TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ENSENADA

MAESTRÍA EN CIENCIAS EN INGENIERÍA MECATRÓNICA

CAMINATA BÍPEDA MEDIANTE PLANEACIÓN Y CONTROL DE CUERPO COMPLETO

TRABAJO DE TESIS PRESENTADO POR:
JOSÉ CRUZ ROJAS RODRÍGUEZ

PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS EN INGENIERÍA MECATRÓNICA

DIRECTORA DE TESIS:
DRA. ANA YAVENI AGUILAR BUSTOS

ENSENADA, B. C., JUNIO 2020

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE ENSENADA
División de Estudios de Posgrado e Investigación

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

Ensenada, Baja California, 04/Junio/2020

No. Oficio. D.E.P.I./086/2020
Asunto: Autorización de impresión de Trabajo de Tesis

C. JOSÉ CRUZ ROJAS RODRÍGUEZ
ESTUDIANTE DE LA MAESTRÍA EN CIENCIAS
EN INGENIERÍA MECATRÓNICA
PRESENTE

Con base en el dictamen de aprobación emitido por el Comité Tutorial de la Tesis con título: **CAMINATA BÍPEDA MEDIANTE PLANEACIÓN Y CONTROL DE CUERPO COMPLETO**, entregado por Usted para su análisis, le informamos atentamente que se **AUTORIZA** la impresión de dicho trabajo de tesis; en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Por la Tecnología de hoy y del Futuro®

EUSEBIO BUGARIN CARLOS
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ESTADOS UNIDOS MEXICANOS
SECRETARÍA DE EDUCACIÓN PÚBLICA
Instituto Tecnológico de Ensenada
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.p. Archivo.
EBC/kfcc

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

MAESTRÍA EN CIENCIAS EN INGENIERÍA MECATRÓNICA
DICTAMEN DEL COMITÉ TUTORIAL

Ensenada, B.C. a 03 de junio de 2020

DATOS DEL ESTUDIANTE			
Nombre:	**José Cruz Rojas Rodríguez**	**No. de Control:**	**M10760303**

DATOS DEL PROYECTO DE TESIS	
Título:	Caminata bípeda mediante planeación y control de cuerpo completo
Línea de Investigación:	Robótica y Control

COMITÉ DE TESIS		
Miembro	**Nombre**	**Firma**
DIRECTORA DE TESIS (Directora del comité)	Ana Yaveni Aguilar Bustos	
MIEMBRO DEL COMITÉ	Eusebio Bugarin Carlos	
MIEMBRO DEL COMITÉ	Salvador Durazo Acevedo	
Dictamen:	Se aprueba el trabajo de tesis, en virtud de que realizó las correcciones correspondientes conforme a las observaciones planteadas por este Comité Tutorial.	

Salvador Durazo Acevedo
Coordinador Académico

Ismael Hernández Capuchin
Presidente del Consejo de Posgrado

STA_10_dictamen_tesis_comite_tutorial_V1

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

CARTA DE CESIÓN DE DERECHOS

Ensenada, Baja California, a 04 de junio de 2020

Bajo protesta de decir verdad, quien suscribe **JOSÉ CRUZ ROJAS RODRÍGUEZ**, con número de control **M10760303**, alumno de la **Maestría en Ciencias en Ingeniería Mecatrónica** y con la dirección de Ana Yaveni Aguilar Bustos, declaro que soy autor del trabajo original de Tesis

CAMINATA BÍPEDA MEDIANTE PLANEACIÓN Y CONTROL DE CUERPO COMPLETO

Y otorgo mi conformidad, sin límite de temporabilidad, para ceder los derechos de reproducir y distribuir copias en su totalidad o en partes al Tecnológico Nacional de México campus Instituto Tecnológico de Ensenada (TecNM-ITE), mencionando la fuente. Lo anterior incluye que la tesis pueda ser publicada y difundida por el TecNM-ITE en los medios que considere convenientes. Asimismo, manifiesto que el TecNM-ITE queda liberado de cualquier conflicto surgido a raíz de la publicación. De igual manera, es de mi conocimiento que la publicación de la tesis no es con finalidad lucrativa, sino académica.

ATENTAMENTE

Jose C. Rojas R.

JOSÉ CRUZ ROJAS RODRÍGUEZ
ALUMNO DE LA MAESTRÍA EN CIENCIAS
EN INGENIERÍA MECATRÓNICA

STA_11_carta_cesion_derechos_V1

RESUMEN

Los problemas de balance de postura y generación de trayectorias para caminata de robots con patas, en general, se pueden formular como problemas de optimización no convexos. Sin embargo, la solución encontrada por estos algoritmos de optimización, que dependen del punto inicial de búsqueda, puede converger a cualquiera de sus múltiples mínimos locales existentes. Además, en la práctica los problemas de optimización resultantes son difíciles de resolver numéricamente. Por lo tanto, el enfoque del presente trabajo de tesis radica en la aplicación de algoritmos de optimización convexos multi-objetivo sobre los problemas de balance de postura y generación de trayectorias para la caminata de un robot bípedo.

Para incrementar la robustez ante perturbaciones en el balance de postura se regula el momento centroidal. Esto se logra empleado control de cuerpo completo mediante programación cuadrática. Además, se propone un algoritmo $O(N)$ para el cómputo del momento centroidal y su derivada temporal multiplicada por las velocidades articulares de manera conjunta. Dicha implementación se logra empleando álgebra espacial y de Lie, lo que permite reducir el costo computacional del controlador.

Finalmente, se propone un algoritmo jerárquico para la planeación de trayectorias con el modelo de la dinámica centroidal y la cinemática de cuerpo completo. El algoritmo jerárquico se desarrolla mediante una descomposición de la formulación original en 2 problemas de optimización que se alternan hasta la convergencia. Esto es, la planeación con la dinámica centroidal linealizada mediante heurísticas y la planeación cinemática de cuerpo completo. De esta forma es posible utilizar algoritmos especializados para resolver cada uno de los problemas abordados. En particular, se emplea programación cuadrática jerárquica en la planeación cinemática para lidiar con múltiples tareas u objetivos deseados.

Palabras claves: Robots con patas, robot humanoide, balance de postura, control de cuerpo completo, programación cuadrática, dinámica centroidal, caminata.

ABSTRACT

The postural balance and the trajectory planner problems for walking of legged robots, in general, can be formulated as non-convex optimization problems. Nevertheless, the solution found by these optimization algorithms, which depends on the initial starting search point, can converge to any of their multiple existing local minima. Also, the resultant optimization problems are difficult to solve numerically in practice. Therefore, the focus of the present thesis work lies in the application of multi-objective convex optimization algorithms on the postural balance and the trajectory planner problems for the walking of a biped robot.

To increase the robustness to perturbations in the postural balance the centroidal momentum is regulated. This is achieved using whole body control with quadratic programming. Moreover, an $O(N)$ algorithm is proposed for the joint computation of the centroidal momentum and its temporal derivative multiplied by the joint velocities. Such implementation is achieved using spatial and Lie algebra, which allows to reduce the computational cost of the controller.

Finally, a hierarchical algorithm is proposed for the planning of trajectories with the centroidal dynamics model and the full kinematics. The hierarchical algorithm is developed with a decomposition of the original formulation into 2 optimization problems that alternate to the convergence. This is, the planning with the lineralized centroidal dynamics with heuristics and the planning of the whole body kinematics. In this way, it is possible to use specialized algorithms to solve each one of the addressed problems. In particular, hierarchical quadratic programming is used in the kinematics planning to deal with multiple tasks or desired objectives.

Key words: Legged robots, humanoid robot, postural balance, whole body control, quadratic programming, centroidal dynamics, walking.

AGRADECIMIENTOS

En especial a mi directora de tesis la Dra. Ana Yaveni Aguilar Bustos por su atención y apoyo brindado durante mis estudios de maestría. Al comité de tesis por permitirme realizar este proyecto de tesis y a mi familia por apoyar mis estudios.

Finalmente, al TecNM-Instituto Tecnológico de Ensenada y al Consejo Nacional de Ciencia y Tecnología CONACyT.

ÍNDICE GENERAL

Contenido

Página

ÍNDICE GENERAL…… viii

LISTA DE FIGURAS……. xiii

LISTA DE TABLAS……... xviii

CAPÍTULO 1. INTRODUCCIÓN……. 1

1.1 Estudio del estado del arte……..... 4

1.1.1 Caminata estática y Punto de Momento Cero…..……. 4

1.1.2 Métodos heurísticos…….. 6

1.1.3 Balance y control de cuerpo completo……. 8

1.1.4 Péndulo elástico invertido…… 14

1.1.5 Cono de contacto…….. 15

1.1.6 Optimización mixta entera……... 17

1.1.7 Planeación con dinámica de cuerpo completo……..... 18

1.1.8 Planeación con la dinámica centroidal y de un cuerpo rígido…….. 20

1.1.9 Planeación con programación dinámica diferencial……. 22

1.2 Planteamiento del problema……... 23

1.2.1 Justificación…….. 23

1.3 Objetivos…….…… 24

1.3.1 Objetivo general…..…….. 24

1.3.2 Objetivos específicos.……... 24

1.4 Organización del documento…….……. 24

CAPÍTULO 2. MARCO TEÓRICO……. 26

2.1 Optimización convexa…… 26

2.1.1 Método del conjunto activo…….. 27

ÍNDICE GENERAL (Continuación)

Página

2.1.2 Método del punto interior……. 28

2.1.3 Método de multiplicadores con direcciones alternadas…..…….. 31

2.2 Álgebra espacial……. 34

2.2.1 Producto cruz espacial…….. 36

2.2.2 Inercia espacial……. 36

2.2.3 Ecuaciones de movimiento…… 37

2.3 Álgebra y grupos de Lie……. 38

2.3.1 Grupo $SO(3)$ y álgebra de Lie $so(3)$……..……... 39

2.3.2 Grupo $SE(3)$ y álgebra de Lie $se(3)$…… 39

2.3.3 Representación adjunta…..……... 40

2.3.4 Cinemática directa…….…….... 42

2.4 Balance de postura…….. 43

2.4.1 Balance de postura con dinámica de un cuerpo rígido…….. 44

2.4.2 Balance de postura con dinámica centroidal……. 46

2.4.3 Balance de postura con dinámica de cuerpo completo…….. 47

2.4.4 Programación cuadrática jerárquica (PCJ)…..…….. 48

2.5 Planeación de caminata…….. 49

2.5.1 Planeación con dinámica de cuerpo completo……... 50

2.5.2 Planeación con dinámica centroidal y cinemática de cuerpo completo……..... 51

2.6 Simulación de cuerpos rígidos articulados……. 53

2.6.1 Problema complementario lineal……... 53

2.6.2 Inclusiones diferenciales……..... 55

2.6.3 Modelo de contacto suave convexo……..... 56

2.7 Cómputo de la matriz de masas…….. 57

2.7.1 Algoritmo recursivo $O(N)$…….……..... 57

ÍNDICE GENERAL (Continuación)

Página

2.7.2 Fuerza unitaria espacial……..... 58

CAPÍTULO 3. BALANCE DE POSTURA EMPLEANDO CONTROL DE CUERPO COMPLETO…… 60

3.1 Introducción…… 60

3.2 Optimización convexa…… 61

3.2.1 Programación cuadrática……... 61

3.3 Balance de postura con control de cuerpo completo…….. 66

3.3.1 Tareas múltiples……. 66

3.3.2 Control de cuerpo completo…….. 71

3.4 Resultados……... 73

3.5 Conclusiones……... 80

CAPÍTULO 4. ALGORITMO $O(N)$ PARA EL CÓMPUTO DE LA DINÁMICA CENTROIDAL…… 81

4.1 Introducción…… 81

4.2 Álgebra espacial y de Lie……... 82

4.2.1 Derivadas de la representación adjunta……..... 83

4.3 Momento centroidal……..... 85

4.4 Tareas múltiples…….. 87

4.5 Control de cuerpo completo jerárquico…….. 91

4.6 Resultados……... 92

4.7 Conclusiones……..... 95

CAPÍTULO 5. PLANEACIÓN JERÁRQUICA DE CUERPO COMPLETO…… 97

5.1 Introducción…… 98

ÍNDICE GENERAL (Continuación)

Página

5.2 Planeación con dinámica centroidal……... 100

5.3 Planeación cinemática……..... 104

5.4 Resultados……..... 107

5.5 Conclusiones……... 112

CAPÍTULO 6. CONCLUSIONES……... 113

6.1 Conclusiones generales……... 113

6.2 Principales contribuciones……..... 114

6.3 Problemas abiertos……..... 114

LITERATURA CITADA……. 116

ANEXOS……..... 129

A Código de Matlab de algoritmos de dinámica y cinemática de cuerpos rígidos articulados…… 129

A.1 Algoritmo del cuerpo compuesto…..……... 129

A.2 Jacobiano aumentado…..……... 130

A.3 Jacobiano de eslabón…..……... 131

A.4 Algoritmo $O(N)$ propuesto.……... 131

A.5 Algoritmo Recursivo de Newton Euler……..……... 133

A.6 Pirámide de fricción……...……... 134

A.7 Cómputo de la posición del centro de masas……..……...……... 134

A.8 Aceleración por defecto de los puntos de contacto……...……... 135

A.9 Posición y velocidad de los puntos de contacto…..……...……... 138

A.10 Posición y velocidad de eslabón…….……...……... 139

A.11 Aceleraciones de los puntos de contacto…….…….. 139

ÍNDICE GENERAL (Continuación)

Página

A.12 Algoritmo del cuerpo articulado……...…….…… 141

B Código de Matlab de funciones elementales……... 142

B.1 Transformaciones espaciales…..……... 142

B.2 Producto cruz espacial……..…..……... 143

B.3 Matrices de rotación……... 144

B.4 Mapeo de índices……... 145

C Simulación del robot humanoide……. 146

C.1 Código de Matlab del simulador……... 146

C.2 Modelo del robot humanoide en Matlab…..……... 149

C.3 Código de Matlab de un escenario de balance con plataformas disparejas... 154

D Optimización convexa…….. 157

D.1 Código de Matlab de algoritmo del punto interior con regularización dual.. 157

LISTA DE FIGURAS

		Página
Figura 1.1	Locomoción animal sobre terrenos no acondicionados: (a) cabra montés y (b) borrego cimarrón (figura tomada de www.northlandartsnatureimages.com) ……...	1
Figura 1.2	Locomoción animal: (a) avestruz y (b) guepardo (figura tomada de www.dissolve.com)…….....	2
Figura 1.3	Metodologías para la planeación de caminata…….	3
Figura 1.4	Modelos reducidos (figura tomada de Orin y Goswami, 2013)……..	3
Figura 1.5	Criterio de estabilidad estático (figura tomada de Kajita y Espiau, 2008)……	4
Figura 1.6	Definición del ZMP (figura tomada de Kajita y Espiau, 2008)……..	5
Figura 1.7	Robots neumáticos (a), simetría durante la locomoción (b) (figura tomada de Raibert, 1986)…….....	7
Figura 1.8	Cuadrúpedo galopando (figura tomada de Krasny y Orin, 2010)……..	7
Figura 1.9	Elementos de la heurística de SIMBICOM (figura tomada de Yin *et al.,* 2007)……..	8
Figura 1.10	Control virtual (figura tomada de Pratt, 1995)…….....	9
Figura 1.11	Gato cayendo controlando su aterrizaje…….....	10
Figura 1.12	Balance de creaturas con diferentes morfologías (figura tomada de Macchietto *et al.,* 2009)……...	11
Figura 1.13	Balance de humanoide en superficies no estacionarias (figura tomada de Lee y Goswami, 2012)……	11
Figura 1.14	Balance de humanoide con diferentes ganancias para especificar la prioridad en las tareas (figura tomada de Abe *et al.,* 2007)……	12
Figura 1.15	Robot Atlas caminando sobre bloques (figura tomada de Feng *et al.,* 2014)……	13

LISTA DE FIGURAS (Continuación)

		Página
Figura 1.16	Control de cuerpo completo en creaturas con diferentes morfologías (figuras tomada de De Lasa et al., 2010)……..	13
Figura 1.17	Humanoide navegando terrenos irregulares (figura tomada de Mordatch *et al.,* 2010)……	14
Figura 1.18	Humanoide corriendo en simulación (figura tomada de Wensing y Orin, 2013)…….	15
Figura 1.19	Margen de estabilidad en el cono de contacto (figura tomada de Dai, 2016)……....	15
Figura 1.20	Robot Atlas caminando en superficies disparejas (figura tomada de Dai y Tedrake, 2016)…….	16
Figura 1.21	Robot LittleDog caminando en superficies disparejas (figura tomada de Aceituno-Cabezas *et al.,* 2017a)...…...……..	16
Figura 1.22	Modelo simplificado del robot LittleDog navegando entorno con brechas (figura tomada de Valenzuela, 2016)……	17
Figura 1.23	Robot Atlas navegando entorno con brechas y superficies disparejas (figura tomada de Koolen, 2019)……....	18
Figura 1.24	Pirueta sintetizada para creatura humanoide (figura tomada de Fang y Pollard, 2003)…….	18
Figura 1.25	Trayectorias optimizadas para el robot FastRunner (figura tomada de Posa y Tedrake, 2013)……	19
Figura 1.26	Locomoción de creaturas terrestres (figura tomada de Wampler y Popović, 2009)……....	20
Figura 1.27	Movimientos sintetizados por algoritmo de Mordatch *et al.,* 2012 (figura tomada de Mordatch *et al.,* 2012)……..	21
Figura 1.28	Robot Atlas corriendo (figura tomada de Dai *et al.,* 2014)……	21

LISTA DE FIGURAS (Continuación)

		Página
Figura 1.29	Planeación en MPC con modelo del Cheetah del MIT (figura tomada de Bledt *et al.,* 2017)……..	22
Figura 1.30	Trayectorias optimizadas para el humanoide por algoritmo de Neunert *et al.,* 2016 (figura tomada de Neunert et al., 2016)……	23
Figura 2.1	Algoritmo del conjunto activo……...	28
Figura 2.2	Idea del método del punto interior (figura tomada de Wadayama, 2008)……..	29
Figura 2.3	Algoritmo del punto interior……..	31
Figura 2.4	Método de multiplicadores con direcciones alternadas...……..	33
Figura 2.5	Algoritmo de Gauss Seidel Proyectado con sobre relajamiento……	56
Figura 3.1	Estructura cinemática del robot humanoide bajo estudio……..	61
Figura 3.2	Algoritmo de Mehrotra para programación cuadrática…….	65
Figura 3.3	(a) Visualización en Matlab del escenario 0, (b) graficas de la posición del centro de masas del escenario 0…….	75
Figura 3.4	(a) Visualización en Matlab del escenario 1, (b) graficas de la posición del centro de masas del escenario 1…….	75
Figura 3.5	(a) Visualización en Matlab del escenario 2, (b) graficas de la posición del centro de masas del escenario 2……..……...	76
Figura 3.6	(a) Visualización en Matlab del escenario 3, (b) graficas de la posición del centro de masas del escenario 3…….	76
Figura 3.7	(a) Visualización en Matlab del escenario 4, (b) graficas de la posición del centro de masas del escenario 4…….	77
Figura 3.8	Errores del COM, momento lineal y angular en el escenario 0…….	77
Figura 3.9	Errores del COM, momento lineal y angular en el escenario 1…….	78
Figura 3.10	Errores del COM, momento lineal y angular en el escenario 2…….	78
Figura 3.11	Errores del COM, momento lineal y angular en el escenario 3…….	78

LISTA DE FIGURAS (Continuación)

		Página
Figura 3.12	Errores del COM, momento lineal y angular en el escenario 4…….	79
Figura 4.1	Estructura cinemática del robot humanoide bajo estudio……..	83
Figura 4.2	Algoritmo para el cómputo de la matriz del momento centroidal…….	86
Figura 4.3	Algoritmo recursivo propuesto……..	88
Figura 4.4	Algoritmo propuesto expresado con operadores del álgebra de Lie..…	89
Figura 4.5	(a) Visualización en Matlab del escenario 1, (b) graficas de la posición del centro de masas del escenario 1…….	94
Figura 4.6	(a) Visualización en Matlab del escenario 2, (b) graficas de la posición del centro de masas del escenario 2…….	94
Figura 4.7	(a) Visualización en Matlab del escenario 3, (b) graficas de la posición del centro de masas del escenario 3……..……...	95
Figura 4.8	(a) Visualización en Matlab del escenario 4, (b) graficas de la posición del centro de masas del escenario 4……..……...	95
Figura 5.1	Estructura cinemática de humanoide (a) con brazos (b) sin brazos…...	100
Figura 5.2	Algoritmo propuesto para la planeación de cuerpo completo……	107
Figura 5.3	Locomoción en el escenario 1……	107
Figura 5.4	Graficas de la fuerza normal y posición del centro de masas del escenario 1…….....	108
Figura 5.5	Locomoción en el escenario 2……	108
Figura 5.6	Graficas de la fuerza normal y posición del centro de masas del escenario 2……..	108
Figura 5.7	Locomoción en el escenario 3……	109
Figura 5.8	Graficas de la fuerza normal y posición del centro de masas del escenario 3……..	109
Figura 5.9	Ciclo límite de la cadera y rodilla de la pierna izquierda en el escenario 3……..	109

LISTA DE FIGURAS (Continuación)

Página

Figura 5.10 Ciclo límite de la cadera y rodilla de la pierna derecha en el escenario 3…….. 110

Figura 5.11 Locomoción en el escenario 4…… 110

Figura 5.12 Graficas de la fuerza normal y posición del centro de masas del escenario 4…….. 110

Figura 5.13 Ciclo límite de la cadera y rodilla de la pierna izquierda en el escenario 4…….. 111

Figura 5.14 Ciclo límite de la cadera y rodilla de la pierna derecha en el escenario 4…….. 111

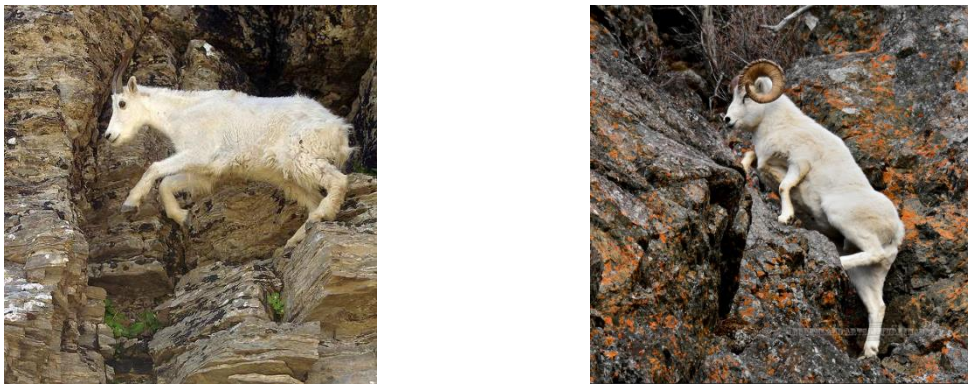
LISTA DE TABLAS

		Página
Tabla 3.1	Propiedades del modelo del robot bípedo……..	74
Tabla 4.1	Algoritmos propuestos en la literatura……...	90
Tabla 4.2	Jerarquía de tareas……..	91
Tabla 4.3	Ganancias empleadas en el controlador y simulador…….	93
Tabla 5.1	Jerarquía de tareas……..	106

CAPÍTULO 1

INTRODUCCIÓN

El interés del presente trabajo de tesis radica en el estudio del balance de postura y locomoción de robots con patas. Una de las principales ventajas de considerar robots con piernas sobre uno con ruedas es la versatilidad para navegar entornos no acondicionados por los humanos. En la naturaleza un ejemplo de la gran versatilidad que poseen los animales con piernas es la cabra montés y el borrego cimarrón (ver figura 1.1) los cuales muestran una gran destreza incluso a pocos días desde su nacimiento en la navegación de montañas rocosas.



(a) (b)

Figura 1.1. Locomoción animal sobre terrenos no acondicionados: (a) cabra montés y (b) borrego cimarrón (figura tomada de *www.northlandartsnatureimages.com*).

Otra de las destrezas que muestran algunos animales con piernas es su gran velocidad de desplazamiento.

Animales como el avestruz y el guepardo (ver figura 1.2) pueden alcanzar velocidades de hasta 70 km/h y 93 km/h respectivamente.

(a)

(b)

Figura 1.2. Locomoción animal: (a) avestruz y (b) guepardo (figura tomada de *www.dissolve.com*).

Las observaciones anteriores han motivado el desarrollo de robots con piernas con el objetivo de alcanzar las mismas destrezas que su contraparte de la naturaleza. En el estudio de robots con piernas se pueden destacar dos problemas importantes: el balance de postura y la planeación de trayectorias.

El problema de planeación para la caminata de robots con patas se suele dividir en dos partes: planeación de los puntos de contacto (pisadas) y posteriormente, planeación de trayectorias dinámicas, restringidas a los puntos de contacto previamente encontrados. Sin embargo, esta subdivisión del problema presenta limitaciones severas dado que en general no se considera la dinámica durante la planeación de los puntos de contacto. Algunas técnicas alternativas realizan la planeación de los puntos de contacto y las trayectorias del robot de forma simultánea. Para ello emplean la dinámica de cuerpo completo o la dinámica centroidal. En la figura 1.3 se muestran las principales metodologías que se utilizan para la planeación de caminata (Valenzuela, 2016).

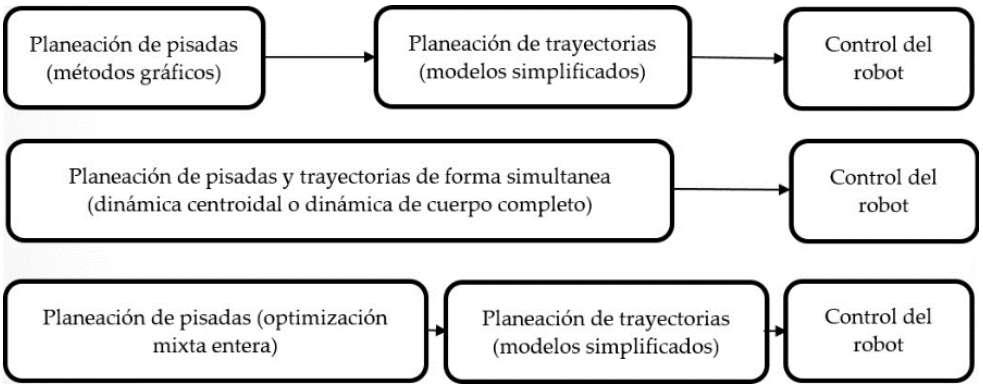


Figura 1.3. Metodologías para la planeación de caminata.

La planeación de trayectorias dinámicas de un robot suele emplear modelos reducidos que captura los aspectos más importantes de la locomoción. Algunos de los modelos simplificados más utilizados en la planeación son: el péndulo rígido invertido, el péndulo telescópico invertido, el carro-mesa, el péndulo lineal invertido (PLI) y el péndulo elástico invertido (PEI) (ver figura 1.4).

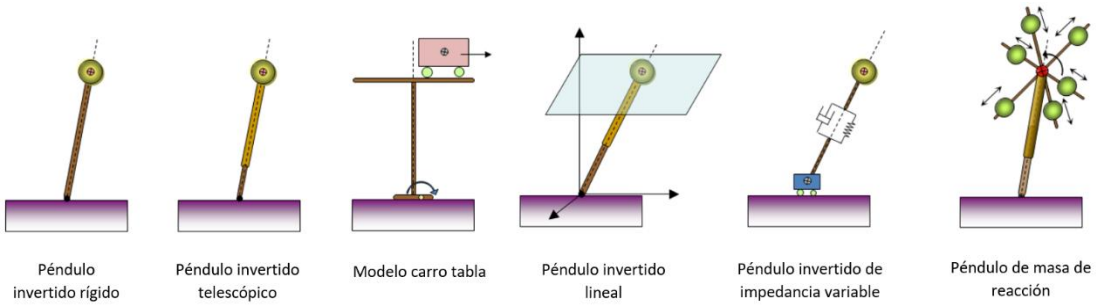


Figura 1.4. Modelos reducidos (figura tomada de Orin y Goswami, 2013).

