales y uno horizontal, los cuales dividen al cuerpo humano. El plano transversal, conformado por los ejes XY, se encuentra ubicado a nivel de la cintura y divide al cuerpo humano en la parte superior e inferior. El plano sagital divide al cuerpo humano en la parte lateral izquierda y derecha, ejes YZ. Mientras que el plano coronal, o frontal, está conformado por los ejes XZ y divide al cuerpo humano en la parte posterior y anterior.

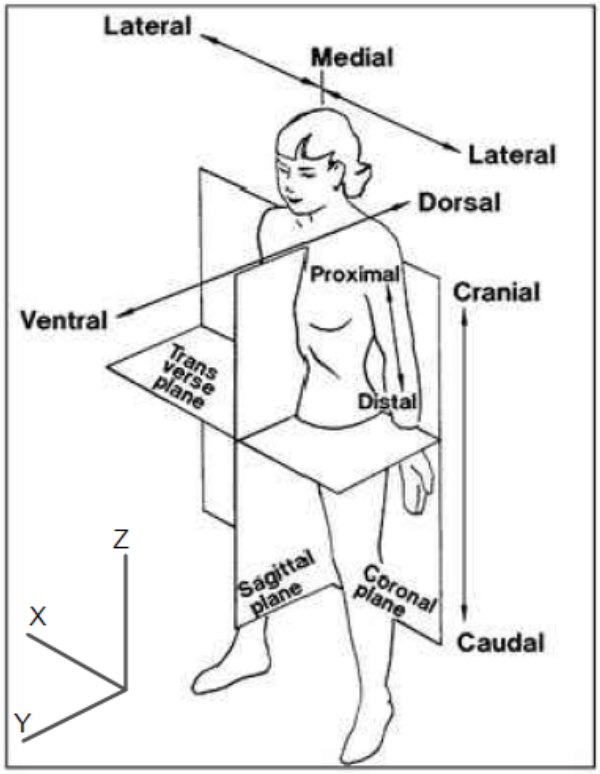

Figura 2.1: Sistema de referencia para describir la marcha bípeda [23].

2.1.3. El Ciclo de Marcha

El ciclo de marcha es el conjunto de acciones sucesivas, alternas y uniformes que comienza cuando un talón hace contacto con el piso y termina cuando el mismo talón hace contacto nuevamente con el piso [22]. En en la figura 2.2 se muestra la secuencia del ciclo de marcha en el plano sagital.

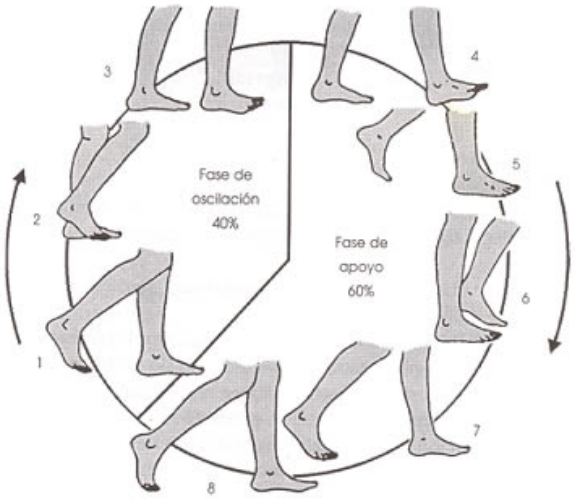

Figura 2.2: El ciclo de la marcha bípeda [24].

Periodos

Según [24] y [25] el Instituto de Biomecánica de Valencia ha identificado durante el ciclo de marcha, a una velocidad normal de 100 a 115 pasos por minuto, dos periodos:

- **Periodo de soporte:** Se refiere al periodo en el que el pie se encuentra en contacto con el piso, comienza con el contacto inicial de talón sobre el suelo y termina con el despegue del antepié. Este periodo representa el 60 % del ciclo de marcha.
- **Periodo de balanceo:** Comienza con el despegue del antepié y termina con el contacto del talón sobre el piso. El pie pierde contacto con el piso para mantenerse en el aire mientras avanza hacia adelante. Este periodo representa el 40 % del ciclo de marcha.

En en la figura 2.3 se muestran los peridos del cliclo de marcha.

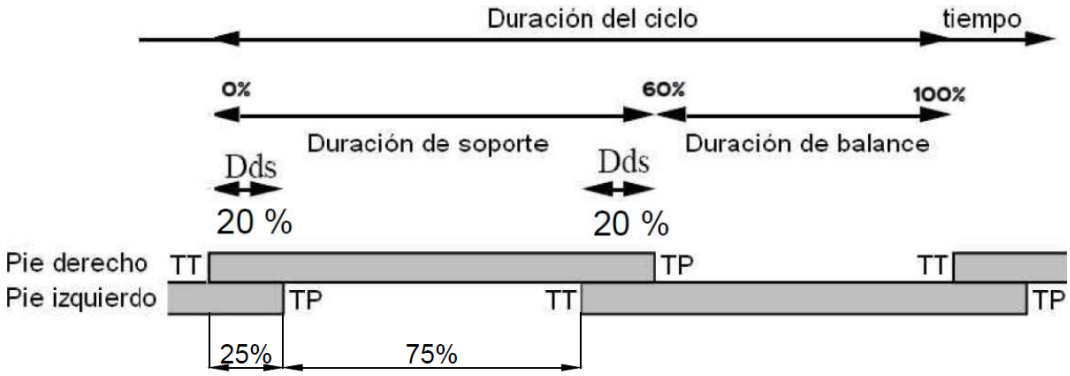

Figura 2.3: Periodos del ciclo de la marcha [21]. Dds.- Duración de doble soporte, TT.- Toque de talón, TP.-Toque de punta.

De acuerdo con la figura 2.3, el periodo de soporte se divide en:

- **Periodo de Soporte Simple(PSS):** El Periodo de Soporte Simple de un pie ocurre durante el periodo de balanceo del otro pie, considerando que los periodos de cada pie se producen simultáneamente.
- **Periodo de Soporte Doble(PSD):** Se produce cuando ambos pies están en contacto con el suelo y ocurre en dos ocasiones durante el ciclo de marcha: al inicio y al final del periodo de soporte. Este periodo corresponde a una tercera parte del Periodo de Soporte Simple.

2.2. Estabilidad

La estabilidad es una característica de un cuerpo en equilibrio que sirve para determinar si éste puede mantenerse en equilibrio, estático o dinámico, ante perturbaciones. Si el sistema puede mantenerse en equilibrio ante perturbaciones se dice que el sistema es estable, por el contrario, si el sistema no puede mantenerse en equilibrio se dice que el sistema es inestable. Cabe destacar que equilibrio y estabilidad son dos conceptos totalmente distintos puesto que el equilibrio depende de la estabilidad. Para poder profundizar en estos conceptos, equilibrio y estabilidad, es necesario revisar algunos términos a continuación.

2.2.1. Polígono de Soporte

Como se estudiará más adelante, el polígono de soporte sirve como base de sustento para mantener el equilibrio estático y dinámico ya que cuando un individuo se encuentra erguido la proyección del centro de gravedad sobre el suelo (línea de gravedad) cae dentro de la base de sustento y cuanto mayor sea ésta, mayor será la fuerza necesaria para mover la línea de gravedad fuera de dicha base [26].

De acuerdo con [27] el polígono de soporte es la superficie poligonal delimitada por los márgenes externos del apoyo, de uno o ambos pies, en contacto con el suelo. Cuando el robot se encuentra en el periodo de balanceo o de soporte simple, el polígono de soporte se define por la silueta del pie en contacto con el suelo. Cuando el robot se encuentra en el periodo de soporte doble (ambos pies en el suelo), el polígono de soporte se define por la figura formada por los márgenes de ambos pies. En la figura 2.4, el área delimitada por la línea azul representa al polígono de soporte doble.

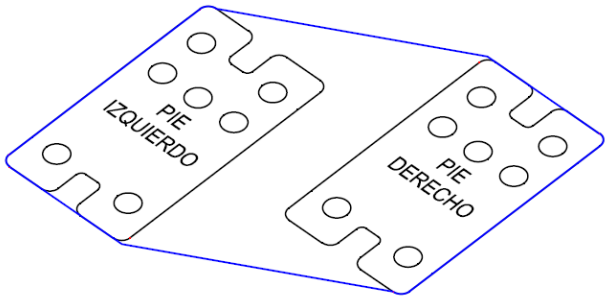

Figura 2.4: Polígono de soporte doble del robot bípedo *Scout*.

2.2.2. Equilibrio

Existen diferentes tipos de equilibrio: mecánico, termodinámico, químico, económico, emocional, etc. En el ámbito del movimiento del cuerpo humano, el equilibrio se refiere a aspectos relacionados con el mantenimiento de la postura. Tomando en cuenta estos aspectos, en [28] se define de forma general al equilibrio como:

> La capacidad de asumir y sostener cualquier posición del cuerpo contra la ley de gravedad.

Por otra parte, desde el punto de vista de la Biomecánica, en [29] se define al equilibrio como:

> Un término genérico que describe la dinámica de la postura corporal para prevenir las caídas, relacionado con las fuerzas que actúan sobre el cuerpo y las características inerciales de los segmentos corporales.

De estas definiciones se puede concluir que el equilibrio o balanceo corporal es la capacidad de adoptar y mantener determinadas posturas, incluso en presencia de perturbaciones como la gravedad, con el propósito de llevar a cabo una determinada tarea sin que ésta se vea afectada por efectos de perturbaciones.

Los factores que intervienen en el equilibrio, sin considerar perturbaciones, son: la fuerza de gravedad, fuerzas inerciales, el centro de gravedad y el polígono de soporte. De esta forma se distinguen dos tipos de equilibrio: equilibrio estático y equilibrio dinámico.

Equilibrio estático

El equilibrio estático se refiere al control de la postura sin desplazamiento, es decir la capacidad de mantener una posición estática frente a la acción de la fuerza de gravedad. En este caso, para evitar el desequilibrio es necesario que el Centro de Gravedad, punto en el que actúa la fuerza resultante de todas las fuerzas de gravedad que actúan sobre el cuerpo (COG), mantenga su proyección sobre el suelo dentro del polígono de soporte. Si en algún momento la proyección del COG sobre el suelo llegase a salir del polígono de soporte (PS), el cuerpo entraría en desequilibrio puesto que se generarían momentos angulares debidos a la energía potencial producida por la altura del centro de gravedad con respecto al suelo [26]. Por lo tanto, mientras más centrado se encuentre la proyección del COG dentro del PS, más estable será el cuerpo puesto que la distancia a cualquier punto de desequilibrio será mayor.

Equilibrio dinámico

Por otra parte el equilibrio dinámico se refiere a la adopción de la postura adecuada, con forme se desplaza el cuerpo, para contrarrestar no sólo la acción de la gravedad como en el caso estático, sino también la de fuerzas inerciales.

Aunque los fundamentos del equilibrio dinámico son los mismos que los del estático, las condiciones de equilibrio cambian. Por ejemplo, la acción de correr puede ser considera, desde el punto de vista de equilibrio estático, como una actividad en continuo desequilibrio puesto que el COG se sitúa sobre el polígono de soporte durante periodos muy cortos de tiempo, es decir que la mayor parte del tiempo el COG se sitúa fuera del PS. Otro ejemplo es la acción de caminar, ya que cuando comienza el desplazamiento la proyección del COG sobre el suelo sale del PS y el cuerpo entra en desequilibrio, por lo que la pierna oscilante debe desplazarse de tal forma que se defina un nuevo PS y se recuperé el equilibrio. Por lo tanto, se dice que la acción de caminar y correr son una sucesión de caídas controladas [27].

Cuando el cuerpo adopta movimiento aparecen aceleraciones, cambios de sentido, de dirección, etc. Por lo que ahora no sólo influye la fuerza de gravedad en el equilibrio, sino también las fuerzas inerciales. Por lo tanto, ahora la fuerza resultante que se debe mantener dentro del polígono soporte es la suma de la resultante de la fuerza de gravedad con la resultante de las fuerzas inerciales.

2.2.3. Zero Moment Point (ZMP)

En las figuras 2.5(a) y 2.5(b) la letra A representa al punto donde el efecto de todas las fuerzas interactuando con el mecanismo puede ser reemplazado por el de una sola fuerza, la cual sirve como un indicador del comportamiento del mecanismo completo. La parte del mecanismo que se encuentra arriba del tobillo se omite, puesto que la fuerza F_A representa la fuerza cuyo efecto reemplaza el efecto de todas las fuerzas interactuando sobre el mecanismo y M_A el momento angular producido por ésta, [7]. El peso del pie se ubica en su centro

gravedad, punto G, y la fuerza de reacción R, del piso sobre la planta del pie, en el punto P.

La reacción total del piso sobre la planta del pie consta de tres componentes de la fuerza de reacción R(R_x, R_y, R_z) y de tres componentes del momento M(M_x, M_y, M_z), donde las componentes de la fuerza R que actúan en el plano horizontal representan la fuerza de fricción, que actúa en el punto de contacto del pie con el piso y que compensa la componentes horizontales de la fuerza F_A. A su vez, el momento de reacción vertical M_z, producido por las fuerzas de reacción, compensa la componente vertical M_{Az} del momento producido por la fuerza F_A, figura 2.5(c), mientras que la componente R_z, que representa la componente vertical de la reacción del piso, compensa las fuerzas verticales debidas a F_A y al peso. Ahora, falta considerar el equilibrio de las componentes horizontales del momento debido a la carga del pie, pero ya que la dirección de la fuerza de reacción inducida por la acción del pie es siempre hacia arriba, las componentes horizontales de todos los momentos activos pueden ser compensados únicamente cambiando de posición la fuerza de reacción dentro del polígono de soporte delimitado por la silueta del pie. Por lo tanto, para balancear la carga adicional, las componentes horizontales del momento M_A provocarán que se cambie a la posición correcta la fuerza de reacción. Lo anterior se ilustra en la figura 2.5(d) para el plano sagital, donde el momento M_{Ax} se balancea recorriendo el punto en donde actúa la fuerza de reacción R_z, una distancia y.

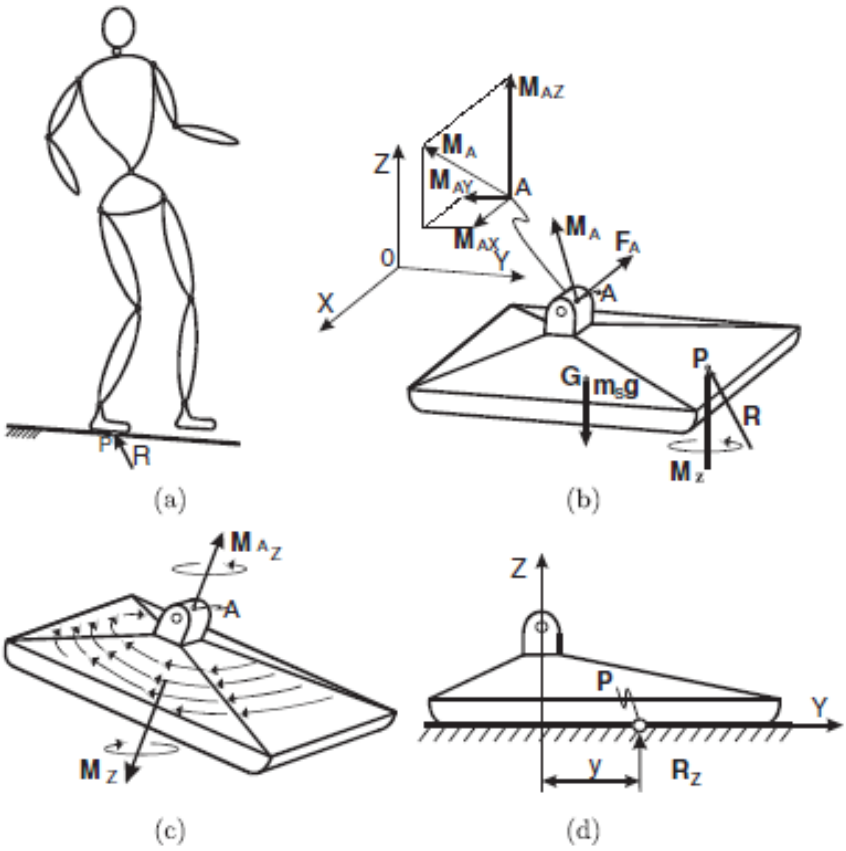

Figura 2.5: Representación de las fuerzas actuando sobre la suela del mecanismo [7].

En resumen, el aumento del momento angular producido en el tobillo del pie se equilibrará cambiando de posición la fuerza de reacción dentro del polígono de soporte, con el objetivo de que no existan componentes horizontales y que el mecanismo no se vuelque. En caso de que el polígono de soporte no sea lo suficientemente amplio como para albergar la posición adecuada de la fuerza de reacción, la fuerza de reacción actuará en el borde del pie y la parte de la componente horizontal del momento que no pueda ser compensada causará que el mecanismo rote con respecto al borde del pie, lo que podría resultar en el desequilibrio del mecanismo. Por lo tanto, la condición necesaria para que el mecanismo esté en equilibrio dinámico, en el punto P dentro del polígono de soporte donde actúa la fuerza de reacción, es que:

$$Mx = 0$$

$$My = 0$$

Al final se reduce el efecto de la fuerza de reacción del piso, debida al reposo del pie del robot sobre éste, a una fuerza R y a un momento de componente vertical M_z, el punto P en el que la fuerza de reacción actúa es conocido como Zero Moment Point (ZMP) [7].

Considerando el ZMP como el centro de presión de la fuerza de reacción del piso sobre la planta del pie se puede realizar un análisis más amplio [30]. Primero se considera la reacción del piso a detalle como se muestra en la figura 2.6. Tomando en cuenta que la

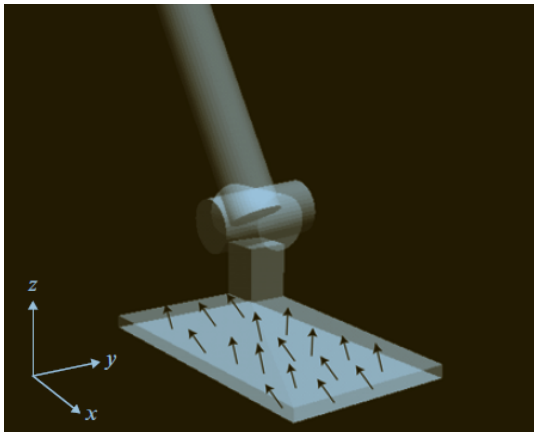

Figura 2.6: Reacción del piso en tres dimensiones [30].

fuerza de reacción actúa en un determinado número de puntos de contacto, del piso sobre la planta del pie, $p_i (i = 1, \ldots, N)$ y que cada vector de fuerza tiene la siguiente forma:

$$f_i = [f_{ix}, f_{iy}, f_{iz}]^T,$$

donde f_{ix}, f_{iy} y f_{iz} son las componentes de la fuerza de reacción del piso en las direcciones x, y y z respectivamente. El ZMP se puede calcular como:

$$p = \frac{\sum_{i=1}^{n} p_i f_{iz}}{\sum_{i=1}^{n} f_{iz}}, \tag{2.1}$$

lo que se puede escribir como:

$$p = \sum_{i=1}^{n} \alpha_i p_i, \tag{2.2}$$

donde,

$$\alpha_i = \frac{f_{iz}}{f_z},$$
$$f_z = \sum_{i=1}^{n} f_{iz}.$$

Puesto que $f_{iz} \geq 0$ para $i = 1, \ldots, N$ se determina que:

$$\begin{cases} \alpha_i \geq 0 \quad para \quad i = 1, \ldots, N, \\ \\ \sum_{i=1}^{n} \alpha_i = 1. \end{cases} \tag{2.3}$$

Si se considera que la reacción de suelo sobre la planta del pie es unilateral, los puntos que satisfacen las ecuaciones (2.2) y (2.3) definen el polígono de soporte. Es decir, se asegura que el ZMP nunca salga del polígono de soporte debido a la restricción unilateral de la fuerza de reacción.

El momento angular resultante en el ZMP puede ser calculado como:

$$\tau = \sum_{i=1}^{n} (p_i - p) \times f_i, \tag{2.4}$$

el cual puede ser definido en términos de vectores componentes como:

$$\tau_x = \sum_{i=1}^{n} (p_{iy} - p_y) f_{iz} - \sum_{i=1}^{n} (p_{iz} - p_z) f_{iy},$$
$$\tau_y = \sum_{i=1}^{n} (p_{iz} - p_z) f_{ix} - \sum_{i=1}^{n} (p_{ix} - p_x) f_{iz},$$
$$\tau_z = \sum_{i=1}^{n} (p_{ix} - p_x) f_{iy} - \sum_{i=1}^{n} (p_{iy} - p_y) f_{ix},$$

donde p_{ix}, p_{iy} y p_{iz} son las componentes del vector de posición p_i y p_x, p_y y p_z las componentes del ZMP.

De acuerdo con la definición de ZMP, ecuación (2.1), y considerando que cuando el piso es horizontal $p_{iz} = p_z$, se obtiene que:

$$\begin{cases} \tau_x &= 0 \\ \\ \tau_y &= 0 \end{cases}$$

Por otra parte se debe considerar que la fuerza de fricción crea un momento vertical diferente de cero, por lo que:

$$\tau_z \neq 0$$

2.2.4. Fictitious Zero Moment Point (FZMP)

El ZMP también sirve como indicador de la estabilidad: si el ZMP se encuentra dentro del polígono de soporte y centrado, el mecanismo estará equilibrado y será estable. Si el ZMP se encuentra en el contorno o fuera del polígono de soporte, el robot perderá el balance y será muy difícil recobrar el equilibrio (inestabilidad). Por ejemplo, en la figura 2.7(a) se ilustra el segundo caso quedando el ZMP calculado fuera del polígono de soporte. En este caso para mantener el equilibrio ($M_x = 0$ y $M_y = 0$) sería necesario ampliar el polígono de soporte a través de un pie más grande, lo que es prácticamente imposible, por lo que el punto calculado es llamado FZMP (*FictitiousZeroMomentPoint*). Si el ZMP calculado está fuera del polígono de soporte (PS), en el caso del FZMP, significa que la fuerza de reacción del suelo se ubica en el borde del PS en un punto P y que el mecanismo comenzará a rotar con respecto a éste, con una magnitud proporcional a la distancia entre el punto P y el FZMP, debido a los momentos que no fueron compensados (figura 2.7(b)).

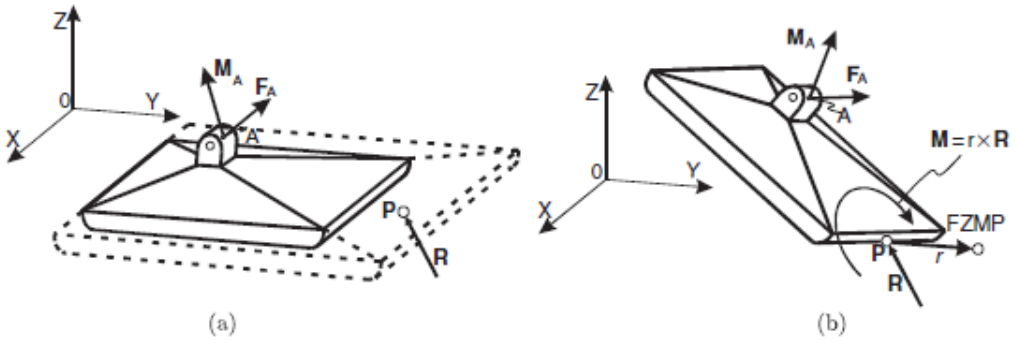

Figura 2.7: *FictitiousZeroMomentPoint* (FZMP) [7].

2.3. Control Óptimo

En el capítulo 4 se describe una solución para el modelo carro-mesa que emplea control óptimo e información futura de la trayectoria de referencia, por lo que es importante describir algunos principios de control óptimo.

El control óptimo es una técnica utilizada en teoría de control para resolver problemas de optimización en sistemas dinámicos (sistemas físicos cuyo estado evoluciona en el tiempo). Se entiende por optimización realizar un trabajo de la mejor forma posible. Un problema de control óptimo se puede identificar al momento de poner en órbita un satélite, para lo cual se deben emplear los controladores adecuados (ángulo de lanzamiento y velocidad de emisión de combustible en el escape) que hagan que el satélite llegue a su destino empleando el mínimo combustible en el menor tiempo posible.

La estrategia de control óptimo dependerá de que se defina como “la mejor forma posible”, por ejemplo, un sistema simple e impreciso pero no costoso y fácil de implementar, con un desempeño adecuado, puede ser considerado óptimo. En contraste, un sistema muy preciso podría ser considerado no óptimo por ser demasiado costoso o porque su implementación no es sencilla.

Para plantear un problema de control óptimo se requiere:

- Definir el modelo matemático del sistema controlado.
- Definir la tarea a realizar.
- Especificar el criterio de optimización y definir la función de costo.

Considerando un sistema dinámico, en forma de variables de estado y en tiempo discreto, definido por:

$$x(k+1) = Ax(k) + Bu(k), \tag{2.5}$$

$$p(x) = Cx(k), \tag{2.6}$$

el problema de control que se desea resolver es operar o manejar el sistema dinámico descrito por las ecuaciones (2.5) y (2.6) con el mínimo costo, por lo que los parámetros del controlador que gobierna un proceso o una máquina deben ser encontrados mediante un algoritmo matemático que minimice la función de costo J.

2.3.1. Función de Costo J

La función de costo J es una función empleada para medir la calidad del desempeño del sistema y es definida como la suma de la desviación del valor deseado de las mediciones de los parámetros que se desean controlar. Por lo que con el algoritmo matemático que minimiza la función de costo se encuentran los parámetros del controlador que minimiza las desviaciones no deseadas.

Un criterio de desempeño común es el del tiempo mínimo, en el cual se desea que el controlador $u(t)$ produzca la trayectoria más rápida para obtener un estado final deseado. Para este caso, la función de costo que se desea minimizar se puede definir como:

$$J = T.$$

Otro criterio de desempeño común es el error entre el valor deseado y valor obtenido en un estado final (error en estado estacionario) en un determinado tiempo T. En este caso la función de costo que se desea minimizar se puede definir como:

$$J = ||x(T)||^2.$$

Un criterio de desempeño aun más, podría ser minimizar el área bajo la curva producida por la trayectoria de la transición de un estado inicial a un estado final, de esta forma se seleccionarían los controladores que producen transiciones eficientes. Para este caso.

$$J = \int_0^T ||x(t)||^2.$$

El criterio cuadrático es un criterio que incluye la acción de control para mantener limitada la energía empleada por la misma acción, al mismo tiempo que se controla el error. En este caso la función de costo empleada se define como:

$$J = \sum_{k=1}^{\infty} \{x_k^T Q x_k + u_k^T R u_k\}. \tag{2.7}$$

En este caso el problema de control a resolver es: encontrar la secuencia óptima de controlador u_k que minimice la función de costo, también llamada índice de desempeño, J. Cabe destacar que Q y R son matrices simétricas positivas de peso, que determinan la importancia relativa entre el error (primer término de la función) y el coste de la energía empleada por el controlador (segundo término de la función). Si la prioridad es minimizar el error se tendrá que asignar un peso mayor a la matriz Q con respecto al de la matriz R y viceversa [31].

De acuerdo con [32] la secuencia óptima de control que minimiza el índice de desempeño J es definida como:

$$u_k = -Kx_k, \tag{2.8}$$

donde:

$$K = (R + B^T PB)^{-1} B^T PA \tag{2.9}$$

y P la única solución definida positiva de la ecuación de Riccati:

$$P = Q + A^T (P - PB(R + B^T PB)^{-1} B^T P)A$$

El conjunto de ecuaciones (2.8) y (2.9) representa el controlador lineal cuadrático para el sistema discreto representado por las ecuaciones (2.5) y (2.6). En $MATLAB$ K y P pueden ser calculadas empleando la siguiente función:

$$[K, P] = lqr(A, B, Q, R).$$

Por otro lado, en $Mathematica$ K y P se pueden calcualar a partir de la función:

$$DiscreteRiccatiSolve[\{A, B\}, \{Q, R\}].$$

En ambos casos A y B son las matrices correspondientes a la ecuación de estados, ecuación (2.5), Q y R las matrices de peso que determinan la importancia relativa entre el error y el coste de la energía empleada por el controlador.

Capítulo 3

Descripción del robot bípedo Scout

3.1. Características

El Robot Lynx Scout (figura 3.1) es un robot bípedo de 12 grados de libertad (6 grados de libertad en cada una de sus piernas) que puede caminar hacia adelante o hacia atrás, girar hacia la izquierda o derecha con velocidad variable. Su estructura está formada de aluminio, tiene una altura de 23$[cm]$, un peso de 0,9$[Kg]$ y cuenta con un servomotor programable (rango de operación, velocidad, precisión y sentido de giro) por cada grado de libertad.

Figura 3.1: Robot Bípedo Scout de Lynx

3.2. Arquitectura

El Robot bípedo Scout consta de un eslabón central (considerado el torso) que une sus dos piernas conformadas por 6 eslabones (en total 13 eslabones considerando el torso y las dos piernas) que se unen mediante juntas rotacionales actuadas por servomotores.

De acuerdo con el modelo simplificado de la figura 3.2, utilizado para obtener el modelo cinemático y dinámico del robot en [19], la nomenclatura empleada es la siguiente: Partiendo del eslabón del torso, el cual se identifica con la letra B, a cada eslabón de las piernas se le identifica con la etiqueta ni, donde $1 \leq n \leq 6$ sirve para identificar los eslabones de arriba (comienza con 1) hacia abajo (termina con 6). Por otra parte i sirve para identificar la pierna, se le asigna 1 para la pierna izquierda y 2 para la pierna derecha. Las juntas rotacionales se identifican por medio de la etiqueta θ_{ni} , donde ni corresponde al segundo eslabón de la junta rotacional n, comenzando de arriba hacia abajo.

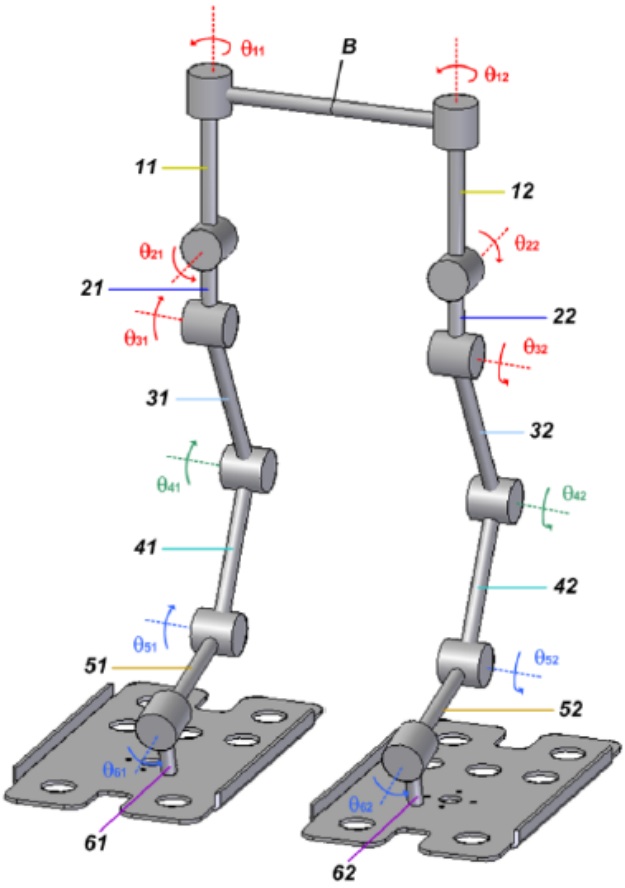

Figura 3.2: Modelo simplificado del robot bípedo Scout [19]

3.3. Interfaz gráfica

La interfaz gráfica de usuario para la manipulación del robot mediante la computadora, figura 3.3, fue programada en *LabVIEW* como parte del trabajo realizado en [16] y publicado en [33]. El propósito de la interfaz gráfica es facilitar la manipulación del robot y prevenir, una vez que se obtienen los ángulos por medio de la cinemática inversa, que ningún eslabón de las 12 juntas choquen ya que la cinemática inversa es obtenida por medio del diagrama unifilar mostrado en la figura 3.2. Por este motivo, el contar con una interfaz gráfica resulta demasiado importante ya que ayuda a simular el comportamiento ideal del robot considerando las dimensiones físicas y reales de los eslabones del robot. La interfaz gráfica también brinda la posibilidad de manipular manualmente los 12 ángulos correspondientes a las 12 grados de libertad del robot Scout, con lo que se puede identificar el sistema de referencia con base en el cual trabaja la simulación ya que la simulación no funciona con el mismo sistema de referencia del modelo físico puesto que el sistema de referencia del modelo físico depende de la caracterización de los servomotores.

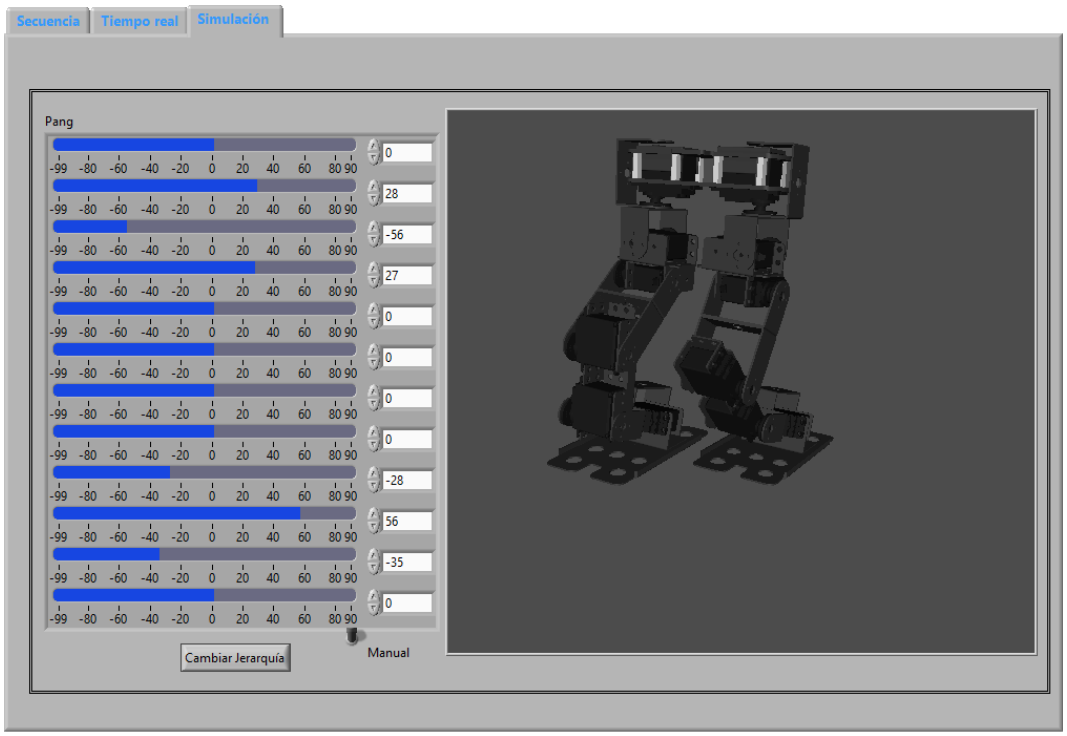

Figura 3.3: Simulación del ciclo de marcha mediante la interfaz gráfica en *LabVIEW* [16, 33].

3.4. Puesta en marcha del robot bípedo Scout

Puesto que el banco de pruebas no había sido utilizado durante un periodo de tiempo antes de comenzar este proyecto, fue necesario ponerlo en marcha. La principal dificultad radica en que el programador de los servomotores del banco de pruebas y los mismos servomotores ($HITECHS - 5475HB$) se encuentran descontinuados, por lo que no se

puede programar el sentido de giro, la velocidad, el rango de operación, la precisión y las posiciones inicial, central y final. Por otra parte, con el programador también se pueden conocer los valores de los parámetros que son programables, lo cual es una gran ventaja, pero puesto que no se contaba con un programador funcional esto representó un gran problema ya que al momento de controlar los servomotores ninguna posición coincidía con la deseada y tampoco avanzaban los grados correspondientes entre dos ángulos deseados, principalmente porque no se conocían el rango y la frecuencia de trabajo a la que debían ser programados.

Por lo tanto se caracterizó por separado cada servomotor para poder determinar el sentido de giro, el rango de trabajo y establecer la posición inicial de cada uno de acuerdo al programa en *Matemathica*® que resuelve la cinemática inversa [19]. Para lograr este propósito se realizó un programa en LabVIEW® (figura 3.4) por medio del cual se controló en tiempo real la posición deseada de los servomotores empleando una comunicación serial con la rajeta de desarrollo Arduino Mega.

Figura 3.4: Interfaz gráfica para controlar en tiempo real un servomotor.

Para la caracterización también se utilizó la librería del software de Arduino que sirve para controlar servomotores (*SoftwareServo.h*), en específico se emplearon las funciones setMinimumPulse y setMaximumPulse que permiten establecer la duración de los pulsos mínimo y máximo correspondientemente, es decir el rango de trabajo del servomotor a controlar. De acuerdo con la documentación de esta librería [34] el pulso mínimo establecido por la función setMinimumPulse corresponde a la posición de 0° y el pulso máximo establecido por la función setMaximumPulse corresponde a la posición de 180°, existiendo una relación lineal entre la posición y la duración del pulso (figura 3.5).

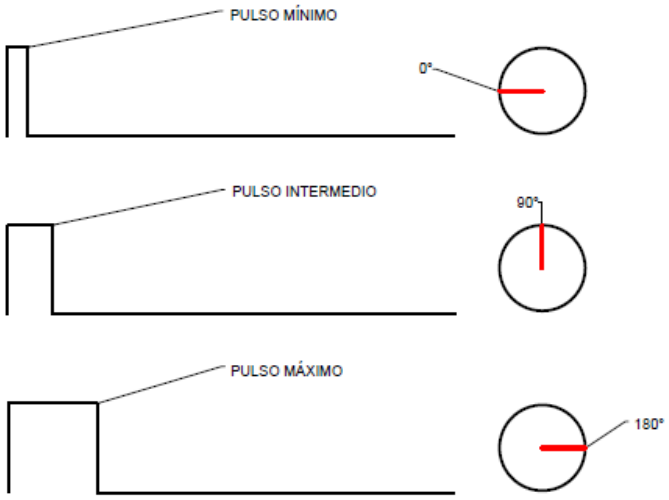

Figura 3.5: Relación lineal entre posición y pulso.

El procedimiento para caracterizar cada servomotor fue el siguiente:

1. Se estableció un pulso de 900 [ms] para 0° y uno de 2100 [ms] para 180° (rango de trabajo establecido por el fabricante).

2. Por medio del programa en $LabVIEW$® se manipuló la posición del servomotor para encontrar la posición de 0° (podría haber sido también 90° ó 180°). Se registró la posición real del servomotor en la que se alineó con 0° o con la posición elegida. A esta posición se le nombró $ang1$.

3. A continuación se ajustó la flecha de un transportador especial para servomotores $HITEC$ (figura 3.6) para que coincidiera con la posición del servomotor definida en el paso anterior y se colocó sobre su eje.

Figura 3.6: Transportador especial para servomotores $HITEC$.

4. Se manipuló nuevamente la posición del eje del servomotor para que la flecha del transportador recorriera 90°. Cuando la flecha del transportador recorrió los 90° se registró nuevamente la posición real del servomotor. A esta posición se le nombró $ang2$.

5. Para determinar la posición real del servomotor en 180° se sumó a $ang2$ la misma cantidad de grados en los que el eje del servomotor recorrió 90° sobre el transportador, considerando que existe una relación lineal. Cabe mencionar que los rangos de algunos servomotores no llegan a 0° y 180°, en este caso se seleccionó como posición inicial 90° y se midió el rango en el que la flecha del transportador recorrió 10° para así poder determinar las posiciones reales del servomotor en las que llegaría a 0° y 180°, aunque realmente esto no fuera posible.

6. Se determinó la siguiente fórmula para poder definir los pulsos que corresponden a las posiciones reales de cada servomotor en 0° y 180°:

$$PULSO = PULSO_{minimo} + angx\left(\frac{PULSO_{maximo} - PULSO_{minimo}}{180}\right) \tag{3.1}$$

considerando la siguiente relación:

$$\begin{aligned} 0° &\rightarrow 900[ms] \\ 90° &\rightarrow 1500[ms] \\ 180° &\rightarrow 2100[ms] \end{aligned}$$

y con base en la ecuación (3.1) se obtiene:

$$PULSO = 900 + angx\left(\frac{2100 - 900}{180}\right) \tag{3.2}$$

donde el valor de $angx$ corresponde al ángulo del que se quiere determinar el pulso correspondiente. En este caso se desean los pulsos correspondientes de los ángulos medidos en los puntos 2 y 4.

7. Una vez que se determinaron los pulsos mencionados, se emplearon para definir el pulso mínimo y máximo entre los que debería trabajar cada servomotor, obedeciendo la relación lineal entre pulso y posición mencionada anteriormente, con lo que se logró caracterizar cada servomotor, saber el sentido de giro y establecer la posición inicial de cada uno para que el robot bípedo adoptara la posición inicial con base en la cual se calculan los ángulos en el programa que resuelve la cinemática inversa [19].

Capítulo 4

Planificación de la Marcha Bípeda

4.1. Planificación con base en el criterio de estabilidad del ZMP

Puesto que la planeación de la marcha bípeda mediante generación de trayectorias en el espacio cartesiano, así como la planeación mediante consecución de metas, no consideran ningún criterio de estabilidad para llevar a cabo la marcha dinámicamente estable, sección 1.1, en este trabajo se decidió llevar a cabo la planificación de la marcha bípeda con base en el criterio de estabilidad del ZMP. Considerando el criterio del ZMP, la marcha bípeda puede llevarse a cabo de dos formas diferentes: estáticamente y dinámicamente estable.

4.1.1. Marcha estáticamente estable

En la marcha estáticamente estable la proyección del centro de gravedad sobre el suelo se mantiene siempre en el centro del polígono de soporte, por lo que es necesario mover el COG de robot sobre el siguiente pie de soporte antes de que el pie oscilante despeje del suelo. En este tipo de marcha se garantiza la estabilidad pero se limita la longitud del paso y la velocidad, además de que el movimiento del COG en el plano frontal aumenta demasiado puesto que oscila de derecha a izquierda, o viceversa, conforme se intercala el pie de soporte. Puesto que la marcha estáticamente estable se realiza a velocidades demasiado lentas, la inercia del mecanismo no influye en el equilibrio, por lo que se desprecia la dinámica del mecanismo y solamente se toman en cuenta los efectos de la gravedad.

4.1.2. Marcha dinámicamente estable

En contraste con la marcha estáticamente estable, en la marcha dinámicamente estable no se limita la proyección del COG a permanecer dentro del polígono de soporte, solamente el ZMP debe permanecer dentro del polígono de soporte, lo que incrementa la velocidad de la marcha puesto que la longitud de paso es mayor, el periodo de soporte doble menor y la oscilación del COG en el plano frontal no es tan pronunciada como en el caso de la marcha estáticamente estable, lo que ahorra tiempo. En la figura 4.1 se muestra la diferencia entre la marcha estáticamente estable y la marcha dinámicamente estable por medio de la comparación de la proyección del COG durante el periodo de soporte simple. En la marcha dinámicamente estable el movimiento del pie oscilante, en tiempo y forma, ayuda a reducir los efectos de la inercia y a no perder el equilibrio. Pero los efectos de la inercia también ayudan a controlar el equilibrio, por lo que la aceleración del COG debe

ser controlada de forma que el ZMP no salga del polígono de soporte.

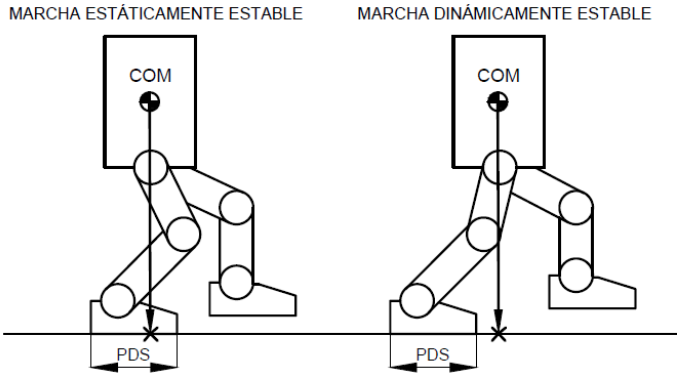

Figura 4.1: Mientras que en la marcha estáticamente estable la proyección del COG debe permanecer dentro del polígono de soporte, la marcha dinámicamente estable permite temporalmente la inestabilidad.

La arquitectura del sistema para generar trayectorias dinámicamente estables considerando la marcha dinámicamente estable, figura 4.2, se define de la siguiente manera:

1. Considerando la estructura física del robot y la longitud de paso, se determina la trayectoria de cada uno de los pies mediante interpolación segmentaria cuadrática.

2. De acuerdo con el polígono de soporte, definido por la trayectoria de los dos pies, se determina la trayectoria del ZMP.

3. Con base en el modelo dinámico del robot bípedo (en este caso se usa la dinámica del modelo carro-mesa) y en la trayectoria del ZMP, definida en el punto anterior, se determina la trayectoria de centro de gravedad del robot.

4. Definidas las trayectorias del centro de gravedad y de los pies, se calculan por medio de cinemática inversa los ángulos que deben adoptar las juntas rotacionales (servomotores) del robot para que los efectores finales (pies y centro de gravedad) sigan las trayectorias deseadas.

5. Por último, se ajustan los ángulos calculados en el paso anterior a los ángulos correspondientes de cada servomotor del robot y son enviados secuencialmente para generar el ciclo de marcha dinámicamente estable.

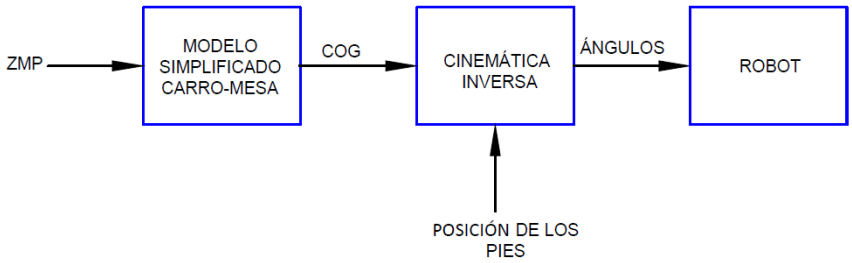

Figura 4.2: Arquitectura de la planeación de trayectorias dinámicamente estables.

Puesto que en la marcha dinámicamente estable no se restringe la proyección del centro de gravedad a permanecer siempre dentro del polígono de soporte, lo que ofrece mayor movilidad, velocidad y movimientos más reales, en este trabajo no se considera la marcha estáticamente estable para mantener el equilibrio dinámico durante la marcha bípeda.

4.2. Modelos de masa concentrada

Como ya se ha mencionado anteriormente, considerar la dinámica del modelo real de un robot bípedo para la planificación de la marcha bípeda implica un alto costo computacional debido al elevado número de grados de libertad que son necesarios para simular la marcha bípeda. Por lo tanto, utilizar el modelo real de un robot bípedo para generar trayectorias dinámicamente estables resulta una tarea demasiado compleja que consume demasiados recursos computacionales, los cuales son necesarios para procesar información y generar patrones de caminata dinámicamente estables en tiempo real. Para simplificar lo máximo posible la dinámica de un robot bípedo se han propuesto los modelos péndulo invertido y carro-mesa, los cuales comparten el mismo principio ya que concentran toda la masa del robot en un solo punto a nivel de la cadera del robot, el cual es considerado como el centro de gravedad del robot completo.

4.2.1. Modelo pendulo invertido lineal

De acuerdo con [12, 13], cuando un robot bípedo se encuentra en periodo de soporte simple, su dinámica puede ser aproximada empleando el modelo de un péndulo invertido que conecta el pie de soporte y el centro de gravedad del robot por medio de una pierna con masa despreciable, figura 4.3. Donde θ_r representa el ángulo entre el plano XZ y el péndulo, θ_p el ángulo entre el plano YZ y el péndulo, m la masa de total del robot, Z la altura del centro de gravedad del robot y r la distancia entre el centro de gravedad y el origen (pie de soporte).

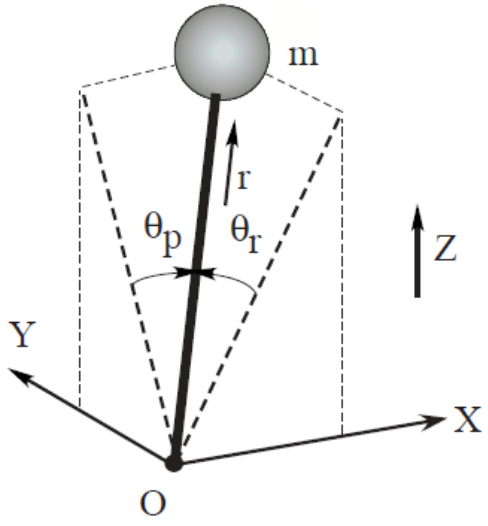

Figura 4.3: Modelo simplificado pendulo invertido. Tomada de [12]

De acuerdo [12], si se consideran (f,τ_r,τ_p) como la fuerza y los momentos angulares que se relacionan respectivamente con las variables de estado (r,θ_r,θ_p), las ecuaciones de movimiento del péndulo invertido se definen como:

$$m(-z\ddot{y} + y\ddot{z}) = \frac{D}{C_r}\tau_r - mgy, \tag{4.1}$$

$$m(-z\ddot{x} + x\ddot{z}) = \frac{D}{C_p}\tau_p - mgx, \tag{4.2}$$

donde:

$$C_r = \cos\theta_r, C_p = \cos\theta_p, D = \sqrt{C_r^2 + C_p^2 - 1}.$$

Partiendo de las ecuaciones (4.1) y (4.2), considerando que el robot camina sobre un terreno plano y que su centro de gravedad se mantiene a una altura constante z_c, es decir que $\ddot{z} = 0$, se obtiene respectivamente que:

$$\ddot{y} = \frac{g}{z_c}y - \frac{D}{mz_cC_r}\tau_r, \tag{4.3}$$

$$\ddot{x} = \frac{g}{z_c}x - \frac{D}{mz_cC_p}\tau_p. \tag{4.4}$$

De acuerdo con las ecuaciones (4.3) y (4.4), las entradas del sistema se vuelven entradas no lineales, puesto que τ_r y τ_p están siendo multiplicadas por un cociente que depende de $\cos\theta_r$ y $\cos\theta_p$, por lo tanto la dinámica del sistema también es no lineal.

Para poder generar las trayectorias del centro de gravedad, con este modelo y empleando indirectamente el criterio de estabilidad del ZMP, se considera que $\tau_r = \tau_p = 0$ y que el origen del péndulo se encuentra dentro del polígono de soporte, por lo que la dinámica del sistema se vuelve lineal:

$$\begin{cases} \ddot{y} = \frac{g}{z_c}y \\ \ddot{x} = \frac{g}{z_c}x. \end{cases} \tag{4.5}$$

De esta forma la generación de trayectorias consiste en establecer la posición del ZMP en los periodos de soporte simple, la cual representa la posición del origen del péndulo invertido a partir de la cual se determina la posición inicial de la masa de éste para resolver el sistema de ecuaciones (4.5), el cual describe el movimiento del masa del péndulo invertido en el espacio únicamente durante el periodo de soporte simple. Cabe mencionar que la condición inicial necesaria para resolver sistema de ecuaciones (4.5) es la posición inicial del centro de gravedad del robot (posición inicial de la masa del péndulo).

En la figura 4.4 se muestra un ejemplo de la generación de la trayectoria del centro de gravedad de un robot bípedo empleando el modelo péndulo invertido lineal. Las curvas hiperbólicas rojas y azules representan la trayectoria del centro de gravedad en los periodos de soporte simple, los puntos rojos la posición del ZMP ubicado en la planta del pie de soporte simple y las líneas rojas discontinuas el sistema de referencia local (ejes x y y) con respecto al cual se generan los movimientos del péndulo.

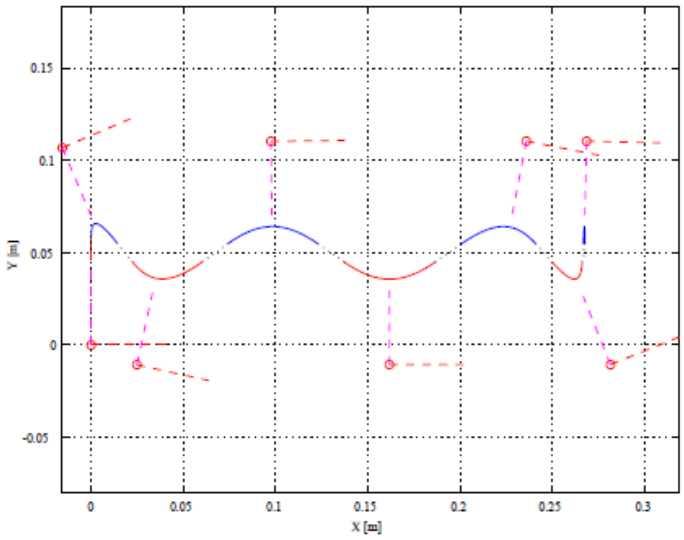

Figura 4.4: Trayectoria del centro de gravedad de un robot bípedo generada a partir del modelo péndulo invertido lineal [12].

4.2.2. Modelo carro-mesa

El modelo carro-mesa [11] es representado por medio de un carro o esfera que se desplaza sobre la superficie horizontal de una mesa, figura 4.5. La masa del carro representa la masa total del robot y su posición representa la posición del centro de gravedad del robot. Es importante destacar que el pie de la mesa representa tanto al polígono de soporte simple como al polígono de soporte doble, dependiendo en qué periodo se encuentre el ciclo de marcha. Por lo tanto, cuando el robot se encuentra en el periodo de soporte simple la base la mesa coincide con la planta del pie de soporte del robot.

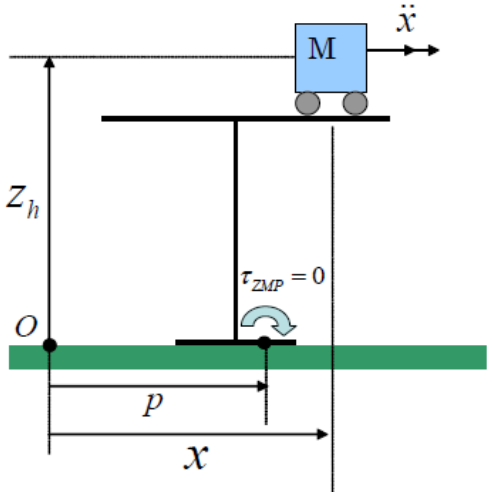

Figura 4.5: Análisis del modelo simplificado carro-mesa [35].

Para comprender el modelo carro-mesa y determinar su relación con el ZMP se parte del esquema mostrado en la figura 4.5, en el cual se muestra el instante en que el sistema

está a punto de perder el equilibrio en el plano frontal o sagital (dependiendo de la orientación del robot). Donde M representa la masa total del modelo real, zh la altura constante del centro de gravedad, $\ddot{x}$ la aceleración del carro sobre la mesa, x la posición horizontal del centro de gravedad con respecto al origen y p la distancia del origen al punto de referencia con respecto al cual se quiere mantener el equilibrio empleando el criterio de estabilidad del ZMP, es decir que no deben existir momentos angulares con respecto al punto p_{zmp}, el cual debe estar localizado dentro del polígono de soporte.

De acuerdo con el esquema de la figura 4.5, la posición y el peso del carro sobre la mesa producen un momento angular con respecto al punto p, por lo que el carro debe acelerar en la dirección correcta y con la aceleración adecuada para producir un momento angular que compense al primero, por lo tanto la sumatoria de momentos debe ser igual a cero:

$$-Mg(x-p)+M\ddot{x}z=0, \tag{4.6}$$

donde el primer término de la ecuación (4.6) representa al momento angular debido al peso y a la posición del carro sobre la mesa, el segundo término representa al momento angular producido por la aceleración del carro sobre la mesa que ejerce la compensación adecuada.