Algunos de los modelos reducidos ignoran información del cuerpo completo, como el momento angular, el cual es indispensable para el balance del robot. Los modelos que consideran información del cuerpo completo son la dinámica de cuerpo completo y la dinámica centroidal. Estos modelos permiten la planeación de movimientos más ágiles y naturales.

1.1 Estudio del estado del arte

1.1.1 Caminata estática y Punto de Momento Cero

Una forma de evaluar la locomoción de un robot con patas es mediante un criterio de estabilidad. Uno de estos criterios es mantener la proyección de su centro de masas (COM) sobre su polígono de soporte (ver figura 1.5). Cumpliendo con este criterio el robot puede desplazarse sin perder el balance (Raibert, 1986).

Debido a que el criterio de estabilidad de centro de masas es muy conservativo, la velocidad de desplazamiento se ve severamente limitada. Sin embargo, en la navegación de terrenos con grandes irregulares, este criterio de estabilidad "estático" es de gran utilidad ya que le permite al robot desplazarse con el mayor cuidado posible. Por ejemplo, en Rebula *et al.* (2007), Kolter *et al.* (2008) y Buchli *et al.* (2009) se emplea una caminata estática en el robot LittleDog para la navegación en terrenos irregulares mientras el control del robot se realiza empleando dinámica inversa de base flotante.

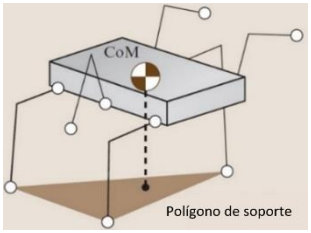


Figura 1.5. Criterio de estabilidad estático (figura tomada de Kajita y Espiau, 2008).

Para lograr una mayor velocidad de desplazamiento es necesario el uso de un criterio de estabilidad dinámico. Uno de estos criterios es mantener el Punto de Momento Cero (ZMP por sus siglas en inglés) dentro del polígono de soporte (el ZMP es un punto en donde actúa la fuerza de reacción resultante y el par horizontal es cero (ver figura 1.6)). El criterio anteriormente descrito fue introducido por Vukobratović y Juričić (1968). En Kajita *et al.* (2003) se utiliza LQR (Regulador Cuadrático Lineal) con el modelo del carro-mesa para

sintetizar una caminata en un humanoide especificando la posición deseada del ZMP a priori. En Wieber (2006) se realiza MPC (Control Predictivo por Modelo) mediante un programa cuadrático para incorporar límites en la posición del ZMP.

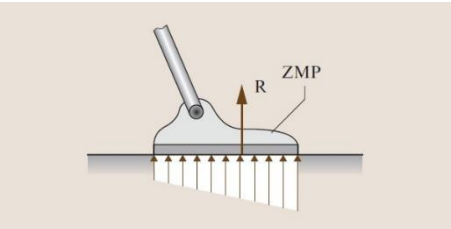


Figura 1.6. Definición del ZMP (figura tomada de Kajita y Espiau, 2008).

El péndulo lineal invertido es un modelo simplificado que típicamente se utiliza relacionando la dinámica del centro de masas y el ZMP. Este modelo supone que la altura del centro de masas es constante y el momento angular es cero, lo cual es una aproximación realista pues experimentos demuestran que los humanos minimizan el momento angular durante la caminata (Popovic *et al.*, 2004). Dado que este modelo es lineal, diversos trabajos formulan la planeación de caminata como un programa cuadrático utilizando la filosofía del control predictivo por modelo; esto permite considerar perturbaciones. En Faraji *et al.* (2014) se utiliza programación cuadrática y MPC para optimizar las pisadas del robot Atlas con el modelo del péndulo lineal invertido especificando los tiempos de contacto y la orientación del pie a priori.

En Kuindersma *et al.* (2014) se utiliza la función de valor cuadrática obtenida mediante LQR con la dinámica del ZMP como función objetivo en un programa cuadrático para el control de cuerpo completo, el cual se resuelve utilizando el método del conjunto activo. En Winkler *et al.* (2017) se introducen restricciones que utilizan los vértices del polígono de soporte para acotar la posición del ZMP. Sin embargo, estas restricciones son no lineales resultando en un programa no lineal. Esto permite optimizar las posiciones de los puntos de contacto y la trayectoria del centro de masas de un robot cuadrúpedo de manera simultánea.

El modelo del péndulo lineal invertido presenta varias limitaciones pero la más severa de éstas es que necesita la especificación del ZMP, el cual es complicado determinar en superficies disparejas (lo que limita su aplicación a solo terrenos parejos). El criterio de estabilidad de mantener el ZMP dentro del polígono de soporte es muy restrictivo e implica que el robot siempre debe tener un pie pegado al suelo durante la caminata, lo cual puede resultar no aceptable para ciertas aplicaciones. Tedrake (2004) muestra que, de hecho, los humanos al caminar violan este criterio de estabilidad.

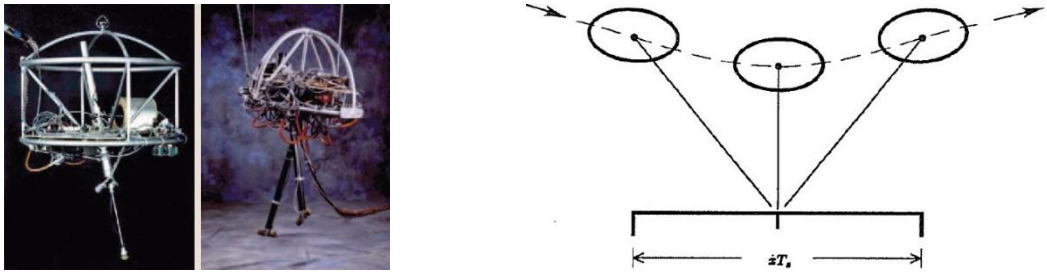
1.1.2 Métodos heurísticos

En Raibert (1986) se presenta una serie de robots de una, dos y cuatro piernas neumáticas (ver figura 1.7a) en el MIT (Massachusetts Institute of Technology). Estos robot son capaces de desplazarse y su metodología para mantener el balance está inspirada en el modelo del péndulo elástico invertido. Una observación importante de estos trabajos fue que la elasticidad de las piernas por si solas crean el movimiento necesario para la locomoción. La función del controlador es regular este movimiento natural creado por las propiedades de las piernas mediante 3 partes:

- Control de la altura: Para inyectar un valor de energía constante a la pierna.
- Control de velocidad: Incluye la predicción de pisadas con una heurística inspirada en la simetría del movimiento durante la fase de soporte (ver figura 1.7b).
- Control de la orientación de la base: Aquí se emplea control PD en la fase de soporte para controlar la orientación de la base a su valor deseado.

Algunos trabajos emplean la heurística de Raibert. En Murphy y Raibert (1985) se aplica en el trote de un cuadrúpedo con un modelo $2D$ en simulación y en Raibert y Hodgins (1991) en la animación con diversos patrones de locomoción de un bípedo, cuadrúpedo y canguro. En Raibert *et al.* (1986) se introduce el concepto de pierna virtual que consiste en reducir un sistema con múltiples patas a un sistema con una sola pata, esto se aplica en el

control de un cuadrúpedo corriendo. En Herr y McMahon (2000 y 2001) se introduce un controlador heurístico para el trote y galope con el modelo de un caballo que consiste en regular la postura y la velocidad de la pata a un valor deseado mediante control PD (Proporcional-Derivativo) y P (Proporcional), respectivamente; los parámetros del controlador se especifican de manera manual o se optimizan con un algoritmo genético. En Krasny y Orin (2010) se propone optimizar los parámetros de un controlador diseñado para el galope de un cuadrúpedo utilizando su modelo $3D$ (ver figura 1.8) con un algoritmo genético multi-objetivo.



(a) (b)

Figura 1.7. Robots neumáticos: (a) Robots de una y dos piernas y (b) simetría durante la locomoción (figura tomada de Raibert, 1986).

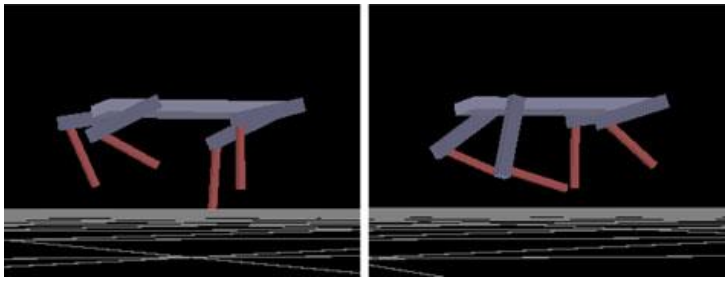


Figura 1.8. Cuadrúpedo galopando (figura tomada de Krasny y Orin, 2010).

En Yin *et al.* (2007) se introduce SIMBICON, un controlador que consiste de una máquina de estados y una heurística para mantener el balance durante la locomoción bípeda. En esta técnica, la posición de la pisada de la pierna que se encuentra en la fase aérea se controla utilizando la velocidad del centro de masas y la distancia del tobillo de la pierna que

se encuentra en la fase de soporte hasta la posición del centro de masas (ver figura 1.9). El sistema resultante muestra una gran robustez ante perturbaciones externas con diferentes patrones de locomoción. Algunos trabajos posteriores que emplean la heurística de SIMBICON son Wang *et al.* (2009) en la síntesis de caminata de una creatura humanoide, Thatte y Geyer (2015) en el desarrollo de una prótesis de pierna y Geijtenbeek *et al.* (2013) en la locomoción de creaturas virtuales con músculos de Hill.

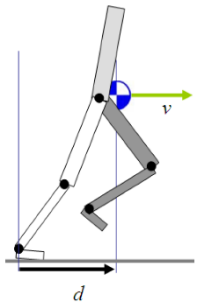


Figura 1.9. Elementos de la heurística SIMBICON (figura tomada de Yin *et al.*, 2007).

En Pratt y Tedrake (2006) se plantea el punto de captura que consiste en el cálculo de la posición del ZMP de forma tal que la velocidad del centro de masas sea cero después de tomar un paso, esta técnica es de utilidad en el balance de postura ante perturbaciones de gran magnitud. Trabajos posteriores extienden el punto de captura a la locomoción. Por ejemplo, en Coros *et al.* (2010) se propone una pequeña modificación a la ecuación del punto de captura y ésta se utiliza para el cómputo de las pisadas de creaturas bípedas virtuales. Mientras que en Coros *et al.* (2011) se utiliza para el cómputo de las pisadas de un cuadrúpedo en simulación.

1.1.3 Balance y control de cuerpo completo

En Pratt (1995) se introduce el control virtual, el cual consiste en definir elementos mecánicos en los puntos deseados a controlar (ver figura 1.10). Estos elementos generan una fuerza que se mapea al cuerpo mediante el Jacobiano transpuesto. Sin embargo, una de las

limitaciones del control virtual es la no capacidad de poder lidiar con los posibles grados de libertad redundantes del sistema al realizar una tarea deseada.

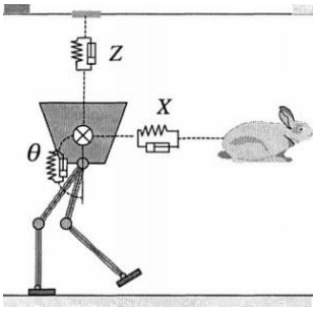


Figura 1.10. Control virtual (figura tomada de Pratt, 1995).

Posteriormente en Pratt *et al.* (2001) se utilizó el control virtual para controlar la caminata de un robot bípedo mediante una máquina de estados. En Coros *et al.* (2010) se propone un controlador genérico para la locomoción bípeda. Esta locomoción consiste en utilizar un control PD para regular las posiciones articulares a una referencia deseada, el control de las pisadas mediante el punto de captura y control virtual por Jacobiano transpuesto. Con esto se agrega compensación por gravedad y se logra controlar la velocidad del centro de masas a un valor deseado. En Coros *et al.* (2011), se extiende el controlador previamente mencionado a la locomoción cuadrúpeda. En Geijtenbeek *et al.* (2013) se emplea control virtual en el balance de creaturas humanoides utilizando trayectorias articulares deseadas que especifican el movimiento a realizar. Las ganancias del controlador se optimizan fuera de línea mediante CMA-ES (Covariance Matrix Adaptation Evolution Strategy).

En Mistry *et al.* (2007) se aborda la locomoción de un robot cuadrúpedo mediante cinemática inversa de base flotante, donde tareas de menor prioridad se proyectan en el espacio nulo de las de mayor prioridad. En Mistry *et al.* (2008) se introduce un algoritmo numérico para el cómputo de la cinemática inversa para sistemas de base flotante y se manejan múltiples tareas en un robot humanoide mediante la proyección iterativa en los espacios nulos. En Mistry *et al.* (2010) se trabaja con la dinámica inversa para sistemas de

base flotante empleando descomposición QR del Jacobiano de los puntos de contacto activos, esto permite proyectar las ecuaciones de movimiento a un espacio reducido y obtener los pares articulares sin la necesidad de conocer las fuerzas de contacto.

Como ya se mencionó, experimentos en biomecánica demuestran que los humanos minimizan el momento angular durante la caminata con un valor aproximadamente de cero (Popovic *et al.*, 2004). Otro ejemplo es el de un gato cayendo, el cual controla su momento centroidal a un valor de cero y re orienta su torso para controlar su aterrizaje ver (figura 1.11). Estas observaciones han motivado el desarrollo de técnicas que controlan el momento centroidal durante el balance y la caminata.

Figura 1.11. Gato cayendo controlando su aterrizaje.

En Macchietto *et al.* (2009) y Lee y Goswami (2010) se estudia el balance de postura regulando el momento lineal y angular empleando la matriz del momento centroidal en creaturas con diversas morfologías (ver figura 1.12) y en superficies disparejas no estacionarias (ver figura 1.13); respectivamente. Mientras que en Liu *et al.* (2019) se realiza control de cuerpo completo regulando el momento centroidal en un robot cuadrúpedo durante la locomoción con trote. En Koolen *et al.* (2016) se desarrolla un controlador empleando la dinámica centroidal y se regula el momento centroidal con un programa cuadrático en el robot Atlas. En los trabajos previamente mencionados, del controlador se obtienen aceleraciones y los pares articulares se calculan en un paso posterior mediante dinámica inversa hibrida de base flotante.

Figura 1.12. Balance de creaturas con diferentes morfologías (figura tomada de Macchietto *et al.*, 2009).

Figura 1.13. Balance de humanoide en superficies no estacionarias (figura tomada de Lee y Goswami, 2012)

Particularmente, para el balance de postura existen diversas técnicas para el balance de postura donde se destacan las realizadas mediante optimización convexa como lo es la programación cuadrática, programación cuadrática jerárquica y programación cónica jerárquica. En Stephens y Atkeson (2010) se analiza el balance de un robot humanoide empleando la dinámica centroidal en un programa cuadrático para el cómputo de las fuerzas de contacto, las cuales se restringen a permanecer dentro de una pirámide de fricción. Estas fuerzas se mapean al cuerpo del robot utilizando la dinámica del cuerpo completo junto con las restricciones cinemáticas (que consisten en mantener los puntos de contacto fijos en una matriz aumentada empleando su pseudo inversa para el cálculo de los pares articulares). En Gehring *et al.* (2015) y Focchi *et al.* (2017) se introduce el balance de un robot cuadrúpedo con la dinámica de un cuerpo rígido empleando control PD para regular la posición y

orientación de la base y se formula un programa cuadrático para el cómputo de las fuerzas de contacto. Estas fuerzas son mapeadas a las piernas del robot mediante control virtual con el Jacobiano transpuesto. En Bledt *et al.* (2018) se agrega un término adicional en la función objetivo penalizando la desviación con respecto a la solución anterior para evitar discontinuidades en los pares articulares durante la locomoción. En Nguyen *et al.* (2019) se utiliza la misma formulación para controlar el aterrizaje después de realizar un salto con el robot Cheetah 3 del MIT.

Otras contribuciones realizan el balance de postura con control de cuerpo completo empleando un programa cuadrático en donde las fuerzas de contracto se restringen a permanecer dentro de un cono de fricción generalizando el criterio de balance del ZMP a superficies irregulares. Por ejemplo, en Abe *et al.* (2007) se introduce un controlador multi-objetivo en donde a cada tarea se le asigna un peso para definir prioridad entre las múltiples tareas deseadas y el controlador resultante se utiliza en el balance de un humanoide en simulación (ver figura 1.14). En Kuindersma *et al.* (2014) se propone un algoritmo eficiente que emplea el método del conjunto activo para la solución de un programa cuadrático durante la caminata en terrenos irregulares con el robot Atlas en simulación. Para incrementar la robustez ante errores de modelado en Feng *et al.* (2014) la dinámica inversa se complementa con la cinemática inversa y ambos problemas se resuelven mediante programación cuadrática, el controlador resultante se utiliza en la caminata del robot Atlas sobre bloques (ver figura 1.15).

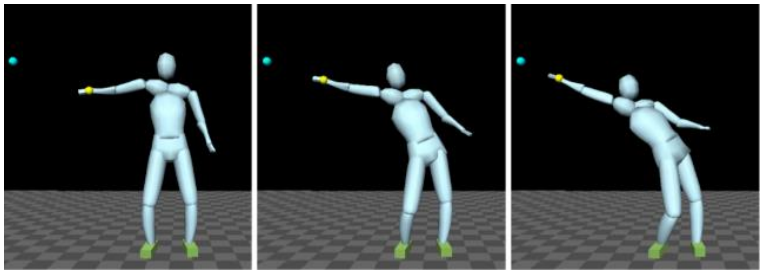


Figura 1.14. Balance de humanoide con diferentes ganancias para especificar la prioridad en las tareas (figura tomada de Abe *et al.,* 2007).

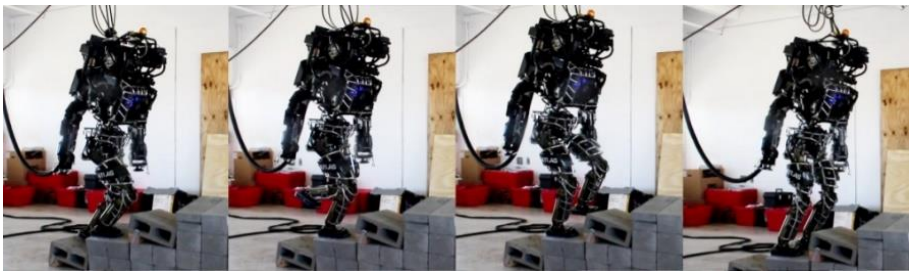


Figura 1.15. Robot Atlas caminando sobre bloques (figura tomada de Feng *et al.*, 2014).

Una de las dificultades en el control de cuerpo completo es el manejo de múltiples tareas que pueden entrar en conflicto. En Sentis y Khatib (2005) se aborda el control de tareas múltiples para sistemas de base flotante mediante proyección iterativa en los espacios nulos. En Herzog *et al.* (2014) el balance de cuerpo completo se formula mediante una jerarquía de tareas y se resuelve mediante programación cuadrática jerárquica (PCJ), en donde las tareas de menor prioridad están restringidas a no modificar las tareas de mayor prioridad; mientras que en Wensing y Orin (2013a) se emplea programación cónica jerárquica. En Escande *et al.* (2014) se reduce el costo computacional de la PCJ proyectando las tareas de menor prioridad en el espacio nulo de las tareas de mayor prioridad mediante descomposición QR, reduciendo así la dimensión de los programas cuadráticos en la jerarquía de tareas; mientras que en De Lasa *et al.* (2010) se emplea descomposición del valor singular y el controlador resultante se utiliza en la locomoción de creaturas con diversas morfologías (ver figura 1.16).

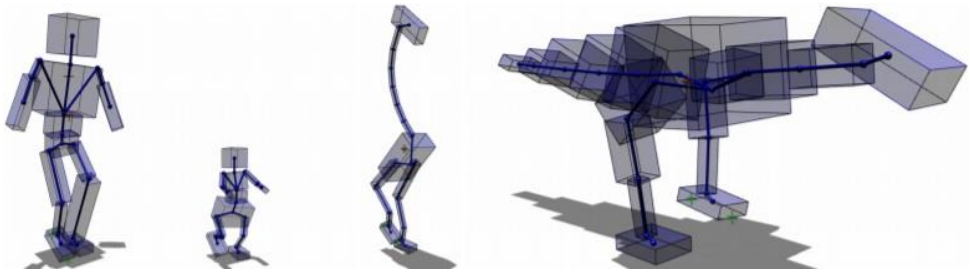


Figura 1.16. Control de cuerpo completo en creaturas con diferentes morfologías (figura tomada de De Lasa *et al.*, 2010).

1.1.4 Péndulo elástico invertido

En Mordatch *et al.* (2010) se propone una simplificación del modelo del péndulo elástico invertido en donde la fase de soporte se modela utilizando un sistema masa resorte en la vertical y el péndulo lineal invertido para el movimiento en el plano paralelo al suelo. Mientras que la fase aérea se modela como un movimiento balístico, las trayectorias se mapean al cuerpo empleando control de cuerpo completo y el controlador resultante se utiliza en la navegación de terrenos irregulares (ver figura 1.17); la optimización de las trayectorias se realiza con CMA-ES. Posteriormente en Mastalli *et al.* (2018) se extiende el trabajo previamente mencionado a la locomoción de un robot cuadrúpedo, mientras que en Apgar *et al.* (2018) la planeación de caminata se formula como un problema de programación no lineal continuo especificando a priori los tiempos de contacto resolviendo mediante el optimizador libre IPOPT.

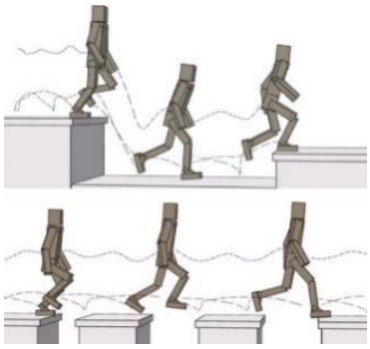


Figura 1.17. Humanoide navegando terrenos irregulares (figura tomada de Mordatch *et al.*, 2010).

En Wensing y Orin (2013b) se introduce la optimización de trayectorias con el modelo del péndulo elástico invertido $3D$, los resultados son mapeados al cuerpo completo mediante programación cónica jerárquica y esto permite el control de un humanoide corriendo a velocidades de hasta $6.5\,m/s$ en simulación (ver figura 1.18).

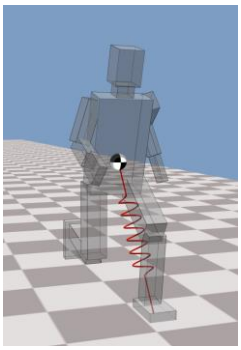


Figura 1.18. Humanoide corriendo en simulación (figura tomada de Wensing y Orin, 2013b).

1.1.5 Cono de contacto

En Hirukawa *et al.* (2006) se propone el cono de contacto como un nuevo criterio de estabilidad (también denominado como "Adiós ZMP"), el cual es aplicable en superficies parejas y disparejas. Este criterio es un reemplazo del ZMP y consiste en mantener la derivada del momento centroidal $\dot{h}_O \in \mathbb{R}^6$ menos la fuerza de gravedad $w_g \in \mathbb{R}^6$ dentro del cono de contacto $\dot{h}_O - w_g \in CWC$. El cono de contacto se define como la suma de Minkowsky de todos los conos de fricción de los puntos de contacto activos (Dai, 2016). En Barthélemy y Bidaud (2008) se introduce un margen de estabilidad $\mathcal{B}_\epsilon$ que mide la robustez del sistema para permanecer dentro del cono de contacto CWC y se define como la máxima perturbación espacial $\epsilon \in \mathbb{R}^6$ de forma tal que la suma de $\dot{h}_O - w_g$ y la perturbación permanece dentro del CWC (ver figura 1.19).

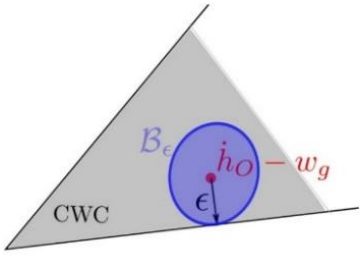


Figura 1.19. Margen de estabilidad en el cono de contacto (figura tomada de Dai, 2016).

En Dai y Tedrake (2016) se aborda la planeación de caminata en superficies irregulares (ver figura 1.20) con optimización convexa empleando el cono de contacto CWC como criterio de estabilidad y se incrementa la robustez de las trayectorias maximizando el margen de estabilidad $\mathcal{B}_\epsilon$, para mantener el problema de optimización convexo se especifican los puntos de contacto a priori.

Figura 1.20. Robot Atlas caminando en superficies disparejas (figura tomada de Dai y Tedrake, 2016).

En Aceituno-Cabezas *et al.* (2017a) se introduce la planeación de trayectorias empleando del cono de contacto CWC como criterio de estabilidad y se optimizan los tiempos y posiciones de los puntos de contacto mediante programación mixta entera, el algoritmo resultante se aplica en la caminata en terrenos irregulares con el modelo del robot LittleDog en simulación (ver figura 1.21).

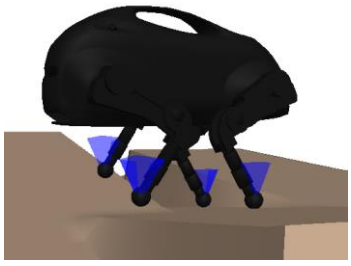


Figura 1.21. Robot LittleDog caminando en superficies disparejas (figura tomada de Aceituno-Cabezas *et al.*, 2017a).

1.1.6 Optimización mixta entera

En Deits y Tedrake (2014) se introduce la planeación de las pisadas de un robot humanoide en terrenos irregulares con un programa cuadrático mixto entero con restricciones cuadráticas, en donde se realiza una aproximación lineal a trozos de las funciones seno y coseno para manejar la orientación del pie; mientras que en Aceituno-Cabezas *et al.* (2016) se extiende el trabajo previamente mencionado a robots con múltiples patas. En Valenzuela (2016) se estudia la planeación sobre plataformas separadas con un modelo simplificado del robot LittleDog (ver figura 1.22) empleando la dinámica de un cuerpo rígido planar y aproximando la no convexidad del momento angular mediante envolturas de McCormick junto con una representación convexa del entorno, se optimizan las posiciones de los puntos de contacto con un programa cuadrático mixto entero.

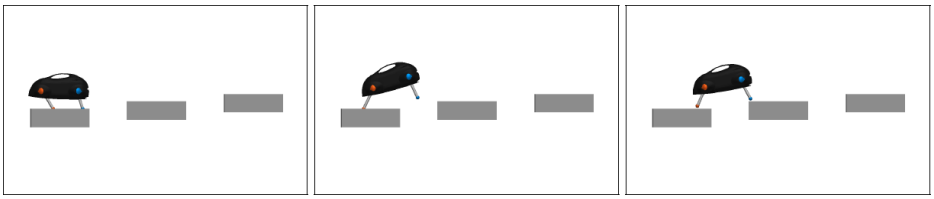


Figura 1.22. Modelo simplificado del robot LittleDog navegando entorno con brechas (figura tomada de Valenzuela, 2016).

En Ponton *et al.* (2016) se introduce un modelo convexo del momento angular centroidal mediante diferencias de funciones cuadráticas, lo que permite formular el problema de planeación como un programa cuadrático con restricciones cuadráticas. Mientras que en Aceituno-Cabezas *et al.* (2017b) se optimiza el patrón de locomoción del robot cuadrúpedo HyQ utilizando el modelo del momento angular propuesto en Ponton *et al.* (2016); de esta manera el problema de planeación es un programa cuadrático mixto entero con restricciones cuadráticas. En Koolen (2019) se propone realizar la planeación de caminata en un entorno con brechas (ver figura 1.23) mediante optimización mixta entera no lineal con el modelo de la dinámica centroidal, las trayectorias se parametrizan mediante

curvas de Bézier y el problema mixto no lineal se resuelve empleando el optimizador libre SCIP con la heurística del vecindario grande adaptivo introducida en Hendel (2018).