Puesto que la aceleración del centro de gravedad es la variable del sistema que lo mantiene en equilibrio, el punto p_{zmp} con respecto al cual se mantiene el equilibrio (ZMP) puede ser calculado a partir de ésta como:

$$p_{zmp}=x-\frac{z}{g}\ddot{x}. \tag{4.7}$$

Para poder analizar el modelo carro-mesa, tanto en el plano frontal como en el sagital, es necesario desacoplar el sistema de la figura 4.6 como se ilustra en la figura 4.7.

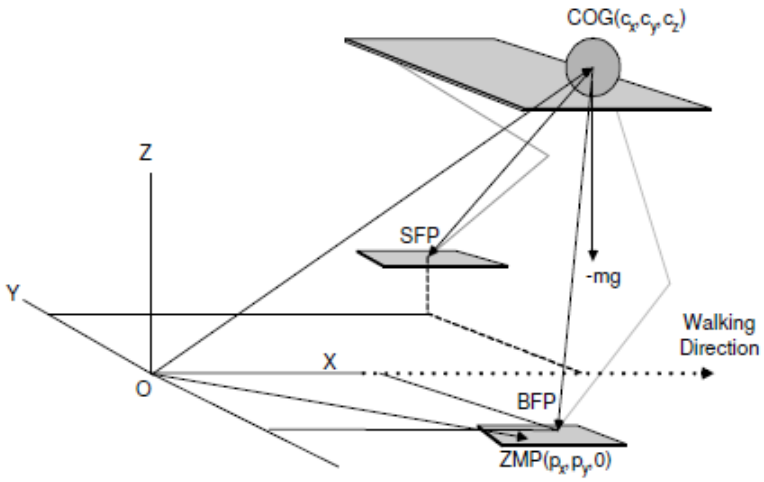

Figura 4.6: Representación del modelo simplificado rolling sphere (carro-mesa) en el espacio [14].

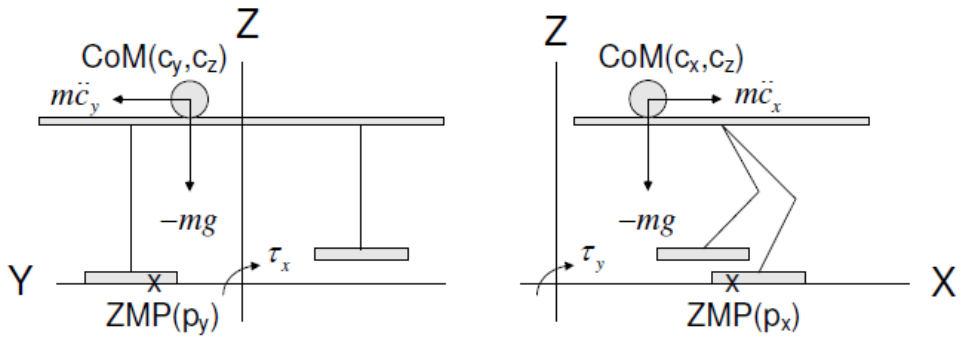

Figura 4.7: Representación del modelo simplificado rolling sphere (carro-mesa), a partir de la figura 4.6, en los planos frontal y sagital respectivamente [36].

Aplicando el análisis de balance de momentos al modelo rolling sphere, en el plano frontal, se obtiene que:

$$-mg(c_y - p_y) + m\ddot{c}_y c_z = 0, \tag{4.8}$$

$$p_y = c_y - \frac{c_z}{g}\ddot{c}_y, \tag{4.9}$$

considerando $c_y = y$ y $c_z = z$, la ecuación (4.9) se puede reescribir como:

$$p_y = y - \frac{z}{g}\ddot{y}. \tag{4.10}$$

En el caso del plano sagital se tiene que:

$$-mg(c_x - p_x) + m\ddot{c}_x c_z = 0, \tag{4.11}$$

$$p_x = c_x - \frac{c_z}{g}\ddot{c}_x, \tag{4.12}$$

considerando $c_x = x$ y $c_z = z$, la ecuación (4.12) se puede reescribir como:

$$p_x = x - \frac{z}{g}\ddot{x}. \tag{4.13}$$

El sistema de ecuaciones conformado por las ecuaciones (4.10) y (4.13) representa al modelo carro-mesa con altura constante:

$$\begin{cases} p_x = x - \frac{z}{g}\ddot{x} \\ \\ p_y = y - \frac{z}{g}\ddot{y}. \end{cases} \tag{4.14}$$

A partir del sistema de ecuaciones (4.14) se destacan dos puntos importantes:

1. Cuando la aceleración del COM es cero ($\ddot{x} = 0$), la proyección del COM corresponde al ZMP resultante.

2. En el sistema de ecuaciones (4.14), el ZMP ($p_{x,y} = ZMP$) resultante no se limita a permanecer dentro del polígono de soporte, por lo que se puede obtener cualquier valor de p fuera de éste.

Por lo tanto, cuando la aceleración del carro es demasiado grande el ZMP resultante puede salir del polígono de soporte como se muestra en la figura 4.8(a), lo que se debe a que el sistema de ecuaciones (4.14) no considera la restricción unilateral, subsección 2.2.3, la cual establece que la reacción del piso debida al apoyo del pie sobre éste tiene solo una dirección.

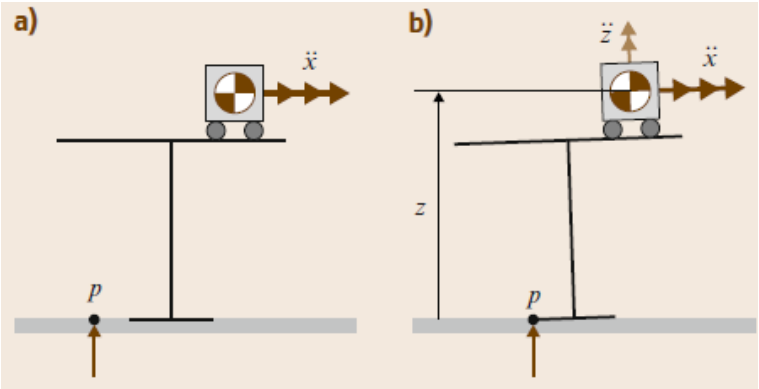

Figura 4.8: Fictitious Zero Moment Point en el modelo carro-mesa [30].

De acuerdo con [30], si se considera la restricción unilateral de $f_{iz} \geq 0$ se obtiene el caso (b) de la figura 4.8. En este caso la mesa comenzará a rotar con respecto al punto p ya que en el caso (a) ($FictitiousZeroMomentPoint$, subsección 2.2.4) no es posible mantenerlo dentro del polígono de soporte, por lo tanto, al ya no mantenerse constante la altura del centro de gravedad, el ZMP resultante en x se puede calcular como:

$$p_x = x - \frac{z}{g + \ddot{z}}\ddot{x}. \tag{4.15}$$

Esta ecuación determina el límite del ZMP resultante en los extremos del polígono de soporte y ya que la altura deja de ser constante, se da pie al modelo carro-mesa paramétrica.

Modelo carro-mesa paramétrica

En [15] se presenta el modelo carro-mesa paramétrica, figura 4.9, en el cual el carro se desplaza sobre una superficie paramétrica de arriba abajo y viceversa. Si se propone la superficie adecuada, se pueden generar patrones para que el robot pueda subir y bajar escaleras, o caminar simplemente sobre un terreno plano. Otra ventaja de este modelo es que la altura del centro de gravedad también se puede variar para ampliar o reducir, de la forma más conveniente, la longitud de paso, incluso se puede reducir el impacto del suelo sobre la planta del pie del robot para evitar impactos que hagan que el robot pierda el equilibrio.

Realizando el análisis de equilibrio de momentos para los ejes coordenados x y y de la figura 4.9, las ecuaciones del punto de equilibrio o ZMP se definen como:

$$\begin{cases} x_{zmp} = x - \frac{z}{g+\ddot{z}}\ddot{x}, \\ \\ y_{zmp} = y - \frac{z}{g+\ddot{z}}\ddot{y}. \end{cases} \tag{4.16}$$

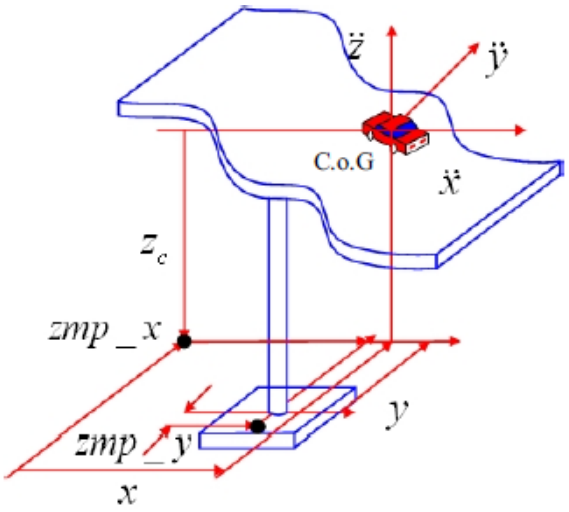

Figura 4.9: Modelo simplificado carro-mesa paramétrica [15].

Propuestos los parámetros u y v en función del tiempo, las coordenadas del carro sobre la mesa paramétrica se definen por el siguiente sistema de ecuaciones:

$$\begin{cases} x = f_x(u(t), v(t)), \\ y = f_y(u(t), v(t)), \\ z = f_z(u(t), v(t)). \end{cases} \tag{4.17}$$

Por lo tanto, si se sustituye el sistema de ecuaciones (4.17) en el sistema de ecuaciones (4.16) se obtiene:

$$\begin{cases} x_{zmp} = f_x - \frac{f_z(f_{xu}\ddot{u}+f_{xv}\ddot{v}+f_{xuu}\dot{u}^2+2f_{xuv}\dot{u}\dot{v}+f_{xvv}\dot{v}^2)}{f_{zu}\ddot{u}+f_{zv}\ddot{v}+f_{zuu}\dot{u}^2+2f_{zu}\dot{u}\dot{v}+f_{zvv}\dot{v}^2+g}, \\ y_{zmp} = f_y - \frac{f_z(f_{yu}\ddot{u}+f_{yv}\ddot{v}+f_{yuu}\dot{u}^2+2f_{yuv}\dot{u}\dot{v}+f_{yvv}\dot{v}^2)}{f_{zu}\ddot{u}+f_{zv}\ddot{v}+f_{zuu}\dot{u}^2+2f_{zu}\dot{u}\dot{v}+f_{zvv}\dot{v}^2+g}. \end{cases} \tag{4.18}$$

La desventaja de este modelo, de acuerdo al sistema de ecuaciones (4.18), es que al existir derivadas parciales el computo empleado para la solución del modelo carro-mesa paramétrica es más exhaustivo que en el caso del modelo carro-mesa simple.

4.2.3. Comparación

Como se aprecia en la figura 4.4, la principal desventaja del modelo péndulo invertido lineal es que la trayectoria generada para el centro de gravedad no es continua, puesto que este modelo sólo sirve para representar al robot bípedo en el periodo de soporte simple, por lo tanto si se quiere generar una trayectoria continua se debe interpolar entre los segmentos de trayectoria del COM en los periodos de soporte simple, lo que produce inexactitud. Otra desventaja del modelo péndulo invertido lineal es que no existe una relación directa con el ZMP puesto que es difícil considerar la no linealidad del sistema, en la generación de trayectorias, cuando los momentos angulares son diferentes de cero. La única manera de establecer una relación con el ZMP es representativa puesto que sólo se considera que los momentos angulares debidos a la gravedad son nulos cuando el origen del péndulo

se encuentra en el ZMP deseado. Al no considerar los momentos angulares también se está despreciando la dinámica del sistema.

Por otra parte, con el modelo carro-mesa sí se pueden generar trayectorias continuas para el centro de gravedad, en la figura 4.10 se muestra un ejemplo de la trayectoria del centro gravedad de un robot bípedo generada a partir del modelo carro-mesa paramétrica. Además de que con este modelo se establece una relación directa del sistema con el ZMP, puesto que el sistema de ecuaciones que representa al modelo carro-mesa se define a partir de la compensación de los momentos angulares en el ZMP. Al contrario del modelo péndulo invertido lineal, el modelo carro-mesa sí considera la dinámica del sistema y la altura del centro de gravedad no tiene que mantenerse constante, lo que amplía las posibilidades de generar trayectorias dinámicamente estables.

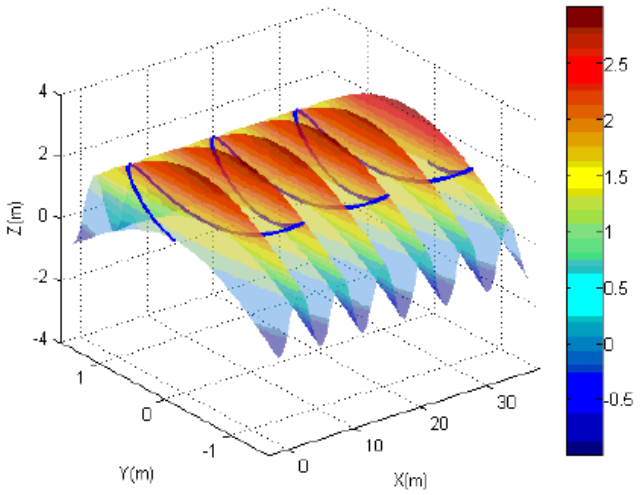

Figura 4.10: Superficie paramétrica y trayectoria del centro de gravedad de un robot bípedo generada a partir del modelo simplificado carro-mesa paramétrica [15].

Es importante destacar que han sido propuestas al menos 3 versiones diferentes del modelo péndulo invertido que tienen la finalidad de brindar mayor robustez al sistema y así evitar las restricciones del modelo péndulo invertido lineal, a cambio de una solución más compleja. El modelo péndulo invertido con punto de soporte virtual [37] así como el modelo con 2 tipos de péndulo invertido [38] y el modelo péndulo invertido móvil [39] son algunas de estas versiones. El más importante de estos modelos es el modelo péndulo invertido móvil, figura 4.11, ya que también considera la dinámica del sistema e incluso permite generar trayectorias cuando el robot camina sobre terrenos discontinuos, para lo cual sería necesario contar con sensores que detecten las irregularidades de la superficie o conocerlas previamente y generar las trayectorias de los pies de acuerdo con éstas. El problema que existe con estos modelos es que no se encuentran lo suficientemente documentados como en el caso de los modelos péndulo invertido lineal y carro-mesa.

En [40] se propone solucionar el modelo péndulo invertido móvil por medio de control óptimo para generar la trayectoria del COM que optimiza el cambio del ZMP con respecto al tiempo. Sin embargo se desea obtener la trayectoria del COM que garantice la estabilidad dinámica del robot, en lugar de optimizar la velocidad del ZMP, lo cual se

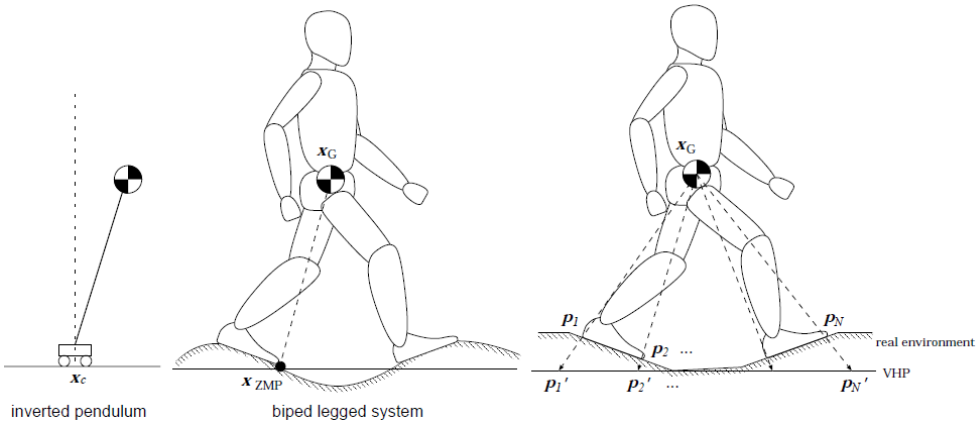

Figura 4.11: El modelo simplificado péndulo invertido móvil permite generar trayectorias incluso sobre terrenos irregulares a través de un plano virtual horizontal [39].

logra optimizando el *jerk* del COM aplicando control óptimo al modelo carro-mesa como se verá en la siguiente sección. Por lo tanto se puede afirmar que el modelo carro-mesa es un modelo robusto que se adapta mejor a la necesidad de generar trayectorias dinámicamente estables para el centro de gravedad, en comparación con el modelo péndulo invertido lineal o péndulo invertido móvil.

4.3. Métodos de solución para el modelo carro-mesa

Como se mencionó anteriormente, en la motivación, se han propuesto muy pocas soluciones para el modelo carro-mesa, de las cuales algunas son soluciones aproximadas y algunas otras consumen demasiados recursos computacionales. Por ejemplo, en [14] se aplica la transformada de Laplace, la transformada inversa de Laplace y las series de Fourier para obtener la trayectoria del COG, además de que el ZMP de referencia (figura 4.12) se restringe ya que debe ser representado como una función impulso unitario periódica, tanto en x como en y, eliminando así una amplia gama de posibles trayectorias a ser ejecutadas por el robot.

En [41] se aplica la convolución discreta al modelo carro-mesa, en este caso la salida del sistema es el COG, el cual es considerado como la respuesta al impulso unitario. Por lo tanto nuevamente se restringe la trayectoria del ZMP al ser representada como una función escalón unitario periódica, tanto en x como en y, eliminado así el periodo de soporte doble y provocando que el centro de gravedad tenga movimientos más bruscos, por lo que el robot estará más propenso a caer en cada intercambio de pie de soporte simple. Puesto que el intercambio de pie de soporte simple ocurre instantáneamente, al eliminar la migración del ZMP de un pie a otro y hacerlo de forma instantánea, la proyección del centro de gravedad se mantiene siempre dentro del polígono de soporte simple antes de migrar al polígono de soporte simple del siguiente pie de soporte, lo que se asemeja más a la marcha estáticamente estable y a sus deficiencias.

Por lo tanto ninguno de estos métodos es considerado óptimo para la generación de trayectorias dinámicamente estables. A continuación se presentan dos métodos que no

restringen la representación del ZMP como una función impulso unitario periódica y que además no involucran operaciones matemáticas complejas en la solución del modelo carromesa.

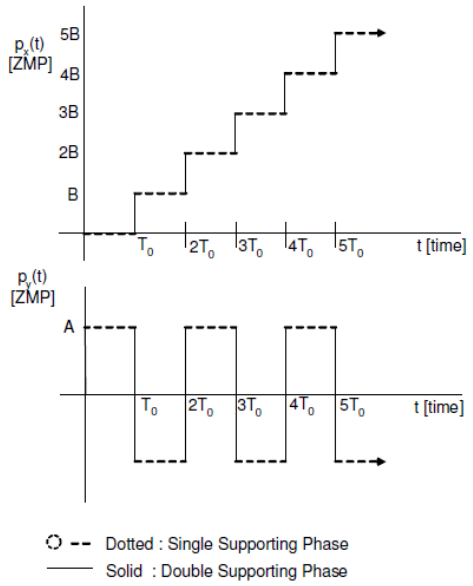

Figura 4.12: Trayectoria del ZMP representada como una función impulso unitario periódica [14].

4.3.1. Solución por discretización de la aceleración

En [18] y [42] se propone un método relativamente sencillo que consiste en discretizar la aceleración para trabajar con una expresión algebraica en lugar de trabajar con una diferencial. Para determinar la aceleración en el tiempo discreto i, con un tiempo de muestreo Δt, es necesario calcular la derivada de la primera derivada de la posición. Para calcular la segunda derivada, en el tiempo discreto i, se calcula la primera derivada en el punto posterior $i + \frac{1}{2}$ y en el punto anterior $i - \frac{1}{2}$, la diferencia entre ambas dividida entre el intervalo de tiempo discreto que hay entre ambas, en este caso Δt, será la segunda derivada de la posición:

$$\frac{dx}{dt}(i + \frac{1}{2}) = \frac{x(i+1) - x(i)}{\Delta t},$$

$$\frac{dx}{dt}(i - \frac{1}{2}) = \frac{x(i) - x(i-1)}{\Delta t},$$

$$\frac{d^2}{dt^2}x(i) = \frac{\frac{x(i+1)-x(i)}{\Delta t} - \frac{x(i)-x(i-1)}{\Delta t}}{\Delta t}$$

$$= \frac{x(i-1) - 2x(i) + x(i+1)}{\Delta t^2}.$$

Por lo tanto, la aceleración discretizada en x y en y se define respectivamente como:

$$\ddot{x_i} = \frac{x_{i-1} - 2x_i + x_{i+1}}{\Delta t^2}, \tag{4.19}$$

$$\ddot{y_i} = \frac{y_{i-1} - 2y_i + y_{i+1}}{\Delta t^2}. \tag{4.20}$$

Si se sustituyen respectivamente las ecuaciones (4.19) y (4.20) en las ecuaciones (4.13) y (4.10), se obtiene que:

$$p_{ix} = x_i - \frac{z}{g}\Big(\frac{x_{i-1} - 2x_i + x_{i+1}}{\Delta t^2}\Big),$$

$$p_{iy} = y_i - \frac{z}{g}\Big(\frac{y_{i-1} - 2y_i + y_{i+1}}{\Delta t^2}\Big),$$

lo que se puede reescribir como:

$$p_{ix} = a_i x_{i-1} + b_i x_i + c_i x_{i+1}, \tag{4.21}$$

$$p_{iy} = a_i y_{i-1} + b_i y_i + c_i y_{i+1}. \tag{4.22}$$

con

$$1 \leq i \leq N$$

donde p_{ix} y p_{iy} son las componentes del ZMP tanto en x como en y en el tiempo discreto i, N el numero máximo de elementos discretos y a_i, b_i y c_i constantes definidas como:

$$a_i = -\frac{z}{g\Delta t^2}$$
$$b_i = \frac{2z}{g\Delta t^2} + 1$$
$$c_i = -\frac{z}{g\Delta t^2}$$

Es importante destacar que el robot parte del reposo, por lo que $x_0 = x_1$, $y_0 = y_1$ y por lo tanto cuando $i = 1$,

$$p_{1x} = (a_1 + b_1)x_1 + c_1 x_2,$$
$$p_{1y} = (a_1 + b_1)y_1 + c_1 y_2,$$

de igual forma $x_{N+1} = x_N$ y $y_{N+1} = y_N$ por lo tanto cuando $i = N$,

$$p_{Nx} = a_N x_{N-1} + (b_N + c_N)x_N,$$

$$p_{Ny} = a_N y_{N-1} + (b_N + c_N)y_N.$$

De esta forma se logra representar al sistema de ecuaciones diferenciales que define al modelo carro-mesa, como un sistema de ecuaciones algebraico. Para resolverlo se comienza por representar a p_{ix} de forma matricial como:

$$p_x = Ax, \tag{4.23}$$

donde los elementos de p_x representan la serie de puntos discretos que conforman la trayectoria del ZMP establecida, los elementos de x son los puntos o elementos que conforman

la trayectoria del COG que se quiere obtener a partir del ZMP de referencia y A es una matriz conformada por los coeficientes que definen a pix, esto es:

$$p_x = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{N-1} \\ p_N \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} \quad y \quad A = \begin{bmatrix} a_1+b_1 & c_1 & 0 & & & & \\ a_2 & b_2 & c_2 & \ddots & & & \\ & & & \ddots & & & \\ & & & \ddots & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & 0 & a_N & b_N+c_N \end{bmatrix}$$

La solución del COG en el plano sagital puede ser obtenida fácilmente multiplicando ambos lados de la ecuación (4.23) por la inversa de la matriz A:

$$x = A^{-1}p_x$$

Puesto que los coeficientes a_i, b_i y c_i son los mismos tanto para p_{ix} como para p_{iy}, la matriz de coeficientes A también es la misma para ambos casos, por lo que la solución del COG en plano frontal puede ser obtenida de forma análoga como:

$$y = A^{-1}p_y$$

4.3.2. Solución por seguimiento de trayectorias

Puesto que en el sistema de ecuaciones (4.14), el cual define al modelo carro-mesa tanto en x como en y, la salida es el ZMP, en [11] se propone controlar el ZMP a partir del movimiento del carro sobre la mesa para que siga una trayectoria de referencia. Por lo tanto, la solución por seguimiento de trayectorias puede ser considerada como el problema inverso de la planeación de trayectorias, es decir se genera la trayectoria del ZMP a partir del movimiento del carro sobre la mesa (COG) en vez de generar la trayectoria del COG a partir de ZMP deseado.

Como se vio en la sección 4.2.2, en el análisis del modelo carro-mesa, la aceleración del carro sobre la mesa es la variable del sistema que define el punto de equilibrio (ZMP) en el que el momento angular debido al peso es compensado para mantener el equilibrio. Por lo tanto, considerando la variación de la aceleración del carro con respecto al tiempo como la entrada de la ecuación (4.13), la cual define al modelo carro-mesa en el plano sagital, se puede trasladar ésta a un sistema dinámico como:

$$\begin{aligned} \tfrac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x \\ p_x &= \begin{bmatrix} 1 & 0 & -\frac{z}{g} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \end{aligned} \tag{4.24}$$

Para el plano frontal se define u_y y se obtiene el sistema dinámico de la misma forma.

Definido el sistema dinámico, se puede generar la trayectoria del COG de forma que el ZMP resultante siga la trayectoria de referencia. Puesto que el sistema dinámico para x es el mismo que para y, solo cambia la variable, se emplean los términos sistema dinámico, COG y ZMP indistintamente.

El ZMP resultante puede ser generado a partir de un sistema de control predictivo que utiliza información futura de la trayectoria de referencia o demanda solicitada.

Control Predictivo

Los requerimientos que debe satisfacer el controlador, mediante el cual se controlará al ZMP resultante, son:

- Que en estado estacionario, el ZMP resultante siga al ZMP de referencia o demanda solicitada.
- Que muestre una respuesta transitoria aceptable.

De acuerdo con [43], para lograr satisfacer estos requerimientos es necesario representar el sistema dinámico (4.24) como un sistema dinámico discreto de la siguiente forma:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ p(k) &= Cx(k) \end{aligned} \tag{4.25}$$

donde:

$$x(k) = \begin{bmatrix} x(kT) \\ \dot{x}(kT) \\ \ddot{x}(kT) \end{bmatrix},$$

$$u(k) = u_x(kT),$$

$$p(k) = p_x(kT),$$

$$A = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & -\frac{z}{g} \end{bmatrix}$$

Ahora se define $p_d(k)$ como la salida deseada, o demanda solicitada, para la cual se asume que existe un valor $\bar{p_d}$ tal que:

$$\lim_{k\to\infty} p_d(k) = \bar{p_d}.$$

Lo que quiere decir que la demanda solicitada es variable con respecto al tiempo, pero que tiende a un valor estable. También se considera que en cualquier tiempo discreto k, se

cuenta con N_L valores futuros de la demanda solicitada, o ZMP de referencia en nuestro caso. Para valores deseados después de la ventana de valores futuros con la que se cuenta a partir de tiempo actual k, es decir para valores a partir de $k + N_L$, se considera que su valor es igual al último valor de la ventana de valores futuros, es decir:

$$p_d(k+i) = p_d(k+N_L), \quad i = N_L + 1, \ldots$$

Puesto que se desea una respuesta transitoria aceptable y que el error en el estado estacionario sea nulo, se debe introducir la acción integradora para reducir el error de seguimiento $e(k) = p(k) - p_d(k)$, y obtener el controlador adecuado mediante control óptimo. De acuerdo con [43], si se introduce el vector de variables de estado incremental en el índice de desempeño J, se podrá regular la respuesta transitoria de las variables de estado. Por otra parte, la acción integradora del controlador es llevada a cabo introduciendo también como término el vector de control incremental. El vector de control incremental es definido como: $\Delta u(k) = u(k) - u(k-1)$ y el vector de variables de estado incremental como: $\Delta x(k) = x(k) - x(k-1)$. Por lo tanto, se desea obtener el controlador óptimo $u(k)$ que minimiza la función de costo

$$J = \sum_{i=k}^{\infty} \left[e^T(i) Q_e e(i) + \Delta x^T(i) Q_x \Delta x(i) + \Delta u^T(i) R \Delta u(i) \right] \tag{4.26}$$

con la cual se mide el desempeño del sistema para asegurar que el ZMP generado siga la trayectoria de referencia sin una velocidad de cambio excesiva tanto del controlador como del vector de estados.