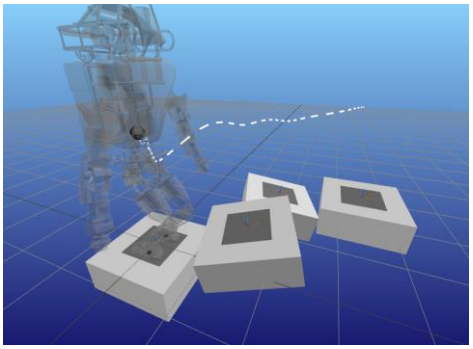


Figura 1.23. Robot Atlas navegando entorno con brechas y superficies disparejas (figura tomada de Koolen, 2019).

1.1.7 Planeación con dinámica de cuerpo completo

En Fang y Pollard (2003) se propone un algoritmo $O(N)$ para el cómputo del Jacobiano de restricciones definidas como la sumas de fuerzas. El algoritmo resultante se utiliza en la planeación de movimientos altamente dinámicos como piruetas y saltos (ver figura 1.24) mediante el método del Lagrangiano aumentado con el optimizador LANCELOT.

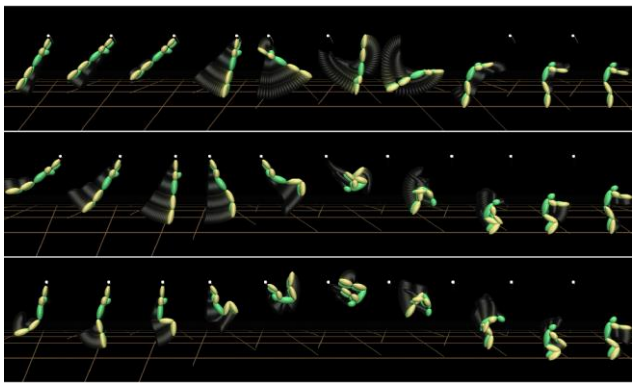


Figura 1.24. Pirueta sintetizada para creatura humanoide (figura tomada de Fang y Pollard, 2003).

En Posa y Tedrake (2013) se introduce la planeación de trayectorias de cuerpo completo con los tiempos y las posiciones de los puntos de contacto mediante restricciones complementarias no lineales inspirado en la formulación para la simulación de cuerpo rígidos como un problema complementario lineal, esto permite sintetizar el patrón de locomoción de robots con patas (ver figura 1.25). El problema de planeación se resuelve utilizando el modo elástico del optimizador SNOPT para la resolución del problema de optimización no lineal.

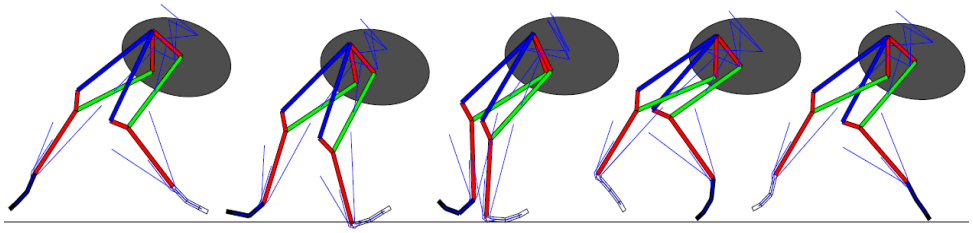


Figura 1.25. Trayectorias optimizadas para el robot FastRunner (figura tomada de Posa y Tedrake, 2013).

En Posa *et al.* (2016) se propone DIRCON, un algoritmo modificado de colocación directa (los estados en cada intervalo de la trayectoria se aproximan con un polinomio cúbico de Hermite) para la síntesis de trayectorias de cuerpo completo en donde la optimización se realiza en la variación de las restricciones de contacto. En Mastalli *et al.* (2016) se propone la planeación jerárquica en donde se utilizan los resultados de la planeación con la dinámica centroidal y la cinemática de cuerpo completo para inicializar la búsqueda en la optimización con la dinámica de cuerpo completo de una de las patas de 3 grados de libertad de un robot cuadrúpedo.

En Hereid *et al.* (2016) se sintetizan trayectorias de cuerpo completo con un método multifase (las transiciones entre las diferentes fases de la locomoción se modelan con la dinámica de impacto), los resultados se experimentan en el robot DURUS, mientras que en Zhao *et al.* (2016) se utiliza la misma metodología en la planeación de trayectorias para una prótesis de pierna. En Pardo *et al.*, 2017 se reduce el costo computacional de la colocación directa en los métodos multifase proyectando la dinámica de cuerpo completo y de impacto

en el espacio tangente a la variedad de las restricciones de contacto. En Katz (2018) se sintetiza una pirueta en el robot mini Cheetah del MIT mientras que en Nguyen *et al.* (2019) se sintetiza un salto en el Cheetah 3 del MIT, en ambos trabajos la trayectoria se divide en 3 fases y en cada fase se minimizan los pares articulares empleando un modelo planar del robot.

1.1.8 Planeación con la dinámica centroidal y de un cuerpo rígido

En Wampler y Popović (2009) se introduce la optimización de trayectorias, tiempos de contacto y las dimensiones de las extremidades de diversas creaturas terrestres (ver figura 1.26) empleando el modelo de la dinámica centroidal; sin embargo, la optimización de los tiempos de contacto es un aspecto discreto no diferenciable, para lidiar con este problema se propone combinar programación cuadrática secuencial para optimizar la parte continua y un optimizador estocástico CMA-ES para lidiar con los tiempos de contacto y los mínimos locales.

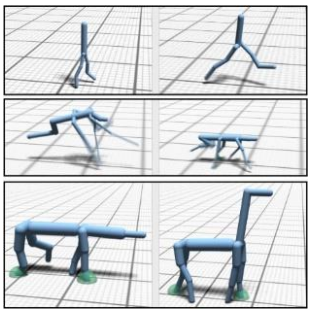


Figura 1.26. Locomoción de creaturas terrestres (figura tomada de Wampler y Popović, 2009).

En Mordatch *et al.* (2012) se aborda la optimización de los eventos de contacto con el modelo de un cuerpo rígido agregando variables adicionales en el espacio de búsqueda que indican cuando un contacto debe estar activo durante el movimiento, lo que permite activar y desactivar las fuerzas de contacto; todas las restricciones se imponen de manera suave en

la función objetivo con una ganancia que se incrementa en cada iteración del algoritmo. Esto permite la síntesis de trayectorias en creaturas con diversas morfologías (ver figura 1.27).

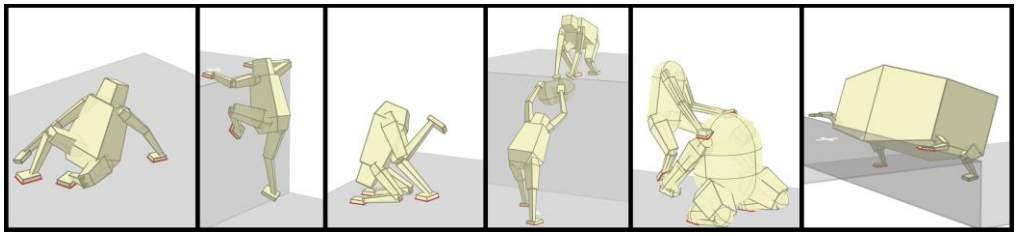


Figura 1.27. Movimientos sintetizados por algoritmo de Mordatch *et al.*, 2012 (figura tomada de Mordatch *et al.*, 2012).

En Dai *et al.* (2014) se introduce la planeación de trayectorias de cuerpo completo empleando el modelo de la dinámica centroidal y la cinemática de cuerpo completo, además se emplean restricciones complementarias para la optimización de los tiempos de contacto. Esto permite la síntesis de trayectorias para el robot Atlas y LittleDog mediante programación cuadrática secuencial con el optimizador comercial SNOPT (ver figura 1.28). Mientras en Herzog *et al.* (2016) se propone una descomposición de la planeación de trayectorias de cuerpo completo en dos problemas que se alternan hasta la convergencia, esos problemas son la planeación con la dinámica centroidal y la planeación cinemática de cuerpo completo.

Figura 1.28. Robot Atlas corriendo (figura tomada de Dai *et al.*, 2014).

En Winkler *et al.* (2018) se optimiza el patrón de locomoción con el modelo $3D$ de un cuerpo rígido mediante la variación del tiempo de soporte en el suelo de las patas con programación no lineal, evitando de esta manera las restricciones complementarias y la

programación entera. En Bledt *et al.* (2017) la dinámica de un cuerpo rígido se linealiza especificando las posiciones de los puntos de contacto mediante heurísticas y los tiempos de contacto a priori, esto permite formular el problema de planeación como un programa cuadrático. El controlador resultante se utiliza en MPC en la locomoción del Cheetah del MIT en simulación (ver figura 1.29). Trabajos posteriores incluyen Di Carlo *et al.* (2018) en donde se experimenta el MPC convexo en el robot Cheetah 3 del MIT permitiendo el control de diversos patrones de locomoción como el trote y el galope, mientras que en Kim *et al.* (2019) las fuerzas obtenidas del MPC se utilizan como referencia en el control de cuerpo completo con programación cuadrática y el controlador resultante se experimenta en el robot Mini-Cheetah del MIT.

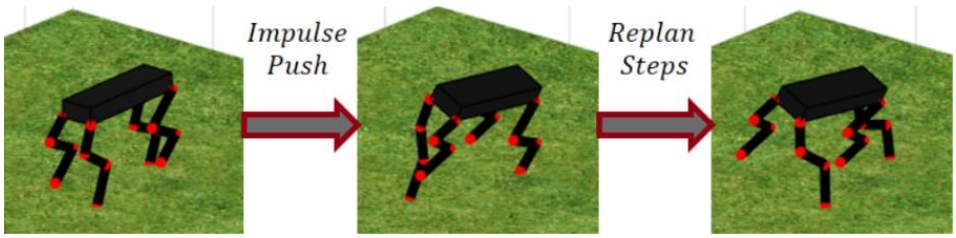


Figura 1.29. Planeación en MPC con el modelo del Cheetah del MIT (figura tomada Bledt *et al.*, 2017).

1.1.9 Planeación con programación dinámica diferencial

En Feng (2016) se emplea programación dinámica diferencial considerando una altura variable en el modelo del péndulo invertido para sintetizar trayectorias del centro de masas, las cuales posteriormente son mapeadas al cuerpo completo mediante programación cuadrática. En Neunert *et al.* (2016) las restricciones complementarias se simplifican especificando los tiempos de contacto a priori, estas restricciones se mueven a la función objetivo mediante el método penalty en conjunto con SQL (control lineal cuadrático secuencial), esto permite sintetizar diferentes patrones de locomoción en un humanoide y cuadrúpedo (ver figura 1.30).

En Neunert *et al.* (2018) se muestra que es posible realizar MPC no lineal con la dinámica del cuerpo completo con experimentos en hardware empleando los robots cuadrúpedos Anymal y HyQ, además se optimizan los tiempos y posiciones de los puntos empleando un modelo de contacto suave mediante LQR iterativo. En Carius *et al.* (2018 y 2019) se propone la optimización de trayectorias de cuerpo completo con los tiempos y posiciones de contacto empleando el integrador (time stepping) de Moreau en conjunto con el LQR iterativo.

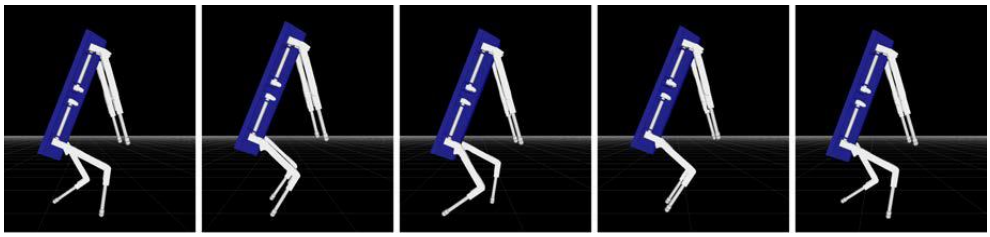


Figura 1.30. Trayectorias optimizadas para el humanoide por algoritmo de Neunert *et al.*, 2016 (figura tomada Neunert *et al.*, 2016).

1.2 Planteamiento del problema

En la planeación de movimientos dinámicos de robots con patas es necesario razonar sobre los aspectos continuos y discretos del entorno y del cuerpo del robot. En general, el robot debe ser capaz de navegar entornos irregulares que contienen superficies disparejas y brechas sin perder el balance. El navegar entornos no acondicionados requiere del control del cuerpo completo y esto a su vez está relacionado con el balance del robot. La problemática actual es el control y la planeación de movimientos dinámicos en terrenos irregulares empleando modelos de cuerpo completo del robot.

1.2.1 Justificación

El objetivo del presente trabajo de tesis consiste en la formulación y desarrollo de nuevos algoritmos para incrementar la robustez en la obtención de la solución de los

problemas de balance de postura y planeación de caminata. Esto debido a que aún no se han resueltos los problemas numéricos asociados al formular el balance de postura y la planeación de caminata como problemas de optimización no convexos. Además la programación no lineal no ofrece garantías y se requiere de la sintonización adecuada de los diversos parámetros del problema y del optimizador para obtener una solución. Por su parte, la formulación del balance de postura y planeación de caminata como problemas de optimización convexos no presentan los problemas numéricos previamente mencionados incrementando robustez y aplicabilidad de los algoritmos.

1.3 Objetivos

1.3.1 Objetivo general.

Analizar diferentes formulaciones de planeación de movimientos dinámicos para la caminata en robots bípedos.

1.3.2 Objetivos específicos.

- Analizar diferentes aproximaciones convexas de la dinámica centroidal.
- Comparar métodos de planeación para la caminata bípeda.
- Simular el control de cuerpo completo utilizando los resultados de la planeación.

1.4 Organización del documento

El presente documento está organizado como se describe a continuación:

En el primer capítulo de esta tesis se detallan los antecedentes y el estado del arte del balance de postura y la planeación de caminata. En el segundo capítulo se introducen las herramientas matemáticas para el desarrollo de los algoritmos propuestos en los capítulos

posteriores, en particular optimización convexa, modelado y simulación de cuerpos rígidos articulados.

En el tercer capítulo se introduce un algoritmo del punto interior con regularización dual en la programación cuadrática para su aplicación en el balance de postura de un robot humanoide empleando control de cuerpo completo. También en este capítulo se proporcionan algunos resultados de simulación de cuerpos rígidos articulados. En el cuarto capítulo se propone un algoritmo $O(N)$ para el cómputo del momento centroidal y su derivada temporal de manera conjunta para su aplicación en el balance de postura con múltiples tareas mediante programación cuadrática jerárquica. El capítulo 4 también presenta simulaciones realizadas mediante Matlab, las cuales permiten realizar algunas conclusiones respecto al desempeño del algoritmo propuesto. En el capítulo 5 se propone un algoritmo jerárquico capaz de lidiar con múltiples tareas en la planeación de trayectorias de cuerpo completo empleando el modelo de la dinámica centroidal y la cinemática de cuerpo completo. Finalmente, en el capítulo 6 se detallan las conclusiones generales, las contribuciones realizadas del presente trabajo de tesis y problemas abiertos como posible trabajo futuro.

CAPÍTULO 2

MARCO TEÓRICO

2.1 Optimización convexa

La optimización convexa considera problemas en donde la función objetivo y las restricciones son funciones convexas, esto incluye un amplio espectro de problemas de optimización como la programación lineal, cuadrática, cónica y semidefinida entre otros (Antoniou y Lu, 2007). Sin embargo, esta sección considera únicamente la programación cuadrática (PC), la cual consiste en minimizar una función objetivo cuadrática sujeta a restricciones lineales de igualdad y desigualdad, esto es:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Qx + c^T x \\ s.t. \quad & Ax = b \\ & Gx \leq d \end{aligned} \tag{2.1}$$

donde $Q \in \mathbb{R}^{n\times n}$ es la matriz Hesiana, $c \in \mathbb{R}^n$ es el vector de constantes del término lineal, $A \in \mathbb{R}^{l\times n}$ es la matriz de las l restricciones de igualdad, $b \in \mathbb{R}^l$ es el vector de constantes de las l restricciones de igualdad, $G \in \mathbb{R}^{m\times n}$ es la matriz de las m restricciones de desigualdad y $d \in \mathbb{R}^m$ es el vector de constantes de las m restricciones de desigualdad. En general se pueden considerar 3 técnicas para la solución de un programa cuadrático, estos son el método del conjunto activo, el método de multiplicadores con direcciones alternadas y el método del punto interior.

2.1.1 Método del conjunto activo

El método del conjunto activo consiste en identificar un subconjunto de desigualdades activas en la solución x^* del programa cuadrático, este subconjunto se denomina el conjunto activo $\mathcal{A}$ y se define como (Nocedal y Wright, 2006):

$$\mathcal{A}(x^*) = \left\{ j \in \mathcal{E} \cup \mathcal{I} \mid G_j x^* = d_j \right\} \tag{2.2}$$

donde $\mathcal{E}$ es el conjunto de los índices de las restricciones de igualdad y $\mathcal{I}$ es el conjunto de los índices de las restricciones de desigualdad. La solución del programa cuadrático (2.1) debe satisfacer las condiciones de optimalidad de Karush Kuhn Tucker, esto es:

$$Qx + A^T y + \sum_{j \in \mathcal{A}} z_j G_j = -c \tag{2.3}$$

$$\begin{aligned} Ax &= b \\ G_j x &= d_j \qquad \forall j \in \mathcal{A} \\ z_j &\geq 0 \qquad \forall j \in \mathcal{A} \\ Gx &\leq d \end{aligned} \tag{2.4}$$

donde $y \in \mathbb{R}^l$ son los multiplicadores de Lagrange de las restricciones de igualdad y $z_j \in \mathbb{R}$ es el multiplicador de Lagrange de la restricción de desigualdad j. El algoritmo identifica el conjunto activo mediante la solución de múltiples programas cuadráticos de igualdad en donde se agrega el índice de la desigualdad $j \notin \mathcal{A}$ al conjunto activo $\mathcal{A}$ si $G_j > d_j$ o se retira si una desigualdad activa $j \in \mathcal{A}$ tiene un multiplicador de Lagrange negativo $z_j < 0$.

El programa cuadrático de igualdad a resolver en cada iteración del algoritmo es equivalente al siguiente sistema de ecuaciones simétrico indefinido:

$$\begin{bmatrix} Q & A^T & G_{\mathcal{A}}^T \\ A & 0 & 0 \\ G_{\mathcal{A}} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ d_{\mathcal{A}} \end{bmatrix} \tag{2.5}$$

donde $G_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}| \times n}$ es el conjunto actual de desigualdades activas y $d_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|}$ es el vector de constantes de las desigualdades activas. En la figura 2.1 se muestra un pseudo código del algoritmo del conjunto activo propuesto en Kuindersma *et al.* (2014).

$k = 0$
while$(k < k_{max})$
$\begin{bmatrix} Q & A^T & G_{\mathcal{A}}^T \\ A & 0 & 0 \\ G_{\mathcal{A}} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ d_{\mathcal{A}} \end{bmatrix}$
$if\ G_j x > d_j$
$\mathcal{A}_{k+1} = \mathcal{A}_k \cup \{j\}$
end
$if\ z_j < 0$
$\mathcal{A}_{k+1} = \mathcal{A}_k \backslash \{j\}$
end
$if\ (Gx < d)\ and\ (z_j \geq 0 \quad \forall j \in \mathcal{A}_k)$
$return\ x$
end
$k = k + 1$
end

Figura 2.1. Algoritmo del conjunto activo.

El algoritmo de la figura 2.1 se itera hasta un número máximo de iteraciones k_{max} o retorna la solución óptima si la solución actual satisface todas las restricciones de desigualdad y todos los multiplicadores de Lagrange de las desigualdades activas son positivos.

2.1.2 Método del punto interior

El método del punto interior encuentra la solución de un problema de optimización convexo tomando una secuencia de pasos desde el interior de la región válida hasta la frontera del conjunto activo mediante el uso de una función de barrera (ver figura 2.2).

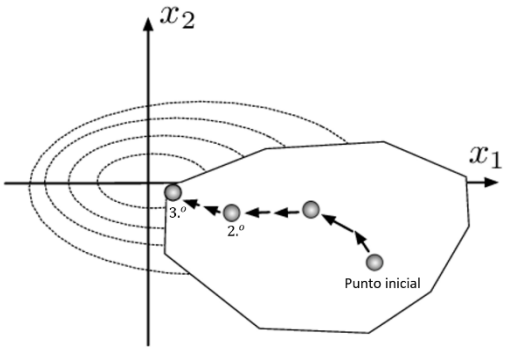


Figura 2.2. Idea del método del punto interior (figura tomada de Wadayama, 2008).

Es posible reformular el programa cuadrático (2.1) al introducir una nueva variable $s \in \mathbb{R}^m$ restringida a tomar valores positivos, esto es:

$$\begin{aligned} \min_{x,s} \quad & \frac{1}{2}x^T Q x + c^T x \\ s.t. \quad & Ax - b = 0 \\ & Gx + s - d = 0 \\ & s \geq 0\,. \end{aligned} \tag{2.6}$$

El método del punto interior convierte un problema restringido a uno sin restricciones al incluir las desigualdades en la función objetivo mediante una función de barrera logarítmica (Nocedal y Wright, 2006), esto es:

$$\begin{aligned} \min_{x,s} \quad & \frac{1}{2}x^T Q x + c^T x - \mu \sum_{j=1}^{m} \log(s_j) \\ s.t. \quad & Ax - b = 0 \\ & Gx + s - d = 0\,. \end{aligned} \tag{2.7}$$

Empleando las condiciones de optimalidad de Karush Kuhn Tucker en (2.7) se obtiene:

$$Qx + c + A^T y + G^T z = 0 \tag{2.8a}$$

$$SZe - \sigma\mu e = 0 \tag{2.8b}$$

$$Ax - b = 0 \tag{2.8c}$$

$$Gx + s - d = 0 \tag{2.8d}$$

$$(s, z) \geq 0 \tag{2.8e}$$

donde $S = diag(s) \in \mathbb{R}^{m\times m}$ es una matriz con s en la diagonal, $Z = diag(z) \in \mathbb{R}^{m\times m}$ es una matriz con z en la diagonal, $I \in \mathbb{R}^{n\times n}$ es la matriz identidad, $e \in \mathbb{R}^m$ es un vector de unos, $\mu = s^T z/m$ es la brecha complementaria y $\sigma \in \mathbb{R}$ es una constante positiva. Reescribiendo las condiciones de optimalidad (2.8) como una función no lineal se obtiene:

$$F(x, s, y, z) = \begin{bmatrix} Qx + c + A^T y + G^T z \\ SZe - \sigma\mu e \\ Ax - b \\ Gx + s - d \end{bmatrix}. \tag{2.9}$$

El método del punto interior consiste en utilizar el método de Newton en (2.9), por lo tanto se tiene que:

$$\begin{bmatrix} Q & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & 0 & 0 \\ G & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} (Qx + c + A^T y + G^T z \\ SZe - \sigma\mu e \\ Ax - b \\ Gx + s - d \end{bmatrix}. \tag{2.10}$$

Una vez resuelto el sistema (2.10) el nuevo punto de búsqueda se obtiene como:

$$(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1}) = (x_k, s_k, y_k, z_k) + \alpha(\Delta x, \Delta s, \Delta y, \Delta z) \tag{2.11}$$

donde la longitud de paso $\alpha \in \mathbb{R}$ se obtiene realizando una búsqueda de línea de forma tal que las condiciones $(s, z) \geq 0$ no se violen. En la figura 2.3 se muestra un pseudo código del algoritmo del punto interior.

$k = 0$
while $E(x_k, s_k, y_k, z_k) \leq \epsilon_{TOL}$
$(x, s, y, z) = (x_k, s_k, y_k, z_k)$
$\mu_k = s^T z / m$
$\sigma_k \in (0, 1)$

$$\begin{bmatrix} Q & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & 0 & 0 \\ G & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} Qx + c + A^T y + G^T z \\ SZe - \sigma \mu_k e \\ Ax - b \\ Gx + s - d \end{bmatrix}$$

$Encontrar\ \alpha\ tal\ que\ (s, z) \geq 0$
$(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1}) = (x_k, s_k, y_k, z_k) + \alpha(\Delta x, \Delta s, \Delta y, \Delta z)$
$k = k + 1$
end

Figura 2.3. Algoritmo del punto interior.

En el algoritmo de la figura 2.3 la función de error se define como:

$$E(x, s, y, z) = \max\{\|Qx + c + A^T y + G^T z\|, \|Sz\|, \|Ax - b\|, \|Gx + s - d\|\} . \quad (2.12)$$

El mayor costo computacional del algoritmo radica en la solución del sistema de ecuaciones indefinido (2.10) el cual se puede resolver mediante factorización simétrica indefinida.

2.1.3 Método de multiplicadores con direcciones alternadas

El método de multiplicadores con direcciones alternadas es una técnica de primer orden y se encuentra inspirado en una combinación del método del ascenso dual y el método del Lagrangiano aumentado, esta última técnica incrementa la robustez del algoritmo al incluir un término cuadrático en la función objetivo del error del cumplimiento de las restricciones. En particular se consideran problemas en los cuales sea posible descomponer

el vector de variables de optimización en partes o sub-vectores. Entonces si se considera una descomposición en 2 partes se tiene que el problema de optimización resultante es:

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ s.t. \quad & Ax + Bz = c \end{aligned} \tag{2.13}$$

donde $x \in \mathbb{R}^n$ y $z \in \mathbb{R}^m$ son las variables de optimización, $f: \mathbb{R}^n \to \mathbb{R}$ y $g: \mathbb{R}^m \to \mathbb{R}$ son las funciones objetivo de las variables x y z respectivamente, $A \in \mathbb{R}^{l \times n}$ y $B \in \mathbb{R}^{l \times m}$ son las matrices de las restricciones de igualdad. Para la solución de (2.13) se considera su Lagrangiano aumentado, esto es:

$$L_\rho = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bx - c\|^2 \tag{2.14}$$

donde $\rho \in \mathbb{R}$ es una ganancia positiva. El método de multiplicadores con direcciones alternadas descompone el problema (2.13) en 2 problemas de optimización que se alternan primero resolviendo por x con la solución previa de z y después por z con la nueva solución de x empleando la función del Lagrangiano aumentado, esto es:

$$x_{k+1} = \underset{x}{\operatorname{argmin}}\, L_\rho(x, z, y_k) \tag{2.15a}$$

$$z_{k+1} = \underset{z}{\operatorname{argmin}}\, L_\rho(x_{k+1}, z, y_k) \tag{2.15b}$$

$$y_{k+1} = y_k + \rho(Ax_{k+1} + Bz_{k+1} - c)\ . \tag{2.15c}$$

El último paso del algoritmo consiste en la actualización de los multiplicadores de Lagrange realizando gradiente ascendiente en el espacio dual con una longitud de paso ρ. La metodología previa se puede generalizar cuando se tiene una descomposición mayor a 2 partes. La ecuación (2.15) se puede reescribir en una forma alternativa que es más conveniente escalando la variable dual (Boyd *et al.*, 2011), esto es:

$$x_{k+1} = \underset{x}{\operatorname{argmin}}(f(x) + (\rho/2)\|Ax + Bz_k - c + u_k\|^2) \tag{2.16a}$$

$$z_{k+1} = \underset{z}{\operatorname{argmin}}(g(z) + (\rho/2)\|Ax_{k+1} + Bz - c + u_k\|^2) \tag{2.16b}$$

$$u_{k+1} = u_k + Ax_{k+1} + Bz_{k+1} - c \tag{2.16c}$$

donde $u = (1/\rho)y$ es la variable dual escalada. En general el método de multiplicadores con direcciones alternadas puede ser aplicado a cualquier problema convexo de la forma:

$$\begin{aligned} \min_{x} \quad & f(x) \\ s.t. \quad & x = \mathcal{C} \end{aligned} \tag{2.17}$$

donde f y $\mathcal{C}$ son convexos. Reescribiendo el problema (2.17) en el formato del método de multiplicadores con direcciones alternadas se obtiene:

$$\begin{aligned} \min_{x} \quad & f(x) + g(z) \\ s.t. \quad & x - z = 0 \end{aligned} \tag{2.18}$$

donde g es la función indicador que se define como $g(z) = 1$ si $z \in \mathcal{C}$ y 0 de otra forma. Por lo tanto, empleando la versión escalada del método de multiplicadores con direcciones alternadas a (2.18) se obtiene el algoritmo que se muestra en la figura 2.4.