El sistema dinámico incremental es definido, a partir del sistema (4.25), como:

$$\begin{aligned} \Delta x(i+1) &= A\Delta x(i) + B\Delta u(i), \\ \Delta p(i) &= C\Delta x(i). \quad i = k, k+1, \ldots \end{aligned}$$

Puesto que el error de seguimiento es definido como $e(k) = p(k) - p_d(k)$, el error de seguimiento incremental puede ser definido como:

$$\Delta e(i) = \Delta p(i) - \Delta p_d(i), \quad i = k, k+1, \ldots$$

donde $\Delta p_d(i) = p_d(i) - p_d(i-1)$. Por lo tanto,

$$\begin{aligned} e(i+1) &= e(i) + \Delta e(i), \\ e(i+1) &= e(i) + CA\Delta x(i) + CB\Delta u(i) - \Delta p_d(i+1). \quad i = k, k+1, \ldots \end{aligned}$$

Combinando $\Delta x(i+1)$ y $e(i+1)$ se obtiene el siguiente sistema:

$$\begin{bmatrix} e(i+1) \\ \Delta x(i+1) \end{bmatrix} = \begin{bmatrix} 1 & CA \\ 0 & A \end{bmatrix} \begin{bmatrix} e(i) \\ \Delta x(i) \end{bmatrix} + \begin{bmatrix} CB \\ B \end{bmatrix} \Delta u(i) + \begin{bmatrix} -1 \\ 0 \end{bmatrix} \Delta p_d(i+1) \tag{4.27}$$

donde $i = k, k+1, \ldots$

Los N_L valores futuros del ZMP de referencia, a partir del tiempo de referencia actual k, pueden ser ordenados por medio de un vector de valores deseados como:

$$x_d(k) = \begin{bmatrix} \Delta p_d(k+1) \\ \Delta p_d(k+2) \\ \vdots \\ \Delta p_d(k+N_L) \end{bmatrix}$$

Por lo que:

$$x_d(i+1) = A_d x_d(i), \tag{4.28}$$

donde $i = k, k+1, \ldots,$

$$A_d = \begin{bmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & & 0 \\ & & & \ddots & \\ & & & \ddots & 1 \\ 0 & & & & 0 \end{bmatrix}$$

A continuación se define el vector de estados incremental como:

$$\bar{x}(i) = \begin{bmatrix} e(i) \\ \Delta x(i) \\ x_d(i) \end{bmatrix}$$

De las ecuaciones (4.27) y (4.28) se obtiene el siguiente sistema:

$$\bar{x}(i+1) = \begin{bmatrix} \begin{bmatrix} 1 & CA \\ 0 & A \end{bmatrix} & \begin{bmatrix} -1 \\ 0 \end{bmatrix} \\ 0 & A_d \end{bmatrix} \bar{x}(i) + \begin{bmatrix} \begin{bmatrix} CB \\ B \end{bmatrix} \\ 0 \end{bmatrix} \Delta u(i), \quad i = k, k+1, \ldots$$

Del cual se obtienen las siguientes matrices:

$$\tilde{F} = \begin{bmatrix} CA \\ A \end{bmatrix}, \ \tilde{I} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \tilde{A} = \begin{bmatrix} \tilde{I} & \tilde{F} \end{bmatrix}, \ \tilde{B} = \begin{bmatrix} CB \\ B \end{bmatrix}$$

Empleando el vector de estados incremental, el índice de desempeño (4.26) se define como:

$$J = \sum_{i=k}^{\infty} \left\{ \bar{x}^T(i) \begin{bmatrix} Q_e & 0 & 0 \\ 0 & Q_x & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{x}(i) + \Delta u^T(i) R \Delta u(i) \right\} \tag{4.29}$$

Por lo tanto el controlador óptimo es obtenido resolviendo el algoritmo matemático que minimiza el índice de desempeño (4.29). En [43] se resuelve el algoritmo matemático que minimiza el índice de desempeño (4.29) empleando la ecuación de Riccati a partir de [32], y se define el controlador óptimo como:

$$u(k) = \overbrace{-G_I \sum_{i=0}^{k} e(i)}^{Termino\ integrativo} - \overbrace{G_x x(k)}^{R.\ de\ estados} - \overbrace{\sum_{l=1}^{N_L} G_d(l) p_d(k+l)}^{Ganancia\ predictiva} \tag{4.30}$$

donde $p(k) = p_d(k) = 0, x(k) = 0$ para $k = 0, -1, \ldots,$

$$G_I = [R + \tilde{B}^T \tilde{P} \tilde{B}]^{-1} \tilde{B}^T \tilde{P} \tilde{I}$$

$$G_x = [R + \tilde{B}^T \tilde{P} \tilde{B}]^{-1} \tilde{B}^T \tilde{P} \tilde{F}$$

$$G_d(l) = [R + \tilde{B}^T \tilde{P} \tilde{B}]^{-1} \tilde{B}^T \tilde{X}(l-1), \quad l = 2, \ldots, N_L; \quad G_d(1) = -G_I$$

$$\tilde{X}(l) = \tilde{A}^T \tilde{X}(l-1), \quad l = 2, \ldots, N_L; \quad \tilde{X}(1) = -\tilde{A}_c^T \tilde{P} \tilde{I}$$

$$\tilde{A}_c = \tilde{A} - \tilde{B}[R + \tilde{B}^T \tilde{P} \tilde{B}]^{-1} \tilde{B}^T \tilde{P} \tilde{A}$$

y P es la solución de la ecuación algebraica de Riccati:

$$\tilde{P} = \tilde{A}^T \tilde{P} \tilde{A} - \tilde{A}^T \tilde{P} \tilde{B}[R + \tilde{B}^T \tilde{P} \tilde{B}]^{-1} \tilde{B}^T \tilde{P} \tilde{A} + \tilde{Q}$$

De acuerdo con la ecuación (4.30), este controlador depende de tres términos: de la acción integral del error entre el ZMP resultante y el deseado (error de seguimiento), de la retroalimentación de estados y de la ponderación (ganancia predictiva) de un N número de valores futuros, del ZMP deseado, que se emplearán en el controlador.

4.3.3. Simulaciones

Para generar la trayectoria del ZMP deseado, implementar los algoritmos de solución por discretización de la aceleración y de solución por control predictivo, obtener la trayectoria del centro de gravedad por medio de estos y generar las trayectorias de la planta de cada pie se utilizó el software de *Mathematica*®. Por cada método de solución se generó un programa diferente, los cuales se encuentran documentados en los apéndices A y B, respectivamente.

Los programas generados en *Mathematica*® se encuentran divididos en secciones, para una mayor organización, las cuales son diferentes para cada método de solución. En la primera sección de cada uno, llamada *Datos*, se ingresan los parámetros deseados para definir el ciclo de marcha, el sistema del modelo carro-mesa y la trayectoria del ZMP deseado. Se ingresan parámetros como la aceleración de la gravedad, la altura del COG, la longitud y altura del paso, el tiempo de discretización y la distancia sobre el eje x del origen al punto dentro de la planta de los pies del robot donde se desea que se encuentre el ZMP en los periodos de soporte simple.

Cabe destacar que una vez que se ingresan estos parámetros, la trayectoria discreta del ZMP se define automáticamente (en la segunda sección de cada programa) para que el robot camine 5 pasos hacia adelante. La trayectoria de los pies también se define automáticamente, mediante interpolación segmentaria cuadrática, considerando únicamente la altura del paso, la longitud de paso y el número de pasos programado. Por otra parte, la duración de los periodos del ciclo de marcha depende del tiempo elegido para la discretización del sistema, manteniendo siempre la misma razón de proporcionalidad entre ambos.

En el caso del programa para la solución por discretización de la aceleración, en la tercera sección se define, de forma automática, la matriz cuadrada de coeficientes los cuales son calculados en la primera sección con base en los parámetros ingresados. La dimensión de la matriz cuadrada es definida con base en el número de elementos empleados para generar la trayectoria discreta del ZMP, los cuales dependen del número de pasos, del tiempo de discretización y de los periodos del ciclo de marcha. Se dice que la matriz cuadrada de coeficientes se define de forma automática porque el programa está diseñado para generar el ZMP, la trayectoria del COG y la trayectoria de los pies para que el robot ejecute 5 pasos, pero si no se desean calcular las trayectorias completas se puede calcular sólo lo correspondiente a los pasos deseados, por lo que la matriz de coeficientes debe reajustarse automáticamente al número de elementos que se desean calcular.

En la cuarta sección del programa para la solución por discretización de la aceleración se procede a calcular, mediante la función $Inverse[A]$, la inversa de la matriz de coeficientes A calculada en la sección 3. Posteriormente, en esta misma sección, se programa el algoritmo de multiplicación de matrices mediante el cual se definen los elementos discretos de la trayectoria del COG.

En el caso del programa para la solución por seguimiento de trayectorias, en la primera sección se definen también las matrices Ac, Bc, Cc y Dc correspondientes al sistema dinámico continuo del modelo carro-mesa, ecuación (4.24).

Por otra parte, en la tercera sección se lleva a cabo la discretización del sistema, empleando las matrices del sistema dinámico continuo (Ac, Bc, Cc y Dc), el cual es definido en mathematica mediante la función $StateSpaceModel[a, b, c, d]$, donde $a = Ac,\ b = Bc,\ c = Cc$ y $d = Dc$. Para llevar a cabo la discretización del sistema continuo se emplea la función $ToDiscreteTimeModel[sys, \tau]$, donde sys es el sistema dinámico continuo definido mediante la función $StateSpaceModel[a, b, c, d]$ y τ el tiempo de muestreo, en este caso se empleó el tiempo de muestro propuesto en [11] y en [40]. Una vez que se ha definido el sistema dinámico discreto y las matrices ad, bd, cd y dd que lo constituyen, se procede a definir las matrices $\tilde{A}$ y $\tilde{B}$ necesarias para resolver la ecuación de Riccati.

En la cuarta sección del programa para la solución por seguimiento de trayectorias se resuelve la ecuación de Riccati mediante las matrices de peso $Q,\ R$ y las matrices $\tilde{A}$ y $\tilde{B}$ calculadas en la sección anterior; para los valores de Q y R se parte de los prupuestos en [11]:

$$Q = \begin{bmatrix} Q_e & 0 & 0 \\ 0 & Q_x & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 \times 10^{-6} \end{bmatrix}$$

Mediante la función $DiscreteRiccatiSolve[\{aR, bR\}, \{Q, R\}]$ se resuelve la ecuación de Riccati, donde $aR = \tilde{A}$ y $bR = \tilde{B}$. Una vez que se ha obtenido la solución de la ecuación de Riccati, se procede a determinar las constantes del controlador G_I, G_x y la ganancia predictiva G_d, programando en $Mathematica$® las ecuaciones que definen a cada una. En la quinta sección se obtiene el controlador con base en las constantes obtenidas anteriormente, en la ganancia predictiva y en la programación de algoritmo que realiza las operaciones

necesarias para calcular los tres términos del controlador. Las dos últimas secciones, de los dos programas, sirven para determinar la trayectoria de los pies, en y y z, por medio de interpolación segmentaria cuadrática. Para el caso del avance en y, los datos necesarios para la interpolación son obtenidos de la longitud de paso deseada, para el caso de z es necesario ingresar en la primera sección la altura de paso máxima.

Con el propósito de simular los métodos de solución del modelo carro-mesa y de establecer un punto de partida para la experimentación con el prototipo físico, considerando la estructura física del robot, se seleccionaron los siguientes parámetros:

Parametro	Valor
Longitud de paso	7 [cm]
Altura del centro de gravedad	23.5 [cm]
Amplitud de paso	4.454 [cm]
Periodo de soporte simple	87.5 %
Periodo de soporte doble	12.5 %
Tiempo de muestreo	5 [ms]

Donde la altura del centro de gravedad elegida, es la altura con base en la cual se desarrollaron los modelos de cinemática y dinámica espacial para el robot en [19], la amplitud de paso elegida es la distancia correspondiente para que en el periodo de soporte simple el ZMP se encuentre en el punto de contacto de la planta del pie con el último eslabón de la pierna correspondiente, ver figura 3.2, y el tiempo de muestreo elegido es el propuesto en [11, 40]. El periodo de soporte doble elegido es del 12.5 % del periodo total de un solo pie (periodo de soporte doble + periodo de soporte simple) cuando, de acuerdo con los preliminares, éste debería ser aproximadamente del 25 %. La razón, como ya se mencionó, es comenzar a experimentar y ver si realmente afecta a la marcha dinámicamente estable un periodo de soporte simple corto, ya que cuando la velocidad del ciclo de marcha aumenta el periodo de soporte doble decrece, tanto que cuando el ser humano corre el periodo de soporte doble desaparece.

En las figuras 4.13, 4.14 y 4.15 se muestran los resultados de la solución por discretización de la aceleración. La trayectoria azul de cada imagen representa la trayectoria del ZMP deseado, mientras que la trayectoria roja representa la posición del COG correspondiente al ZMP deseado.

En cuanto a la solución por seguimiento de trayectorias, se consideró una ventana de valores futuros del ZMP deseado de 270 elementos ya que el peso, determinado por la ganancia predictiva, del elemento 270 tiene un valor aproximado a cero y no influye en el controlador al igual que los elementos siguientes. En la figura 4.16 se muestra la gráfica de la ganancia predictiva y para mostrar el efecto de ésta sobre el controlador, en la figura 4.17 se muestra el ZMP deseado en x (color azul) y el generado mediante seguimiento de trayectorias (color rojo) considerando el peso de 120 elementos futuros del ZMP deseado para calcular el termino predictivo del controlador. En las figuras 4.18 y 4.19 se muestran el ZMP deseado en x (color azul) y el generado (color rojo) empleando el peso de 270 y 420 elementos futuros del ZMP deseado, respectivamente.

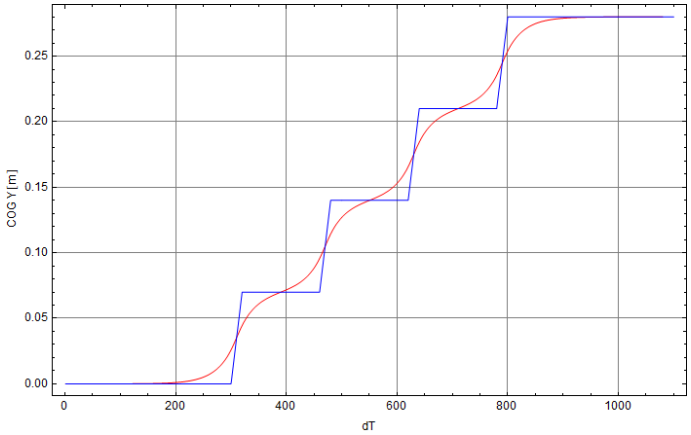

Figura 4.13: Trayectoria del centro de gravedad (roja) correspondiente al ZMP (azul) deseado en y, obtenida mediante la solución por discretización de la aceleración.

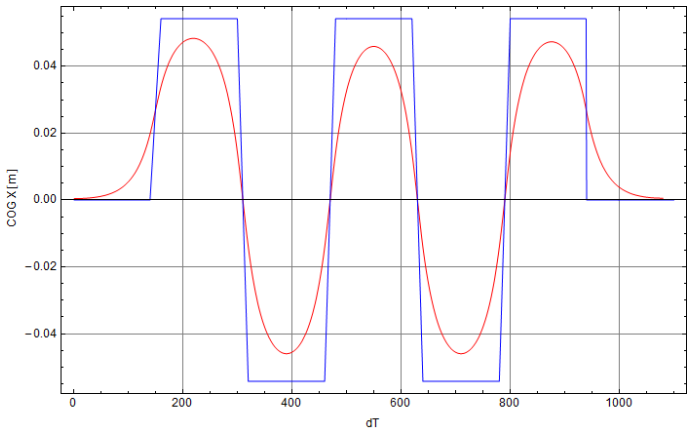

Figura 4.14: Trayectoria del centro de gravedad (roja) correspondiente al ZMP (azul) deseado en x, obtenida mediante la solución por discretización de la aceleración.

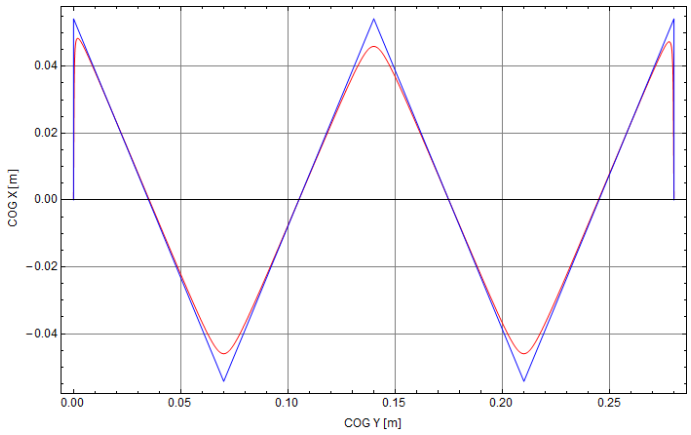

Figura 4.15: Trayectoria vista desde el plano transversal, del centro de gravedad (roja) correspondiente al ZMP (azul) deseado, obtenida mediante la solución por discretización de la aceleración.

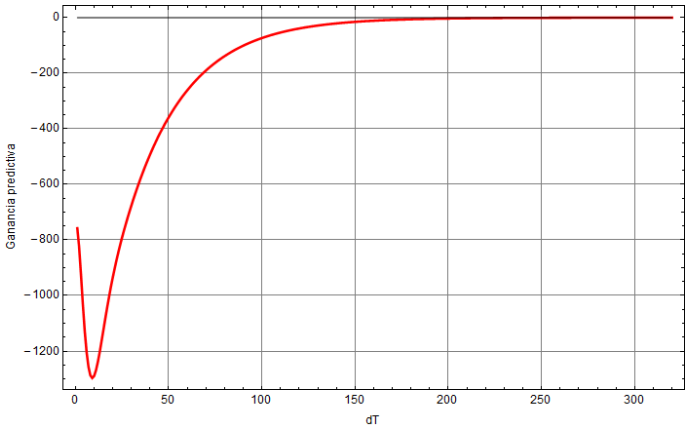

Figura 4.16: La ganancia predictiva determina el peso de los valores futuros del ZMP deseado en el controlador. Una ganancia predictiva cercana a cero producirá un efecto nulo sobre el controlador.

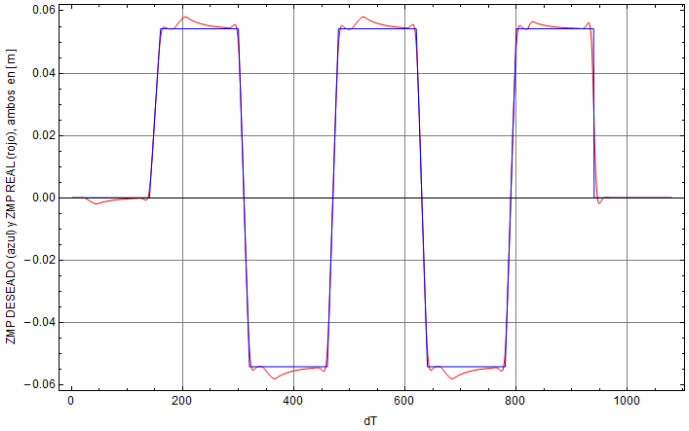

Figura 4.17: Trayectoria del ZMP (roja) generada mediante control por seguimiento de trayectorias, con la ponderación de 120 elementos futuros del ZMP deseado en x (azul).

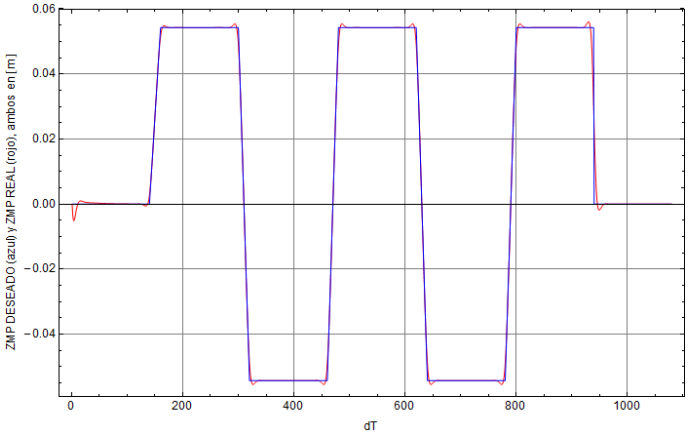

Figura 4.18: Trayectoria del ZMP (roja) generada mediante control por seguimiento de trayectorias, con la ponderación de 270 elementos futuros del ZMP deseado en x (azul).

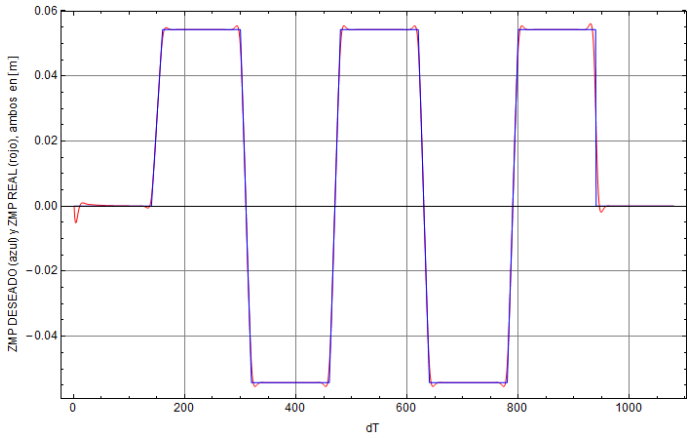

Figura 4.19: Trayectoria del ZMP (roja) generada mediante control por seguimiento de trayectorias, con la ponderación de 420 elementos futuros del ZMP deseado en x (azul).

Como se puede apreciar en la figura 4.17, la discrepancia entre el ZMP deseado y el generado es mayor cuando no se emplea una ventana de valores futuros del ZMP lo suficientemente extensa. Por otra parte, cuando se emplean 270 elementos futuros del ZMP deseado (figura 4.18) se obtiene una buena aproximación del ZMP generado, además de que la ganancia predictiva para el elemento 270 es aproximadamente cero, por lo que emplear más de 270 elementos futuros, de la demanda solicitada, para calcular el termino predictivo del controlador resulta redundante ya que el ZMP generado con 420 elementos es demasiado similar al generado con 270 elementos, tal y como se muestra en la figura 4.19. Para los pesos propuestos en [11], se obtuvieron en promedio los siguientes errores relativos entre el ZMP deseado y el generado:

$$e_x = 0{,}441\,\%,$$
$$e_y = 0{,}164\,\%,$$

por lo que se considera que, en cuanto a exactitud, esta solución ofrece una aproximación bastante aceptable. Empleando una ventana de 420 elementos futuros del ZMP deseado, con su respectiva ganancia predictiva, se obtuvieron en promedio los siguientes errores relativos:

$$e_x = 0{,}438\,\%,$$
$$e_y = 0{,}154\,\%,$$

con lo que se demuestra que la aproximación que se obtiene con una ventana predictiva de 270 valores es prácticamente igual a la aproximación que se obtiene empleando 420 elementos, por lo que no vale la pena emplear más de 270 valores futuros del ZMP deseado, ni calcular su respectiva ganancia.

En las figuras 4.20 y 4.21 se muestran las trayectorias del ZMP generado (azul) y las del centro de gravedad (rojo) calculado a partir de control por seguimiento de trayectorias para que el ZMP generado siga el ZMP de referencia en x y en y respectivamente, empleando una ventana de valores futuros de 270 elementos con su respectiva ganancia.

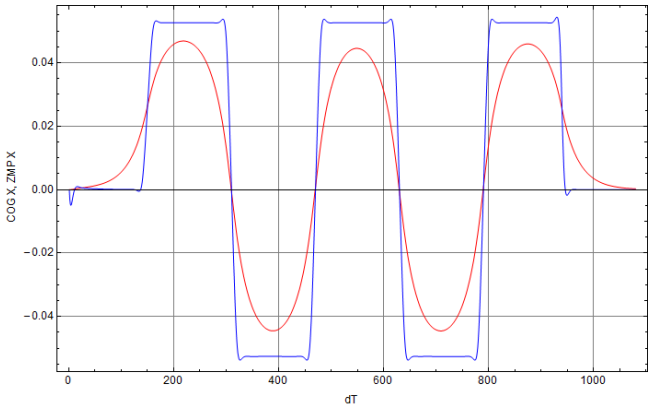

Figura 4.20: Trayectoria del COG (roja) generada mediante control por seguimiento de trayectorias para controlar el ZMP de salida en x(azul).

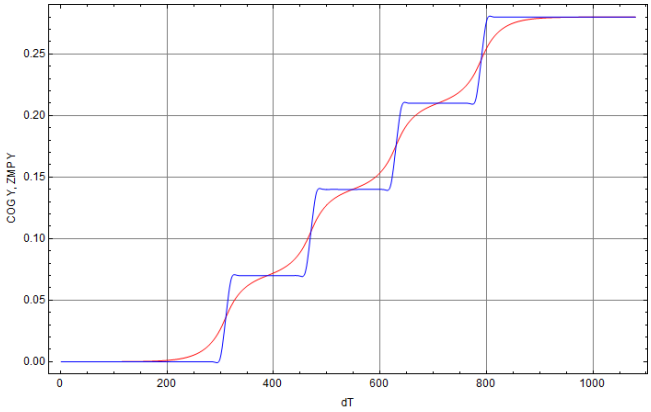

Figura 4.21: Trayectoria del COG (roja) generada mediante control por seguimiento de trayectorias para controlar el ZMP de salida en y(azul).

En cuanto a las trayectorias del avance de cada pie, generadas mediante interpolación segmentaria cuadrática, en las figuras 4.22 y 4.23 se muestra el avance de cada uno en Z, mientras que en la figura 4.24 se muestra la trayectoria de cada uno en Y .

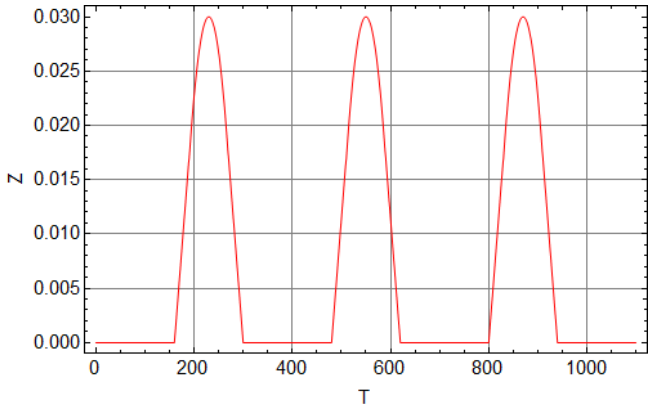

Figura 4.22: Trayectorias del pie derecho en Z.

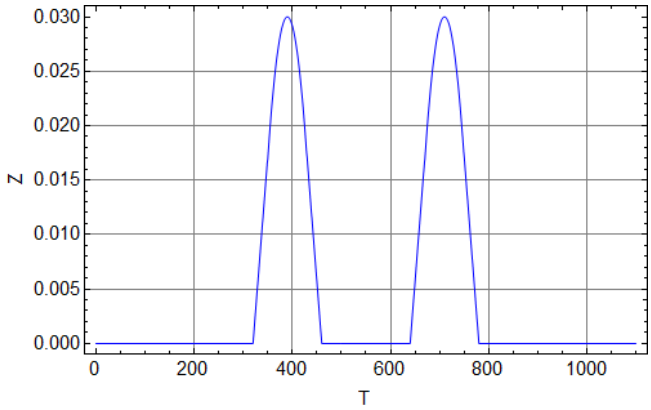

Figura 4.23: Trayectorias del pie izquierdo en Z.

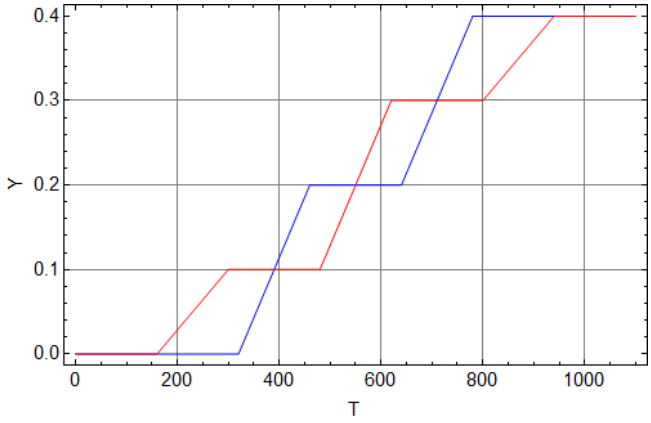

Figura 4.24: Trayectorias del avance de cada pie en Y. Trayectoria roja para el pie derecho y azul para el pie izquierdo.

Ajuste de parámetros

Actualmente el termino *jerk* se define como el cambio de la aceleración de un cuerpo con respecto al tiempo, es decir $jerk = \dddot{x}$. Puesto que la aceleración de un cuerpo produce una fuerza $F = ma$, el *jerk* se puede interpretar como la variación con respecto al tiempo de la fuerza aplicada a un cuerpo. Ahora bien, la aceleración del centro de gravedad juega un papel muy importante en el equilibrio dinámico del mecanismo cuando se emplea el modelo carro-mesa, por lo que no es conveniente tener variaciones demasiado pronunciadas en la aceleración del centro de gravedad, lo cual se puede lograr optimizando el *jerk*. En el caso del modelo carro-mesa, el *jerk* se puede optimizar aplicando control óptimo ya que $u(k) = \dddot{x}$.

Puesto que se desea optimizar el *jerk* del centro de gravedad y que también la trayectoria del ZMP generada mediante el controlador se aproxime lo máximo posible a la trayectoria de referencia, lo primero que se hizo fue verificar como influyen los pesos del error de seguimiento y del controlador en el índice de desempeño J. Por lo tanto, se procedió a dar un mayor peso al controlador, con respecto a los valores propuestos en [11] y empleados

en la simulación inicial, definiendo así:

$$Q = \begin{bmatrix} Q_e & 0 & 0 \\ 0 & Q_x & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 \times 10^{-3} \end{bmatrix},$$

para cuyos valores se obtuvieron en promedio los siguientes errores relativos del ZMP generado con respecto al ZMP deseado:

$$e_x = 6{,}398\,\%$$
$$e_y = 3{,}429\,\%$$

Es congruente que el error, tanto en x como en y, aumentará con respecto a los calculados anteriormente, ya que al incrementar la prioridad del controlador de cierta forma se le está quitando prioridad al error de seguimiento, por lo tanto se definieron los siguientes pesos:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 \times 10^{-8} \end{bmatrix}$$

obteniendo en promedio los siguientes errores relativos:

$$e_x = 0{,}12\,\%$$
$$e_y = 0{,}03\,\%$$

Con estos pesos se obtuvo una aproximación bastante aceptable, sin embargo la prioridad del controlador con respecto a la del error de seguimiento es muy baja, por lo tanto se eligieron de forma definitiva los siguientes pesos:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 \times 10^{-5} \end{bmatrix}$$

ya que la prioridad del controlador es más alta y también se obtiene una buena aproximación (error promedio de seguimiento aceptable):

$$e_x = 1{,}110\,\%$$
$$e_y = 0{,}487\,\%$$

Una vez que se verificó el funcionamiento de los programas de los métodos para la solución del modelo carro-mesa, se procedió a verificar que ninguna variable coincidiera con el nombre de alguna variable definida en el programa que resuelve la cinemática inversa [19], también elaborado en *Mathematica*, con el propósito de insertar las secciones de cualquiera de los programas que resuelven el modelo carro-mesa en éste, y de tener todo en un solo programa sin la necesidad de importar datos como se venía haciendo. Es por

este motivo que el nombre de algunas variables empleadas en los programas que resuelven el modelo carro-mesa no coincide con el nombre empleado en la teoría de la solución de cada método, por ejemplo, el caso de las constantes del controlador predictivo.

También se realizaron modificaciones al programa original que resuelve la cinemática inversa, ya que en éste se considera que la posición inicial (altura y posición del COG) a partir de la cual se resuelve la cinemática inversa es constante, pero ya que en los programas realizados para resolver el modelo carro-mesa se puede modificar la altura y la posición inicial del COG con respecto al centro de la planta de los pies, se tuvo que modificar esta parte. Por otra parte, también se modificaron las variables empleadas en la solución de la cinemática inversa, ya que la trayectoria del centro de gravedad se encontraba restringida para que sólo se desplazara sobre el plano sagital (a lo largo del eje y) y para que en el plano frontal se mantuviera constante y centrada ($x = 0$), pero de acuerdo con el capítulo anterior, la solución del COG con el modelo carro-mesa tiene componentes tanto en x como en y.

4.3.4. Comparación

Para realizar la comparación entre los dos métodos empleados para la solución del modelo carro-mesa, se consideró la precisión de cada método y el cómputo empleado en la resolución de estos. Puesto que ambos métodos fueron programados en *mathematica*, se empleó la función *AbsoluteTiming* para medir el tiempo empleado por la computadora en evaluar cada método. Cabe mencionar que no siempre se obtiene el mismo resultado, en cuanto al tiempo, puesto que la computadora siempre se encuentra empleando diferentes recursos computacionales en diferentes procesos; por lo que el resultado depende del procesador de la computadora y de sus operaciones.

En el caso de la solución por seguimiento de trayectorias para definir el sistema continuo, discretizarlo, definir las matrices necesarias para resolver la ecuación de Riccati, calcular las constantes de controlador, la ganancia predictiva de 270 elementos y obtener mediante el controlador el COG correspondiente a 1080 puntos discretos de la trayectoria deseada del ZMP, se emplearon en promedio 1,4755$[s]$. Aunque cabe mencionar que no es necesario realizar todo el proceso del cálculo de las constantes del controlador y de la ganancia predictiva cada que se obtiene la trayectoria del COG, es decir que una vez que se conocen estas se puede modificar la trayectoria de referencia del ZMP y calcular, con la misma ganancia predictiva y con las mismas constantes, la trayectoria correspondiente del COG, con lo que se ahorran recursos computacionales.

En el caso de la solución por discretización de la aceleración, para la misma trayectoria deseada del ZMP, discretizada con 1080 puntos, se emplearon en la generación de la matriz de coeficientes, en el cálculo de la matriz inversa y en el algoritmo de la multiplicación de matrices mediante el cual se obtiene el COG correspondiente: 4,3439$[s]$ en promedio. Es decir que con esta solución se emplean exactamente 2.944 veces el cómputo que se emplea en la solución por seguimiento de trayectorias, pero es importante recordar que esta última es una aproximación en comparación con la solución por discretización de aceleración, la cual es una solución exacta.

En la figura 4.25 se muestra la carga computacional empleada para resolver el modelo carro-mesa tanto para el método de solución por discretización de la aceleración (trayectoria azul) como para el método de solución por seguimiento de trayectorias (trayectoria roja). En el eje de las abscisas se muestra el número de elementos discretos, mientras que en el eje de las ordenadas el computo estimado para resolver el modelo carro-mesa para el número de elementos discretos correspondiente.

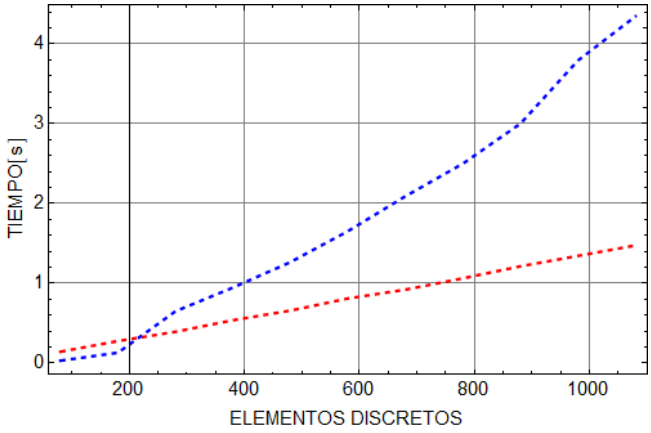

Figura 4.25: Carga computacional empleada para resolver el modelo carro-mesa, trayectoria roja para el método de solución por seguimiento de trayectorias, y azul para el método de solución por discretización de la aceleración.

Empleando una computadora con diferentes características y la misma trayectoria deseada del ZMP discretizada con 1080 elementos, se obtuvo que para la solución por seguimiento de trayectorias se emplean en promedio 1,853$[s]$, y que con la solución por discretización de la aceleración 5,4547$[s]$, obteniendo así que este último método de solución emplea exactamente 2.9437 veces el cómputo empleado en la solución por seguimiento de trayectorias. Con esto se resalta que aunque la carga computacional para resolver cada método difiere de un equipo a otro, la razón de proporcionalidad entre cómputo empleado se mantiene aproximadamente constante.

Otro factor que hay que tomar en consideración es la complejidad de cada método, ya que la solución por discretización de la aceleración es una solución relativamente sencilla que termina siendo una expresión algebraica, en contraste con la solución por seguimiento de trayectorias, la cual involucra conceptos de control óptimo y de control predictivo, inclusive se tiene que discretizar el sistema dinámico y resolver la ecuación de Riccati, lo que representa un gran problema ya que no todos los programas tienen programada la función para resolverla, además se tienen que proporcionar, en el índice de desempeño J, los factores de peso del error y de la energía empleada por el controlador para así poder comparar los resultados obtenidos con los deseados y seleccionar los pesos adecuados, ya que en la mayoría de los casos no existe o no se ha establecido un criterio para su selección.

Por otra parte, es importante resaltar que una vez seleccionada la longitud adecuada de la ventana de valores futuros del ZMP deseado a emplear en el controlador, la exactitud de la solución por seguimiento de trayectorias depende de los pesos seleccionados para definir el índice de desempeño J.

4.3.5. Resultados

Cabe destacar que en [11] se propone la solución por seguimiento de trayectorias utilizando control predictivo, pero sólo se presenta la ecuación del índice de desempeño y la ecuación del controlador, ecuación 4.12, sin definir lo que significa el índice de desempeño empleado o de donde salió, ni a que son igual G_I, G_x y G_d, lo que no sirve de algo si se desea emplear esta solución. Los fundamentos para desarrollar un controlador óptimo para un sistema discreto sujeto a una demanda predictiva se presentan en [43], con base en el cual se definieron las matrices $\tilde{I}$, $\tilde{A}$, $\tilde{B}$ y $\tilde{F}$ para el sistema dinámico del modelo carro-mesa considerando una entrada y una salida, ya que en [43] se considera un sistema de múltiples entradas y múltiples salidas. De esta forma fue posible comprender en qué consiste el controlador, lo que significa cada término y definir G_I, G_x y G_d para poder calcular el controlador óptimo. Además se comprendió el efecto de la ganancia predictiva sobre el controlador con el propósito de determinar la longitud adecuada de la ventana de valores futuros del ZMP a emplear, para así evitar realizar operaciones de sobra calculando ganancias innecesarias, multiplicándolas por su respectivo ZMP deseado, y también para evitar emplear pocos valores y no lograr obtener una buena aproximación del ZMP generado.

Por otra parte, se generaron dos programas debidamente documentados (apéndices A y B) con los cuales sólo es necesario especificar los parámetros mediante los cuales los programas generaran trayectorias dinámicamente estables, con base en el modelo simplificado carro-mesa, para cualquier robot bípedo; sólo es necesario que se definan, en la primera sección de cada programa, la longitud y amplitud de paso adecuadas para que la trayectoria deseada del ZMP, generada por el programa, se mantenga dentro del polígono de soporte el cual es definido por los parámetros físicos de los pies de cada robot. Aunque la trayectoria (5 pasos hacia adelante) y la relación entre los periodos de soporte doble y de soporte simple se mantienen restringidas por los programas.

Por lo tanto, entendiendo en qué consiste cada programa, se pueden generar trayectorias dinámicamente estables por medio del modelo carro-mesa para cualquier robot bípedo sin la necesidad de involucrarse a fondo con la base teórica de este modelo simplificado y tampoco con la de los métodos de solución. También se hace la comparación de las ventajas y desventajas de los dos métodos de solución empleados en este trabajo, para así contar con un criterio más amplio a la hora de seleccionar que método emplear.

Capítulo 5

Simulación y experimentación con el robot Scout

5.1. Simulación del ciclo de marcha

Las trayectorias generadas para los efectores finales, pies y COG, sirvieron como referencia para proceder a resolver la cinemática inversa y puesto que en $Mathematica$ se muestra algún error cuando se detecta alguna singularidad al momento de resolver la cinemática inversa, esto se empleó como una delimitante para seleccionar los parámetros del ciclo de marcha adecuados.

Para evitar singularidades se encontró, a prueba y a error, la siguiente relación entre la altura del COG y la longitud de paso (LP):

$$\begin{aligned} 17[cm] \leq altura \leq 19[cm] &\rightarrow 5[cm] \leq LP \leq 9[cm] \\ 19[cm] \leq altura \leq 21[cm] &\rightarrow 5[cm] \leq LP \leq 8[cm] \\ 21[cm] \leq altura \leq 23{,}5[cm] &\rightarrow LP \equiv 5[cm] \end{aligned}$$

Para la altura del paso se emplearon indistintamente $3[cm]$ y $5[cm]$, sin tener mayor problema en cuanto a singularidades.

Una vez que se obtuvieron las secuencias de los 12 ángulos, mediante la solución de la cinemática inversa de cada una de las posiciones que debe adoptar el mecanismo del robot bípedo para seguir las trayectorias de sus efectores finales, y que se verificó que no existieran singularidades se procedió a verificar que los eslabones del modelo real no chocaran entre ellos mediante la simulación del modelo virtual en $LabVIEW$, para finalmente acoplar los ángulos obtenidos de la cinemática inversa con los ángulos de los servomotores correspondientes en el modelo real, para lo cual fue necesario exportarlos y modificarlos mediante Excel. También se verificó en Excel que los ángulos ajustados al modelo real no fueran negativos o mayores a 180°.

En la figura 5.1 se muestran algunos resultados obtenidos, mediante la simulación en $Mathematica$ del avance del robot, una vez que se resolvió la cinemática inversa.

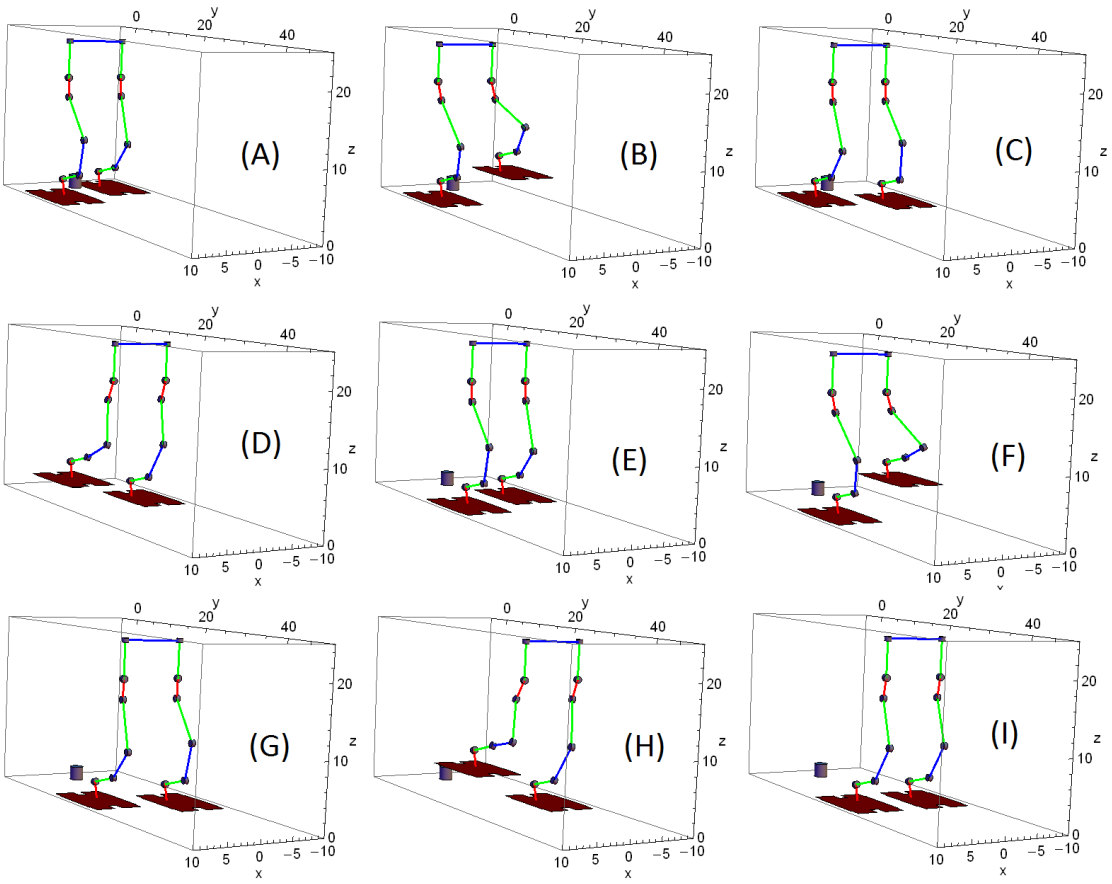

Figura 5.1: Simulación del ciclo de marcha con *Mathematica*.

5.2. Simulación del ZMP considerando la dinámica total del robot

De acuerdo con [30] para obtener la trayectoria del ZMP que se genera a partir de la locomoción de un robot bípedo, considerando que el robot está conformado por N eslabones rígidos y tridimensionales en lugar del modelo simplificado carro-mesa, es necesario calcular no solamente la posición de los efectores finales (pies y centro de la cadera), sino también la velocidad y aceleración. También se debe calcular la velocidad y aceleración angular de cada articulación, así como el centro de masa total del robot, considerando la masa y posición del COM de cada eslabón (figura 5.2) de la siguiente manera:

$$M = \sum_{j=1}^{N} m_j,$$

$$c = \sum_{j=1}^{N} m_j c_j / M,$$

donde m_j y c_j son la masa y el COM del eslabon j, respectivamente.

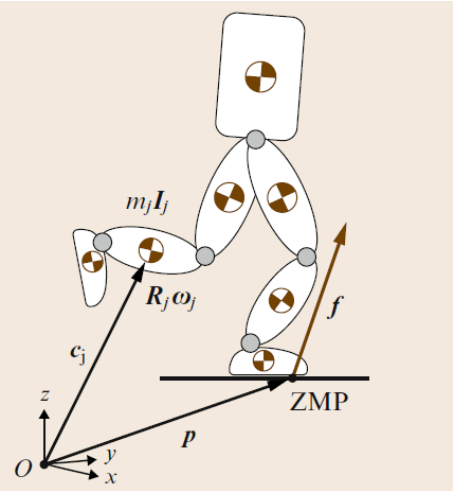

Figura 5.2: Cálculo del ZMP a partir del modelo tridimensional del robot [30].

El momento lineal total está dado por:

$$P = \sum_{j=1}^{N} m_j \dot{c_j} \tag{5.1}$$

Por lo tanto, el momento angular total L con respecto al origen se define como:

$$L = \sum_{j=1}^{N} [c_j \times (m_j \dot{c_j}) + R_j I_j R_j^T \omega_j], \tag{5.2}$$

donde R_j, I_j y ω_j son la matriz de rotación de 3×3, el tensor de inercia y la velocidad angular del eslabón j, respectivamente. $R_j I_j R_j^T$ da como resultado el tensor de inercia con respecto al sistema de referencia global.

De acuerdo con [30], el ZMP real se puede calcular a partir de (5.1) y (5.2), considerando que:

$$P = [P_x, P_y, P_z]^T,$$
$$L = [L_x, L_y, L_z]^T,$$
$$c = [x, y, z]^T,$$

como:

$$p_x = \frac{Mgx + p_z \dot{P}_x - \dot{L}_y}{Mg + \dot{P}_z}, \tag{5.3}$$

$$p_y = \frac{Mgy + p_z \dot{P}_y - \dot{L}_x}{Mg + \dot{P}_z}, \tag{5.4}$$

donde p_z es la altura del piso con respecto al marco de referencia.

Cuando el robot permanece en estado estacionario, el ZMP resultante coincide con la proyección del COM sobre el piso. Para definir P y L es necesario contar con la posición,

velocidad y aceleración del COM de cada eslabón, para lo cual es necesario contar con los vectores de posición del centro de masa de cada eslabón y con la solución de la velocidad y aceleración de cada articulación para así poder determinar la velocidad y aceleración del COM de cada eslabón a partir de la primera y segunda derivada de estos vectores. En el caso del COM del eslabón de la cadera se tomaron los resultados de la solución por seguimiento de trayectorias, ya que las variables de estado son x, $\dot{x}$ y $\ddot{x}$ del COM ideal, el cual coincide con el COM del eslabón de la cadera.

En las figuras 5.3, 5.4 y 5.5 se muestra en color azul el ZMP real calculado a partir de las ecuaciones (5.3) y (5.4) y considerando la posición, velocidad y aceleración del COM ideal del robot, las cuales se generan a partir del ZMP de referencia (de color rojo en las figuras 5.3, 5.4 y 5.5) y del modelo simplificado carro-mesa.

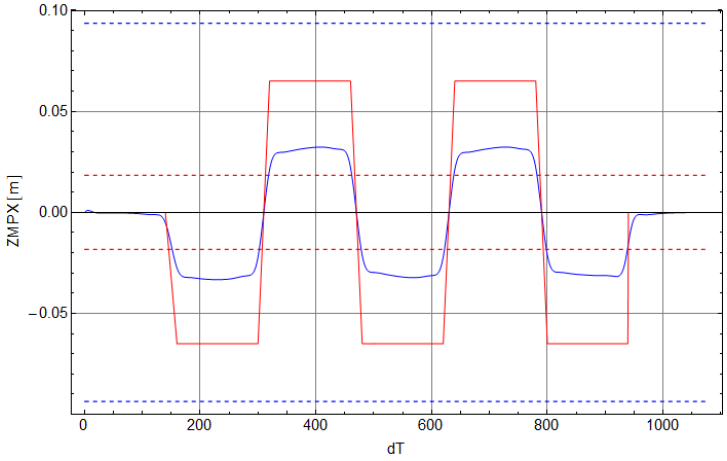

Figura 5.3: ZMP resultante en x (azul) considerando la dinámica total del robot y los resultados de su COM ideal, obtenidos del ZMP propuesto (rojo) y el modelo carro-mesa.

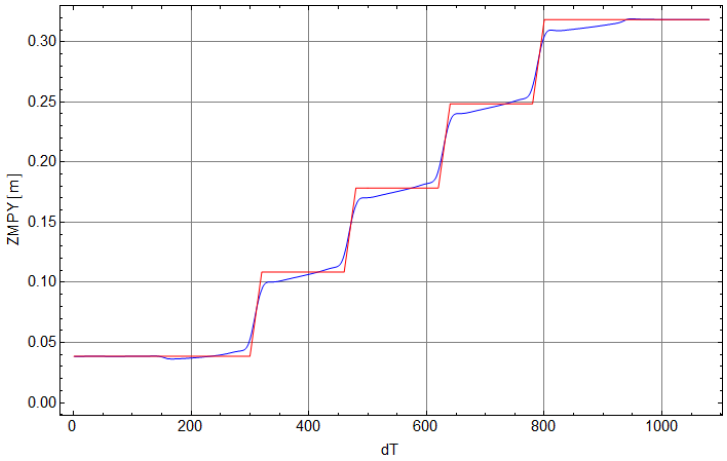

Figura 5.4: ZMP resultante en y (azul) considerando la dinámica total del robot y los resultados de su COM ideal, obtenidos del ZMP propuesto (rojo) y el modelo carro-mesa.

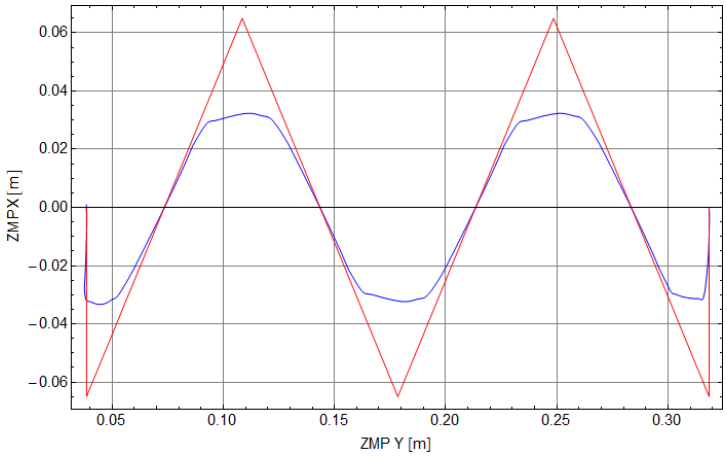

Figura 5.5: ZMP resultante en x y en y (azul) considerando la dinámica total del robot y los resultados de su COM ideal, obtenidos del ZMP propuesto (rojo) y el modelo carro-mesa.

En estas figuras (5.3, 5.4 y 5.5) se aprecia una gran diferencia entre el ZMP correspondiente al modelo carro-mesa y el ZMP calculado a partir de la dinámica total del robot, lo que se debe en parte a la diferencia entre la dinámica total del robot y la dinámica del modelo carro-mesa. Sin embargo el ZMP calculado a partir de la dinámica total del robot se mantiene dentro del polígono de soporte delimitado por la planta de los pies, por ejemplo, en la figura 5.3 las 2 líneas discontinuas rojas representan los límites interiores del polígono de soporte simple de cada pie, mientras que las 2 líneas discontinuas azules representan los límites exteriores y se aprecia que el ZMP calculado a partir de la dinámica total (trayectoria continua azul) nunca se encuentra fuera de estos límites en los periodos de soporte simple.

Para reducir el error entre el ZMP calculado a partir de la dinámica total del robot y el ZMP correspondiente al modelo carro-mesa, se calcula la diferencia entre ambos y se aplica como compensación en el término integrativo y en la ponderación del N número de valores futuros del ZMP deseado que se empleara en el controlador de la solución por seguimiento de trayectorias. En las figuras 5.6, 5.7 y 5.8 se aprecian los resultados correspondientes a la compensación calculada para el ZMP de las figuras 5.3, 5.4 y 5.5.

Los resultados de estas figuras muestran claramente que existe una mejor aproximación del ZMP calculado a partir de la dinámica total del robot al ZMP correspondiente al modelo carro-mesa, aunque al calcular la dinámica total del robot para determinar la compensación adecuada se consumen demasiados recursos computacionales, y lo que se ahorra en tiempo con el modelo carro-mesa ya no es significativo. Por lo tanto, una vez calculada la compensación adecuada, se puede guardar ésta en memoria y aplicar a otras trayectorias que tengan periodos de soporte iguales, evitando así la necesidad de calcular nuevamente la dinámica total del robot para determinar la composición adecuada. De esta manera se puede obtener una buena aproximación a la dinámica total del robot y generar trayectorias dinámicamente estables a partir del modelo simplificado carro-mesa, empleando recursos computacionales realmente mínimos comparados con los que se emplearían si se utilizara el modelo multicuerpo.

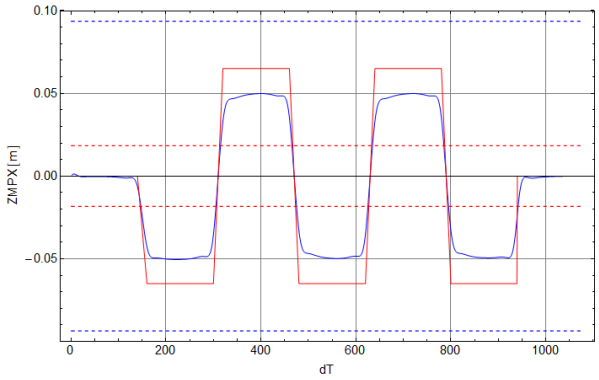

Figura 5.6: ZMP resultante en x (azul) considerando la dinámica total del robot y los resultados de su COM ideal, obtenidos del modelo carro-mesa y la compensación del ZMP de la figura 5.3.