$k = 0$
while$(k < k_{max})$
    $x_{k+1} = \underset{x}{\operatorname{argmin}}(f(x) + (\rho/2)\|x - z_k + u_k\|^2)$
    $z_{k+1} = \Pi_{\mathcal{C}}(x_{k+1} + u_k)$
    $u_{k+1} = u_k + x_{k+1} - z_{k+1}$
    $k = k + 1$
end

Figura 2.4. Método de multiplicadores con direcciones alternadas.

En el algoritmo de la figura 2.4 la función $\Pi_{\mathcal{C}}$ realiza una proyección al conjunto convexo $\mathcal{C}$. Dado que el método del multiplicadores es una técnica de primer orden su ritmo de convergencia es $O(1/k)$; sin embargo, es posible acelerar la convergencia del algoritmo mediante sobre relajamiento (Stellato, 2017) y conseguir un ritmo de convergencia de $O(1/k^2)$.

2.2 Álgebra espacial

El álgebra espacial provee una notación compacta para el modelado de la dinámica y cinemática de cuerpos rígidos, consiste de vectores y matrices $6D$, esto permite combinar 2 ecuaciones en una sola (Featherstone, 2008). Los vectores espaciales combinan la parte lineal y angular en una única entidad. En especial se considera la existencia de 2 espacios vectoriales duales M^6 y F^6 en donde los vectores espaciales que denotan movimiento como la velocidad y la aceleración son elementos de M^6, mientras que los vectores espaciales que denotan fuerza o momento son elementos de F^6.

La velocidad espacial $\hat{v} \in \mathbb{R}^6$ (para evitar confusión con notación repetida cuando sea necesario se agregará el símbolo "$\widehat{\ }$" para destacar que se refiere a un vector espacial) se define como:

$$\hat{v} = \begin{bmatrix} \omega \\ v \end{bmatrix} \tag{2.19}$$

donde $\omega \in \mathbb{R}^3$ es la velocidad angular y $v \in \mathbb{R}^3$ es la velocidad lineal. Si se tienen 2 marcos coordenados A y B en donde $v_A \in \mathbb{R}^6$ esta expresado en A, entonces la velocidad espacial $\hat{v}_B \in \mathbb{R}^6$ en el marco B esta dada por:

$$\begin{bmatrix} \omega_B \\ v_B \end{bmatrix} = \begin{bmatrix} R\omega_A \\ (R\omega_A) \times p + Rv_A) \end{bmatrix} \tag{2.20a}$$

$$\begin{bmatrix}\omega_B\\ v_B\end{bmatrix} = \begin{bmatrix}R & 0\\ -p\times R & R\end{bmatrix}\begin{bmatrix}\omega_A\\ v_A\end{bmatrix} \tag{2.20b}$$

donde $p \in \mathbb{R}^3$ es el vector de posición del origen del marco B expresado en B y $R \in SO(3)$ es la matriz de rotación que transforma vectores del marco A al B. De (2.20) se observa que la matriz que transforma velocidades espaciales entre marcos coordenados es:

$$^{B}X_A = \begin{bmatrix}R & 0\\ -p\times R & R\end{bmatrix} \tag{2.21}$$

donde $^{B}X_A \in \mathbb{R}^{6\times 6}$. Por lo tanto la velocidad $\hat{v}_A$ en el nuevo marco coordenado B es:

$$\hat{v}_B = {}^{B}X_A\hat{v}_A\ . \tag{2.22}$$

La misma transformación de (2.22) aplica a las aceleraciones espaciales. De igual manera la fuerza $\hat{f} \in \mathbb{R}^6$ espacial se define como:

$$\hat{f} = \begin{bmatrix}\tau\\ f\end{bmatrix} \tag{2.23}$$

donde $\tau \in \mathbb{R}^3$ es el par y $f \in \mathbb{R}^3$ es la fuerza lineal. Las fuerzas espaciales se transforman de forma diferente a las velocidades y aceleraciones espaciales, por lo tanto la transformación de la fuerza $\hat{f}_A$ al marco B es:

$$\begin{bmatrix}\tau_B\\ f_B\end{bmatrix} = \begin{bmatrix}R^T(\omega_A + p\times f_A)\\ R^T f_A\end{bmatrix} \tag{2.24a}$$

$$\begin{bmatrix}\tau_B\\ f_B\end{bmatrix} = \begin{bmatrix}R^T & R^T p\times\\ 0 & R^T\end{bmatrix}\begin{bmatrix}\tau_A\\ f_A\end{bmatrix}. \tag{2.24b}$$

De (2.24) se tiene que la matriz que transforma fuerzas espaciales entre los marcos coordenados es:

$$ {}^{B}X_{A}^{*} = \begin{bmatrix} R^{T} & R^{T}p\times \\ 0 & R^{T} \end{bmatrix}. \tag{2.25} $$

Expresando (2.24) de forma compacta se obtiene:

$$ \hat{f}_{B} = {}^{B}X_{A}^{*}\hat{f}_{A}\ . \tag{2.26} $$

2.2.1 Producto cruz espacial

Es posible generalizar el operador producto cruz a M^6 y F^6, para ello se definen dos nuevos operadores $\times$ y $\times^*$ que deben cumplir las siguientes ecuaciones (Featherstone, 2008):

$$ \dot{\hat{m}} = \hat{v} \times \hat{m} \tag{2.27a} $$

$$ \dot{\hat{f}} = \hat{v} \times^{*} \hat{f} \tag{2.27b} $$

donde $\hat{v} \in M^6$, $\hat{m} \in M^6$ y $\hat{f} \in F^6$. La representación matricial de los operadores $\hat{v}\times$ y $\hat{v}\times^*$ es la siguiente:

$$ \hat{v}\times = \begin{bmatrix} \omega\times & 0 \\ v\times & \omega\times \end{bmatrix} \tag{2.28a} $$

$$ \hat{v}\times^{*} = \begin{bmatrix} \omega\times & v\times \\ 0 & \omega\times \end{bmatrix}. \tag{2.28b} $$

De (2.28) se observa que $\hat{v}\times^{*} = -(\hat{v}\times)^{T}$.

2.2.2 Inercia espacial

La inercia espacial de un cuerpo rígido define la relación entre la velocidad y momento espacial. Por lo tanto, la inercia espacial se puede considerar un mapeo de M^6 a

F^6. Si se considera un cuerpo rígido con velocidad espacial $v \in M^6$ e inercia espacial $I: M^6 \to F^6$, entonces su momento espacial se define como:

$$h = Iv \tag{2.29}$$

donde $h \in F^6$. La inercia espacial en cualquier punto O de un cuerpo rígido se define como:

$$I_O = \begin{bmatrix} \bar{I}_C + mc \times c \times^T & mc \times \\ mc \times^T & mI \end{bmatrix} \tag{2.30}$$

donde $\bar{I}_C \in \mathbb{R}^{3\times3}$ es la inercia del cuerpo rígido expresado en el centro de masas, $m \in \mathbb{R}$ es la masa y $c \in \mathbb{R}^6$ es el vector de posición del origen de O hasta C. La expresión (2.30) se puede simplificar si la inercia espacial se expresa con respecto al centro de masas, esto es:

$$I_C = \begin{bmatrix} \bar{I}_C & 0 \\ 0 & mI \end{bmatrix}. \tag{2.31}$$

2.2.3 Ecuaciones de movimiento

La ecuación de movimiento espacial de un cuerpo rígido implica que el ritmo de cambio del momento es igual a las fuerzas totales actuando en el cuerpo rígido, esto es:

$$f = \frac{d}{dt}(Iv) = Ia + v \times^* Iv \tag{2.32}$$

donde $a \in \mathbb{R}^6$ es la aceleración espacial. Las ecuaciones $3D$ de Newton Euler se pueden obtener de (2.32). Esto es, si el origen se posiciona en el centro de masas se tiene que:

$$\begin{bmatrix} \tau_C \\ f_C \end{bmatrix} = \begin{bmatrix} \bar{I}_C & 0 \\ 0 & mI \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \ddot{c} - \omega \times v_C \end{bmatrix} + \begin{bmatrix} \omega \times & v_C \times \\ 0 & \omega \times \end{bmatrix} \begin{bmatrix} \bar{I}_C & 0 \\ 0 & mI \end{bmatrix} \begin{bmatrix} \omega \\ v_C \end{bmatrix} \tag{2.33a}$$

$$\begin{bmatrix}\tau_C\\ f_C\end{bmatrix} = \begin{bmatrix}\bar{I}_C\dot{\omega}\\ m\ddot{c} - m\omega \times v_C\end{bmatrix} + \begin{bmatrix}\omega \times \bar{I}_C\omega\\ m\omega \times v_C\end{bmatrix} \tag{2.33b}$$

$$\begin{bmatrix}\tau_C\\ f_C\end{bmatrix} = \begin{bmatrix}\bar{I}_C\dot{\omega} + \omega \times \bar{I}_C\omega\\ m\ddot{c}\end{bmatrix} \tag{2.33c}$$

donde $v_C = \dot{c}$.

2.3 Álgebra y grupos de Lie

Un grupo de Lie G se define como una variedad diferenciable con un elemento identidad e y dos funciones continuas y diferenciables, esto es:

$$mult: G \times G \to G \tag{2.34a}$$

$$inv: G \to G \tag{2.34b}$$

donde $mult$ es una operación binaria denominada operación de grupo y la imagen de un elemento bajo el mapeo inv es la inversa de ese mismo elemento (Selig, 2005). Para simplificar la notación las funciones $mult$ y inv se pueden abreviar como:

$$mult(g_1, g_2) = g_1 g_2 \tag{2.35a}$$

$$inv(g) = g^{-1} \tag{2.35b}$$

donde $g, g_1, g_2 \in G$. Empleando (2.35) se definen los siguientes axiomas para un grupo de Lie:

$$eg = ge = g \quad \text{(Identidad)} \tag{2.36a}$$

$$g^{-1}g = gg^{-1} = e \quad \text{(Inversa)} \tag{2.36b}$$

$$g_1(g_2g_3) = (g_1g_2)g_3 = g_1g_2g_3 \quad \text{(Asociatividad)} \tag{2.36c}$$

donde $g_3 \in G$. El espacio tangente en el elemento identidad del grupo G se denomina el álgebra de Lie del grupo G. El álgebra de Lie junto con el corchete de Lie (un mapeo bilineal), forman un espacio vectorial. El corchete de Lie tiene las siguientes propiedades:

$$\begin{aligned} [x,y] &= -[y,x] && \text{(Simetría)} \\ \big[x,[y,z]\big] + \big[z,[x,y]\big] + \big[y,[z,x]\big] &= 0 && \text{(Identidad de Jacobi)} \end{aligned} \tag{2.37}$$

donde x, y, z son elementos de su respectiva álgebra de Lie.

2.3.1 Grupo SO(3) y álgebra de Lie so(3)

Las matrices de rotación que definen la orientación de un cuerpo rígido $\mathrm{R} \in \mathrm{SO}(3)$ forman parte del grupo ortogonal especial $SO(3)$ y su álgebra de Lie $so(3)$ es el espacio tangente a la identidad $I \in \mathbb{R}^{3\times3}$. Si la función $\gamma(t) = \mathrm{R(t)}$ define una curva de $SO(3)$ parametrizada por $t \in \mathbb{R}$ que pasa por la identidad, donde $\mathrm{R}(0) = \mathrm{I}$ y $\mathrm{R(t)R(t)}^T = I$. Diferenciando la última relación y evaluando en $t = 0$ resulta en $\dot{\mathrm{R}}(0) + \dot{\mathrm{R}}(0)^T = 0$. Por lo tanto, el álgebra de Lie de $SO(3)$ consiste en matrices anti-simétricas de la forma:

$$\omega \times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.38}$$

donde $\omega \in \mathbb{R}^3$ es la velocidad angular.

2.3.2 Grupo SE(3) y álgebra de Lie se(3)

Las transformaciones espaciales de los cuerpos rígidos expresadas como matrices homogéneas forman un grupo de Lie denominado el grupo Euclidiano $SE(3)$. Si se tiene la

matriz de rotación R ∈ SO(3) y el vector de translación $p \in \mathbb{R}^3$, $SE(3)$ esta dado por matrices de la forma:

$$T = \begin{bmatrix} \mathrm{R} & p \\ 0 & 1 \end{bmatrix} \tag{2.39}$$

donde $T \in SE(3)$. El álgebra de Lie de $SE(3)$ denominada $se(3)$ esta dado por matrices de la forma:

$$g = \begin{bmatrix} \omega \times & v \\ 0 & 1 \end{bmatrix} \tag{2.40}$$

donde $g \in se(3)$ y $v \in \mathbb{R}^3$ es la velocidad lineal. También es posible expresar elementos de $se(3)$ mediante vectores espaciales, esto es:

$$g = \begin{bmatrix} \omega \\ v \end{bmatrix} \tag{2.41}$$

donde $g \in \mathbb{R}^6$. En particular los vectores de velocidad y aceleración espacial son elementos del álgebra de Lie $se(3)$ ($[\omega^T \quad v^T]^T \in se(3)$), mientras que las fuerzas y los momentos se consideran que pertenecen al espacio dual $se(3)^*$ ($[\tau^T \quad f^T]^T \in se(3)^*$), esto es debido a que las fuerzas se transforman de manera diferente ante un cambio de coordenadas.

2.3.3 Representación adjunta

El mapeo de representación adjunta (Adjoint) realiza una transformación de coordenadas en $se(3)$ empleando el mapeo $SE(3)$ entre marcos coordenados, esto permite transformar elementos de $se(3)$ de un sistemas de referencia a otro. El mapeo de representación adjunta, que se denota $Ad_G(h): SE(3) \times se(3) \rightarrow se(3)$ se define como:

$$Ad_G(h) = GhG^{-1} \tag{2.42}$$

donde $G \in SE(3)$ es una transformación entre marcos coordenados y $h \in se(3)$. El operador de representación adjunta puede operar en vectores espaciales $h = [\omega_h^T \quad v_h^T]^T \in \mathbb{R}^6$, esto es:

$$Ad_G(h) = \begin{bmatrix} \mathrm{R} & 0 \\ \mathrm{p} \times \mathrm{R} & \mathrm{R} \end{bmatrix} \begin{bmatrix} \omega_h \\ v_h \end{bmatrix}. \tag{2.43}$$

De igual manera, el operador de representación adjunta dual Ad^* realiza una transformación de coordenadas en el espacio dual $se(3)^*$. El mapeo de representación adjunta dual $Ad_G^*(h^*): SE(3) \times se(3)^* \to se(3)^*$ se define como:

$$Ad_G^*(h^*) = Gh^*G^{-1} \tag{2.44}$$

donde $h^* \in se(3)^*$. También es posible operar en vectores espaciales $h^* = [\tau_h^T \quad f_h^T]^T \in \mathbb{R}^6$, esto es:

$$Ad_G^*(h^*) = \begin{bmatrix} \mathrm{R}^T & \mathrm{R}^T \mathrm{p} \times^T \\ 0 & \mathrm{R}^T \end{bmatrix} \begin{bmatrix} \tau_h \\ f_h \end{bmatrix}. \tag{2.45}$$

Se observa de (2.43) y (2.45) que la matriz de transformación de Ad_G^* es la transpuesta de Ad_G. El corchete de Lie es un mapeo lineal sobre sí mismo. Si se denota como $ad_g(h): se(3) \to se(3)$ al corchete de Lie entonces se tiene que:

$$ad_g(h) = [g, h] = gh - hg \tag{2.46}$$

donde $g, h \in se(3)$. Una expresión alternativa empleando los vectores espaciales $g \in \mathbb{R}^6$ y $h \in \mathbb{R}^6$ es:

$$ad_g(h) = \begin{bmatrix} \omega_g \times & 0 \\ v_g \times & \omega_g \times \end{bmatrix} \begin{bmatrix} \omega_h \\ v_h \end{bmatrix}. \tag{2.47}$$

De igual manera el corchete de Lie dual $ad_g^*(h^*): se(3)^* \to se(3)^*$ se define como:

$$ad_g^*(h^*) = [g, h^*] = gh^* - h^* g \ . \tag{2.48}$$

La otra forma equivalente de (2.48) es:

$$ad_g^*(h^*) = \begin{bmatrix} \omega_g \times^T & v_g \times^T \\ 0 & \omega_g \times^T \end{bmatrix} \begin{bmatrix} \omega_h \\ v_h \end{bmatrix}. \tag{2.49}$$

Los mapeos $ad_g(h)$ y $ad_g^*(h^*)$ se pueden considerar como una generalización del producto cruz a los espacios $se(3)$ y $se(3)^*$.

2.3.4 Cinemática directa

La cinemática directa de un robot se puede modelar mediante la composición de una secuencia de transformaciones homogéneas de los respectivos marcos coordenados articulares. Si $T_{i-1,i} \in SE(3)$ denota la transformación entre los marcos coordenados $i-1$ al i, entonces la cinemática directa del órgano terminal de un robot con N eslabones esta dada por:

$$T_e = T_{0,1} T_{1,2} \dots T_{N-2,N-1} T_{N-1,N} \tag{2.50}$$

donde $T_e \in SE(3)$ contiene la orientación y posición del órgano terminal con respecto al marco coordenado global. La transformación homogénea $T_{i-1,i}$ se puede representar como:

$$T_{i-1,i} = M_i e^{S_i q_i} \tag{2.51}$$

donde $M_i \in \mathbb{R}^{4\times4}$ es una translación entre los marcos coordenados $i-1$ al i, $S_i \in se(3)$ es el eje de giro del eslabón i y $q_i \in se(3)$ es la posición articular del eslabón i. El utilizar (2.51) para realizar las transformaciones entre los diversos marcos coordenados del robot permite obtener sus derivadas de manera relativamente sencilla, esto último se puede utilizar en el cálculo las derivadas analíticas de las ecuaciones de dinámica o cinemática del robot.

2.4 Balance de postura

En el balance de postura o en la misma caminata es necesario un criterio de estabilidad, estos pueden ser tan sencillos como el mantener la posición del centro de masas o el ZMP dentro del polígono de soporte; sin embargo, estos últimos criterios presentan limitaciones severas ya que ignoran la fricción en los contactos (además el criterio del ZMP no es aplicable en superficies irregulares). Un criterio alternativo consiste en restringir las fuerzas de contacto a permanecer dentro de sus conos de fricción (este criterio es una generalización del ZMP a superficies irregulares (Abe *et al.,* 2007)). Las fuerzas de contacto se suelen modelar con la fricción de Coulomb, por lo tanto, las fuerzas están restringidas por un cono de fricción que se define como:

$$\mathcal{K} = \{\lambda \in \mathbb{R}^3 \mid n^T\lambda \geq 0, \|\lambda - (n^T\lambda)n\| \leq \mu n^T\lambda\} \tag{2.52}$$

donde $n \in \mathbb{R}^3$ es un vector tangente a la superficie de contacto y $\mu \in \mathbb{R}$ es el coeficiente de fricción. El cono $\mathcal{K}$ es normalmente remplazado por una pirámide de fricción, esto es:

$$\mathcal{P} = \left\{ \sum_{j=1}^{N_d} \beta_j w_j \mid \beta_j \geq 0 \right\} \tag{2.53}$$

donde $w_j = n + \mu d_j$ es uno de los vectores base, $d_j \in \mathbb{R}^3$ es un vector tangente a la superficie de contacto y $N_d \in \mathbb{Z}$ es el número de vectores base empleados para aproximar el cono de fricción de Coulomb. Dado que (2.53) es una aproximación de (2.52) se cumple que $\mathcal{P} \subset \mathcal{K}$. En la literatura existen diversas técnicas para el balance de postura empleando diversos modelos como la dinámica de cuerpo completo, dinámica centroidal y dinámica de un cuerpo rígido. En general, se suele formular un problema de optimización convexo con el modelo dinámico como una restricción junto con las fuerzas de contacto restringidas a permanecer dentro de un cono o pirámide de fricción como criterio de estabilidad.

2.4.1 Balance de postura con dinámica de un cuerpo rígido

Este tipo de balance en la mayoría de los casos se suele aplicar en robots cuadrúpedos en donde se supone que la masa de las patas del robot es despreciable en comparación con la masa de la base. Por lo tanto, el robot se puede modelar como un cuerpo rígido. Su dinámica está dada por las siguientes ecuaciones (Valenzuela, 2016):

$$I_G \dot{\omega} + \omega \times I_G \omega = \sum_{j=1}^{N_c} (c_j - r) \times \lambda_j \tag{2.54a}$$

$$m\ddot{r} = mg + \sum_{j=1}^{N_c} \lambda_j \tag{2.54b}$$

donde $m \in \mathbb{R}$ es la masa del robot, $I_G \in \mathbb{R}^{3\times3}$ es la matriz de inercia, $g \in \mathbb{R}^3$ es la aceleración de gravedad, $\lambda_j \in \mathbb{R}^3$ es la fuerza en el punto de contacto j, $\omega \in \mathbb{R}^3$ es la velocidad angular, $c_j \in \mathbb{R}^3$ es la posición del punto de contacto j y $r \in \mathbb{R}^3$ es la posición del centro de masas y $N_c \in \mathbb{Z}$ es el número de puntos de contacto. Si se supone que el sistema está en equilibrio estático, (2.54) se simplifica a:

$$\begin{bmatrix} I_G\dot{\omega} \\ m\ddot{r} - mg \end{bmatrix} = \begin{bmatrix} (c_1 - r)\times & \cdots & (c_{N_c} - r)\times \\ I & \cdots & I \end{bmatrix} \lambda \tag{2.55}$$

donde $I \in \mathbb{R}^{3\times3}$ es la matriz identidad, $\lambda \in \mathbb{R}^{3N_c}$ es el vector de las N_c fuerzas de contacto. Supóngase que el objetivo del controlador de balance es regular la posición y orientación de la base mediante control PD, esto es:

$$\dot{\omega}_d = K_{p,\omega} log(R_d R^T) + K_{v,\omega}(\omega_d - \omega) \tag{2.56a}$$

$$\ddot{r}_d = K_{p,r}(r_d - r) + K_{v,r}(\dot{r}_d - \dot{r}) \tag{2.56b}$$

donde $K_{p,\omega} \in \mathbb{R}^{3\times3}$, $K_{v,\omega} \in \mathbb{R}^{3\times3}$, $K_{p,r} \in \mathbb{R}^{3\times3}$ y $K_{v,r} \in \mathbb{R}^{3\times3}$ son matrices definidas positivas de ganancias proporcional y derivativa, respectivamente. De (2.55) y (2.56) se define la siguiente tarea:

$$A = \begin{bmatrix} (c_1 - r)\times & \cdots & (c_{N_c} - r)\times \\ I & \cdots & I \end{bmatrix} \tag{2.57a}$$

$$b = \begin{bmatrix} I_G\dot{\omega}_d \\ m\ddot{r}_d - mg \end{bmatrix} \tag{2.57b}$$

donde $A \in \mathbb{R}^{6\times3N_c}$ y $b \in \mathbb{R}^{3N_c}$. Entonces el problema de optimización a resolver en cada ciclo de control para el balance de postura es (Gehring *et al.*, 2015):

$$\begin{aligned} \min_{\lambda} \quad & (A\lambda - b)^T(A\lambda - b) + \alpha\lambda^T\lambda \\ s.t. \quad & \forall j \;\; \lambda_j \in \mathcal{P} \end{aligned} \tag{2.58}$$

donde $\alpha \in \mathbb{R}$ es una constante positiva que regulariza la solución de (2.58). Una vez que se obtienen las fuerzas de contacto mediante el PC (2.58) éstas se mapean a las patas del robot empleando control virtual mediante el Jacobiano transpuesto, esto es:

$$\tau = J^T\lambda \tag{2.59}$$

donde $J \in \mathbb{R}^{3\times(6+N_J)}$ es el Jacobiano aumentado de todos los puntos de contacto, $\tau \in \mathbb{R}^{N_J}$ los pares articulares y n_J es el número de grados de libertad del sistema sin tener en cuenta la posición y orientación de la base.

2.4.2 Balance de postura con dinámica centroidal

Un modelo alternativo que captura información del cuerpo completo es la dinámica centroidal. En este caso la dinámica centroidal se expresará empleando vectores espaciales. El momento centroidal se define como:

$$h = \begin{bmatrix} k \\ l \end{bmatrix} \tag{2.60}$$

donde $l \in \mathbb{R}^3$ es el momento lineal y $k \in \mathbb{R}^3$ es el momento angular. Igualando la derivada temporal del momento centroidal a la sumatoria de todas las fuerzas espaciales se obtiene la dinámica centroidal, esto es:

$$\dot{h} = W_g + A\lambda \tag{2.61}$$

donde $W_g = [0^T \quad mg^T]^T \in \mathbb{R}^6$ es la fuerza de gravedad y $A \in \mathbb{R}^{6\times 3N_c}$ se define como en (2.57a). En Orin y Goswami (2013) se muestra que existe una relación lineal entre el momento centroidal y las velocidades articulares, esto es:

$$h = A_G(q)\dot{q} \tag{2.62}$$

donde $A_G \in \mathbb{R}^{6\times(6+N_J)}$ es la matriz del momento centroidal. Sustituyendo la derivada temporal de (2.62) en (2.61) se obtiene:

$$A_G(q)\ddot{q} + \dot{A}_G(q)\dot{q} = W_g + A\lambda \ . \tag{2.63}$$

Utilizar (2.63) en el problema de balance en lugar de la dinámica del cuerpo completo permite reducir el costo computacional del controlador. Para evitar que los puntos de contacto se desplacen con respecto a la superficie de contacto es necesario especificar la siguiente tarea:

$$\forall j \ J_j(q)\ddot{q} + \dot{J}_j(q)\dot{q} = 0 \tag{2.64}$$

donde $J_j \in \mathbb{R}^{3\times(6+N_J)}$ es el Jacobiano del punto de contacto j. Si se define f como la función objetivo, entonces de (2.63) y (2.64) el problema de optimización a resolver en cada ciclo de control para el problema de balance es (Koolen *et al.*, 2016):

$$\begin{aligned} \min_{\ddot{q},\lambda,\beta} \quad & f(\ddot{q},\lambda,\beta) \\ s.t. \quad & A_G(q)\ddot{q} + \dot{A}_G(q)\dot{q} = W_g + A\lambda \\ & \forall j \ J_j(q)\ddot{q} + \dot{J}_j(q)\dot{q} = 0 \\ & \forall j \ \lambda_j \in \mathcal{P} \ . \end{aligned} \tag{2.65}$$

Una vez obtenidas las aceleraciones articulares del PC (2.65) estas son mapeadas a pares articulares mediante el algoritmo recursivo de Newton Euler para sistemas de base flotante.