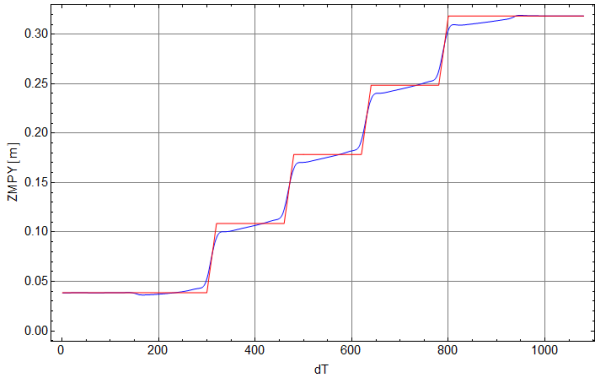

Figura 5.7: ZMP resultante en y (azul) considerando la dinámica total del robot y los resultados de su COM ideal, obtenidos del modelo carro-mesa y la compensación del ZMP de la figura 5.4.

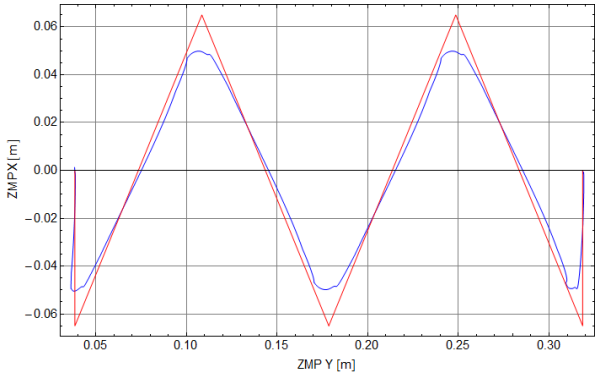

Figura 5.8: ZMP resultante en x y en y (azul) considerando la dinámica total del robot y los resultados de su COM ideal, obtenidos del modelo carro-mesa y la compensación del ZMP de la figura 5.5.

En la figura 5.9 se aprecia el resultado de aplicar la compensación calculada para la trayectoria del ZMP de la figura 5.5 a una trayectoria de ZMP con una amplitud de paso menor. Lo importante aquí es destacar que se obtuvo una buena aproximación, del ZMP real al correspondiente al modelo carro-mesa, sin la necesidad de calcular la compensación correspondiente, en la figura 5.10 se muestra el resultado sin la compensación.

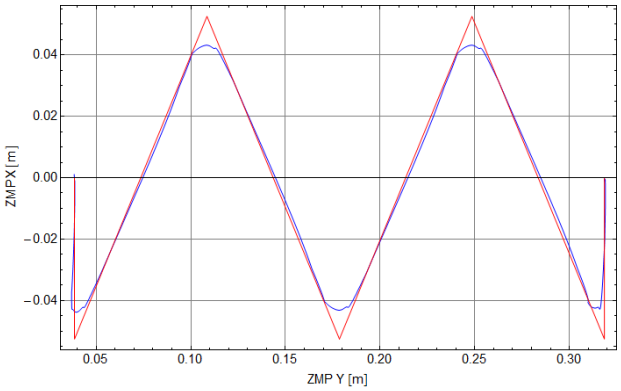

Figura 5.9: ZMP resultante (trayectoria azul) del cálculo de la dinámica total del robot, que se hace a partir del COM obtenido del modelo carro-mesa considerando la compensación del ZMP de la figura 5.5.

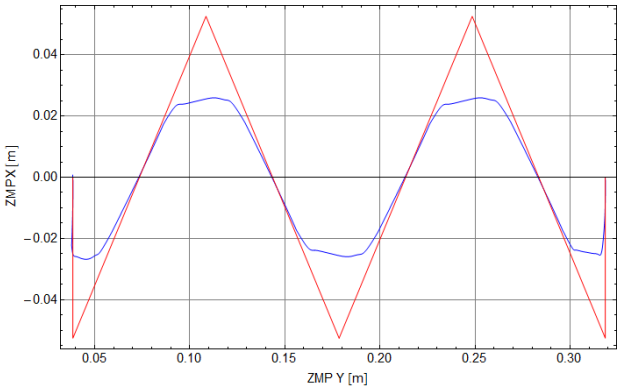

Figura 5.10: ZMP resultante (trayectoria azul) del cálculo de la dinámica total del robot, que se hace a partir del COM obtenido del modelo carro-mesa sin considerar compensación aluna del ZMP.

En resumen, para poder calcular el ZMP total en x y en y es necesario que se calcule la primera derivada del momento lineal total y del momento angular total, para lo cual es necesario calcular, para cada una de las configuraciones del mecanismo del robot que definen el ciclo de marcha, los vectores de posición del centro de gravedad de cada eslabón, el centro de gravedad del robot y la primera y segunda derivada de los vectores de posición del centro de gravedad de cada eslabón a partir de los modelos de cinemática y dinámica espacial desarrollados en [19]. Para calcular el ZMP real de la figura 5.5 se emplearon aproximadamente 15 minutos, sin tomar en cuenta que para hacer los cálculos anteriores es necesario contar con la solución de la posición, velocidad y aceleración, para lo que se empleó en promedio 1, 2 y 10 minutos respectivamente.

Para generar el ZMP a partir del modelo carro-mesa y de la matriz de coeficientes empleada en la solución por discretización de la aceleración, subsección 4.3.1, pero realizando ahora el proceso inverso, es decir calculando el ZMP a partir de la trayectoria del COM del robot utilizando la ecuación (5.5), se emplearon en promedio 1 minuto con 20 segundos, por lo que es importante emplear un modelo simplificado ya que el modelo de la dinámica total consume demasiados recursos computacionales.

$$p_x = Ax, \tag{5.5}$$

donde:

$$p_x = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{N-1} \\ p_N \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} \quad y \quad A = \begin{bmatrix} a_1 + b_1 & c_1 & 0 & & & & \\ a_2 & b_2 & c_2 & \ddots & & & \\ & & & \ddots & & & \\ & & & \ddots & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & 0 & a_N & b_N + c_N \end{bmatrix}$$

5.3. Interfaz Gráfica de Usuario y Aplicación Móvil

En cuanto a la simulación con $LabVIEW$, en un principio se tuvieron algunos problemas ya que la interfaz gráfica emplea su propio sistema de referencia para generar los movimientos de los eslabones del modelo virtual, de lo cual no se tenía información y por tal motivo se obtuvieron incongruencias en las primeras simulaciones realizadas. También se detectó que el archivo que se carga en la interfaz gráfica, con la secuencia de los ángulos de los 12 servomotores debidamente ordenados en 12 columnas, debía contener 2 columnas más (en total 14 columnas) de las cuales no se conocía el propósito.

Para poder solucionar este problema se comenzó por manipular el modelo virtual manualmente, es decir sin simular el ciclo de marcha a partir de los ángulos obtenidos mediante cinemática inversa. Lo que se hizo fue encontrar en el modelo virtual los ángulos de cada servomotor que corresponden a la posición inicial a partir de la cual se resuelve la cinemática inversa, así como los sentidos de giro de cada servomotor. La figura 3.2 muestra el sistema de referencia y la posición inicial mediante la cual se resuelve la cinemática inversa, cabe destacar que un ángulo de cero obtenido mediante cinemática inversa para las 12 juntas rotacionales del robot corresponde a ésta posición. A continuación se ajustaron los ángulos obtenidos mediante cinemática inversa a los ángulos del sistema de referencia del modelo virtual tomando en consideración que algunas juntas rotacionales del modelo virtual no giran en el mismo sentido que las juntas del sistema de referencia mostrado en la figura 3.2.

Posteriormente se determinó el propósito de las dos columnas extras que deben ser incluidas en el archivo de los ángulos que definen el ciclo de marcha. Para esto se inspeccionó el código en el diagrama de bloques de *LabVIEW*, lo cual no fue sencillo ya que el programa principal emplea 25 programas (SubVI´s) para su funcionamiento. De esta forma y prestando atención a que se construyeron dos árboles jerárquicos para indicar quien es el pie de apoyo en la simulación del bípedo [33], se dedujo que la primera columna extra sirve para indicar quien es el pie de soporte simple para así mantenerlo fijo a la referencia (piso)

mientras que la otra pierna adopta los ángulos de forma libre. En cuanto a la segunda columna, se detectó que su propósito es indicar si se desea repetir la simulación, en qué punto repetirla, hasta qué punto o posición regresar cuando se repite y cuando debe parar. Durante éste proceso también se detectó que se suman o restan algunas constantes a cada uno de los ángulos que son extraídos del archivo, las cuales corresponden al ajuste entre los ángulos obtenidos de la cinemática inversa y la caracterización de los servomotores anterior.

Por lo tanto, una vez que se ajustaron los ángulos obtenidos de la cinemática inversa al sistema de referencia del prototipo virtual, se procedió a sumar y a restar (sumar en caso de resta y restar en caso de suma) las constantes que se suman y restan en la simulación. Como resultado se logró simular el ciclo de marcha en el prototipo virtual y verificar que ningún eslabón chocara con otro, también se realizó una plantilla en Excel para ingresar los ángulos obtenidos de la cinemática inversa y obtener los ángulos ajustados a la simulación.

Cabe mencionar que el procedimiento anterior llevo mucho tiempo y como no se sabía si la parte de la simulación afectaba a la parte del envió de los datos o si se lograría hacer funcionar a la interfaz gráfica, se optó por explorar otras alternativas al mismo tiempo que se trabajaba con la puesta en marcha del robot bípedo, con la interfaz gráfica y con la generación de las trayectorias dinámicamente estables mediante el modelo carro-mesa.

Aplicación Móvil

Como alternativa a la interfaz gráfica en *LabVIEW* se decidió emplear una solución novedosa e intuitiva que sólo enviará los datos al banco de pruebas, por lo que se desarrolló una aplicación móvil y se emplearon algunas de las herramientas que ofrecen los dispositivos móviles, como los sensores. Para enviar los ángulos, adecuados a los servomotores del modelo físico, la aplicación móvil ingresa a los ángulos guardados en un archivo de texto, ubicado en la memoria interna del celular o en una memoria externa SD, para leerlos y enviarlos vía *BlueTooth* a la placa de desarrollo *Arduino Mega*.

Por otra parte, se trató de que al menos la comunicación con el robot fuera inalámbrica, lo cual se logró empleando un módulo *BlueTooth* e instalando la placa de desarrollo *Arduino Mega* sobre el robot. Como resultado se logró que ahora el robot sólo ocupe el cable de la alimentación para los servomotores, eliminando así el cable de la comunicación serial entre la computadora y la placa *Arduino Mega*, y el cable entre la placa *Arduino Mega* y los 12 servomotores.

Por lo tanto, ahora que la comunicación es inalámbrica, es más fácil manipular el banco de pruebas ya que el robot se tiene que desplazar y en ocasiones se ve limitado el espacio de trabajo empleando cables, mientras menos cables mejor, además de que la aplicación móvil proporciona una mayor movilidad a quien está manipulando el robot.

En la figura 5.11 se muestra un esquemático de la aplicación móvil desarrollada en *Android*, la cual cuenta con dos actividades que se muestran en esta figura (a cada actividad le corresponde una pantalla en primer plano). La actividad inicial (izquierda) interactúa con el usuario mostrando un mensaje donde se debe ingresar el número de la trayectoria a ser ejecutada por el robot, se toma en cuenta la posibilidad de que el usuario no use de forma correcta la aplicación, para lo cual se muestran mensajes guía y se restringe el

ingreso de datos, por ejemplo, no permitiendo que el usuario ingrese caracteres donde se deben ingresar números.

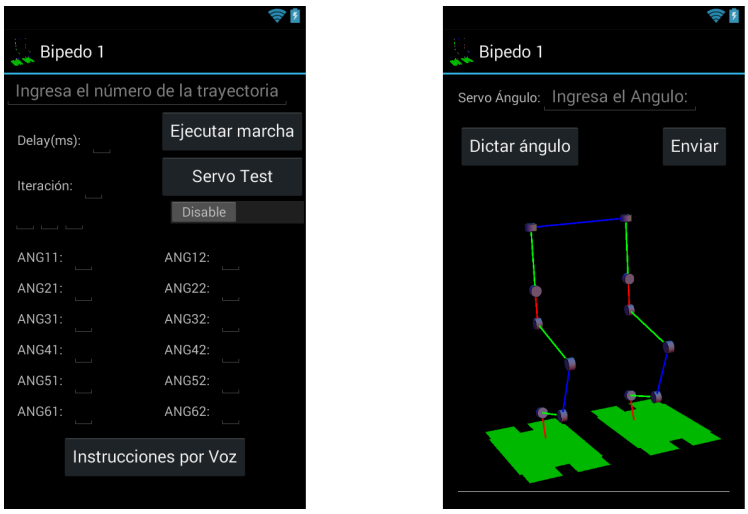

Figura 5.11: Aplicación móvil desarrollada en *Android* para controlar el banco de pruebas.

Mediante la aplicación también se ingresa, en milisegundos, el tiempo deseado para el retardo en el envío de los ángulos, además de que se cuenta con indicadores que muestran en todo momento el número de la solución a la que corresponde la postura actual del robot, así como sus respectivos ángulos divididos en dos columnas, una por cada pierna, siguiendo la nomenclatura descrita en la sección 3.2.

Por otra parte, también se cuenta con un Switch (barra deslizante) mediante el cual se activa y desactiva el robot, de esta forma es posible desactivar el robot durante el ciclo de marcha, verificar los ángulos correspondientes a una determinada posición del ciclo de marcha y reanudar el ciclo de marcha activándolo nuevamente. Mediante el botón etiquetado como *Servo Test* se ingresa a la siguiente actividad (pantalla de la derecha, figura 5.11) la cual tiene la misma función que el programa desarrollado en *LabVIEW* para poner en marcha el robot, sección 3.4. La finalidad es que se pueda probar el funcionamiento de los servomotores y verificar su rango de funcionamiento, por separado, cuando se detecte alguna falla. En este caso se puede ingresar el ángulo deseado y presionar el botón con la etiqueta *Enviar* o se puede presionar el botón etiquetado como *Dictar ángulo* e ingresar el ángulo mediante voz para que una vez que la aplicación reconozca el ángulo dictado, lo envíe al microcontrolador.

En cuanto a los sensores, se emplearon el acelerómetro y el sensor de proximidad sólo para activar y desactivar el avance del robot con el propósito de mostrar la integración de algunas herramientas que ofrecen los dispositivos móviles para que en el futuro sean consideradas a lo hora de hacer mejoras al proyecto. Para obtener las lecturas del acelerómetro se empleó un filtro paso bajo propuesto en la página de *Android*, la cual contiene la documentación acerca de los sensores [44]. El sistema de referencia de los sensores de posición en los dispositivos móviles *Android* se muestra en la figura 5.12.

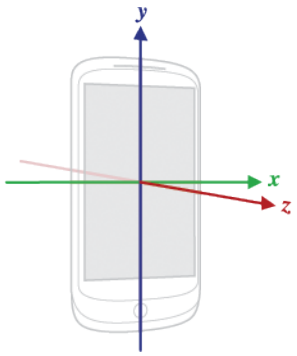

Figura 5.12: Sistema de referencia de los sensores de posición en los dispositivos móviles *Android* [45].

Como el acelerómetro detecta la aceleración de la gravedad, el filtro paso bajos empleado consiste en aislar ésta para poder detectar sus componentes en x, y y z. Para integrar esto a la aplicación con la que se controla el banco de pruebas se restringió la componente de la aceleración sobre el eje y a permanecer menor o igual a 0 $[m/s^2]$, de acuerdo con el sistema de referencia mostrado en la figura 5.12, para que el robot pudiera avanzar. Para la posición del celular mostrada en la figura 5.12, la aceleración en $x = 0$, en $y = 9{,}8$ y en $z = 0$ $[m/s^2]$.

5.4. Pruebas y Resultados

Las pruebas con la aplicación móvil consistieron principalmente en corroborar los resultados obtenidos con la interfaz gráfica en *LabVIEW* . La ventaja de la aplicación móvil fue el control sobre el retardo en el envío de los datos al prototipo físico, ya que con la interfaz en *LabVIEW* no se tenía control total sobre éste. En la figura 5.13 se muestra la trayectoria inicial, con referencia a la planta de los pies del robot Scout, del ZMP con base en el cual se generó la primera trayectoria del COM en el capítulo anterior, subsección 4.3.3. Los puntos amarillos representan al ZMP en los periodos de soporte simple y al ZMP final, el punto verde representa el ZMP inicial así como el origen del sistema de referencia, tanto en x como en y y en z, con base en el cual se resuelve la cinemática inversa en [19]. La línea verde representa el ZMP en los periodos de soporte doble.

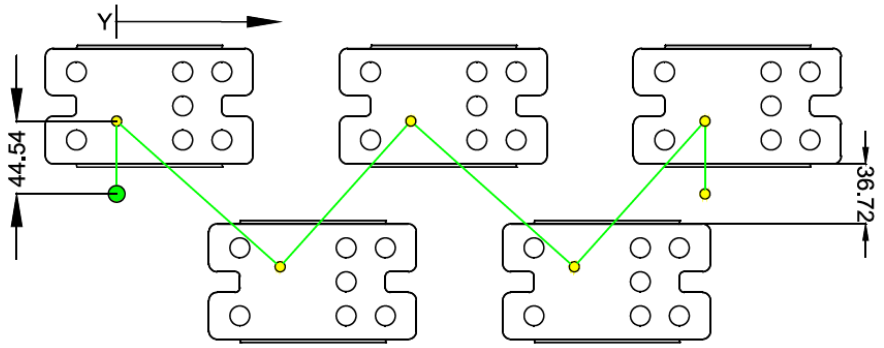

Figura 5.13: Referencia inicial del ZMP deseado, medidas en milímetros.

Como la trayectoria del centro de gravedad que se genera, ya sea por discretización de la aceleración o por control predictivo, se mantiene muy próxima a la trayectoria del ZMP deseado (ver figura 4.15), la posición de éste último con respecto a la planta del pie influye demasiado en el equilibrio dinámico. Por lo que durante las pruebas realizadas con la primera trayectoria del ZMP deseado el robot no alzó lo suficiente las plantas de los pies en los periodos de soporte simple, los arrastraba y también mostraba una ligera tendencia a volcarse hacia atrás, más no lo hizo. Por lo tanto, se modificaron los parámetros iniciales que definen el ZMP en los programas de los apéndices A y B, para que el ZMP se mantuviera centrado en la planta de cada pie, con respecto al eje x, en los periodos de soporte simple y para que en el eje y avanzará conforme a la posición erguida inicial con base en la cual se resuelve la cinemática inversa, figura 5.14.

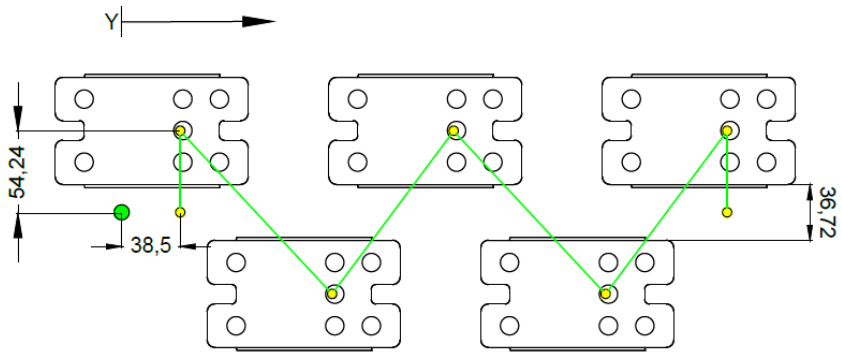

Figura 5.14: Referencia del ZMP deseado con respecto a la posición erguida con base en la cual se resuelve la cinemática inversa, ver figura 3.2. Medidas en milímetros.

Con el ZMP de referencia de la figura 5.14 se obtuvo un mejor movimiento de los pies, sin embargo la posición del ZMP con respecto a la planta de los pies hizo que el robot definitivamente perdiera el equilibrio y se volcara hacia enfrente durante las pruebas. Por lo tanto, se realizaron pruebas con diferentes posiciones relativas del ZMP con respecto a la planta de los pies para definir la zona de mayor seguridad en la cual el robot no se vuelque hacia los extremos durante el ciclo de marcha y arrastre lo menos posible los pies durante los periodos de soporte simple, figura 5.15.

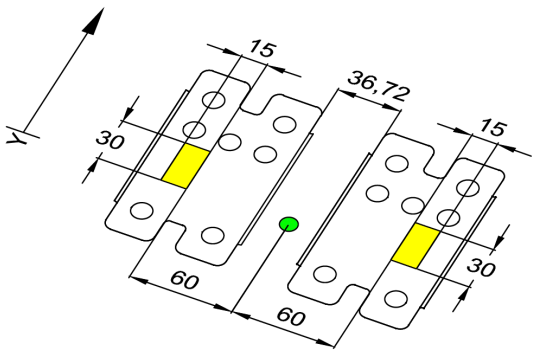

Figura 5.15: Área de mayor estabilidad (amarilla) donde debe mantenerse el ZMP para que el robot no se vuelque, medidas en milímetros.

En los programas de los apéndices A y B, que resuelven el modelo carro-mesa, con la variable off se modifica la posición relativa del ZMP en y y con la variable amp se modifica la posición relativa del ZMP en x, las dos con respecto al origen, para que con los valores adecuados de off y amp se pueda mantener la trayectoria del ZMP deseado dentro del área de mayor estabilidad de cada pie.

Modificación de los periodos de soporte

Por otra parte, se detectó que el periodo de soporte doble propuesto era demasiado corto ya que el robot realizaba el intercambio de pie de soporte simple demasiado rápido, provocando así que el robot no pudiera estabilizar bien la planta del siguiente pie de soporte simple al finalizar el periodo de soporte doble, por lo que no se podía desarrollar el ciclo de marcha como se había planificado. Por lo tanto, se realizaron las modificaciones correspondientes para que el periodo de soporte doble fuera del 22.22 % y el periodo de soporte simple del 77.78 %. Estos porcentajes resultaron porque para los periodos del ciclo de marcha anterior se emplearon 140 puntos discretos para definir el periodo de soporte simple y 20 para definir el de soporte doble, por lo que sólo se duplicaron los puntos del periodo de soporte doble para definir uno más extenso, resultando así los porcentajes de 22.22 % para el periodo de soporte doble y 77.78 % para el periodo soporte simple.

También se realizaron las modificaciones correspondientes para definir un ciclo de marcha de ya no sólo 5 pasos, sino ahora de 10 pasos hacia delante. En las figuras 5.16 y 5.17 se muestran los resultados obtenidos mediante seguimiento de trayectorias, la trayectoria azul representa al ZMP resultante y la trayectoria roja al COM generado.

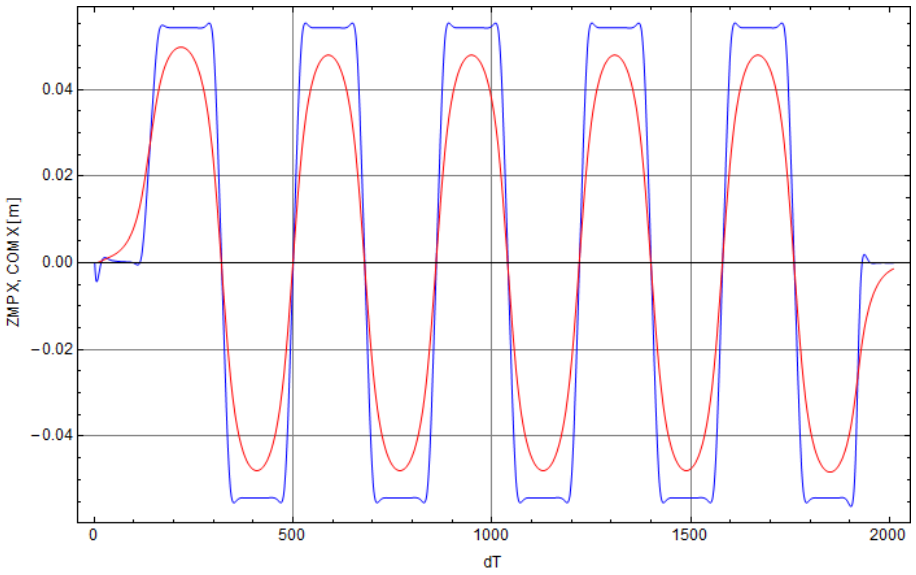

Figura 5.16: Trayectoria del COG (roja) generada mediante control por seguimiento de trayectorias para controlar el ZMP de salida en x(azul).

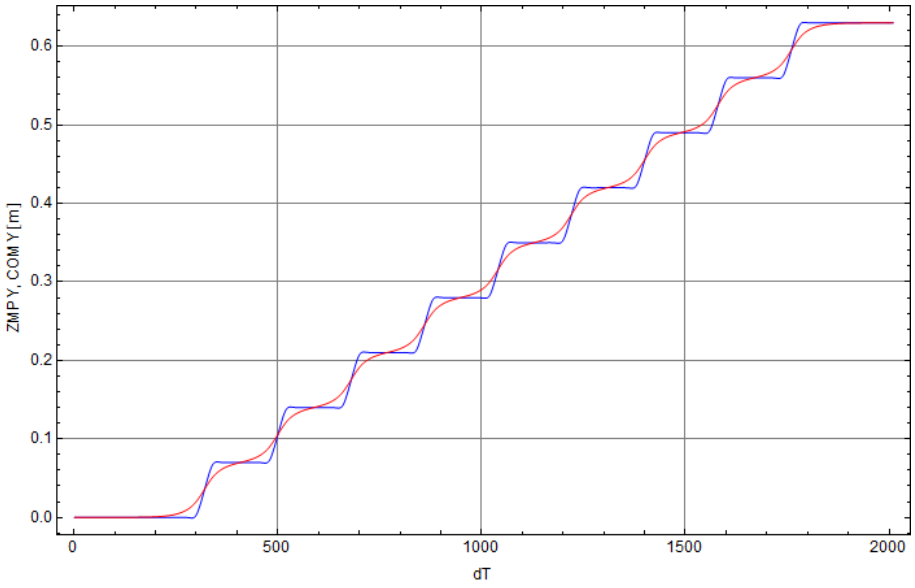

Figura 5.17: Trayectoria del COG (roja) generada mediante control por seguimiento de trayectorias para controlar el ZMP de salida en y(azul).

Los resultados de las figuras 5.16 y 5.17 se obtuvieron considerando la ganancia predictiva de 270 elementos de la ventana de valores futuros de la demanda solicitada y los pesos definitivos obtenidos en la sección 4.3.3:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} 1 \times 10^{-5} \end{bmatrix}$$

Con estos periodos se logró que el robot siguiera con mayor fidelidad las trayectorias propuestas, al poder estabilizar correctamente la planta del siguiente pie de soporte simple al final de cada periodo de soporte doble. En cuanto al cómputo empleado para solucionar cada método, para la solución por seguimiento de trayectorias se emplearon 2.657 [s] y para la solución por discretización de la aceleración 12.195 [s]. Es decir que para la solución por discretización de la aceleración en esta ocasión se emplearon 4.5 veces el computo empleado en la solución por seguimiento de trayectorias.

En las figuras 5.18 y 5.19 se muestran fragmentos de la implementación del ciclo de marcha dinámicamente estable con el prototipo físico, considerando diferentes perspectivas. Estas imágenes fueron extraídas de videos que se gravaron durante la experimentación, algunos de los cuales se pueden encontrar en los siguientes enlaces:

- `http://youtu.be/E_0dQhEmjXk`
- `http://youtu.be/HtLGCBT1DHo`

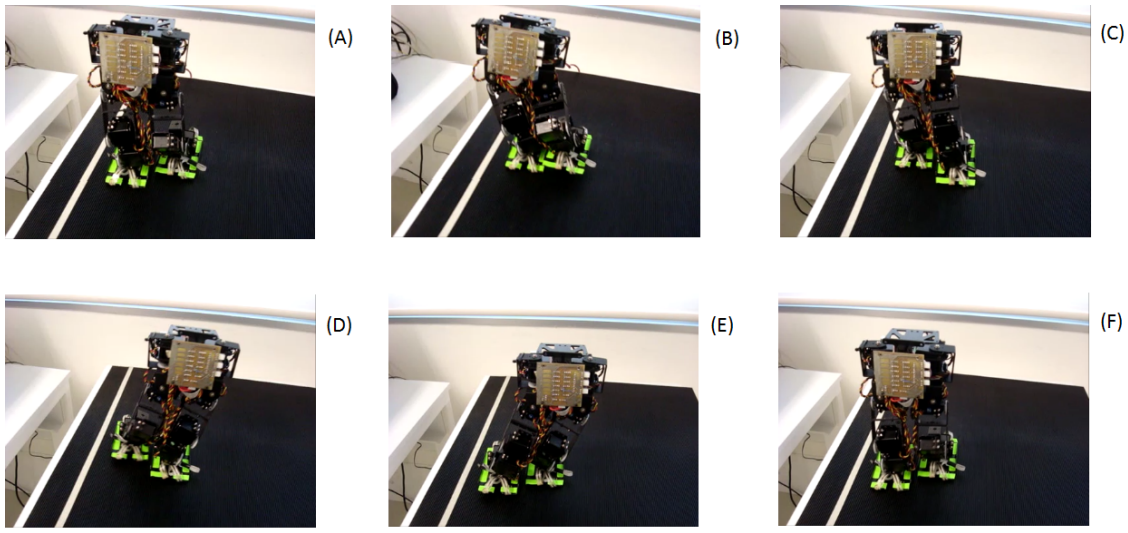

Figura 5.18: Implementación del ciclo de marcha con el prototipo físico. Perspectiva frontal.

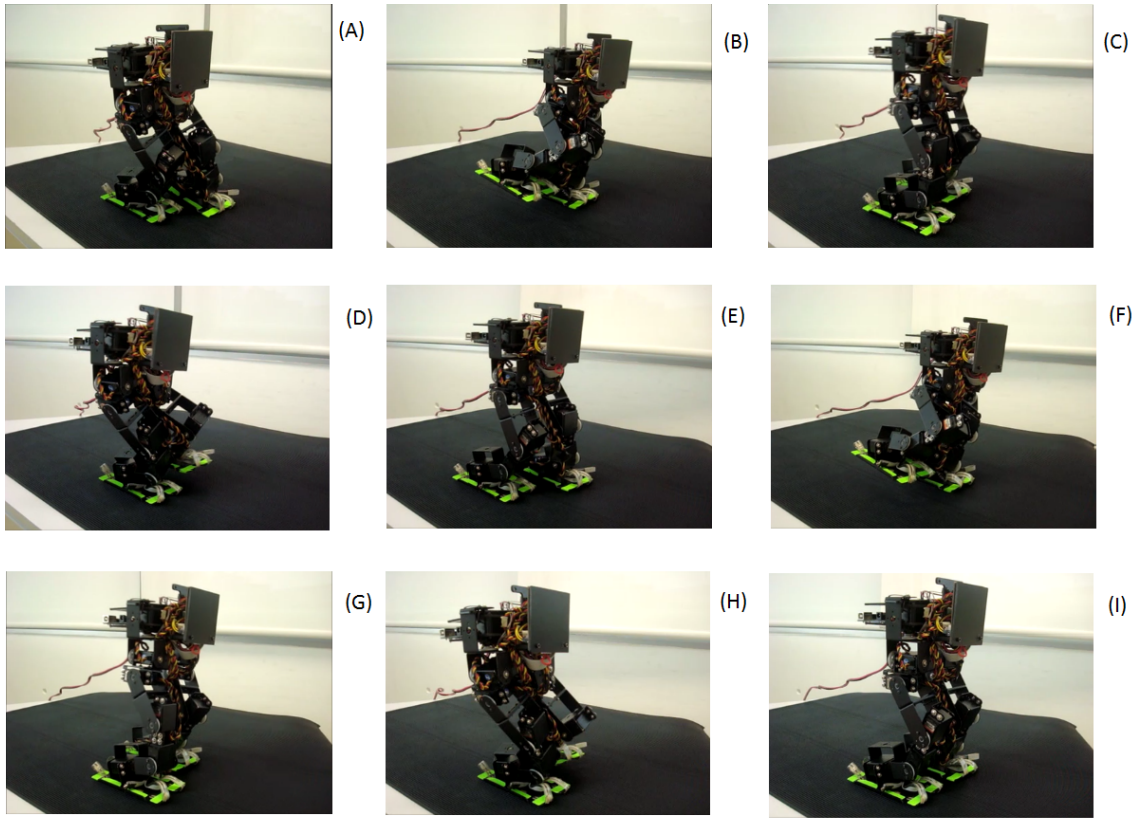

Figura 5.19: Implementación del ciclo de marcha con el prototipo físico. Perspectiva lateral.

Por otra parte, se tuvieron algunos problemas durante la implementación del ciclo de marcha ya que los engranes de un servomotor se barrieron en dos ocasiones. Para solucionar esto, la primera vez se intercambiaron engranes con otro servomotor que ya no servía, la segunda vez como ya no se contaba con otro engrane de repuesto se tuvo que ubicar el

rango de ángulos en el que trabaja ese servomotor e insertar el engrane barrido de tal forma que la parte barrida no hiciera contacto con el siguiente engrane para el rango de trabajo de ese servomotor. En la figura 5.20 se aprecian los servomotores y su arreglo de engranes.

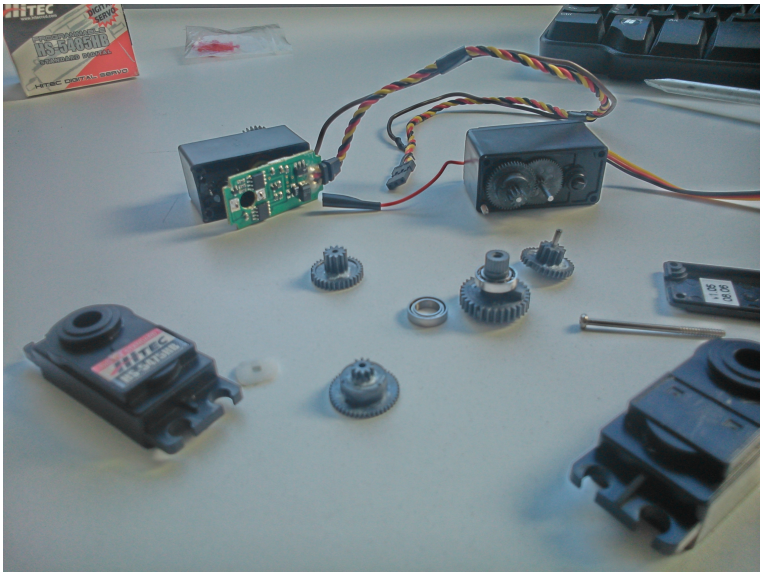

Figura 5.20: Arreglo de engranes de los servomotores.

Capítulo 6

Conclusiones y Trabajo a Futuro

6.1. Conclusiones

En este trabajo se lograron simular e implementar trayectorias dinámicamente estables a partir de un modelo simplificado, de la dinámica real de un robot bípedo, que hasta el momento no ha sido estudiado en la Facultad de Ingeniería de la UNAM tanto como el modelo simplificado péndulo invertido lineal. Mediante las comparaciones correspondientes con el modelo péndulo invertido lineal, se determino que el modelo carro-mesa es un modelo más robusto que se adapta mejor a la necesidad de generar trayectorias dinámicamente estables, ya que se pueden generar dichas trayectorias de forma continua y a la vez optimizar el $jerk$, parte fundamental para el equilibrio dinámico.

Por otra parte, se lograron determinar y simular los métodos más eficientes para la solución del modelo carro-mesa que no hacen consideraciones que acarrean errores, y que tampoco involucran procesos u operaciones matemáticas complejas que consumen demasiados recursos computacionales. En cuanto a las simulaciones, se obtuvo que la solución por seguimiento de trayectorias es una solución más eficiente que la solución por discretización de la aceleración ya que se puede obtener una buena aproximación empleando una carga computacional mucho menor, proporcionando los parámetros adecuados que definen el índice de desempeño J y estableciendo la longitud necesaria para la ventana de valores futuros de la demanda solicitada.

Con la simulación del ZMP, calculado a partir de la dinámica total del banco de pruebas, se corroboró que lo que se gana en tiempo con un modelo simplificado se pierde en exactitud, pero que al menos se obtiene una aproximación que se mantiene dentro de los límites del polígono de soporte, lo cual es un indicador de estabilidad, además se logró calcular una compensación a partir del error para obtener una mejor aproximación. Por otra parte, se pudo obtener experimentalmente el área de mayor estabilidad para el robot bípedo Scout empleando el modelo carro-mesa, corroborando así la diferencia entre la teoría y la experimentación, ya que la marcha bípeda no fue estable para todas las posiciones del ZMP propuestas dentro del polígono de soporte.

En cuanto a la aplicación móvil desarrollada en este trabajo, es claro que cada dispositivo o herramienta tiene un propósito, así como sus ventajas y desventajas, por lo que no en muchos casos una herramienta sustituirá a otra completamente. Pero es muy importan-

te buscar integrar nuevas herramientas que, aunque son diseñadas para otros fines, bien las podemos tomar y adaptar a nuestras necesidades para mantenerse a la vanguardia y desarrollar soluciones más novedosas. Por ejemplo, en este trabajo se realizo con un dispositivo móvil la misma tarea que se venía realizando con una computadora, a excepción de la simulación, e inclusive se emplearon algunas herramientas con las que no cuentan las computadoras.

En cuanto a la integración de la aplicación de reconocimiento de voz para controlar un servomotor, se considera que esto es sólo una demostración de lo que se puede llegar a hacer con una herramienta como ésta, por ejemplo, esto se podría utilizar para controlar una prótesis inteligente con un dispositivo móvil, en especial de miembro superior ya que en ocasiones es difícil poder manipularlas. Por otra parte, el empleo del acelerómetro y del sensor de proximidad demuestra que se puede sacar provecho de los dispositivos móviles para emplearlos en la robótica, por ejemplo, algunos dispositivos móviles que cuentan con los sensores que integran una IMU (giroscopio, magnetómetro y acelerómetro) pueden ser empleados para ayudar a controlar un cuadricoptero proporcionando los sensores y ayudando a los microcontroladores con el procesamiento de la información necesaria para el controlador.

Personalmente lo que a mí dejo este trabajo es la satisfacción de haber retomado y dado continuidad a un proyecto, especialmente porque había una cierta discontinuidad en el mismo, para así poder hacer nuevas aportaciones que servirán como punto de partida para nuevos compañeros que como en mi caso no tendrán que partir de cero y lidiar con los problemas a los cuales yo me enfrente al retomar este proyecto. Por otra parte, para mí fue un logro haber escrito un libro de al menos dos de los más importantes que espero escribir en la vida.

6.2. Trabajo a futuro

Como este trabajo trató básicamente acerca de la generación de trayectorias dinámicamente estables empleando la menor carga computacional posible, el paso siguiente podría ser tratar de generar dichas trayectorias en tiempo real, para lo cual sería necesario evaluar cambiar la tarjeta de desarrollo *Arduino Mega* por un microcontrolador o dispositivo con mayor capacidad y rapidez como la tarjeta-computadora *Raspberry Pi*, aunque con este dispositivo no se pueden realizar lecturas analógicas y los pines de salida y de entrada son limitados. Por otra parte ya se cuenta con una comunicación inalámbrica, por lo que se podría desarrollar una interfaz en algún lenguaje de programación como *Java* o *C #* que se comunique directamente y en tiempo real con *Matlab* o *Mathematica* para resolver la cinemática inversa.

En cuanto al prototipo físico se detectó que la velocidad y la precisión de los servomotores no son las que se especifican por el fabricante, lo que se debe a que estos servomotores en algún momento fueron programados para tener un menor desempeño y aumentar así su tiempo de vida, sin embargo esto afecta al desempeño del robot, por lo que se propone encontrar la forma de programarlos, cambiar de servomotores (lo cual es más caro) o implementar un controlador de posición propio, utilizando los servomotores como motores de corriente directa.

Bibliografía

[1] C. Zhou and Q. Meng. Reinforcement learning with fuzzy evaluative feedback for a biped robot. *Publicación desconocida*, 2000.

[2] M. Vukobratovic and O. Timcenko. Stability analysis of certain class of bipedal walking robots with hybridization of classical and fuzzy control. *International Conference on Advanced Robotics and Intelligent Automation*, 1995.

[3] A. W. Salatian and Y. F. Zheng. Gait synthesis for a biped robot climbing sloping surfaces using neural networks. part 1: Static learning. *In IEEE International Conference*, 1992.

[4] M. Y. Cheng and C. S. Lin. Genetic algorithm for control design of biped locomotion. *Journal of Robotic Systems*, 1997.

[5] Ambarish Goswami. Postural stability of biped robots and the foot rotation indicator (fri) point. *The International Journal of Robotics Research*, 1999.

[6] Jerzy Mrozowski, Jan Awrejcewicz, and Piotr Bamberski. Analysis of stability of the human gait. *Journal of Theoretical and Applied Mechanics*, 2007.

[7] Miomir Vukobratovic and Branislav Borovac. Zero moment point − thirty five years of its life. *International Journal of Humanoid Robotics*, 2004.

[8] J. Pratt, C. Chee-Meng, A. Torres, P. Dilworth, and G. Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 2001.

[9] H. Hirukawa, S. Hattori, S. Kajita, K. Harada, K. Kaneko, and F. Kanehiro. A pattern generator of humanoid robots. *IEEE International Conference on Robotics*, 2007.

[10] A. Albert and W. Gerth. Analytic path planning algorithms for bipedal. *Journal of Intelligent and Robotic Systems*, 2003.

[11] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. *2003 IEEE lnternational Conference on Robotics and Automation*, 2003.

[12] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[13] Shuuji Kajita, Osamu Matsumoto, and Muneharu Saigo. Real-time 3d walking pattern generation for a biped robot with telescopic legs. *In IEEE International Conference on Robotics and Automation*, 2001.

[14] Youngjin Choi, Bum-Jae You, and Sang-Rok Oh. On the stability of indirect zmp controller for biped robot systems. *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[15] WU Wei-guo and HOU Yue-yang. Research on rapid walking of biped robot based on parametric surface table cart model. *In IEEE International Conference on Robotics and Biomimetics*, 2009.

[16] Rafael López García. Planificación y optimización de la caminata de un robot bípedo. Master's thesis, Universidad Nacional Autónoma de México, 2012.

[17] Javier Pliego Jiménez. Análisis del movimiento de un robot bípedo de siete grados de libertad. Master's thesis, Universidad Nacional Autónoma de México, 2010.

[18] Yasuhisa Hasegawa, Junho Jang, and Yoshiyuki Sankai. Cooperative walk control of paraplegia patient and assistive system. *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[19] Octavio Narváez Aroche. Modelos cinemático y dinámio de un robot bípedo de doce grados de libertad internos. Master's thesis, Universidad Nacional Autónoma de México, 2010.

[20] Ernesto Villalobos Guerrero. Instrumentación de un robot bípedo de 12 gdl: sensores de posición, presión e inercial. Master's thesis, Universidad Nacional Autónoma de México, 2013.

[21] Pedro M. Vera Luna. *Biomecánica de la marcha humana normal y patalógica.* Instituto de Biomecánica de Valencia, 1999.

[22] Carmen Marco Sanz. Cinesiología de la marcha humana normal. Technical report, Universidad de Zaragoza, 2011.

[23] NASA. Man-systems integration standards. Technical report, Administración Nacional de la Aeronáutica y del Espacio, 1995.

[24] Francisco Alburquerque. Fundamentos de fisioterapia, disponible en http://www.fisiofundamental.com/guia/tema9.html.

[25] Francisco Hernández Stengele. Diseño y construcción de prototipo neumático de prótesis de pierna humana. Master's thesis, Universidad de las Américas Puebla, 2008.

[26] Barbara A. Gowitzke and Morris Milner. El cuerpo y sus movimiento. bases científicas. In *Colección medicina depotiva.* Paidotribo, 1999.

[27] Miguel Izquierdo. *Biomecánica y Bases Neuromusculares de la Actividad Física y el Deporte.* Médica Panamericana, 2008.

[28] Muska Mosston. Gimnasia dinámica. *Pax-México*, 1968.

[29] D A Winter. Human balance and posture control during standing walking. Technical report, Department of Kinesiology, University of Waterloo, 1995.

[30] Bruno Siciliano and Oussama Khatib. Springer handbook of robotics, 2008.

[31] Ogata Katsuhiko. *Ingeniería de Control Moderna*. Pearson, 1998.

[32] Huibert Kwakernaak and Raphael Silvan. *Linear Optimal Control Systems*. John Wiley and Sons, Inc., 1972.

[33] R. López García, Octavio Narvaéz Aroche, and E. Rocha Cózatl. Interfáz gráfica de usuario para pruebas de marcha en un robot bípedo. *Memorias del XVII Congreso Internacional Anual de la SOMIM*, 2011.

[34] http://playground.arduino.cc/ComponentLib/Servo.

[35] Shuuji Kajita. Overview of zmp-based biped walking. Technical report, National Institute of Advance Industrial Science and Technology, 2008.

[36] Youngjin Choi and Doik Kim. *Humanoid Robot Balancing, Advances in Robotics, Automation and Control*. InTech, 2008.

[37] Toshiaki Tsuji and Kouhei Qhnishi. A control of biped robot which applies inverted pendulum mode with virtual supporting point. *Proceedings of 2002 IEEE*, 2002.

[38] Tomoyuki Suzuki and Kouhei Ohnishi. Trajectory planning of biped robot with two kinds of inverted pendulums. *Proceedings of 2006 IEEE*, 2006.

[39] Tomomichi Sugihara, Yoshihiko Nakamura, and Hirochika Inoue. Realtime humanoid motion generation through zmp manipulation based on inverted pendulum control. *2002 IEEE lnternational Conference on Robotics and Automation*, 2002.

[40] Shuuji Kajita, Mitsuharu Morisawa, Kensuke Harada, Kenji Kaneko, Fumio Kanehiro, Kiyoshi Fujiwara, and Hirohisa Hirukawa. Biped walking pattern generator allowing auxiliary zmp control. *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[41] Jung-Hoon Kim. Walking pattern generation of a biped walking robot using convolution sum. *2007 IEEE*, 2007.

[42] Shengjun Peng, Haitao Shui, Guang Li, and Hongxu Ma ulises. Walking gait planning of humanoid soccer robot based on the desired zmp trajectories. *IEEE 2nd International Conference on Industrial Mechatronics and Automation*, 2010.

[43] Tohru Katayama, Takahira Ohki, Toshio Inoue, and Tomoyuki Kato. Design of an optimal controller for a discrete-time system subject to previewable demand. *Int. J. Control*, 1985.

[44] API Guides. Using the accelerometer, disponible en `http://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-accel`, 2014.

[45] API Guides. Sensor coordinate system, disponible en `http://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords`.

Apéndice A

Programa para la solución por discretización de la aceleración

Solución del modelo carro - mesa mediante discretización de la aceleración

Datos

```
off = 0.0385; (* off-set necesario para el ZMP en Y para que
  el bipedo avance de acuerdo a la posición erguida: 0.0385 metros
    sí queremos que el COM permanezca en la junta de la plana de cada pie,
el off-set debe ser cero *)
g = 9.81;  (* aceleración de la gravedad *)
zh = 0.235;  (* altura del centro de masa *)
dT = 0.005;  (* tiempo de muestreo: 5 [ms] *)

(*Parametros para definir automáticamente el ZMP para 5 pasos *)

lp = 0.07;   (* longitud de paso en metros *)
hp = 0.03;    (* altura del pie en metros *)
amp = 0.05424;  (* 5. 424 cm es la distancia horizontal en metros,
sobre el eje x, del centro del robot al centro de una planta del robot *)
no = 1080;    (* numero de puntos para la solución de 5 pasos 1080 *)

(* Matrizes del sistema continuo que representa al modelo carro-mesa *)
ac = {{0, 1, 0}, {0, 0, 1}, {0, 0, 0}};
bc = {{0}, {0}, {1}};
cc = {{1, 0, -zh / g}};
dc = {{0}};

(* Constantes que representan al modelo carro - mesa discretizado,
definidas en la subsección 4.3.1 *)
ad = -zh/(g * dT^2);
bd = 1 + 2 * zh/(g * dT^2);
cd = -zh/(g * dT^2);
(*Parametros de posicion y tiempo para la interpolación mediate splines*)

(*PARA LA ALTURURA Y AVANCE DEL PIE*)
t0 = 160;  z0 = 0;           y0 = 0;
t1 = 195;  Z1 = 0.6667 * hp;  Y1 = 0.25 * lp;
t2 = 230;  Z2 = hp;           Y2 = 0.5 * lp;
t3 = 265;  z3 = 0.6667 * hp;  y3 = 0.75 lp;
t4 = 300;  z4 = 0;           y4 = lp;
```

Generación arbitraria de la trayectoria del ZMP en X y Y (Lpx=750mm y Wp=100mm)

```
(* Periodo de soporte simple ss=140*dT=0.7,
periodo de soporte doble=20*dT=0.1, periodo total=0.8 *)
For[k = 1, k ≤ 1500, k += 1,

  (*Trayectoria del ZMP en Y*)
  If[k ≥ 1 && k < 300, ZmpY[k] = 0 + off];
  If[k ≥ 300 && k < 320, ZmpY[k] = (k * lp / 20) + (-300 * lp / 20) + off];
  If[k ≥ 320 && k < 460, ZmpY[k] = lp + off];
  If[k ≥ 460 && k < 480, ZmpY[k] = (k * lp / 20) + (lp - (460 * lp / 20)) + off];
  If[k ≥ 480 && k < 620, ZmpY[k] = 2 * lp + off];
  If[k ≥ 620 && k < 640, ZmpY[k] = (k * lp / 20) + (2 * lp - (620 * lp / 20)) + off];
  If[k ≥ 640 && k < 780, ZmpY[k] = 3 * lp + off];
  If[k ≥ 780 && k < 800, ZmpY[k] = (k * lp / 20) + (3 * lp - (780 * lp / 20)) + off];
  If[k ≥ 800 && k ≤ 1500, ZmpY[k] = 4 * lp + off];

  (*Trayectoria del ZMP en X*)
  If[k ≥ 1 && k < 140, ZmpX[k] = 0];
  If[k ≥ 140 && k < 160, ZmpX[k] = k * (amp / 20) + (-140 * amp / 20)];
  If[k ≥ 160 && k < 300, ZmpX[k] = amp];
  If[k ≥ 300 && k < 320, ZmpX[k] = -k * (2 * amp / 20) + (amp + (300 * 2 * amp / 20))];
  If[k ≥ 320 && k < 460, ZmpX[k] = -amp];
  If[k ≥ 460 && k < 480, ZmpX[k] = k * (2 * amp / 20) + (-amp - (460 * 2 * amp / 20))];
  If[k ≥ 480 && k < 620, ZmpX[k] = amp];
  If[k ≥ 620 && k < 640, ZmpX[k] = -k * (2 * amp / 20) + (amp + (620 * 2 * amp / 20))];
  If[k ≥ 640 && k < 780, ZmpX[k] = -amp];
  If[k ≥ 780 && k < 800, ZmpX[k] = k * (2 * amp / 20) + (-amp - (780 * 2 * amp / 20))];
  If[k ≥ 800 && k < 940, ZmpX[k] = amp];
  If[k ≥ 940 && k ≤ 1500, ZmpX[k] = 0];

 ];(*Cierra For*)
(* Graficación de la trayectoria en X *)
Tabla1 = Table[{i, ZmpX[i]}, {i, 1, 1100, 1}];
Fig1 = ListPlot[Tabla1,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"dT", "ZmpX"},
    GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria en Y *)
Tabla2 = Table[{i, ZmpY[i]}, {i, 1, 1100, 1}];
Fig2 = ListPlot[Tabla2,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"dT", "ZmpY"},
```

```
    GridLines → Automatic, PlotRange → All
  ];
(* Graficación de la trayectoria en X y Y*)
Tabla3 = Table[{ZmpY[i], ZmpX[i]}, {i, 1, 1100, 1}];
Fig3 = ListPlot[Tabla3,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.001], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"ZmpY", "ZmpX"},
    GridLines → Automatic, PlotRange → All
  ];
Show[Fig1]
Show[Fig2]
Show[Fig3]
```

Generación de la matriz de coeficientes

```
AbsoluteTiming[

 A = ConstantArray[0, {no, no}];

 (* valores iniciales *)
 A[[1, 1]] = ad + bd;
 A[[1, 2]] = cd;

 (* valores finales *)
 A[[no, no - 1]] = ad;
 A[[no, no]] = bd + cd;

 For[k = 2, k ≤ no - 1, k += 1,

  A[[k, k - 1]] = ad;
  A[[k, k]] = bd;
  A[[k, k + 1]] = cd;

 ];(*Cierra For*)

]
```

Solución del COM, en X y Y, mediante la matriz inversa de coeficientes

```
AbsoluteTiming[

 Ainv = Inverse[A]; (* generación de la matriz inversa *)

 For[j = 1, j ≤ no, j++,
  (* Solución del COM mediante la matriz inversa en x *)
  comx = {Table[Ainv[[j, i]], {i, 1, no}].Transpose[{Table[ZmpX[i], {i, 1, no}]}]};
```

```
  comX[j] = comx[[1, 1]];
  (* Solución del COM mediante la matriz inversa en y *)
  comy = {Table[Ainv[[j, i]], {i, 1, no}].Transpose[{Table[ZmpY[i], {i, 1, no}]}]};
  comY[j] = comy[[1, 1]];

 ];(*cierre For*)

]

(* Graficación de la trayectoria del COM en X *)
Tabla4 = Table[{i, comX[i]}, {i, 1, no, 1}];
Fig4 = ListPlot[Tabla4,
   ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
   Joined → True, PlotStyle → {Thickness[0.002], RGBColor[1, 0, 0]}, Frame → True,
   FrameLabel → {"dT", "COG X"}, LabelStyle → Directive[Blue, Black],
   GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria del COM en Y *)
Tabla5 = Table[{i, comY[i]}, {i, 1, no, 1}];
Fig5 = ListPlot[Tabla5,
   ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
   Joined → True, PlotStyle → {Thickness[0.002], RGBColor[1, 0, 0]}, Frame → True,
   FrameLabel → {"dT", "COG Y"}, LabelStyle → Directive[Blue, Black],
   GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria del COM en X y Y*)
Tabla6 = Table[{comY[i], comX[i]}, {i, 1, no, 1}];
Fig6 = ListPlot[Tabla6,
   ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
   Joined → True, PlotStyle → {Thickness[0.002], RGBColor[1, 0, 0]}, Frame → True,
   FrameLabel → {"COG Y", "COG X"}, LabelStyle → Directive[Blue, Black],
   GridLines → Automatic, PlotRange → All
  ];

Show[Fig4, Fig1]
Show[Fig5, Fig2]
Show[Fig6, Fig3]

Export["izmpx.png", Show[Fig4, Fig1]]
Export["izmpy.png", Show[Fig5, Fig2]]
Export["izmpxy.png", Show[Fig6, Fig3]]
```