2.4.3 Balance de postura con dinámica de cuerpo completo

Otra forma de realizar el balance de postura es emplear la dinámica del cuerpo completo, la cual se define como:

$$M(q)\ddot{q} + h(q,q) = S^T\tau + J^T(q)\lambda \tag{2.66}$$

donde $M(q) \in \mathbb{R}^{(6+n_J)\times(6+n_J)}$ es la matriz de inercia de base flotante, $h_b(q,\dot{q}) \in \mathbb{R}^{6+n_J}$ son las fuerzas de Coriolis, centrífugas y gravedad, $S = [I_{n_J \times n_J} \quad 0_{n_J \times 6}]$ es la matriz de selección de pares actuados y $\tau \in \mathbb{R}^{n_J}$ es el vector de pares articulares. Combinando la restricción cinemática (2.64) y la dinámica (2.66) el problema de optimización a resolver en cada ciclo de control para el problema de balance es:

$$\begin{aligned} \min_{\ddot{q},\tau,\lambda,\beta} \quad & f(\ddot{q},\tau,\lambda,\beta) \\ s.t. \quad & M(q)\ddot{q} + h\big(q,\ \dot{q}\big) = S^T\tau + J^T(q)\lambda \\ & \forall j \ \ J_j(q)\ddot{q} + \dot{J}_j(q)\dot{q} = 0 \\ & \forall j \ \ \lambda_j \in \mathcal{P} \\ & \tau_l \leq \tau \leq \tau_u \end{aligned} \tag{2.67}$$

donde $\tau_l \in \mathbb{R}^{N_J}$ y $\tau_u \in \mathbb{R}^{N_J}$ son los límites inferiores y superiores en los pares articulares, respectivamente. Una de las ventajas de la formulación (2.67) con respecto de (2.65) es la capacidad de poder penalizar e introducir límites en los pares articulares.

2.4.4 Programación cuadrática jerárquica (PCJ)

Cuando un robot es redundante es posible ejecutar múltiples tareas de manera simultánea. Para lograr esto, se define una jerarquía de tareas con prioridades de forma tal que las tareas de menor prioridad se ejecutan en el espacio nulo de las tareas de mayor prioridad. La jerarquía de tareas se puede resolver mediante proyección iterativa en los espacios nulos o PCJ; sin embargo, una de las desventajas de la primera técnica es su incapacidad para incluir límites cinemáticos o dinámicos en el proceso de solución lo que limita su aplicación, mientras que la PCJ no posee esta desventaja. En la PCJ es necesaria la solución de una cascada de PC, si se considera una jerarquía de n tareas, entonces es necesario la solución de una serie de PC de la forma (Rocchi *et al.*, 2015):

$$
\begin{aligned}
x_n^* = \quad & \underset{x}{\operatorname{argmin}} \quad \frac{1}{2}\|A_n x - b_n\|^2 \\
& s.t. \quad A_1 x = A_1 x_1^* \\
& \qquad \vdots \\
& \qquad A_{n-1} x = A_{n-1} x_{n-1}^* \\
& \qquad l_1 \leq A_1 x \leq u_1 \\
& \qquad \vdots \\
& \qquad l_{n-1} \leq A_{n-1} x \leq u_{n-1} \\
& \qquad l_n \leq A_n x \leq u_n
\end{aligned}
\tag{2.68}
$$

donde x_n^* es la solución de la jerarquía de tareas, x_j^* es la solución de la tarea j, l_j denota un límite inferior en la tarea j, u_j denota un límite superior de la tarea j, A_j y b_j definen la tarea j. Es posible reducir el costo computacional de la formulación (2.68) optimizando en el espacio nulo de las tareas de mayor prioridad mediante descomposición QR (Escande *et al.,* 2014).

2.5 Planeación de caminata

La planeación de trayectorias de un robot con patas se puede formular como un problema de control óptimo. En general, se pueden considerar dos categorías para la solución del problema de planeación: disparo (shooting) y colocación directa (direct collocation). El método de disparo consiste en optimizar una parametrización de la señal de control en donde los estados se obtienen mediante integración numérica (Remy, 2011). Mientras la colocación directa consiste en la optimización simultánea de los estados y entradas de control. El problema de control óptimo para la planeación de caminata se puede expresar de manera canónica como (Valenzuela, 2016):

$$
\begin{aligned}
\min_{x,u} \quad & \Psi(x(T)) + \int_0^T \psi(x(t), u(t),) dt \\
s.t. \quad & \dot{x} = f(x(t), u(t)) \\
& x \in \mathcal{X} \\
& u \in \mathcal{U}
\end{aligned}
\tag{2.69}
$$

donde ψ y Ψ son las funciones costo intermedia y final, $f(x,u)$ es la dinámica del sistema, $T \in \mathbb{R}$ es la longitud del horizonte de planeación, $\mathcal{X} \in \mathbb{R}^{6\times(6+N_J)}$ y $\mathcal{U} \in \mathbb{R}^{6\times(6+N_J)}$ representan los conjuntos de los estados y entradas de control admisibles por el sistema. En la práctica el problema de optimización (2.69) se resuelve discretizando la dinámica del sistema empleando algún método de integración numérica o interpolación por polinomios y la función objetivo se aproxima por una sumatoria.

2.5.1 Planeación con dinámica de cuerpo completo

Es posible sintetizar trayectorias articulares de cuerpo completo empleando la dinámica (2.66) en la formulación del problema de control óptimo. Sin embargo, para mantener el problema de optimización como continuo es necesario definir el patrón de locomoción a priori ya que la síntesis del patrón de locomoción convierte el problema de planeación, en su caso más general, en una optimización mixta entera no lineal. Una alternativa para manejar el aspecto combinatorial de la planeación de caminata como un problema de optimización no lineal es el introducido en Posa y Tedrake (2013) mediante el uso de restricciones complementarias, esto es:

$$\phi_j(q)\lambda_j^N = 0 \tag{2.70}$$

donde $\phi(q)\colon \mathbb{R}^{6+N_J} \to \mathbb{R}$ es la función cinemática directa que define la altura del punto de contacto j con respecto a la superficie de contacto y $\lambda_j^N \in \mathbb{R}$ es la fuerza normal. En la formulación del problema de planeación de caminata como un problema de optimización no lineal, la dinámica (2.66) se discretiza empleando integración de Euler implícita, esto es:

$$q_{k+1} - q_k - \dot{q}_{k+1}\Delta t = 0 \tag{2.71a}$$

$$H_{k+1}(\dot{q}_{k+1} - \dot{q}_k) + (h_{k+1} - S^T\tau_{k+1} - J_{k+1}^T\lambda_{k+1})\Delta t = 0 \tag{2.71b}$$

donde $\Delta t \in \mathbb{R}$ es el paso de integración. Si se define el conjunto de variables a optimizar:

$$\Gamma = \{q_k, \dot{q}_k, \tau_k, \lambda_k | \; k = 1, \dots, N\} \tag{2.72}$$

donde $\lambda_k = \left[\lambda_1^T, \dots, \lambda_{N_c}^T\right]^T \in \mathbb{R}^{3N_c}$. Entonces, de (2.70) y (2.71) el problema de planeación con la dinámica del cuerpo completo se muestra en (P1):

$$\begin{aligned} \underset{\Gamma}{min} \quad & f(\Gamma) \\ s.t. \quad & H_{k+1}(\dot{q}_{k+1} - \dot{q}_k) + (h_{k+1} - S^T\tau_{k+1} - J_{k+1}^T\lambda_{k+1})\Delta t = 0 \qquad \forall\, k \in \{1, \dots, N\} \\ & q_{k+1} - q_k - \dot{q}_{k+1}\Delta t = 0 \\ & \forall j \;\; \phi_{j,k}(q) \geq 0 \\ & \forall j \;\; \phi_{j,k}(q)\lambda_{j,k}^N = 0 \\ & \forall j \;\; \lambda_{j,k} \in \mathcal{K} \end{aligned} \tag{P1}$$

La formulación (P1) es general y es aplicable para la síntesis de diversos tipos de movimientos como saltos, piruetas, etc. El problema (P1) se puede resolver empleando cualquier técnica de programación no lineal como programación cuadrática secuencial (Posa *et al.*, 2016), métodos del punto interior no lineal (Mastalli *et al.*, 2016) y Lagrangiano Aumentado (Fang y Pollard, 2003).

2.5.2 Planeación con dinámica centroidal y cinemática de cuerpo completo

Una formulación alternativa para la planeación de trayectorias de cuerpo completo introducida en Dai *et al.* (2014) consiste en emplear la dinámica centroidal y la cinemática de cuerpo completo en lugar de (2.66) en el problema de control óptimo. El modelo de la dinámica centroidal se define como:

$$m\ddot{r} = mg + \sum_{j=1}^{N_c} \lambda_j \tag{2.73a}$$

$$\dot{k} = \sum_{j=1}^{N_c} (c_j - r) \times \lambda_j \tag{2.73b}$$

donde $r \in \mathbb{R}^3$ es la posición del centro de masas. Dado que el vector de configuración $q \in \mathbb{R}^{6+N_J}$ y su derivada $\dot{q} \in \mathbb{R}^{6+N_J}$ son variables de optimización es necesario introducir restricciones de consistencia cinemática para el momento angular, posiciones de los puntos de contacto y el centro de masas, esto es:

$$k = A_G^u(q)\dot{q} \tag{2.74a}$$

$$r = COM(q) \tag{2.74b}$$

$$c_j = FK_j(q) \tag{2.74c}$$

donde $A_G^u \in \mathbb{R}^{3\times(6+N_J)}$ es la parte superior de la matriz del momento centroidal, $COM(q)\colon \mathbb{R}^{6+N_J} \to \mathbb{R}^3$ es la función cinemática directa del centro de masas y $FK_j(q)\colon \mathbb{R}^{6+N_J} \to \mathbb{R}^3$ es la función cinemática directa del punto de contacto j. El nuevo conjunto de variables a optimizar es:

$$\Gamma = \left\{q_k, \dot{q}_k, h_k, \dot{h}_k, c_k, \lambda_k, r_k, \dot{r}_k, r_k \mid k = 1, \dots, N\right\} \tag{2.75}$$

donde $c_k = \left[c_1^T, \dots, c_{N_c}^T\right]^T \in \mathbb{R}^{3N_c}$. Empleando integración de Euler implícita en la dinámica centroidal y utilizando restricciones complementarias para optimizar el patrón de locomoción el problema de planeación se muestra en (P2).

Utilizar la formulación (P2) en lugar de (P1) resulta en un problema de optimización más pequeño e incrementa la robustez a la sensibilidad de la variación de los diversos parámetros del problema o el optimizador.

$$\min_{\Gamma} \quad f(\Gamma) \tag{P2}$$

$$s.t. \qquad \forall\, k \in \{1, \dots, N\}$$

$$m\ddot{r}_k = mg + \sum_{j=1}^{N_c} \lambda_{j,k}$$

$$\dot{k}_k = \sum_{j=1}^{N_c} (c_{j,k} - r_k) \times \lambda_{j,k}$$

$$k_k = A_G(q_k)\dot{q}_k$$

$$r_k = COM(q_k)$$

$$\forall j \;\; c_{j,k} = FK_j(q_k)$$

$$\dot{r}_{k+1} = \dot{r}_k + \ddot{r}_{k+1}\Delta t$$

$$r_{k+1} = r_k + \dot{r}_{k+1}\Delta t$$

$$k_{k+1} = k_k + \dot{k}_{k+1}\Delta t$$

$$q_{k+1} = q_k + \dot{q}_{k+1}\Delta t$$

$$\forall j \;\; \phi_{j,k}(q) \geq 0$$

$$\forall j \;\; \phi_{j,k}(q)\lambda_{j,k}^{N} = 0$$

$$\forall j \;\; \lambda_{j,k} \in \mathcal{K}$$

2.6 Simulación de cuerpos rígidos articulados

En la simulación de cuerpos rígidos articulados se pueden considerar 3 paradigmas, estos varían dependiendo de la forma de modelar la naturaleza híbrida de los contactos; ya sea mediante un problema complementario lineal o no lineal, modelos de contacto suave con optimización o resortes con amortiguadores y mediante inclusiones diferenciales.

2.6.1 Problema complementario lineal

El problema de la dinámica directa con contactos unilaterales se suele formular al nivel de la velocidad y la aceleración como un problema complementario lineal (Stewart y Trinkle, 1996). Sin embargo, la formulación al nivel de la aceleración no siempre tiene solución, una alternativa en la cual siempre existe solución es formular la dinámica directa al nivel de la velocidad mediante la discretización de la dinámica y a esto se le conoce como

"time stepping" (Studer, 2009). Realizando la discretización de la dinámica con integración de Euler se obtiene:

$$\dot{q}_{i+1} = \dot{q}_i + M^{-1}(S^T\tau - h)\Delta t + M^{-1}J^T p \tag{2.76a}$$

$$J\dot{q}_{i+1} = 0 \tag{2.76b}$$

donde $p = f\Delta t \in \mathbb{R}^{3N_c}$ son los impulsos en los puntos de contacto. Sustituyendo (2.76a) en la restricción cinemática (2.76b) se tiene:

$$J\dot{q}_i + JM^{-1}(S^T\tau - h)\Delta t + JM^{-1}J^T p = 0 \tag{2.77}$$

donde $JM^{-1}J^T$ es la matriz de inercia proyectada en el espacio de los contactos y está relacionada con la inversa de la matriz de inercia del espacio operacional desarrollada por Khatib (1987). Si se define $A = JM^{-1}J^T$ y $v^- = J\dot{q}_i + JM^{-1}(S^T\tau - h)\Delta t$, entonces es posible definir el siguiente problema complementario lineal sin fricción:

$$Ap + v^- = v \tag{2.78a}$$

$$0 \leq p \perp v \geq 0 \tag{2.78b}$$

donde $v \in \mathbb{R}^{3N_c}$ es una variable adicional. Una vez obtenidos los impulsos de (2.78) posteriormente se obtienen las nuevas velocidades de (2.76a) y por último las coordenadas articulares se obtienen con el integrador de Euler simpléctico, esto es:

$$q_{i+1} = q_i + \dot{q}_{i+1}\Delta t\ . \tag{2.79}$$

El problema (2.77) se puede resolver empleando el algoritmo de Danzing, Lemke o Gauss Seidel proyectado, también es posible transformar el problema a un programa cuadrático y resolverlo mediante el método del punto interior. De igual manera es posible

incorporar fricción en la formulación del problema complementario lineal mediante una pirámide de fricción.

2.6.2 Inclusiones diferenciales

Es posible formular las restricciones complementarias mediante inclusiones diferenciales al nivel de la velocidad (Carius *et al.*, 2018), esto es:

$$-\dot{g}_j \in \mathcal{N}_{\mathcal{K}_j}(p_j) \quad \forall j \in \mathcal{A} \tag{2.80}$$

donde $\mathcal{A}$ es el conjunto de los puntos de contacto activos y $\mathcal{N}_{\mathcal{K}_j}$ es el cono normal de $\mathcal{K}_j$. El cono normal de un conjunto convexo ζ se define como el conjunto de todos los vectores que forman un ángulo obtuso con todos los otros vectores que emanan de $\mathrm{x} \in \zeta$ a cualquier $\hat{\mathrm{x}} \in \zeta$ (Studer, 2009), esto es:

$$\mathcal{N}_\zeta(\mathrm{x}) = \{\mathrm{y} |\ \mathrm{y}^{\mathrm{T}}(\mathrm{x}^* - \mathrm{x}) \leq 0;\ \forall \mathrm{x}^* \in \zeta;\ \mathrm{x} \in \zeta\}\ . \tag{2.81}$$

Mediante el principio de mínima acción se tiene que los impulsos de contacto son la solución del siguiente problema de optimización:

$$\begin{aligned} \min_{p} \quad & \frac{1}{2} p^T A p + p^T v^- \\ s.t. \quad & p \in \mathcal{K}\ . \end{aligned} \tag{2.82}$$

El problema de optimización cuadrático con restricciones cónicas (2.82) se puede resolver un impulso a la vez con el algoritmo de Gauss Seidel o Jacobi con sobre relajamiento y esta la base del "time stepping" de Moreau. Además en Gehring *et al.* (2014) se muestra que el "time stepping" de Moreau es aplicable al problema de simulación de robots con patas. El algoritmo para resolver (2.82) con el algoritmo de Gauss Seidel con sobre relajamiento se muestra en pseudo código en la figura 2.5.

En el algoritmo de la figura 2.5 la función $\text{prox}_{\mathcal{K}_i}$ proyecta el impulso i en el cono de fricción $\mathcal{K}_i$, $r_i \in \mathbb{R}$ es una constante positiva de sobre relajamiento y $k_{max} \in \mathbb{Z}$ es el número máximo de iteraciones.

for $k = 1\ to\ k_{max}$
    $for\ i\ in\ \mathcal{A}$
        $$p_i = \text{prox}_{\mathcal{K}_i}\left[p_i - r_i\left(\sum_{j\in\mathcal{A}} A_{ij}p_j + v_i^-\right)\right]$$
    end
end

Figura 2.5. Algoritmo de Gauss Seidel proyectado con sobre relajamiento.

2.6.3 Modelo de contacto suave convexo

En Todorov (2014) se propone un modelo de contacto suave el cual no necesita de restricciones complementarias y es la base del simulador MuJoCo. La justificación de esta elección está basada en las observaciones realizadas en Chatterjee (1999) donde se muestra que las restricciones complementarias al nivel de la velocidad no son físicamente precisas. La formulación de la dinámica directa con contactos unilaterales de Todorov (2014) está inspirada en el principio de Gauss de menor restricción en donde se minimiza una función análoga a la energía sujeta a las restricciones cinemáticas; sin embargo, en este caso se aplica este principio a las restricciones de contacto resultando en un programa cuadrático empleando una pirámide de fricción $\mathcal{P}$ o un programa cónico con el cono de fricción $\mathcal{K}$ (Drumwright *et al.*, 2010), esto es:

$$\begin{aligned} \min_{p} \quad & \frac{1}{2}p^T(A + \varepsilon I)p + p^T(v^- - v^*) \\ s.t. \quad & p \in \mathcal{P} \end{aligned} \tag{2.83}$$

donde $\varepsilon \in \mathbb{R}$ es una constante positiva análogo al "Constraint Force Mixing" de Smith (2005) y $v^* \in \mathbb{R}^{3N_c}$ es una velocidad deseada que permite lidiar con la penetración de los puntos de contacto con el suelo debido a los errores de integración y se determina en base a la estabilidad de Baumgarte (1972). Al emplear ε y v^* en (2.83) se obtiene un modelo de contacto suave que es equivalente a un sistema de resortes y amortiguadores (método penalty); sin embargo, en este caso las ganancias de los resortes se obtienen de manera adaptiva teniendo en cuenta la masa de los objetos en contacto lo que incrementa la estabilidad del simulador a diferencia del método penalty en donde las ganancias de los resortes son siempre constantes.

2.7 Cómputo de la matriz de masas

Durante la resolución de las restricciones de contacto es necesario el cómputo de la matriz de masas en el espacio de los contactos. En general, se puede considerar 2 técnicas para el cómputo de la matriz de masas, por algoritmos recursivos y métodos directos.

2.7.1 Algoritmo recursivo $O(N)$

En Mirtich (1996) se propone un algoritmo $O(N)$ para el cómputo de la matriz de masas A simplificando el algoritmo del cuerpo articulado presentado en Featherstone (1984) eliminando los términos irrelevante durante una colisión. Los impulsos durante las colisiones se resuelven con la siguiente relación:

$$\Delta v = JM^{-1}J^T p \tag{2.84}$$

donde $\Delta v \in \mathbb{R}^3$ es la velocidad relativa de los puntos de contacto en colisión. Mientras en Kokkevis y Metaxas (1998) se propone un algoritmo similar a Mirtich (1996); sin embargo, en este caso es un algoritmo al nivel de la aceleración que utiliza la siguiente relación lineal:

$$a^x - a^0 = JM^{-1}J^T f^x \tag{2.85}$$

donde $f^x \in \mathbb{R}^{3N}$ son las N fuerzas de contacto externas, $a^0 = [a_1^0 \quad ... \quad a_N^0] \in \mathbb{R}^{3N}$ son las aceleraciones por defecto sin considerar las fuerzas externas f^x de los N puntos de contacto y $a^x = [a_1^x \quad ... \quad a_N^x] \in \mathbb{R}^{3N}$ son las aceleraciones de los N puntos de contacto después de aplicar una fuerza f^x en los puntos de contacto.

Si se define δ_j como un vector que contiene 1 en su elemento j y ceros en las demás componentes, entonces el algoritmo de Kokkevis y Metaxas (1998) calcula la columna j de la matriz de masas A mediante la siguiente relación:

$$a^j - a^0 = ABM\left(q, q, \tau, \delta_j\right) - ABM(q, q, \tau, 0) \tag{2.86}$$

donde la función ABM es el algoritmo del cuerpo articulado que retorna las aceleraciones de todos los puntos de contacto y toma como argumentos las coordenadas articulares, sus derivadas, los pares articulares y fuerzas externas. El cómputo (2.86) se repite hasta calcular todas las columnas de A.

2.7.2 Fuerza unitaria espacial

En Stępień (2012) se extiende el algoritmo presentado en Lilly (1993) a múltiples puntos de contacto. El primer paso del algoritmo consiste en el cómputo de todas las columnas de la matriz $M^{-1}J^T$, este proceso se realiza mediante la aplicación de fuerzas unitarias δ_j al robot ignorando todas las fuerzas externas e internas con j tomando valores del conjunto $\mathcal{J} = \{1, \ldots, 6 + N_J\}$, esto es:

$$M\ddot{q}_j = J^T \delta_j \tag{2.87}$$

donde $\ddot{q}_j$ es la columna j de la matriz $M^{-1}J^T$. Es necesario resolver el sistema de ecuaciones (2.87) múltiples veces; sin embargo, es posible reducir el costo computacional del algoritmo obteniendo los factores de Cholesky de M una sola vez y reutilizarlos en la solución de todos los sistemas de ecuaciones (2.87) mediante sustitución hacia adelante y atrás.

Por último, la matriz de masas A se obtiene multiplicando directamente la matriz $M^{-1}J^T$ por el Jacobiano J, esto es:

$$A = J\begin{bmatrix}\ddot{q}_1 & \cdots & \ddot{q}_{6+N_J}\end{bmatrix} \tag{2.88}$$

donde $\begin{bmatrix}\ddot{q}_1 & \cdots & \ddot{q}_{6+N_J}\end{bmatrix} = M^{-1}J^T$. El costo computacional del algoritmo es $O(N^3)$; sin embargo, puede ser más eficiente que el algoritmo recursivo $O(N)$ en la práctica cuando existen muchos contactos activos (Stępień, 2012).

CAPÍTULO 3

BALANCE DE POSTURA EMPLEANDO CONTROL DE CUERPO COMPLETO

En este capítulo se introduce un algoritmo para el balance de postura de un robot humanoide con control de cuerpo completo regulando el momento centroidal (lineal y angular) y empleando regularización dual (este último método para la solución de un programa cuadrático con la técnica del punto interior para incrementar la eficiencia y robustez ante configuraciones singulares e inconsistencias en la ejecución de múltiples tareas). Para verificar la teoría propuesta se consideran 4 escenarios de simulación en donde se aplican perturbaciones en el torso durante el balance en una plataforma y en superficies disparejas en movimiento.

3.1 Introducción

Trabajos previamente propuestos en la literatura que emplean programación cuadrática no pueden manejar tareas singulares eficientemente ya que la técnica de solución empleada es el método del conjunto activo. Este método presenta varias limitaciones, como por ejemplo sensibilidad al condicionamiento del problema y un costo computacional que incrementa exponencialmente con el número de desigualdades. Por lo tanto, el objetivo del presente capítulo es manejar configuraciones singulares de manera eficiente y robusta en la ejecución de múltiples tareas durante el balance de postura con control de cuerpo completo

de un robot humanoide con 18 articulaciones (ver figura 3.1) mediante programación cuadrática empleando el método del punto interior con regularización dual.

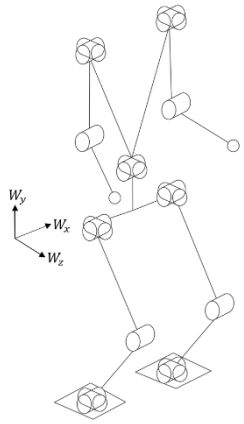


Figura 3.1. Estructura cinemática del robot humanoide bajo estudio.

3.2 Optimización convexa

3.2.1 Programación cuadrática

El problema del balance de postura con control de cuerpo completo se formula como un programa cuadrático, esto es:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x + c^T x \\ s.t. \quad & Ax = b \\ & Gx \leq d \end{aligned} \tag{3.1}$$

donde $Q \in \mathbb{R}^{n\times n}$ es la matriz Hesiana, $c \in \mathbb{R}^n$ es el vector de constantes del término lineal, $A \in \mathbb{R}^{l\times n}$ es la matriz de las l restricciones de igualdad, $b \in \mathbb{R}^l$ es el vector de constantes de las l restricciones de igualdad, $G \in \mathbb{R}^{m\times n}$ es la matriz de las m restricciones de desigualdad y $d \in \mathbb{R}^m$ es el vector de constantes de las m restricciones de desigualdad. En general, como se ha mencionado, se pueden considerar 3 métodos de solución para (3.1) estos son el método del conjunto activo, el método del punto interior y el método de multiplicadores. El método

del conjunto activo es eficiente cuando se inicializa con una buena estimación del conjunto activo óptimo requiriendo en este caso pocas iteraciones para converger; sin embargo, este método es sensible al condicionamiento del problema y su costo computacional incrementa de manera considerable con el número de desigualdades y también cuando el conjunto activo inicial defiere de manera significativa del conjunto activo óptimo (Maes, 2011). El método de multiplicadores es robusto y se puede inicializar empleando una solución previa; sin embargo, es un algoritmo ineficiente dado que es un método de primer orden requiriendo un número considerable de iteraciones para converger, aunque es posible acelerar la convergencia utilizando sobre relajamiento y pre condicionamiento como la equilibración de matrices (Stellato, 2017).

En términos de robustez y eficiencia un optimizador basado en el método del punto interior es superior, además el número de iteraciones para converger es aproximadamente constante; sin embargo, es difícil inicializar este método con una solución previa ya que tiende a volverse inestable. Esta deficiencia del método del punto interior no presenta un problema en la práctica ya que es extremadamente eficiente incluso con una iniciación por ceros (Wang y Boyd, 2009). Por ende este es el método que se utilizará para el balance de cuerpo completo. En el método del punto interior las desigualdades se convierten a igualdades introduciendo una nueva variable $s \in \mathbb{R}^m$ restringida a ser positiva (Nocedal y Wright, 2006) y empleando regularización dual en (3.1) se tiene que:

$$\begin{aligned} \min_{x,s} \quad & \frac{1}{2}x^T Q x + c^T x + \frac{\delta}{2}(x^T x + y^T y + z^T z) \\ s.t. \quad & Ax - b = 0 \\ & Gx + s - d = 0 \\ & s \geq 0 \end{aligned} \tag{3.2}$$

donde $\delta \in \mathbb{R}$ es una constante positiva que regulariza las variables duales, $y \in \mathbb{R}^l$ son los multiplicadores de Lagrange de las restricciones de igualdad y $z \in \mathbb{R}^m$ son los multiplicadores de Lagrange de las restricciones de desigualdad. Empleando las condiciones de optimalidad de Karush Kuhn Tucker en (3.2) se obtiene:

$$(Q + \delta I)x + c + A^T y + G^T z = 0 \quad (3.3a)$$
$$SZe - \mu e = 0 \quad (3.3b)$$
$$Ax - b + \delta y = 0 \quad (3.3c)$$
$$Gx + s - d + \delta z = 0 \quad (3.3d)$$
$$(s, z) \geq 0 \quad (3.3e)$$

donde $S = diag(s) \in \mathbb{R}^{m \times m}$ es una matriz con s en la diagonal, $Z = diag(z) \in \mathbb{R}^{m \times m}$ es una matriz con z en la diagonal, $I \in \mathbb{R}^{n \times n}$ es la matriz identidad, $e \in \mathbb{R}^m$ es un vector de unos, $\mu = s^T z/m$ es la brecha complementaria y $\sigma \in \mathbb{R}$ es una constante positiva.