Cálculo de los splines para la trayectoria de los 2 pies en z

```
Clear[a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4];
a = NSolve[
   {
```

```
    a1 * (t0)^2 + b1 * t0 + c1 == z0,
    a1 * (t1)^2 + b1 * t1 + c1 == Z1,

    a2 * (t1)^2 + b2 * t1 + c2 == Z1,
    a2 * (t2)^2 + b2 * t2 + c2 == Z2,

    a3 * (t2)^2 + b3 * t2 + c3 == Z2,
    a3 * (t3)^2 + b3 * t3 + c3 == z3,

    a4 * (t3)^2 + b4 * t3 + c4 == z3,
    a4 * (t4)^2 + b4 * t4 + c4 == z4,

    2 * a1 * t1 + b1 == 2 * a2 * t1 + b2,
    2 * a2 * t2 + b2 == 2 * a3 * t2 + b3,
    2 * a3 * t3 + b3 == 2 * a4 * t3 + b4,

    a1 == 0
   },
   {a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4}
  ];
a11 = a1 /. a[[1]];
b11 = b1 /. a[[1]];
c11 = c1 /. a[[1]];
a22 = a2 /. a[[1]];
b22 = b2 /. a[[1]];
c22 = c2 /. a[[1]];
a33 = a3 /. a[[1]];
b33 = b3 /. a[[1]];
c33 = c3 /. a[[1]];
a44 = a4 /. a[[1]];
b44 = b4 /. a[[1]];
c44 = c4 /. a[[1]];

For[k = 0, k ≤ 1500, k += 1,

 If[k ≥ 0 && k < 160, pzd[k] = 0];

 If[k ≥ 160 && k < 190, pzd[k] = a11 * k^2 + b11 * k + c11];
 If[k ≥ 190 && k < 220, pzd[k] = a22 * k^2 + b22 * k + c22];
 If[k ≥ 220 && k < 270, pzd[k] = a33 * k^2 + b33 * k + c33];
 If[k ≥ 270 && k < 300, pzd[k] = a44 * k^2 + b44 * k + c44];

 If[k ≥ 300 && k < 480, pzd[k] = 0];
 If[k ≥ 480 && k < 620, pzd[k] = pzd[k - 320]];
 If[k ≥ 620 && k < 800, pzd[k] = 0];
 If[k ≥ 800 && k < 940, pzd[k] = pzd[k - 640]];
 If[k ≥ 940 && k ≤ 1500, pzd[k] = 0];

 If[k ≥ 0 && k < 320, pzi[k] = 0];
 If[k ≥ 320 && k < 350, pzi[k] = a11 * (k - 160)^2 + b11 * (k - 160) + c11];
```

```
 If[k ≥ 350 && k < 380, pzi[k] = a22 * (k - 160)^2 + b22 * (k - 160) + c22];
 If[k ≥ 380 && k < 430, pzi[k] = a33 * (k - 160)^2 + b33 * (k - 160) + c33];
 If[k ≥ 430 && k < 460, pzi[k] = a44 * (k - 160)^2 + b44 * (k - 160) + c44];
 If[k ≥ 460 && k < 640, pzi[k] = 0];
 If[k ≥ 640 && k < 780, pzi[k] = pzi[k - 320]];
 If[k ≥ 780 && k ≤ 1500, pzi[k] = 0];
]
(* Graficación de la trayectoria del pie *)
Tablazd = Table[{i, pzd[i]}, {i, 0, 1500, 1}];
Figzd = ListPlot[Tablazd,
   ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
    {Thickness[0.01], RGBColor[0, 0, 1]}, Frame → True, FrameLabel → {"T", "zd"},
   GridLines → Automatic, PlotRange → All
  ];
(* Graficación de la trayectoria del pie *)
Tablazi = Table[{i, pzi[i]}, {i, 0, 1500, 1}];
Figzi = ListPlot[Tablazi,
   ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
    {Thickness[0.01], RGBColor[1, 0, 0]}, Frame → True, FrameLabel → {"T", "zi"},
   GridLines → Automatic, PlotRange → All
  ];
Show[Figzi, Figzd]
```

Cálculo de los splines para la trayectoria de los 2 pies en y

```
Clear[a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4];
b = NSolve[
   {
    a1 * (t0)^2 + b1 * t0 + c1 == y0,
    a1 * (t1)^2 + b1 * t1 + c1 == Y1,

    a2 * (t1)^2 + b2 * t1 + c2 == Y1,
    a2 * (t2)^2 + b2 * t2 + c2 == Y2,

    a3 * (t2)^2 + b3 * t2 + c3 == Y2,
    a3 * (t3)^2 + b3 * t3 + c3 == y3,

    a4 * (t3)^2 + b4 * t3 + c4 == y3,
    a4 * (t4)^2 + b4 * t4 + c4 == y4,

    2 * a1 * t1 + b1 == 2 * a2 * t1 + b2,
    2 * a2 * t2 + b2 == 2 * a3 * t2 + b3,
    2 * a3 * t3 + b3 == 2 * a4 * t3 + b4,

    a1 == 0
   },
   {a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4}
  ];
ax1 = a1 /. b[[1]];
```

```
bx1 = b1 /. b[[1]];
cx1 = c1 /. b[[1]];
ax2 = a2 /. b[[1]];
bx2 = b2 /. b[[1]];
cx2 = c2 /. b[[1]];
ax3 = a3 /. b[[1]];
bx3 = b3 /. b[[1]];
cx3 = c3 /. b[[1]];
ax4 = a4 /. b[[1]];
bx4 = b4 /. b[[1]];
cx4 = c4 /. b[[1]];

For[k = 0, k ≤ 1500, k += 1,

  If[k ≥ 0 && k < 160, pyd[k] = 0];

  If[k ≥ 160 && k < 190, pyd[k] = ax1 * k^2 + bx1 * k + cx1];
  If[k ≥ 190 && k < 220, pyd[k] = ax2 * k^2 + bx2 * k + cx2];
  If[k ≥ 220 && k < 270, pyd[k] = ax3 * k^2 + bx3 * k + cx3];
  If[k ≥ 270 && k < 300, pyd[k] = ax4 * k^2 + bx4 * k + cx4];

  If[k ≥ 300 && k < 480, pyd[k] = lp];
  If[k ≥ 480 && k < 620, pyd[k] = 2 * pyd[k - 320] + lp];
  If[k ≥ 620 && k < 800, pyd[k] = 3 * lp];
  If[k ≥ 800 && k < 940, pyd[k] = pyd[k - 640] + 3 * lp];
  If[k ≥ 940 && k ≤ 1500, pyd[k] = 4 * lp];

  If[k ≥ 0 && k < 320, pyi[k] = 0];
  If[k ≥ 320 && k < 460, pyi[k] = 2 * pyd[k - 160]];
  If[k ≥ 460 && k < 640, pyi[k] = 2 * lp];
  If[k ≥ 640 && k < 780, pyi[k] = pyi[k - 320] + 2 * lp];
  If[k ≥ 780 && k ≤ 1500, pyi[k] = 4 * lp];

 ];

Tablayd = Table[{i, pyd[i]}, {i, 0, 1500, 1}];
Figyd = ListPlot[Tablayd,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
     {Thickness[0.01], RGBColor[0, 0, 1]}, Frame → True, FrameLabel → {"T", "xd"},
    GridLines → Automatic, PlotRange → All
  ];
Tablayi = Table[{i, pyi[i]}, {i, 0, 1500, 1}];
Figyi = ListPlot[Tablayi,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
     {Thickness[0.01], RGBColor[1, 0, 0]}, Frame → True, FrameLabel → {"T", "xi"},
    GridLines → Automatic, PlotRange → All
  ];
Show[Figyi, Figyd]
Show[Figyi]
Show[Figyd]
```

Apéndice B

Programa para la solución por seguimiento de trayectorias

Generación de trayectorias con el modelo carro - mesa y control predictivo

Datos

In[1]:=
```
off = 0.04;(* off-set necesario en el ZMPy para que
   el bipedo avance de acuerdo a la posición erguida: 0.0385 metros
     sí queremos que el COM permanezca en la junta de la plana de cada pie,
el off-set debe ser cero *)

np = 270;    (* número de valores conocidos del ZMP de
 referencia para definir la ventana del control predictivo *)
no = 1080;   (* número de puntos requeridos para la
 generación de la trayectoria de la cadera *)
g = 9.81;     (* aceleración de la gravedad *)
zh = 0.235;   (* altura del centro de masa *)
dT = 0.005;  (* tiempo de muestreo en segundos *)

(*Parametros para definir automáticamente el ZMP*)

lp = 0.07;  (* longitud de paso en metros *)
hp = 0.03;   (* altura del pie en metros *)
amp = 0.05424; (*5.424 centimetros,
distancia horizontal del centro del robot al centro de una planta del robot*)

(* Matrizes del sistema continuo que definen al modelo carro-
 mesa en variables de estado *)
ac = {{0, 1, 0}, {0, 0, 1}, {0, 0, 0}};
bc = {{0}, {0}, {1}};
cc = {{1, 0, -zh / g}};
dc = {{0}};

(* Parametros para resolver la ecuación de Riccati *)
Q = {{1, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
R = {{0.00001}};

(*PARA LA ALTURURA Y AVANCE DEL PIE*)
t0 = 160;  z0 = 0;          Y0 = 0;
t1 = 195;  Z1 = 0.6667 * hp;  Y1 = 0.25 * lp;
t2 = 230;  Z2 = hp;          Y2 = 0.5 * lp;
t3 = 265;  z3 = 0.6667 * hp;  y3 = 0.75 lp;
t4 = 300;  z4 = 0;          y4 = lp;
```

Generación arbitraria de la trayectoria del ZMP en X y Y (Lpx=750mm y Wp=100mm)

```
(* Periodo de soporte simple ss=140*dT=0.7,
periodo de soporte doble=20*dT=0.1, periodo total=0.8 *)
For[k = 1, k ≤ 1500, k += 1,

  (* Trayectoria del ZMP en Y *)
  If[k ≥ 1 && k < 300, ZmpY[k] = 0 + off];
  If[k ≥ 300 && k < 320, ZmpY[k] = (k * lp / 20) + (-300 * lp / 20) + off];
  If[k ≥ 320 && k < 460, ZmpY[k] = lp + off];
  If[k ≥ 460 && k < 480, ZmpY[k] = (k * lp / 20) + (lp - (460 * lp / 20)) + off];
  If[k ≥ 480 && k < 620, ZmpY[k] = 2 * lp + off];
  If[k ≥ 620 && k < 640, ZmpY[k] = (k * lp / 20) + (2 * lp - (620 * lp / 20)) + off];
  If[k ≥ 640 && k < 780, ZmpY[k] = 3 * lp + off];
  If[k ≥ 780 && k < 800, ZmpY[k] = (k * lp / 20) + (3 * lp - (780 * lp / 20)) + off];
  If[k ≥ 800 && k ≤ 1500, ZmpY[k] = 4 * lp + off];

  (*Trayectoria del ZMP en Y*)
  If[k ≥ 1 && k < 140, ZmpX[k] = 0];
  If[k ≥ 140 && k < 160, ZmpX[k] = k * (amp / 20) + (-140 * amp / 20)];
  If[k ≥ 160 && k < 300, ZmpX[k] = amp];
  If[k ≥ 300 && k < 320, ZmpX[k] = -k * (2 * amp / 20) + (amp + (300 * 2 * amp / 20))];
  If[k ≥ 320 && k < 460, ZmpX[k] = -amp];
  If[k ≥ 460 && k < 480, ZmpX[k] = k * (2 * amp / 20) + (-amp - (460 * 2 * amp / 20))];
  If[k ≥ 480 && k < 620, ZmpX[k] = amp];
  If[k ≥ 620 && k < 640, ZmpX[k] = -k * (2 * amp / 20) + (amp + (620 * 2 * amp / 20))];
  If[k ≥ 640 && k < 780, ZmpX[k] = -amp];
  If[k ≥ 780 && k < 800, ZmpX[k] = k * (2 * amp / 20) + (-amp - (780 * 2 * amp / 20))];
  If[k ≥ 800 && k < 940, ZmpX[k] = amp];
  If[k ≥ 940 && k ≤ 1500, ZmpX[k] = 0];

 ];(*Cierra For*)
(* Graficación de la trayectoria en Y *)
Tabla1 = Table[{i, ZmpY[i]}, {i, 1, 1500, 1}];
Fig1 = ListPlot[Tabla1,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.001], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"dT", "ZmpY"},
    GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria en X *)
Tabla2 = Table[{i, ZmpX[i]}, {i, 1, no, 1}];
Fig2 = ListPlot[Tabla2,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.001], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"dT", "ZmpX"},
```

```
    GridLines → Automatic, PlotRange → All
   ];
(* Graficación de la trayectoria en X y Y*)
Tabla3 = Table[{ZmpY[i], ZmpX[i]}, {i, 1, no, 1}];
Fig3 = ListPlot[Tabla3,
    ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.001], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"ZmpY", "ZmpX"},
    GridLines → Automatic, PlotRange → All
   ];

Show[Fig1]
Show[Fig2]
Show[Fig3]
```

Discretización del sistema continuo

In[31]:=

```
AbsoluteTiming[

 dS = StateSpaceModel[{ac, bc, cc, dc}]; (* definición del sistema dinámico continuo *)
d = ToDiscreteTimeModel[dS, 0.005, z, Method → "ForwardRectangularRule"];
 (* discretización del sistema dinámico continuo *)
{ad, bd, cd, dd} = Normal[d];

 (* generación de las matrices para resolver la ecuación de Riccati *)
 aP = cd.ad;
 aR = {{1, aP[[1, 1]], aP[[1, 2]], aP[[1, 3]]},
   {0, ad[[1, 1]], ad[[1, 2]], ad[[1, 3]]}, {0, ad[[2, 1]], ad[[2, 2]], ad[[2, 3]]},
   {0, ad[[3, 1]], ad[[3, 2]], ad[[3, 3]]}}; (* aR = Ã *)

 bP = cd.bd;
 bR = {{bP[[1, 1]]}, {bd[[1, 1]]}, {bd[[2, 1]]}, {bd[[3, 1]]}}; (* bR = B̃ *)
]

(*ad//MatrixForm
bd//MatrixForm
cd//MatrixForm
dd//MatrixForm*)

aR // MatrixForm
bR // MatrixForm
```

Solución de la ecuación de Riccati y calculos de k y fi

In[34]:=
```
AbsoluteTiming[

 P = DiscreteRiccatiSolve[{aR, bR}, {Q, R}];
 (* P es la solución de la ecuación de Riccati *)

 (* Cálculo de las constantes que definen el controlador predictivo *)

 k = Inverse[R + Transpose[bR].P.bR].Transpose[bR].P;

 ke = k.{{1}, {0}, {0}, {0}};  (* ke = G_I *)

 kx = k.{{aP[[1, 1]], aP[[1, 2]], aP[[1, 3]]},
    {ad[[1, 1]], ad[[1, 2]], ad[[1, 3]]}, {ad[[2, 1]], ad[[2, 2]], ad[[2, 3]]},
    {ad[[3, 1]], ad[[3, 2]], ad[[3, 3]]}};  (* kx = G_x *)

 (* Cálculo de la ganancia predictiva para cada valor futuro del ZMP de referencia *)

 acR = aR - (bR.(Inverse[R + ((Transpose[bR]).P.bR)])).(Transpose[bR]).P.aR;
 (* acR = Ãc*)
 xR[1] = -(Transpose[acR]).P.{{1}, {0}, {0}, {0}};  (* xR[1] = X̃(1) *)
 fi[1] = -ke[[1, 1]];  (* fi[1] = G_d(1) = -G_I *)

 For[i = 2, i ≤ np, i++,
  xR[i] = (Transpose[acR]).xR[i - 1];
  (*vec=-(Inverse[R+((Transpose[bR]).P.bR)]).(Transpose[bR]).
       (MatrixPower[Transpose[acR],i-1]).P.{{1},{0},{0},{0}};*)
  vec = (Inverse[R + ((Transpose[bR]).P.bR)]).(Transpose[bR]).xR[i - 1];
  fi[i] = vec[[1, 1]]; (* fi[i] = G_d(i) *)

 ];
 (*Cierra For*)
]
P // MatrixForm
k.aR // MatrixForm
ke // MatrixForm
kx // MatrixForm
fi[np] (* para verificar hasta que valor calcular la ganancia predictiva *)
```

Solución del COM, en X y Y, mediante la entrada del controlador

```
(* condiciones iniciales *)
AbsoluteTiming[
 (* condiciones iniciales del COM *)
 x0 = {{0}, {0}, {0}};
 y0 = {{off}, {0}, {0}};
```

```
 xi = x0;
 yi = y0;
 (* variables para medir el error de seguimiento tanto en x como en y *)
 ex = {{0}};
 ey = {{0}};

 comX = ConstantArray[0, {no, 2}];
 (* Generación de una Matriz de no x 2 elementos = "0" para almacenar el COM en X *)
 comY = ConstantArray[0, {no, 2}];
 (* Generación de una Matriz de no x 2 elementos = "0" para almacenar el COM en Y *)

 For[j = 1, j ≤ no, j++,

  px = cd.xi; (* zmp de salida en x *)
  ex = ex + (px - {{ZmpX[j]}}); (* error en x entre el zmp generado y el deseado *)
  uix =
   -kx.xi - ke.ex - {Table[fi[i], {i, 1, np}].Transpose[{Table[ZmpX[j + i], {i, 1, np}]}]};
  (* controlador en x *)


  py = cd.yi; (* zmp de salida en y *)
  ey = ey + (py - {{ZmpY[j]}}); (* error en y entre el zmp generado y el deseado *)
  uiy =
   -kx.yi - ke.ey - {Table[fi[i], {i, 1, np}].Transpose[{Table[ZmpY[j + i], {i, 1, np}]}]};
  (* controlador en y *)


  comX[[j, 1]] = j;
  comX[[j, 2]] = xi[[1, 1]]; (* Trayectoria resultante del COG en x *)
  zmpXR[j] = px[[1, 1]];     (* ZMP generado, mediante el controlador, en x *)

  comY[[j, 1]] = j;
  comY[[j, 2]] = yi[[1, 1]];  (* Trayectoria resultante del COG en y *)
  zmpYR[j] = py[[1, 1]];      (* ZMP generado, mediante el controlador, en y *)

  (* cálculo del siguiente estado *)

  xi = ad.xi + bd.uix;
  yi = ad.yi + bd.uiy;
 ];(*cierre For*)

]

(* Graficación de la ganancia preventiva (fi) *)
previewGain = Table[{i, fi[i]}, {i, 1, np, 1}];
FigPG = ListPlot[previewGain,
   ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
   Joined → True, PlotStyle → {Thickness[0.004], RGBColor[1, 0, 0]},
   Frame → True, FrameLabel → {"dT", "Ganancia predictiva"},
   GridLines → Automatic, PlotRange → All
  ];
```

```
line = Table[{i, 0}, {i, 1, np, 1}];
FigL = ListPlot[line,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"}, Joined → True,
    PlotStyle → {Thickness[0.00001], RGBColor[0, 0, 0]}, Frame → True,
    GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria del COM en X *)
Tabla4 = Table[{comX[[i, 1]], comX[[i, 2]]}, {i, 1, no, 1}];
Fig4 = ListPlot[Tabla4,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[1, 0, 0]},
    Frame → True, FrameLabel → {"dT", "COG X, ZMP X"},
    GridLines → Automatic, PlotRange → All
  ];
(* Graficación de la trayectoria del COM en Y *)
Tabla5 = Table[{comY[[i, 1]], comY[[i, 2]]}, {i, 1, no, 1}];
Fig5 = ListPlot[Tabla5,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[1, 0, 0]},
    Frame → True, FrameLabel → {"dT", "COG Y, ZMP Y"},
    GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria del COM en X y Y*)
Tabla6 = Table[{comY[[i, 2]], comX[[i, 2]]}, {i, 1, no, 1}];
Fig6 = ListPlot[Tabla6,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[1, 0, 0]},
    Frame → True, FrameLabel → {"COG Y", "COG X"},
    GridLines → Automatic, PlotRange → All
  ];

(* Graficación de la trayectoria del ZMP resultante en X *)
Tabla41 = Table[{i, zmpXR[i]}, {i, 1, no, 1}];
Fig41 = ListPlot[Tabla41,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"dT", "zmpX"},
    GridLines → Automatic, PlotRange → All
  ];
(* Graficación de la trayectoria del ZMP resultante en Y *)
Tabla51 = Table[{i, zmpYR[i]}, {i, 1, no, 1}];
Fig51 = ListPlot[Tabla51,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.002], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"dT", "ZMP DESEADO y ZMP REAL"},
    GridLines → Automatic, PlotRange → All
  ];
(* Graficación de la trayectoria del ZMP resultante en X y Y*)
Tabla61 = Table[{zmpYR[i], zmpXR[i]}, {i, 1, no, 1}];
```

```
Fig61 = ListPlot[Tabla61,
    ImageSize → 300, BaseStyle → {12, FontFamily → "Arial"},
    Joined → True, PlotStyle → {Thickness[0.001], RGBColor[0, 0, 1]},
    Frame → True, FrameLabel → {"ZMPY", "ZMPX"},
    GridLines → Automatic, PlotRange → All
   ];


Show[FigPG, FigL]
Show[Fig4, Fig41]
Show[Fig5, Fig51]
Show[Fig6, Fig61]
Show[Fig51, Fig1]
Export["gain.png", Show[FigPG, FigL]]
Export["pczmpx.png", Show[Fig4, Fig41]]
Export["pczmpy.png", Show[Fig5, Fig51]]
Export["gain420y.png", Show[Fig51, Fig1]]
```

Cálculo de los splines para la trayectoria de los 2 pies en z

In[66]:=

```
Clear[a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4];
a = NSolve[
    {
     a1 * (t0)^2 + b1 * t0 + c1 == z0,
     a1 * (t1)^2 + b1 * t1 + c1 == Z1,

     a2 * (t1)^2 + b2 * t1 + c2 == Z1,
     a2 * (t2)^2 + b2 * t2 + c2 == Z2,

     a3 * (t2)^2 + b3 * t2 + c3 == Z2,
     a3 * (t3)^2 + b3 * t3 + c3 == z3,

     a4 * (t3)^2 + b4 * t3 + c4 == z3,
     a4 * (t4)^2 + b4 * t4 + c4 == z4,

     2 * a1 * t1 + b1 == 2 * a2 * t1 + b2,
     2 * a2 * t2 + b2 == 2 * a3 * t2 + b3,
     2 * a3 * t3 + b3 == 2 * a4 * t3 + b4,

     a1 == 0
    },
    {a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4}
   ];
a11 = a1 /. a[[1]];
b11 = b1 /. a[[1]];
c11 = c1 /. a[[1]];
a22 = a2 /. a[[1]];
b22 = b2 /. a[[1]];
```

```
c22 = c2 /. a[[1]];
a33 = a3 /. a[[1]];
b33 = b3 /. a[[1]];
c33 = c3 /. a[[1]];
a44 = a4 /. a[[1]];
b44 = b4 /. a[[1]];
c44 = c4 /. a[[1]];

For[k = 0, k ≤ 1500, k += 1,

 If[k ≥ 0 && k < 160, pzd[k] = 0];

 If[k ≥ 160 && k < 190, pzd[k] = a11 * k^2 + b11 * k + c11];
 If[k ≥ 190 && k < 220, pzd[k] = a22 * k^2 + b22 * k + c22];
 If[k ≥ 220 && k < 270, pzd[k] = a33 * k^2 + b33 * k + c33];
 If[k ≥ 270 && k < 300, pzd[k] = a44 * k^2 + b44 * k + c44];

 If[k ≥ 300 && k < 480, pzd[k] = 0];
 If[k ≥ 480 && k < 620, pzd[k] = pzd[k - 320]];
 If[k ≥ 620 && k < 800, pzd[k] = 0];
 If[k ≥ 800 && k < 940, pzd[k] = pzd[k - 640]];
 If[k ≥ 940 && k ≤ 1500, pzd[k] = 0];

 If[k ≥ 0 && k < 320, pzi[k] = 0];
 If[k ≥ 320 && k < 350, pzi[k] = a11 * (k - 160)^2 + b11 * (k - 160) + c11];
 If[k ≥ 350 && k < 380, pzi[k] = a22 * (k - 160)^2 + b22 * (k - 160) + c22];
 If[k ≥ 380 && k < 430, pzi[k] = a33 * (k - 160)^2 + b33 * (k - 160) + c33];
 If[k ≥ 430 && k < 460, pzi[k] = a44 * (k - 160)^2 + b44 * (k - 160) + c44];
 If[k ≥ 460 && k < 640, pzi[k] = 0];
 If[k ≥ 640 && k < 780, pzi[k] = pzi[k - 320]];
 If[k ≥ 780 && k ≤ 1500, pzi[k] = 0];
]
(* Graficación de la trayectoria del pie *)
Tablazd = Table[{i, pzd[i]}, {i, 0, 1500, 1}];
Figzd = ListPlot[Tablazd,
   ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
    {Thickness[0.01], RGBColor[0, 0, 1]}, Frame → True, FrameLabel → {"T", "zd"},
   GridLines → Automatic, PlotRange → All
  ];
(* Graficación de la trayectoria del pie *)
Tablazi = Table[{i, pzi[i]}, {i, 0, 1500, 1}];
Figzi = ListPlot[Tablazi,
   ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
    {Thickness[0.01], RGBColor[1, 0, 0]}, Frame → True, FrameLabel → {"T", "zi"},
   GridLines → Automatic, PlotRange → All
  ];
Show[Figzi, Figzd]
```

Cálculo de los splines para la trayectoria de los 2 pies en y

```
Clear[a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4];
b = NSolve[
    {
     a1 * (t0)^2 + b1 * t0 + c1 == Y0,
     a1 * (t1)^2 + b1 * t1 + c1 == Y1,

     a2 * (t1)^2 + b2 * t1 + c2 == Y1,
     a2 * (t2)^2 + b2 * t2 + c2 == Y2,

     a3 * (t2)^2 + b3 * t2 + c3 == Y2,
     a3 * (t3)^2 + b3 * t3 + c3 == y3,

     a4 * (t3)^2 + b4 * t3 + c4 == y3,
     a4 * (t4)^2 + b4 * t4 + c4 == y4,

     2 * a1 * t1 + b1 == 2 * a2 * t1 + b2,
     2 * a2 * t2 + b2 == 2 * a3 * t2 + b3,
     2 * a3 * t3 + b3 == 2 * a4 * t3 + b4,

     a1 == 0
    },
    {a1, b1, c1, a2, b2, c2, a3, b3, c3, a4, b4, c4}
  ];
ax1 = a1 /. b[[1]];
bx1 = b1 /. b[[1]];
cx1 = c1 /. b[[1]];
ax2 = a2 /. b[[1]];
bx2 = b2 /. b[[1]];
cx2 = c2 /. b[[1]];
ax3 = a3 /. b[[1]];
bx3 = b3 /. b[[1]];
cx3 = c3 /. b[[1]];
ax4 = a4 /. b[[1]];
bx4 = b4 /. b[[1]];
cx4 = c4 /. b[[1]];

For[k = 0, k ≤ 1500, k += 1,

  If[k ≥ 0 && k < 160, pyd[k] = 0];

  If[k ≥ 160 && k < 190, pyd[k] = ax1 * k^2 + bx1 * k + cx1];
  If[k ≥ 190 && k < 220, pyd[k] = ax2 * k^2 + bx2 * k + cx2];
  If[k ≥ 220 && k < 270, pyd[k] = ax3 * k^2 + bx3 * k + cx3];
  If[k ≥ 270 && k < 300, pyd[k] = ax4 * k^2 + bx4 * k + cx4];

  If[k ≥ 300 && k < 480, pyd[k] = lp];
```

```
  If[k ≥ 480 && k < 620, pyd[k] = 2 * pyd[k - 320] + lp];
  If[k ≥ 620 && k < 800, pyd[k] = 3 * lp];
  If[k ≥ 800 && k < 940, pyd[k] = pyd[k - 640] + 3 * lp];
  If[k ≥ 940 && k ≤ 1500, pyd[k] = 4 * lp];

  If[k ≥ 0 && k < 320, pyi[k] = 0];
  If[k ≥ 320 && k < 460, pyi[k] = 2 * pyd[k - 160]];
  If[k ≥ 460 && k < 640, pyi[k] = 2 * lp];
  If[k ≥ 640 && k < 780, pyi[k] = pyi[k - 320] + 2 * lp];
  If[k ≥ 780 && k ≤ 1500, pyi[k] = 4 * lp];

 ];

Tablayd = Table[{i, pyd[i]}, {i, 1, 1500, 1}];
Figyd = ListPlot[Tablayd,
   ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
    {Thickness[0.01], RGBColor[0, 0, 1]}, Frame → True, FrameLabel → {"T", "xd"},
   GridLines → Automatic, PlotRange → All
  ];
Tablayi = Table[{i, pyi[i]}, {i, 1, 1500, 1}];
Figyi = ListPlot[Tablayi,
   ImageSize → 300, BaseStyle → {15, FontFamily → "Arial"}, Joined → True, PlotStyle →
    {Thickness[0.01], RGBColor[1, 0, 0]}, Frame → True, FrameLabel → {"T", "xi"},
   GridLines → Automatic, PlotRange → All
  ];
Show[Figyi, Figyd]
Show[Figyi]
Show[Figyd]
```