Reescribiendo (3.3) como una función no lineal se tiene:

$$F(x, s, y, z) = \begin{bmatrix} (Q + \delta I)x + c + A^T y + G^T z \\ SZe - \sigma\mu e \\ Ax - b + \delta y \\ Gx + s - d + \delta z \end{bmatrix}. \quad (3.4)$$

Empleando el método de Newton en (3.4) se obtiene:

$$\begin{bmatrix} Q + \delta I & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & \delta I & 0 \\ G & I & 0 & \delta I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} (Q + \delta I)x + c + A^T y + G^T z \\ SZe - \sigma\mu e \\ Ax - b + \delta y \\ Gx + s - d + \delta z \end{bmatrix}. \quad (3.5)$$

Emplear regularización dual permite introducir 2 matrices de la forma δI en la diagonal inferior de (3.5) con lo que se puede resolver el sistema mediante cualquier técnica de factorización ya que de otra forma el sistema (3.5) sería indefinido. El método del punto interior consiste en la solución de múltiples sistemas de la forma (3.5) disminuyendo en cada iteración el valor de la brecha complementaria μ mediante la constante σ hasta el valor de cero. Una vez resuelto (3.5) se realiza una búsqueda de línea de forma tal que las condiciones $(s, z) \geq 0$ no se violen. El nuevo punto de búsqueda se obtiene como:

$$(x_{k+1}, s_{k+1}) = (x_k, s_k) + \alpha_\tau^{pri}(\Delta x, \Delta s) \tag{3.6a}$$
$$(y_{k+1}, z_{k+1}) = (y_k, z_k) + \alpha_\tau^{dual}(\Delta y, \Delta z) \tag{3.6b}$$

donde la longitud de los pasos $\alpha_\tau^{pri} \in \mathbb{R}$ y $\alpha_\tau^{dual} \in \mathbb{R}$ se obtienen de la siguiente forma:

$$\alpha_\tau^{pri} = max\{\alpha \in (0,1]: s + \alpha\Delta s \geq (1-\tau)s\} \tag{3.7a}$$
$$\alpha_\tau^{dual} = max\{\alpha \in (0,1]: z + \alpha\Delta z \geq (1-\tau)z\} \tag{3.7b}$$

donde $\tau \in (0,1)$ es una constante positiva que define la cercanía de las variables s y z de sus valores mínimos de 0, un valor típico de τ es 0.995. En el presente trabajo se emplea la estrategia de Andrei (2017) $\tau_k = \max(0.995, 1-\mu_k)$ para controlar el valor de τ. La estrategia de resolver (3.5) en secuencia no necesariamente resulta en un algoritmo eficiente, las implementaciones modernas del método del punto interior suelen emplear una estrategia alternativa basada en el algoritmo de Mehrotra, el cual consiste en calcular un paso predictor de (3.5) con $\sigma = 0$; esto es:

$$\begin{bmatrix} Q+\delta I & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & \delta I & 0 \\ G & I & 0 & \delta I \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta s^{aff} \\ \Delta y^{aff} \\ \Delta z^{aff} \end{bmatrix} = - \begin{bmatrix} (Q+\delta I)x + c + A^T y + G^T z \\ SZe \\ Ax - b + \delta y \\ Gx + s - d + \delta z \end{bmatrix}. \tag{3.8}$$

Posteriormente se calcula un paso corrector mediante el siguiente sistema:

$$\begin{bmatrix} Q+\delta I & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & \delta I & 0 \\ G & I & 0 & \delta I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} (Q+\delta I)x + c + A^T y + G^T z \\ SZe - \sigma\mu_k e + \Delta S^{aff}\Delta Z^{aff} e \\ Ax - b + \delta y \\ Gx + s - d + \delta z \end{bmatrix} \tag{3.9}$$

donde $\Delta S^{aff} = diag(\Delta s^{aff}) \in \mathbb{R}^{m\times m}$ es una matriz con Δs^{aff} en la diagonal y $\Delta Z^{aff} = diag(\Delta z^{aff}) \in \mathbb{R}^{m\times m}$ es una matriz con Δz^{aff} en la diagonal. En la figura 3.2 se muestra un pseudo código del algoritmo de Mehrotra para la solución de programas cuadráticos.

$k = 0$

while $E(x_k, s_k, y_k, z_k) \leq \epsilon_{TOL}$

$(x, s, y, z) = (x_k, s_k, y_k, z_k)$

$$\begin{bmatrix} Q+\delta I & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & \delta I & 0 \\ G & I & 0 & \delta I \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta s^{aff} \\ \Delta y^{aff} \\ \Delta z^{aff} \end{bmatrix} = - \begin{bmatrix} (Q+\delta I)x + c + A^T y + G^T z \\ SZe \\ Ax - b + \delta y \\ Gx + s - d + \delta z \end{bmatrix}$$

$\mu_k = s^T z / m$

$\hat{\alpha}_{aff} = max\{\alpha \in (0,1] : (s,z) + \alpha(\Delta s^{aff}, \Delta z^{aff}) \geq 0\}$

$\mu_{aff} = \left(s + \hat{\alpha}_{aff} \Delta s^{aff}\right)^T (z + \hat{\alpha}_{aff} \Delta z^{aff}) / m$

$\sigma = (\mu_{aff} / \mu_k)^3$

$$\begin{bmatrix} Q+\delta I & 0 & A^T & G^T \\ 0 & Z & 0 & S \\ A & 0 & \delta I & 0 \\ G & I & 0 & \delta I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} (Q+\delta I)x + c + A^T y + G^T z \\ SZe - \sigma \mu_k e + \Delta S^{aff} \Delta Z^{aff} e \\ Ax - b + \delta y \\ Gx + s - d + \delta z \end{bmatrix}$$

$\tau_k = \max(0.995, 1 - \mu_k)$

$\alpha_{\tau_k}^{pri} = max\{\alpha \in (0,1] : s + \alpha \Delta s \geq (1 - \tau_k) s\}$

$\alpha_{\tau_k}^{dual} = max\{\alpha \in (0,1] : z + \alpha \Delta z \geq (1 - \tau_k) z\}$

$\hat{\alpha} = min(\alpha_{\tau_k}^{pri}, \alpha_{\tau_k}^{dual})$

$(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1}) = (x_k, s_k, y_k, z_k) + \hat{\alpha}(\Delta x, \Delta s, \Delta y, \Delta z)$

$k = k + 1$

end

Figura 3.2. Algoritmo de Mehrotra para programación cuadrática.

El algoritmo de Mehrotra selecciona la constante σ de manera dinámica empleando la heurística propuesta por Mehrotra (1992) con $\sigma = (\mu_{aff} / \mu_k)^3$ en donde $\mu_{aff} \in \mathbb{R}$ se obtiene como:

$$\mu_{aff} = \left(s + \hat{\alpha}_{aff} \Delta s^{aff}\right)^T (z + \hat{\alpha}_{aff} \Delta z^{aff}) / m \tag{3.10}$$

donde $\hat{\alpha}_{aff} \in \mathbb{R}$ es una constante que se obtiene realizando una búsqueda de línea, esto es:

$$\hat{\alpha}_{aff} = max\{\alpha \in (0,1]: (s,z) + \alpha(\Delta s^{aff}, \Delta z^{aff}) \geq 0\}\,. \tag{3.11}$$

Por último, en el algoritmo de la figura 3.2 la función de error se define como:

$$E(x,s,y,z) = \max\{\|Q + \delta I + c + A^T y + G^T z\|, \|Sz\|, \|Ax - b + \delta y\|, \\ \|Gx + s - d + \delta z\|\}\,. \tag{3.12}$$

Tradicionalmente los sistemas de ecuaciones (3.8) y (3.9) se resuelven mediante factorización simétrica indefinida con librerías como MA57 (Duff, 2004), algunos optimizadores que emplean factorización simétrica indefinida son OOQP (Gertz y Wright, 2002), LOQO (Vanderbei, 1999), KNITRO (Waltz *et al.*, 2006) y IPOPT (Wächter, 2002); sin embargo, en nuestra formulación al emplear regulación dual es posible resolver los sistemas (3.8) y (3.9) mediante descomposición LU disminuyendo así el costo computacional del algoritmo.

3.3 Balance de postura con control de cuerpo completo

3.3.1 Tareas múltiples

Cuando un robot es redundante la ejecución de tareas secundarias son posibles realizarlas en el espacio nulo de las tareas de mayor prioridad. En la formulación del problema de balance se consideran tareas de dinámica y cinemática. El balance se efectúa empleando las ecuaciones del cuerpo completo, esto es:

$$M(q)\ddot{q} + h_b(q,\dot{q}) = S^T\tau + J^T(q)\lambda \tag{3.13}$$

donde $M(q) \in \mathbb{R}^{(6+n_J)\times(6+n_J)}$ es la matriz de inercia de base flotante, $h_b(q,\dot{q}) \in \mathbb{R}^{6+n_J}$ son las fuerzas de Coriolis, centrífugas y gravedad, $S = [I_{n_J \times n_J} \quad 0_{n_J \times 6}]$ es la matriz de selección de pares actuados, $\tau \in \mathbb{R}^{n_J}$ es el vector de pares articulares, $J(q) \in \mathbb{R}^{3n_c \times (6+n_J)}$ es el

Jacobiano aumentado de los n_c puntos de contacto, $\lambda \in \mathbb{R}^{3n_c}$ es el vector de las n_c fuerzas de contacto y n_J es el número de grados de libertad del sistema sin tener en cuenta la posición y orientación de la base. Las fuerzas de contacto se modelan con la fricción estática de Coulomb, por lo tanto la fuerza de contacto λ_j está restringida por:

$$\left\|\lambda_j - (n_j^T \lambda_j) n_j\right\| \leq \mu n_j^T \lambda_j \tag{3.14}$$

donde $n_j \in \mathbb{R}^3$ es el vector normal a la superficie de contacto y $\mu \in \mathbb{R}$ es el coeficiente de fricción. Sin embargo (3.14) es una restricción cónica lo que convierte el problema de balance en una optimización cónica de segundo orden, por lo tanto, el cono de fricción de Coulomb se aproxima mediante una pirámide compuesta de 4 vectores base $w_{ij} \in \mathbb{R}^3$ que se definen como:

$$w_{ij} = n_j + \mu d_{ij} \tag{3.15}$$

donde $d_{ij} \in \mathbb{R}^3$ es un vector tangente a la superficie de contacto. Por lo tanto, la pirámide de fricción es:

$$\forall j \quad \lambda_j = W_j \beta_j \tag{3.16a}$$

$$\forall j \quad \beta_j \geq 0 \tag{3.16b}$$

donde $W_j = [w_{1j} \quad w_{2j} \quad w_{3j} \quad w_{4j}] \in \mathbb{R}^{3\times 4}$ define una base vectorial de la pirámide de fricción y $\beta_j \in \left[\beta_{1j} \quad \beta_{2j} \quad \beta_{3j} \quad \beta_{4j}\right]^T \in \mathbb{R}^4$ es un vector de constantes positivas. El uso de (3.16) en lugar de (3.14) permite abordar el problema de balance mediante programación cuadrática. El restringir a las fuerzas de contacto dentro del cono de fricción por (3.16) y obedecer las ecuaciones de movimiento (3.13) generaliza el criterio de balance del ZMP a superficies irregulares (Abe *et al.*, 2007) y esta es la tarea de mayor prioridad en la

formulación del problema de balance. La ecuación de movimiento es lineal en las variables $\ddot{q}$, τ y λ, por lo tanto (3.13) y (3.16a) se pueden expresar en una matriz aumentada, esto es:

$$\begin{bmatrix} M & -S^T & -J^T & 0 \\ 0 & 0 & I & -W \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \\ \beta \end{bmatrix} = -\begin{bmatrix} h_b \\ 0 \end{bmatrix} \tag{3.17}$$

donde $W = \begin{bmatrix} W_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & W_8 \end{bmatrix} \in \mathbb{R}^{3n_c \times 4n_c}$ y $I \in \mathbb{R}^{3n_c \times 3n_c}$ es la matriz identidad. Ahora si se define el vector de optimización $x = [\ddot{q}^T \quad \tau^T \quad \lambda^T \quad \beta^T]^T$ entonces (3.17) se puede reescribir como:

$$A_1 x = b_1 \tag{3.18}$$

donde $A_1 = \begin{bmatrix} M & -S^T & -J^T & 0 \\ 0 & 0 & E & -W \end{bmatrix}$ y $b_1 = -\begin{bmatrix} h_b \\ 0 \end{bmatrix}$.

Durante el balance se especifica una tarea que restringe el movimiento de los puntos contacto con respecto a la superficie de contacto, estas restricciones se representan mediante la siguiente función:

$$c_j(q) = p_j^d \tag{3.19}$$

donde $c_j(q)$ es la función cinemática directa del punto de contacto j y $p_j^d \in \mathbb{R}^3$ es la posición deseada del punto de contacto j. Se deriva (3.19) dos veces con respecto al tiempo para expresar la restricción cinemática al nivel de la aceleración:

$$J_j(q)\ddot{q} + \dot{J}_j(q)\dot{q} = 0 \tag{3.20}$$

donde $J_j(q)$ es el Jacobiano de la restricción cinemática j. La restricción cinemática (3.20) se reescribe en términos del vector de optimización x, por lo tanto se tiene:

$$[J_j \quad 0 \quad 0 \quad 0]\begin{bmatrix}\ddot{q}\\ \tau\\ \lambda\\ \beta\end{bmatrix} = -\dot{J}_j\dot{q}\ . \tag{3.21}$$

Ahora cuando se tienen n puntos de contacto (3.21) se generaliza a:

$$\begin{bmatrix}J_1 & 0 & 0 & 0\\ \vdots & \vdots & \vdots & \vdots\\ J_n & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\ddot{q}\\ \tau\\ \lambda\\ \beta\end{bmatrix} = -\begin{bmatrix}\dot{J}_0\\ \vdots\\ \dot{J}_n\end{bmatrix}\dot{q} \tag{3.22a}$$

$$A_2 x = b_2 \tag{3.22b}$$

donde $A_2 = \begin{bmatrix}J_1 & 0 & 0 & 0\\ \vdots & \vdots & \vdots & \vdots\\ J_n & 0 & 0 & 0\end{bmatrix}$ y $b_2 = -\begin{bmatrix}\dot{J}_0\\ \vdots\\ \dot{J}_n\end{bmatrix}\dot{q}$.

Para incrementar la robustez del balance se considera la regulación del momento lineal y angular. El momento centroidal está relacionado con las velocidades articulares por (Orin *et al.*, 2013):

$$h_G = A_G(q)\dot{q} \tag{3.23}$$

donde $h_G \in \mathbb{R}^6$ es el momento centroidal y $A_G(q) \in \mathbb{R}^{6\times(6+n_J)}$ es la matriz del momento centroidal. Tomando la derivada temporal de (3.23) se tiene:

$$\dot{h} = A_G(q)\ddot{q} + \dot{A}_G(q)\dot{q} \ . \tag{3.24}$$

Reescribiendo (3.24) en términos de x se obtiene:

$$[A_G \quad 0 \quad 0 \quad 0] \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \\ \beta \end{bmatrix} = \dot{h}_{des} - \dot{A}_G(q)\dot{q} \tag{3.25a}$$

$$A_3 x = b_3 \tag{3.25b}$$

donde $\dot{h}_{des} \in \mathbb{R}^6$ es una referencia deseada, $A_3 = [A_G \quad 0 \quad 0 \quad 0]$ y $b_3 = \dot{h}_{des} - \dot{A}_G(q)\dot{q}$. A continuación, se emplea la siguiente referencia deseada:

$$\dot{h}_{des} = \begin{bmatrix} -k_G h \\ mk_p(r_d - r) - k_v l \end{bmatrix} \tag{3.26}$$

donde $l \in \mathbb{R}^3$ es el momento lineal, $r_d \in \mathbb{R}^3$ es la posición deseada del centro de masas, $k_G \in \mathbb{R}^{3\times3}$ es un matriz de ganancias para la regulación del momento angular, $k_p \in \mathbb{R}^{3\times3}$ y $k_v \in \mathbb{R}^{3\times3}$ son matrices definidas positivas de ganancias proporcional y derivativa, respectivamente.

Por último, se especifica una postura de referencia deseada, esto es:

$$\ddot{q}_{des} = k_p(q_d - q) - k_v\dot{q} \tag{3.27}$$

donde $q_d \in \mathbb{R}^{6+n_J}$ es la referencia deseada. De (3.27) se define la siguiente tarea:

$$\ddot{q} - \ddot{q}_{des} = 0 \tag{3.28a}$$

$$\begin{bmatrix} I & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \\ \beta \end{bmatrix} = \ddot{q}_{des} \tag{3.28b}$$

$$A_4 x = b_4 \tag{3.28c}$$

donde $A_4 = [I \quad 0 \quad 0 \quad 0]$ y $b_4 = \ddot{q}_{des}$.

3.3.2 Control de cuerpo completo

El problema de balance se formula como la solución a una jerarquía de 4 tareas; sin embargo, esto requiere la solución de múltiples programas cuadráticos, por lo tanto, en el presente trabajo se le asigna un peso $w_j \in \mathbb{R}$ a cada tarea de forma tal que sólo es necesario la solución de un único programa cuadrático reduciendo así el costo computacional del control de cuerpo completo. La jerárquica de tareas se obtiene de (3.18), (3.22b), (3.25b) y (3.28c), ahora se crea un sistema aumentado concatenando las 4 tareas, esto es:

$$\begin{bmatrix} w_1 A_1 \\ w_2 A_2 \\ w_3 A_3 \\ w_4 A_4 \end{bmatrix} x = \begin{bmatrix} w_1 b_1 \\ w_2 b_2 \\ w_3 b_3 \\ w_4 b_4 \end{bmatrix} \tag{3.29a}$$

$$Ax = b \tag{3.29b}$$

donde $A = [w_1 A_1^T \quad w_2 A_2^T \quad w_3 A_3^T \quad w_4 A_4^T]^T$ y $b = [w_1 b_1^T \quad w_2 b_2^T \quad w_3 b_3^T \quad w_4 b_4^T]^T$. Por último, las desigualdades (3.16b) se expresan en términos del vector de optimización x y se multiplican por la ganancia w_1, esto es:

$$w_1 [0 \quad 0 \quad 0 \quad -I] \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \\ \beta \end{bmatrix} \leq 0 \tag{3.30a}$$

$$w_1 G x \leq w_1 d \tag{3.30b}$$

donde $I \in \mathbb{R}^{4n_c \times 4n_c}$ es la matriz identidad, $G = [0 \quad 0 \quad 0 \quad -I]$ y $d = 0$.

Por lo tanto, de (3.29b) y (3.30b) el problema de optimización a resolver en cada ciclo de control para el balance de cuerpo completo es:

$$\begin{aligned} \min_{x,u_1,u_2,v_1,v_2} \quad & \frac{1}{2} x^T Q x + c^T x + \gamma e_1^T (u_1 + v_1) + \gamma e_2^T (u_2 + v_2) \\ s.t. \quad & Ax - b + u_1 = v_1 \\ & Gx - d + u_2 \leq v_2 \\ & u_1, u_2, v_1, v_2 \geq 0 \end{aligned} \tag{3.31}$$

donde $u_1 \in \mathbb{R}^l, u_2 \in \mathbb{R}^m, v_1 \in \mathbb{R}^l, v_2 \in \mathbb{R}^m$ son variables adicionales, $\gamma \in \mathbb{R}$ es una constante positiva que penaliza la violación de las restricciones. El optimizador del punto interior de la figura 2 agrega regulación a las variables duales y de optimización del programa cuadrático (3.31) y esta es la razón por la cual estos términos adiciones no aparecen en la función objetivo de (3.31). La función objetivo cuadrática que se emplea en (3.31) consiste en minimizar los pares articulares, esto es:

$$\frac{1}{2} x^T Q x + c^T x = \frac{1}{2} \tau^T \tau \tag{3.32}$$

donde $c = 0$ es un vector de ceros y $Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

Para incrementar la eficiencia durante la solución de (3.31) todas las tareas A_j y la matriz Hesiana Q se representan mediante matrices dispersas. La formulación del programa cuadrático (3.31) está inspirado en el manejo de problemas cuadráticos inconsistentes en la programación cuadrática secuencial y se denomina programa elástico o modo elástico

(Andrei, 2017). La inclusión de variables adicionales en (3.31) permite lidiar con posibles tareas singulares y el uso de regularización dual incrementa la robustez y eficiencia del optimizador.

3.4 Resultados

Se consideran 5 escenarios de simulación en donde las ecuaciones de movimiento se integran empleando el método de Euler simpléctico durante 8000 pasos de integración ($\Delta t = 1 \times 10^{-3}$). Para realizar las simulaciones se desarrolla un simulador en Matlab empleando el algoritmo del cuerpo articulado para sistemas de base flotante de Featherstone (2008) en el cálculo de las aceleraciones articulares dados los pares articulares y fuerzas externas, también se implementa el algoritmo descrito en Kokkevis y Metaxas (1998) para el cómputo de las fuerzas de contacto.

De (13) la matriz de inercia $M(q)$ se calcula mediante el algoritmo del cuerpo compuesto, el vector $h_b(q, \dot{q})$ se obtiene mediante el algoritmo recursivo de Newton Euler. De (3.24) la matriz del momento angular centroidal $A_G(q)$ se obtiene mediante el algoritmo descrito en Orin *et al.* (2013) y para el cálculo del término $\dot{A}_G(q)\dot{q}$ se desarrolla un algoritmo $O(N)$ empleando álgebra de Lie. El número de puntos de contacto que se consideran en los 5 escenarios es $n_c = 8$ (4 para cada pie) y el número de grados de libertad actuados en el robot es $n_J = 18$. Las propiedades del modelo del robot bajo estudio se inspiran en Geyer y Herr (2010) para las piernas mientras que para la parte superior del cuerpo se adaptan de forma tal que la masa total del sistema sea de 80kg, los datos empleados se muestran en la tabla 3.1.

Primero se considera un escenario 0 para visualizar el comportamiento del sistema sin ningún tipo de perturbación. El sistema comienza (al igual que en todos los escenarios) con una postura inicial con los brazos flexionados similar a la postura que se muestra en la figura 3.1 y su postura deseada en este escenario consiste en terminar con los brazos estirados

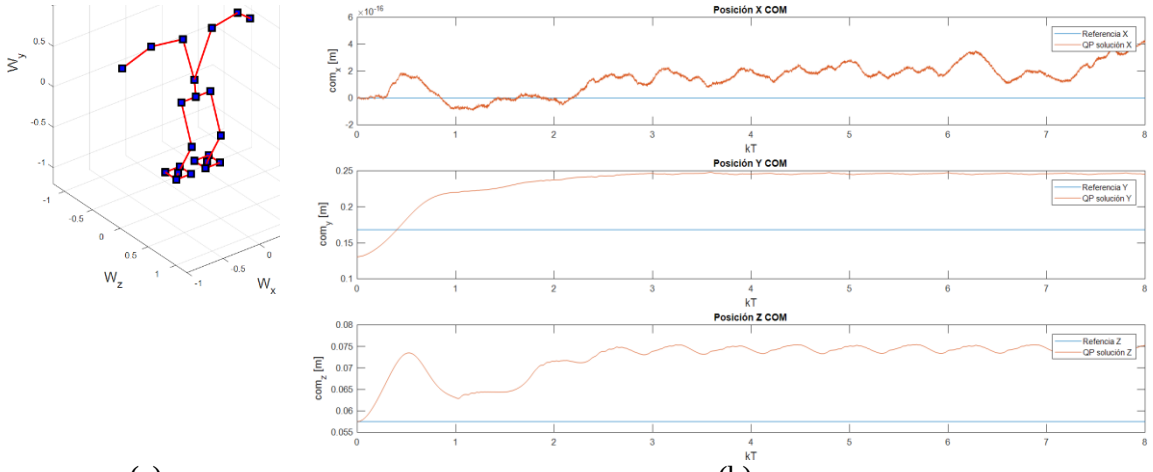
levantados, los resultados del escenario 0 se presentan en la figura 3.3. Observe que la variación en "x" es insignificante y que levanta los brazos al solicitarle una posición 4 cm por arriba del centro de masas inicial. Hay errores aceptables en "y" y en "z" pues el robot se mantiene de pie.

Tabla 3.1. Propiedades del modelo del robot bípedo.

Extremidad	Masa	Longitud
Torso	30.125kg	0.6m
Pelvis	13.375kg	0.2m
Muslo	8.5kg	0.5m
Pierna	3.5kg	0.5m
Pie	1.25kg	0.1m
Brazo derecho	2.5kg	0.4m
Antebrazo derecho	2.5kg	0.4m
Brazo izquierdo	2.5kg	0.4m
Antebrazo izquierdo	2.5kg	0.4m

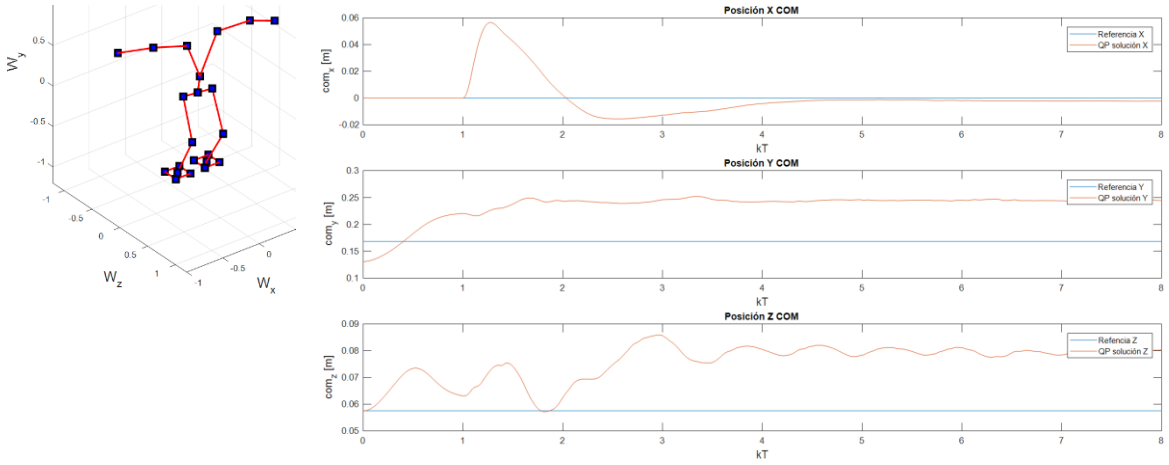
El escenario 1 consiste en mantener el balance aplicando una perturbación sobre el torso de $f^E = [200 \quad 0 \quad 100]^T N$ a partir del primer segundo durante 80 pasos de integración, los resultados del escenario 1 se muestran en la figura 3.4. Note que el robot reaccionó manteniéndose siempre de pie.

El escenario 2 consiste en el balance del robot puesto sobre una plataforma que alterna su movimiento con una velocidad de $0.4\,m/s$ en el eje x, los resultados del escenario 2 se muestran en la figura 3.5. Nuevamente, a pesar de que el piso se mueve, el robot mantiene su balance erguido.



(a) (b)

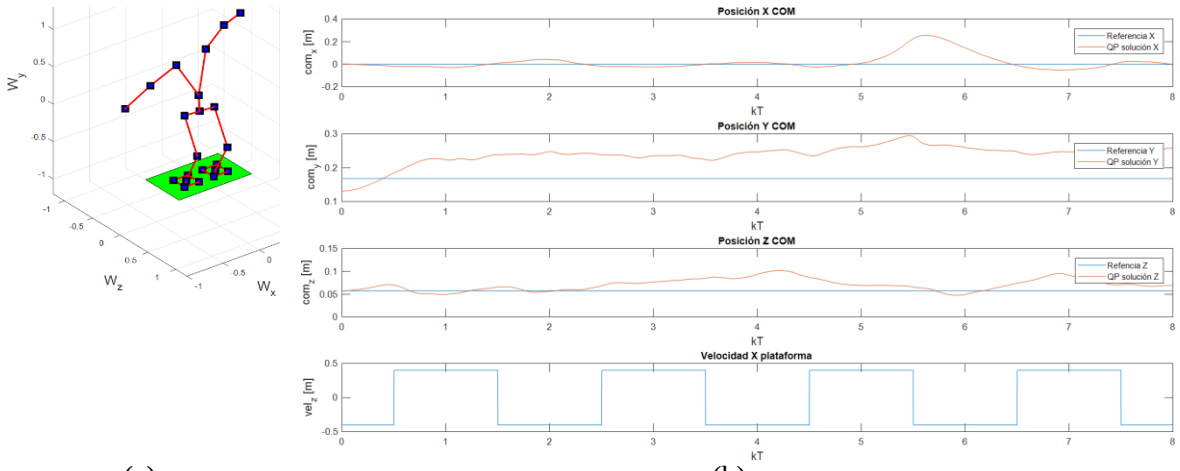
Figura 3.3. (a) Visualización en Matlab del escenario 0, (b) gráficas de la posición del centro de masas del escenario 0.



(a) (b)

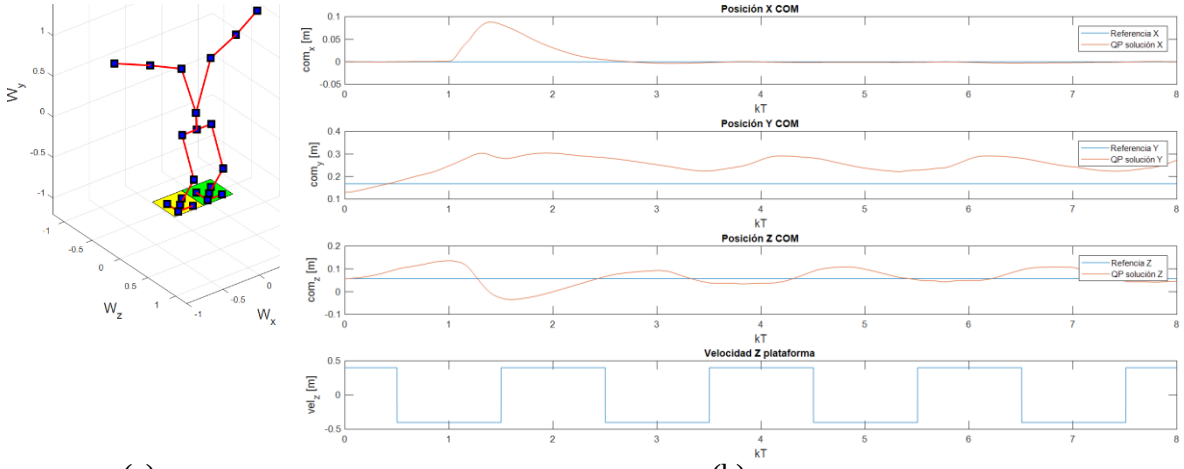
Figura 3.4. (a) Visualización en Matlab del escenario 1, (b) gráficas de la posición del centro de masas del escenario 1.

El escenario 3 al igual que el 2 consiste en mantener el balance en una plataforma que alterna su movimiento con una velocidad de $0.4\,m/s$; sin embargo, en este caso es sobre el eje z, también se aplica una perturbación sobre el torso de $f^E = [200 \quad 0 \quad 0]^T N$ a partir del primer segundo durante 80 pasos de integración, los resultados del escenario 3 se muestran en la figura 3.6.



(a) (b)

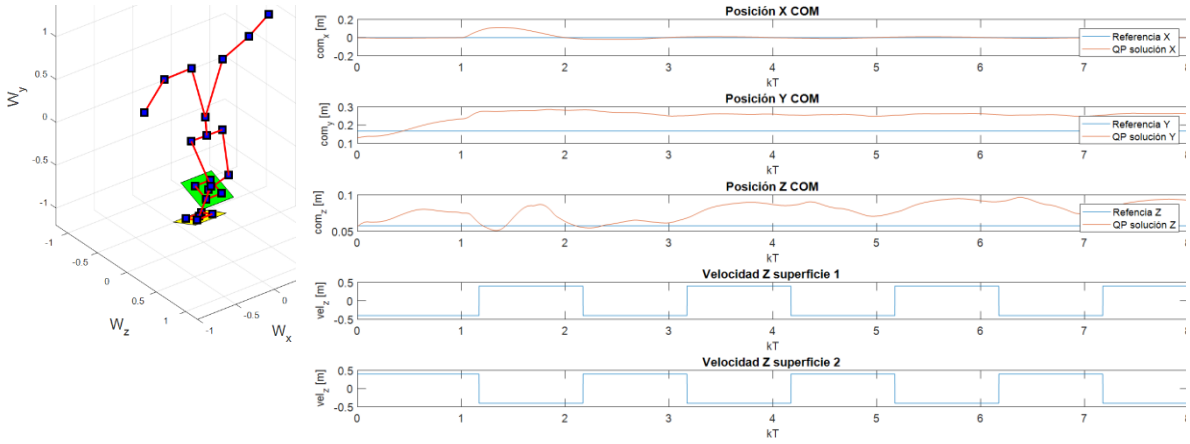
Figura 3.5. (a) Visualización en Matlab del escenario 2, (b) gráficas de la posición del centro de masas del escenario 2.



(a) (b)

Figura 3.6. (a) Visualización en Matlab del escenario 3, (b) gráficas de la posición del centro de masas del escenario 3.

Por último, en el cuarto escenario se considera el balance con el robot puesto sobre dos plataformas disparejas con inclinación de $20°$ y $-20°$ grados que se desplazan en direcciones opuestas sobre el eje z de manera alternante con una velocidad de $0.4\,m/s$, también se aplica una perturbación sobre el torso de $f^E = [400 \quad 0 \quad 0]^T N$ a partir del primer segundo durante 80 pasos de integración, los resultados del escenario 4 se ilustran en la figura 3.7.



(a) (b)

Figura 3.7. (a) Visualización en Matlab del escenario 4, (b) gráficas de la posición del centro de masas del escenario 4.

A continuación, en la figuras 3.8, 3.9, 3.10, 3.11 y 3.12 se presentan los errores con respecto a la referencia deseada del centro de masas, momento lineal y momento angular de los 5 escenarios.

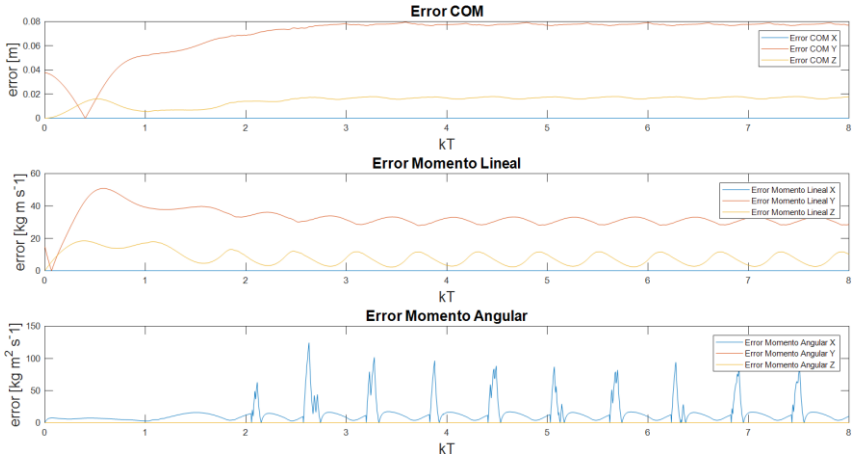


Figura 3.8. Errores del COM, momento lineal y angular en el escenario 0.

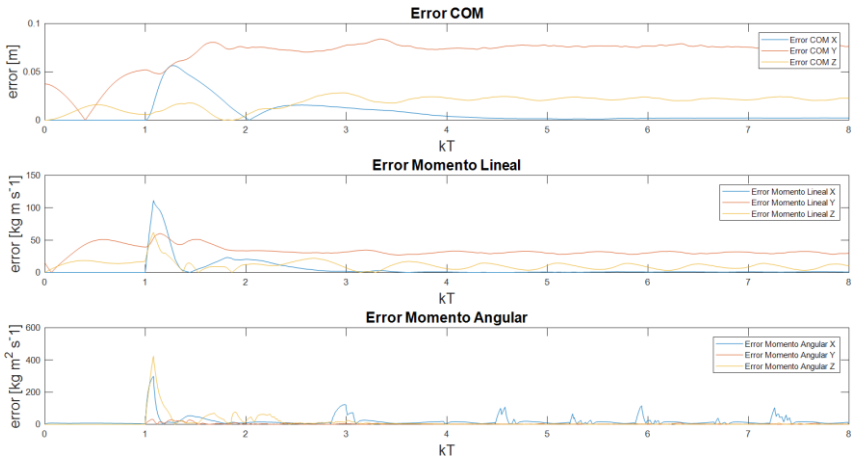


Figura 3.9. Errores del COM, momento lineal y angular en el escenario 1.

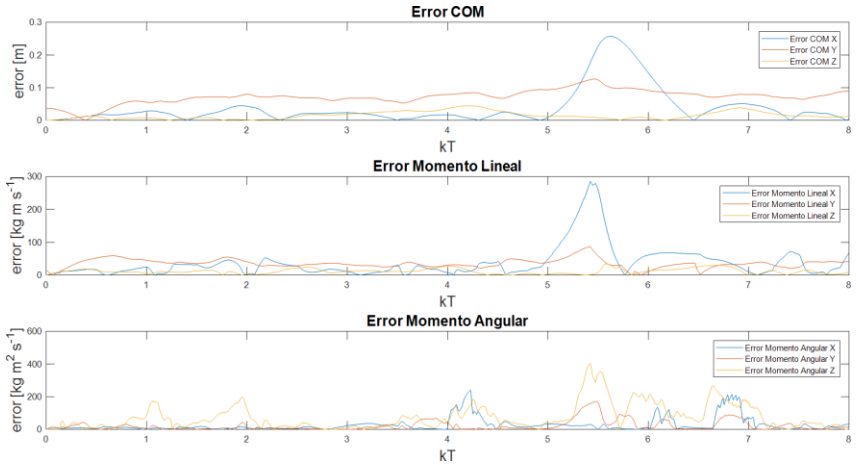


Figura 3.10. Errores del COM, momento lineal y angular en el escenario 2.

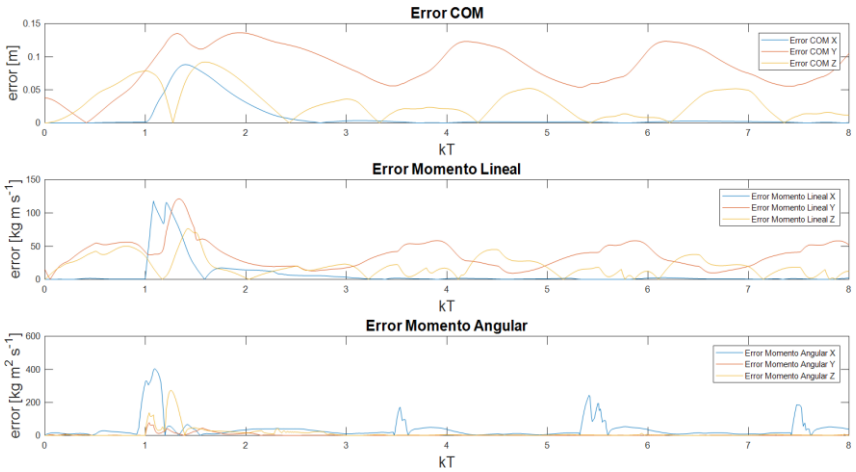


Figura 3.11. Errores del COM, momento lineal y angular en el escenario 3.

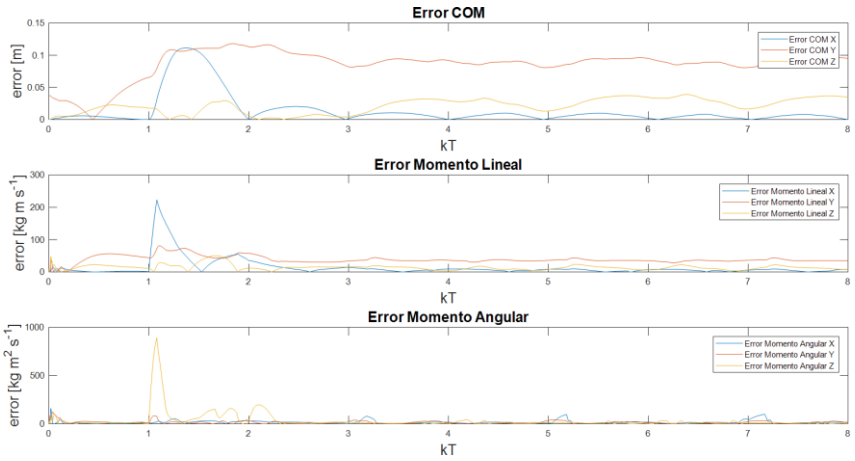


Figura 3.12. Errores del COM, momento lineal y angular en el escenario 4.

En los 5 escenarios se observa que la posición del centro de masas se mantiene en valores bastante aceptables mostrando así que el sistema es capaz de mantener el balance ante la fuerza externa en el torso y el movimiento de la plataforma. De igual manera se observa que la magnitud del error del centro de masas, momento lineal y angular se mantiene con algunas fluctuaciones aceptables. El instante en donde se aprecia el mayor error en los escenarios es donde se aplica la perturbación en el torso (a partir del primer segundo).

El uso de las ganancias para especificar prioridades en las tareas incrementa la robustez del control del cuerpo completo ya que sin éstas el sistema tiende a converger a configuraciones que se desvían de manera considerable de la postura deseada, además no fue difícil especificar los valores de las ganancias en los 5 escenarios considerados en este trabajo, los valores que se emplearon fueron potencias de 10. Sin embargo, diferentes ganancias en las tareas producen diferentes comportamientos del sistema y el grado de cumplimiento de las tareas dependen de éstas. También existe el riesgo de inestabilidad numérica si la diferencia entre las ganancias es muy grande, en los experimentos numéricos se observa inestabilidad con una diferencia mayor a 1000.

3.5 Conclusiones

En este capítulo se ha introducido un algoritmo para el balance de postura de un robot humanoide con control de cuerpo completo capaz de lidiar con múltiples tareas y configuraciones singulares de manera eficiente y robusta. El número de iteraciones que le toma al algoritmo en converger durante cada ciclo de control en los 5 escenarios considerados es de aproximadamente 10. Los escenarios simulados en este capítulo demuestran el potencial de este algoritmo para el balance en superficies irregulares, esto permite visualizar su aplicación en la caminata de robots con patas en terrenos irregulares.

CAPÍTULO 4

ALGORITMO $O(N)$ PARA EL CÓMPUTO DE LA DINÁMICA CENTROIDAL

En el presente capítulo se introduce un algoritmo $O(N)$ para el cómputo de la matriz del momento centroidal y su derivada temporal multiplicada por el vector de velocidades articulares de manera conjunta empleando álgebra espacial y de Lie. Para verificar la efectividad de la regulación del momento centroidal, el algoritmo desarrollado se aplica en el balance de postura con control de cuerpo completo mediante programación cuadrática jerárquica. Las tareas empleadas en el control de cuerpo completo únicamente requieren de la matriz del momento centroidal y su derivada sin la necesidad de la matriz de inercia y el vector de Coriolis reduciendo así el costo computacional del controlador.

4.1 Introducción

Recientemente la regulación del momento centroidal ha emergido como una técnica robusta para el balance de postura y durante la caminata de robots con patas. En Orin *et al.* (2013) se demuestra que existe una relación lineal entre las velocidades articulares y el momento centroidal (lineal y angular) mediante la matriz del momento centroidal y se introduce un algoritmo $O(N)$ para su cómputo mediante álgebra espacial. En Macchietto *et al.* (2009) y De Lasa *et al.* (2010) se presentan algoritmos $O(N^2)$ empleando notación global para el cómputo de la derivada temporal de la matriz del momento centroidal. En Wensing y

Orin (2016) la matriz del momento centroidal y su derivada temporal se expresan en términos de la matriz de inercia, el vector de Coriolis y la cinemática de la base flotante, reduciendo así el costo computacional del control de cuerpo completo.

Por otro lado, en Park *et al.* (1995) las ecuaciones de movimiento de un robot manipulador serial se formulan mediante álgebra de Lie de forma recursiva y en forma cerrada. En Kim *et al.* (1995) y Lee *et al.* (2005) se obtienen las segundas derivadas de la dinámica inversa formulada con operadores del álgebra de Lie; los algoritmos resultantes se utilizan en la planeación de movimiento en cadenas cinemáticas abiertas y con ciclos cerrados mediante optimización no lineal con el método de Newton y Quasi-Newton Memoria Limitada BFGS. En Lo *et al.* (2002) se establece el isomorfismo entre el álgebra de Lie $se(3)$ de las matrices homogéneas y $\mathbb{R}^6$ permitiendo tomar las derivadas analíticas de las ecuaciones recursivas del algoritmo de Newton Euler presentado en Featherstone (1987); el algoritmo resultante se emplea en la planeación de movimiento de una figura humanoide con el método del Lagrangiano Aumentado.

El algoritmo aquí propuesto se obtiene derivando las ecuaciones recursivas del algoritmo de Orin *et al.* (2013) mediante el uso de álgebra de Lie; logrando así un algoritmo $O(N)$ para el cómputo de la matriz del momento centroidal y su derivada temporal de manera conjunta. Por último, el algoritmo resultante se aplica en el balance de postura con control de cuerpo completo empleando programación cuadrática jerárquica en un robot humanoide con 18 articulaciones (ver figura 4.1).

4.2 Álgebra espacial y de Lie

Como se ha mencionado, en el álgebra espacial moderna se considera la existencia de 2 espacios vectoriales duales M^6 y F^6. Es posible realizar una conexión de los espacios M^6 y F^6 con el álgebra de Lie $se(3)$ y $se(3)^*$ del grupo $SE(3)$ para de esta forma obtener las derivadas analíticas de ecuaciones expresadas con álgebra espacial, esto es:

$$M^6 \equiv \left\{ \begin{pmatrix} \omega \\ v \end{pmatrix} \middle| \, \omega \in \mathbb{R}^3, v \in \mathbb{R}^3 \right\} \cong \left\{ \begin{pmatrix} \omega \times & v \\ 0 & 0 \end{pmatrix} \right\} \tag{4.1a}$$

$$F^6 \equiv \left\{ \begin{pmatrix} \tau \\ f \end{pmatrix} \middle| \, \tau \in \mathbb{R}^3, f \in \mathbb{R}^3 \right\} \cong \left\{ \begin{pmatrix} \tau \times & f \\ 0 & 0 \end{pmatrix} \right\} \tag{4.1b}$$

donde M^6 es isomorfo al algebra de Lie $se(3)$ del grupo $SE(3)$ de las matrices homogéneas (Lo *et al.*, 2002) y F^6 es isomorfo al álgebra de Lie $se(3)^*$.

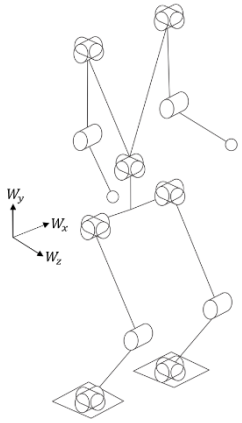


Figura 4.1. Estructura cinemática del robot humanoide bajo estudio.

4.2.1 Derivadas de la representación adjunta

En la deducción del algoritmo es necesario obtener las derivadas de las representaciones adjunta $Ad_G(h): SE(3) \times se(3) \to se(3)$ y $Ad_G^*(h^*): SE(3) \times se(3)^* \to se(3)^*$ que se definen como:

$$Ad_G(h) = GhG^{-1} \tag{4.2a}$$

$$Ad_G^*(h^*) = G^{-1}h^*G \; . \tag{4.2b}$$

La derivada temporal de la representación adjunta se obtiene derivando (4.2a), esto es:

$$\frac{d}{dt}\left(Ad_{T_{\lambda(j),j}}\right)h_{\lambda(j)} = \frac{d}{dt}(e^{-S_j q_j})M_j^{-1}h_{\lambda(j)}M_j e^{S_j q_j} + e^{-S_j q_j}M_j^{-1}h_{\lambda(j)}M_j\frac{d}{dt}(e^{S_j q_j}) \quad (4.3a)$$

$$\frac{d}{dt}\left(Ad_{T_{\lambda(j),j}}\right)h_{\lambda(j)} = \left(Ad_{T_{\lambda(j),j}}h_{\lambda(j)}\right)S_j\dot{q}_j - S_j\dot{q}_j\left(Ad_{T_{\lambda(j),j}}h_{\lambda(j)}\right). \quad (4.3b)$$

La ecuación (4.3b) se simplifica mediante el corchete de Lie empleando la propiedad de simetría $[a,b] = -[b,a]$, esto es:

$$\frac{d}{dt}\left(Ad_{T_{\lambda(j),j}}\right)h_{\lambda(j)} = -ad_{S_j\dot{q}_j}(Ad_{T_{\lambda(j),j}}h_{\lambda(j)}) \quad (4.4a)$$

$$\frac{d}{dt}\left(Ad_{T_{\lambda(j),j}}\right) = -ad_{S_j\dot{q}_j}Ad_{T_{\lambda(j),j}}. \quad (4.4b)$$

De igual manera la derivada temporal de la representación adjunta dual es:

$$\frac{d}{dt}\left(Ad^*_{T_{j,\lambda(j)}}\right) = Ad^*_{T_{j,\lambda(j)}}ad^*_{S_j\dot{q}_j}. \quad (4.5)$$

Las expresiones de las derivadas de la representación adjunta (4.4b) y (4.5) se pueden expresar con notación del álgebra espacial, esto es:

$$\frac{d}{dt}\left(Ad_{T_{\lambda(j),j}}\right) = \frac{d}{dt}\left({}^{j}X_{\lambda(j)}\right) = -S_j\dot{q}_j \times {}^{j}X_{\lambda(j)} \quad (4.6a)$$

$$\frac{d}{dt}\left(Ad^*_{T_{j,\lambda(j)}}\right) = \frac{d}{dt}\left({}^{\lambda(j)}X^*_j\right) = {}^{\lambda(j)}X^*_j S_j\dot{q}_j \times^* \quad (4.6b)$$

Finalmente se consideran las siguientes relaciones en la deducción del algoritmo, esto es:

$$\frac{d}{dt}\left(Ad^*_{T_{j,\lambda(j)}}h^*_j\right) = Ad^*_{T_{j,\lambda(j)}}\left(ad^*_{S_j\dot{q}_j}h^*_j + \dot{h}^*_j\right) \quad (4.7)$$

$$\frac{d}{dt}\left({}^{\lambda(j)}X_j^* h_j^*\right) = {}^{\lambda(j)}X_j^*\left(S_j\dot{q}_j \times^* h_j^* + \dot{h}_j^*\right). \tag{4.8}$$

4.3 Momento centroidal

El momento centroidal se relaciona con las velocidades articulares por (Orin *et al.*, 2013):

$$h_G = A_G(q)\dot{q} \tag{4.9}$$

donde $h_G \in \mathbb{R}^6$ es el momento centroidal, $A_G \in \mathbb{R}^{6\times(6+n_J)}$ es la matriz del momento centroidal, $\dot{q} \in \mathbb{R}^{6+n_J}$ son las velocidades articulares junto con la velocidad (lineal y angular) de la base y n_J es el número de articulaciones. En el balance de postura se considera la dinámica de segundo orden de (4.9), esto es:

$$\dot{h}_G = A_G\ddot{q} + b_G \tag{4.10}$$

donde $b_G = \dot{A}_G\dot{q} \in \mathbb{R}^6$. El objetivo del presente trabajo es la deducción de un algoritmo eficiente para al cálculo del término b_G. El momento centroidal y su matriz se obtiene empleando el algoritmo recursivo de Orin *et al.* (2013) (ver figura 4.2).

El algoritmo de Orin *et al.* (2013) consiste en el cómputo recursivo de la inercia compuesta y la sumatoria del momento espacial de cada uno de los cuerpos compuestos, las ecuaciones recursivas son las siguientes:

$$I_{\lambda(j)}^c = I_{\lambda(j)}^c + {}^{\lambda(j)}X_j^* I_j^c\ {}^jX_{\lambda(j)} \tag{4.11a}$$

$$h_G = h_G + A_{G,j}\dot{q}_j \tag{4.11b}$$

donde $A_{G,j} = {}^{G}X_j^* I_j^c S_j$. Dado que el objetivo es calcular $b_G = \dot{A}_G \dot{q}$, entonces la derivada temporal de la recursión (4.11b) se reescribe ignorando el término $A_{G,j}\ddot{q}_j$:

$$b_G = b_G + \dot{A}_{G,j}\dot{q}_j \tag{4.12}$$

donde $b_G = \dot{h}_G$.

$for\ j = 1\ to\ N_B$
    $I_j^c = I_j$
end
$for\ j = N_B\ to\ 2$
    $I_{\lambda(j)}^c = I_{\lambda(j)}^c + {}^{\lambda(j)}X_j^* I_j^c\ {}^{j}X_{\lambda(j)}$
end
$h_G = 0$
$for\ j = 1\ to\ N_B$
    $A_{G,j} = {}^{G}X_j^* I_j^c S_j$
    $h_G = h_G + A_{G,j}\dot{q}_j$
end

Figura 4.2. Algoritmo para el cómputo de la matriz del momento centroidal.

La derivada temporal del término $A_{G,j}$ se obtiene empleando la relación (4.8), esto es:

$$\dot{A}_{G,j} = \frac{d}{dt}\left({}^{G}X_j^* I_j^c S_j\right) = {}^{G}X_j^*\left(\dot{I}_j^c S_j + S_j\dot{q}_j \times^* I_j^c S_j\right). \tag{4.13}$$

Ahora se toma la derivada temporal de la recursión para el cómputo de la inercia compuesta (4.11a), esto es:

$$\dot{I}_{\lambda(j)}^c = \dot{I}_{\lambda(j)}^c + \frac{d}{dt}\left({}^{\lambda(j)}X_j^*\right) I_j^c\ {}^{j}X_{\lambda(j)} + {}^{\lambda(j)}X_j^* I_j^c \frac{d}{dt}\left({}^{j}X_{\lambda(j)}\right). \tag{4.14}$$

Sustituyendo la derivada temporal de las transformaciones espaciales de la ecuación (4.6a) y (4.6b) en (4.14) se obtiene:

$$\dot{I}^c_{\lambda(j)} = \dot{I}^c_{\lambda(j)} + {}^{\lambda(j)}X^*_j S_j \dot{q}_j \times^* I^c_j \; {}^jX_{\lambda(j)} - {}^{\lambda(j)}X^*_j I^c_j S_j \dot{q}_j \times \; {}^jX_{\lambda(j)} \tag{4.15a}$$

$$\dot{I}^c_{\lambda(j)} = \dot{I}^c_{\lambda(j)} + {}^{\lambda(j)}X^*_j \left[S_j \dot{q}_j \times^* I^c_j - I^c_j S_j \dot{q}_j \times\right] {}^jX_{\lambda(j)} \, . \tag{4.15b}$$

Por lo tanto, de (4.11), (4.12), (4.13) y (4.15) se obtiene el algoritmo propuesto (ver figura 4.3). Dado que M^6 es isomorfo a $se(3)$ y F^6 es isomorfo a $se(3)^*$ es posible reescribir el algoritmo de la figura 4.3 empleando operadores del álgebra de Lie (ver figura 4.4).

$for\ j = 1\ to\ N_B$
$\quad I^c_j = I_j$
$\quad \dot{I}^c_j = 0$
end
$for\ j = N_B\ to\ 2$
$\quad I^c_{\lambda(j)} = I^c_{\lambda(j)} + {}^{\lambda(j)}X^*_j I^c_j \; {}^jX_{\lambda(j)}$
$\quad \dot{I}^c_{\lambda(j)} = \dot{I}^c_{\lambda(j)} + {}^{\lambda(j)}X^*_j \left[S_j \dot{q}_j \times^* I^c_j - I^c_j S_j \dot{q}_j \times\right] {}^jX_{\lambda(j)}$
end
$h_G = 0, b_G = 0$
$for\ j = 1\ to\ N_B$
$\quad A_{G,j} = {}^GX^*_j I^c_j S_j$
$\quad h_G = h_G + A_{G,j}\dot{q}_j$
$\quad \dot{A}_{G,j} = {}^GX^*_j (\dot{I}^c_j S_j + S_j \dot{q}_j \times^* I^c_j S_j)\dot{q}_j$
$\quad b_G = b_G + \dot{A}_{G,j}\dot{q}_j$
end

Figura 4.3. Algoritmo recursivo propuesto.

4.4 Tareas múltiples

Es posible utilizar la dinámica del cuerpo completo en el problema de balance de postura pero esto requiere del cómputo de la matriz de inercia; sin embargo, es posible reducir la dimensión del problema de optimización y el costo computacional del controlador

explotando el algoritmo propuesto de la figura 4.3 mediante el uso de la dinámica centroidal (ya que de esta manera sólo es necesario el cómputo de los términos A_G y $\dot{A}_G\dot{q}$). Por lo tanto, el modelo que se utiliza en el presente trabajo es la dinámica centroidal, esto es:

$$\dot{h}_G = W_g + \sum_{j=1}^{N_c} W_{c,j} \tag{4.16}$$

donde $\dot{h}_G \in \mathbb{R}^6$ es el momento lineal y angular, $W_g = [0_{3\times1}^T \quad mg^T]^T$ es la fuerza debido a la gravedad y $W_{c,j} \in \mathbb{R}^6$ es la fuerza y par debido al punto de contacto j. Sustituyendo (4.10) en (4.16) se obtiene:

$$A_G\ddot{q} + \dot{A}_G\dot{q} = W_g + \sum_{j=1}^{N_c} W_{c,j}\,. \tag{4.17}$$

$for\ j = 1\ to\ N_B$
    $I_j^c = I_j$
    $\dot{I}_j^c = 0$
end
$for\ j = N_B\ to\ 2$
    $I_{\lambda(j)}^c = I_{\lambda(j)}^c + Ad_{T_{j,\lambda(j)}}^* I_j^c Ad_{T_{\lambda(j),j}}$
    $\dot{I}_{\lambda(j)}^c = \dot{I}_{\lambda(j)}^c + Ad_{T_{j,\lambda(j)}}^* \left[ad_{S_j\dot{q}_j}^* I_j^c - I_j^c ad_{S_j\dot{q}_j}\right] Ad_{T_{\lambda(j),j}}$
end
$h_G = 0, b_G = 0$
$for\ j = 1\ to\ N_B$
    $A_{G,j} = Ad_{T_{j,G}}^* I_j^c S_j$
    $h_G = h_G + A_{G,j}\dot{q}_j$
    $\dot{A}_{G,j} = Ad_{T_{j,G}}^* (\dot{I}_j^c S_j + ad_{S_j\dot{q}_j}^* I_j^c S_j)\dot{q}_j$
    $b_G = b_G + \dot{A}_{G,j}\dot{q}_j$
end

Figura 4.4. Algoritmo propuesto expresado con operadores del álgebra Lie.

El utilizar (4.17) en lugar de la dinámica del cuerpo completo en el balance de postura permite reducir la dimensión del problema de optimización por lo que sólo es necesario optimizar las aceleraciones articulares $\ddot{q}$ y las fuerzas de contacto λ que se restringen a permanecer dentro de una pirámide de fricción. Los pares articulares se obtienen en un paso posterior a la optimización mediante el algoritmo recursivo de Newton Euler para sistemas de base flotante.

El algoritmo propuesto en Wensing y Orin (2016) para el cómputo de $A_G(q)$ y $\dot{A}_G(q)\dot{q}$ tiene complejidad computacional de $O(N)$ y $O(1)$ respectivamente cuando se tiene conocimiento previo de la matriz de inercia $H(q)$ y el término de Coriolis $h(q,\dot{q})$; sin embargo, cuando esta información no se conoce a priori el costo computacional es $O(N^2)$ y $O(N)$, respectivamente. La complejidad computacional del algoritmo propuesto de la figura 4.3 es $O(N)$, en la tabla 4.1 se muestran diversos algoritmos propuestos en la literatura y su complejidad computacional.

Durante el balance se considera una tarea que restringe el movimiento de los puntos de contacto con respecto a la superficie de contacto mediante la siguiente restricción cinemática:

$$J\ddot{q} + \dot{J}\dot{q} = 0 \tag{4.18}$$

donde $J \in \mathbb{R}^{3\times(6+n_J)}$ es el Jacobiano aumentado de los n_c puntos de contacto. Para incrementar la robustez del balance se considera la regulación del momento lineal y angular, empleando la siguiente tarea:

$$\dot{h}_{des} = A_G\ddot{q} + b_G \tag{4.19}$$

donde $\dot{h}_{des} \in \mathbb{R}^6$ es una referencia deseada que se define como en (3.26). Por último, se especifica una tarea para regular la postura del robot a una referencia deseada mediante control PD, esto es:

$$\ddot{q}_a - \ddot{q}_{des} = 0 \tag{4.20}$$

donde $\ddot{q}_{des} \in \mathbb{R}^{N_J}$ es la referencia deseada que se define como en (3.27).

Tabla 4.1. Algoritmos propuestos en la literatura.

Término	Algoritmo	Complejidad
$M(q)$	Cuerpo compuesto (Featherstone, 2014)	$O(N^2)$
$h(q, \dot{q})$	Recursivo de Newton Euler (Featherstone, 2014)	$O(N)$
$A_G(q)$	Recursivo matriz del momento centroidal (Orin et al., 2013)	$O(N)$
$A_G(q)$	Propuesto en (Wensing y Orin, 2016)	$O(N^2)$
$\dot{A}_G(q)\dot{q}$	Propuesto en (Wensing y Orin, 2016)	$O(N)$
$\dot{A}_G(q)\dot{q}$	No recursivo (Macchietto et al., 2009; De Lasa et al., 2010)	$O(N^2)$
$A_G(q)$ y $\dot{A}_G(q)\dot{q}$	Figura 4.3	$O(N)$

4.5 Control de cuerpo completo jerárquico

El problema de balance se formula como la solución a una jerarquía de tareas y se resuelve mediante programación cuadrática jerárquica. Sin embargo, cuando se presentan tareas singulares el optimizador se puede quedar ciclado sin converger. Para lidiar de manera robusta ante posibles tareas singulares el problema de optimización se puede reformular empleando el modo elástico permitiendo resolver el problema mediante programación lineal; sin embargo esto incrementa la dimensión del problema de optimización. En el presente trabajo se sigue la metodología propuesta en Herzog *et al.* (2014), por lo tanto cada programa cuadrático a resolver en la jerarquía de tareas es:

$$\begin{aligned} \min_{x,u,v} \quad & \frac{1}{2}(u^T u + v^T v) \\ s.t. \quad & Ax - b = u \\ & Gx - d \leq v \end{aligned} \tag{4.21}$$

donde $u \in \mathbb{R}^l$ y $v \in \mathbb{R}^m$ son vectores de variables adicionales que suavizan las l y m restricciones de igualdad y desigualdad, respectivamente. El programa cuadrático (4.21) se resuelve mediante el método del punto interior con regulación dual siguiendo la metodología de Rojas *et al.* (2019). La jerarquía de tareas que se emplea en el presente trabajo se muestra en la tabla 4.2.

Tabla 4.2. Jerarquía de tareas.

Prioridad	Tarea (Descripción)	No. Ecuación
1	Dinámica centroidal	4.17
2	Aceleración nula de los puntos de contacto	4.18
3	Regulación del momento centroidal	4.19
4	Postura de referencia	4.20

4.6 Resultados

Se consideran 4 escenarios de simulación en donde las ecuaciones de movimiento se integran empleando el método de Euler simpléctico durante 16000 pasos de integración ($\Delta t = 1 \times 10^{-3}$). Para realizar las simulaciones se desarrolla un simulador en Matlab para sistemas de base flotante resolviendo el siguiente sistema de ecuaciones:

$$\begin{bmatrix} M(q_i) & -J^T(q_i) \\ J(q_i) & \beta I \end{bmatrix} \begin{bmatrix} \dot{q}_{i+1} \\ \lambda \end{bmatrix} = \begin{bmatrix} M(q_i)\dot{q}_i + \left(S^T\tau - h(q_i, \dot{q}_i)\right)\Delta t \\ -\alpha c(q_i) \end{bmatrix} \tag{4.22}$$

donde $\alpha \in \mathbb{R}$ y $\beta \in \mathbb{R}$ son constantes positivas que suavizan el contacto y se denominan estabilización de Baumgarte (Baumgarte, 1972) y "Constraint Force Mixing" (Smith, 2005), $c(q_i)$: $\mathbb{R}^{6+n_J} \rightarrow \mathbb{R}^m$ es la función de las m restricciones cinemáticas de igualdad, $\Delta t \in \mathbb{R}$ es el paso de integración y $\lambda \in \mathbb{R}^{3n_c}$ ahora son los impulsos de los n_c puntos de contactos activos. La formulación que se presenta en (4.22) se elige en lugar de la formulación en el espacio de los contactos comúnmente empleada en la simulación de cuerpo rígidos en donde primeramente sólo se optimizan las fuerzas de contacto y posteriormente se obtienen las velocidades; sin embargo, el problema (4.22) es superior numéricamente ya que se obtiene un problema de optimización con matrices más dispersas y un número de condición más pequeño (Redon *et al.,* 2002).

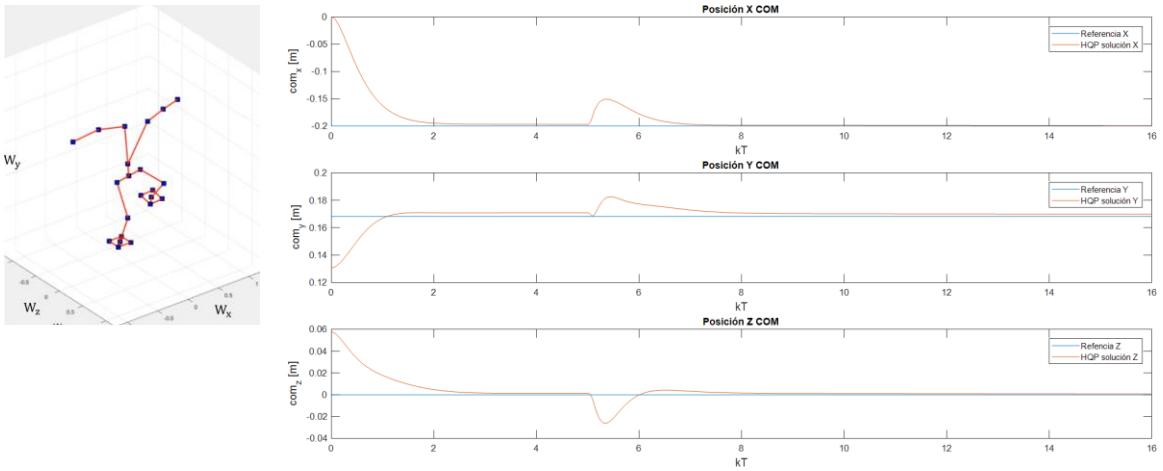
Para el cómputo de la matriz del momento centroidal $A_G(q)$ y el término $\dot{A}_G(q)\dot{q}$ se emplea el algoritmo propuesto de la figura 4.3. El número de puntos de contacto que se consideran en los escenarios 1, 2 y 3 es $n_c = 4$, mientras que en el escenario 4 se considera $n_c = 8$ y el número de grados de libertad actuados es $n_J = 18$. Las ganancias empleadas en el controlador y el simulador se ilustran en la tabla 4.3. Por último, las propiedades del modelo del robot humanoide se basan en la tabla 3.1 del capítulo 3.

Tabla 4.3. Ganancias empleadas en el controlador y simulador.

Descripción	Ganancias	Valor
Regulación del momento angular	K_G	10
Regulación de la posición del COM	K_p y K_v	10 y $2\sqrt{10}$
Postura de referencia	K_p y K_v	10 y $2\sqrt{10}$
Estabilización de Baumgarte	α	25
Constraint Force Mixing	β	1×10^{-3}
Coeficiente de fricción	μ	0.6

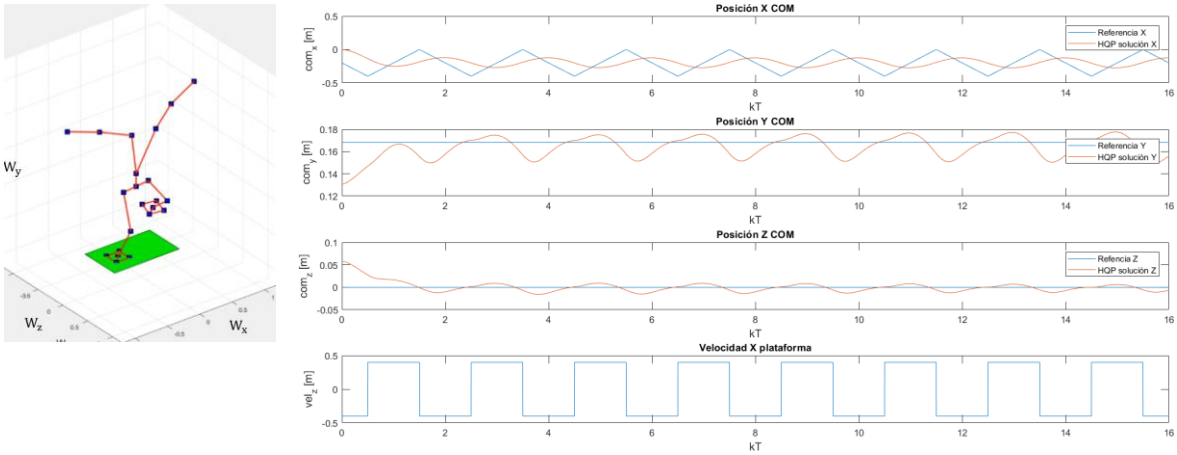
El escenario 1 consiste en mantener el balance utilizando sólo una pierna y se aplica una perturbación en el torso de $f^E = [150 \quad 0 \quad 0]^T N$ a partir del segundo 5 durante 100 ms. Los resultados del escenario 1 se muestran en la figura 4.5. Mientras que el escenario 2

consiste en mantener el balance utilizando también sólo una pierna pero ahora sobre una plataforma que alterna su movimiento con una velocidad constante de $0.4\,m/s$ en el eje x, los resultados se ilustran en la figura 4.6.



(a) (b)

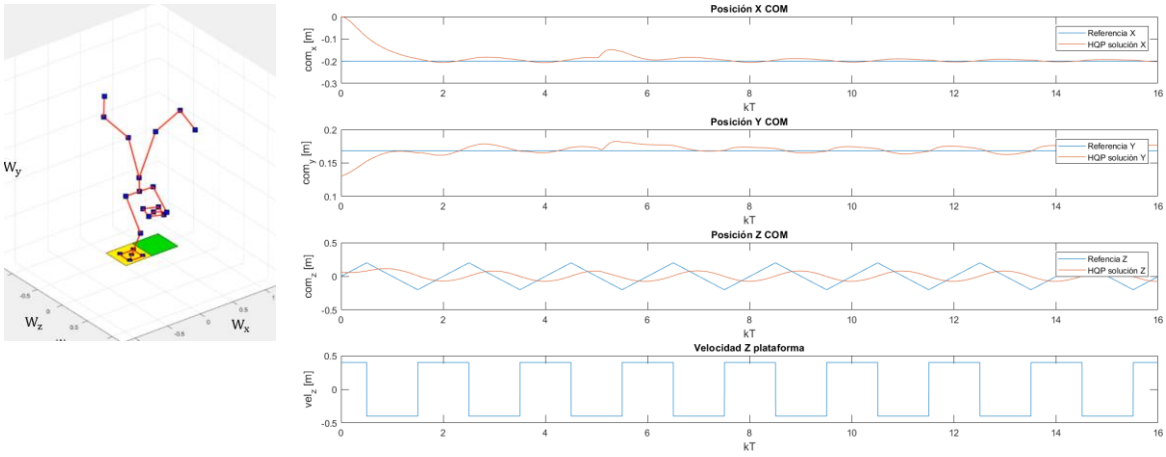
Figura 4.5. (a) Visualización en Matlab del escenario 1, (b) gráficas de la posición del centro de masas del escenario 1.



(a) (b)

Figura 4.6. (a) Visualización en Matlab del escenario 2, (b) gráficas de la posición del centro de masas del escenario 2.

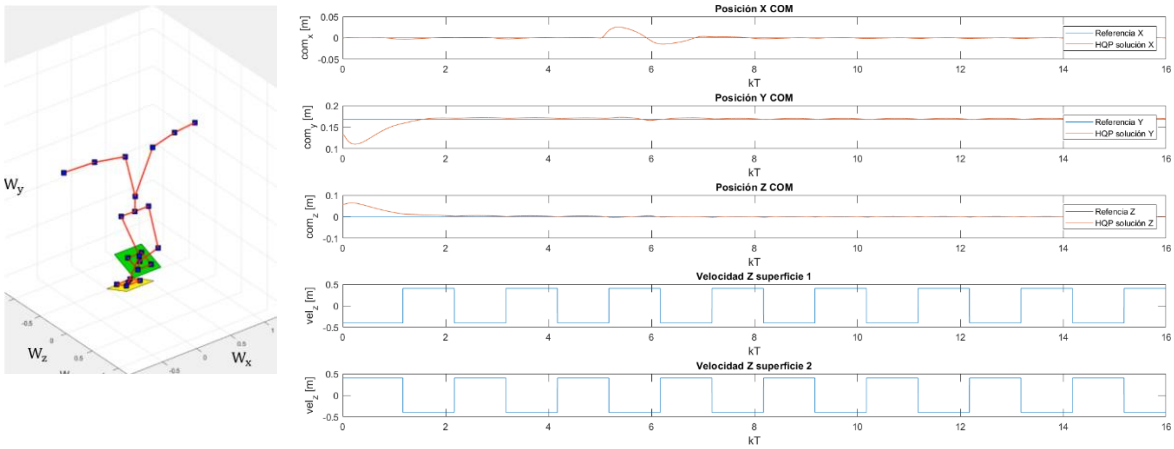
El escenario 3, al igual que el segundo, consiste en mantener el balance en un plataforma que alterna su movimiento con una velocidad constante de $0.4\,m/s$; sin embargo, en este caso el movimiento es sobre el eje z. También se aplica una perturbación sobre el torso de $f^E = [150 \quad 0 \quad 0]^T N$ a partir del segundo 5 durante $100\,ms$, los resultados se ilustran en la figura 4.7.



(a) (b)

Figura 4.7. (a) Visualización en Matlab del escenario 3, (b) gráficas de la posición del centro de masas del escenario 3.

Por último, en el escenario 4 se considera el balance con el robot puesto sobre dos plataformas disparejas con inclinación de 20° y -20° y que se desplazan en direcciones opuestas sobre el eje z de manera alternante con una velocidad constante de $0.4\,m/s$. También se aplica una perturbación sobre el torso de $f^E = [150 \quad 0 \quad 0]^T N$ a partir del segundo 5 durante 80 pasos de integración, los resultados se muestran en la figura 4.8.



(a) (b)

Figura 4.8. (a) Visualización en Matlab del escenario 4, (b) gráficas de la posición del centro de masas del escenario 4.

4.7 Conclusiones

En este capítulo se ha introducido un algoritmo $O(N)$ para el cómputo de la matriz del momento centroidal y su derivada temporal de manera conjunta con un menor costo computacional que las diferentes versiones propuestas en la literatura. El controlador presentado no requiere del conocimiento de la matriz de inercia y el vector de Coriolis; además se explota el algoritmo propuesto requiriendo únicamente la matriz del momento centroidal y su derivada para definir la tarea de la dinámica del sistema y la tarea que regula el momento centroidal. Posteriormente, los pares articulares se obtienen mediante el algoritmo recursivo de Newton Euler, de esta manera se reduce el costo computacional dado que ambos algoritmos (algoritmo propuesto y Newton Euler) tienen una complejidad $O(N)$ evitando el algoritmo del cuerpo compuesto para el cálculo de la matriz de inercia con un costo de $O(N^2)$.

CAPÍTULO 5

PLANEACIÓN JERÁRQUICA DE CUERPO COMPLETO

En el presente capítulo se introduce una reformulación del problema de la planeación de trayectorias de cuerpo completo mediante una jerarquía de tareas; de forma tal que es posible lidiar con posibles conflictos entre las tareas mostrando mayor robustez que algoritmos previamente propuestos en la literatura que utilizan programación no lineal. El modelo empleado utiliza la dinámica centroidal y la cinemática del cuerpo completo. El problema de planeación se descompone en 2 partes que se alternan hasta la convergencia. La primera parte consiste en la planeación con la dinámica centroidal, la cual se linealiza empleando heurísticas y se resuelve mediante programación cuadrática. Y la segunda parte consiste en la planeación cinemática de cuerpo completo la cual se resuelve mediante la programación cuadrática jerárquica. Para verificar la efectividad del algoritmo propuesto se consideran 4 escenarios de planeación con dos modelos y velocidades de desplazamiento diferentes. Finalmente, de acuerdo con las gráficas de posición del centro de masas y de la fuerza normal se concluye que el desempeño del algoritmo propuesto es satisfactorio.

5.1 Introducción

En la literatura existen diversas técnicas para abordar el problema de planeación de caminata. En general se pueden destacar 2 vertientes: métodos que emplean la dinámica y cinemática del cuerpo completo y métodos que emplean modelos reducidos como el péndulo lineal invertido. Independientemente del modelo dinámico utilizado es posible formular el problema de planeación de caminata como un problema de optimización. Los modelos reducidos y la dinámica del cuerpo completo se pueden considerar los casos extremos de un espectro de posibles modelos dinámicos y cinemáticos que pueden ser empleados en el problema de planeación (Dai *et al.*, 2014).

Optimizar el patrón de locomoción requiere optimizar los tiempos de los eventos de contacto, esto implica el modelado de la naturaleza discreta de los puntos de contacto (el cual es un fenómeno binario). En la literatura existen diversas técnicas para la optimización del patrón de locomoción, una de ellas consiste en modelar los contactos mediante resortes y amortiguadores (método "penalty"), restricciones complementarias no lineales y programación entera. Sin embargo, el método "penalty" requiere ganancias muy elevadas para evitar penetración con el suelo lo que empeora el condicionamiento del problema de optimización resultante; mientras que el uso de las restricciones complementarias incrementa la no linealidad del problema de optimización. La optimización mixta entera provoca una explosión combinatorial, por ende esta técnica posee un costo computacional enorme.

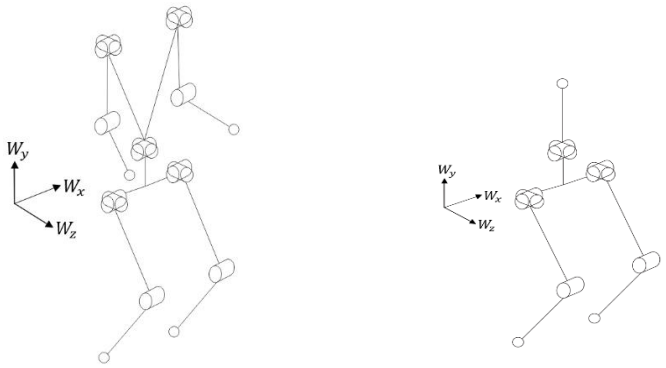
En Posa y Tedrake (2013) se introduce la planeación de los tiempos y las posiciones de los puntos de contacto mediante restricciones complementarias no lineales, esto permite sintetizar el patrón de locomoción de robots con patas. La dinámica de cuerpo completo se discretiza utilizando integración de Euler implícita y el problema de planeación se resuelve mediante el modo elástico del optimizador SNOPT (Gill *et al.*, 2005). En Dai *et al.* (2014) se introduce la planeación de trayectorias con la dinámica centroidal y la cinemática del cuerpo completo, además se emplean restricciones complementarias para la optimización de

los tiempos de contacto. Esto permite la síntesis de la locomoción del robot Atlas y Little Dog mediante PCS. En Herzog *et al.* (2016) se propone una descomposición de la planeación de cuerpo completo en 2 dos problemas que se alternan hasta la convergencia, esos problemas son la planeación con la dinámica centroidal y la planeación cinemática de cuerpo completo. En Bledt *et al.* (2017) se propone realizar MPC en el Cheetah del MIT en simulación con la dinámica de un cuerpo rígido linealizada mediante heurísticas.

Por otro lado, en Siciliano y Slotine (1991) se introduce la ejecución de múltiples tareas en robots redundantes mediante la proyección de las tareas de menor prioridad en el espacio nulo de las tareas de mayor prioridad. En Mistry *et al.* (2007) y Mistry *et al.* (2008) se aborda la cinemática inversa de cuerpo completo mediante proyección iterativa en los espacios nulos (utilizando Jacobianos de base flotante) en un robot cuadrúpedo y bípedo, respectivamente. En Rocchi *et al.* (2014) se utiliza programación cuadrática jerárquica (PCJ) para el cómputo de la cinemática inversa de cuerpo completo del robot humanoide COMAN. En Escande *et al.* (2014) se reduce el costo computacional de la PCJ mediante descomposición ortogonal de las tareas de mayor prioridad reduciendo así la dimensión de los programas cuadráticos en la jerarquía de tareas. En Hoffman *et al.* (2018) se introducen restricciones dinámicas en la cinemática inversa de cuerpo completo con programación cuadrática (en particular límites en los pares articulares al nivel de la velocidad). Mientras que en Dai *et al.* (2019) la cinemática inversa se formula como un problema de optimización mixto entero convexo mediante la aproximación de la restricción no lineal $SO(3)$ de las matrices de rotación con restricciones mixtas enteras.

El presente trabajo está inspirado en la descomposición del problema de planeación de cuerpo completo propuesta por Herzog *et al.* (2016); sin embargo, el trabajo previamente mencionado considera todas las tareas con la misma prioridad en la planeación cinemática (lo cual puede resultar en problemas de convergencia). El algoritmo propuesto en el presente capítulo es capaz de manejar conflictos entre las múltiples tareas mediante PCJ. La planeación dinámica se encuentra inspirado en la linealización de la dinámica de un cuerpo

rígido propuesta en Bledt *et al.* (2017). Sin embargo, en el presente trabajo se linealiza la dinámica centroidal y se extiende al cuerpo completo. Por último, el algoritmo propuesto se aplica en la planeación de caminata de cuerpo completo, en la figura 5.1 se muestra la estructura cinemática de los modelos bajo estudio.



(a) (b)

Figura 5.1. Estructura cinemática humanoide bajo estudio (a) con brazos y (b) sin brazos.

5.2 Planeación con dinámica centroidal

Como se ha venido mencionando, la dinámica centroidal está dada por:

$$m\ddot{r} = mg + \sum_j \lambda_j \tag{5.1a}$$

$$\dot{k} = \sum_j (c_j - r) \times \lambda_j \tag{5.1b}$$

donde $m \in \mathbb{R}$ es la masa del robot, $g \in \mathbb{R}^3$ es la aceleración de gravedad, $\lambda_j \in \mathbb{R}^3$ es la fuerza en el punto de contacto j, $\dot{k} \in \mathbb{R}^3$ es la derivada del momento angular, $c_j \in \mathbb{R}^3$ es la posición del punto de contacto j y $r \in \mathbb{R}^3$ es la posición del centro de masas.

Utilizar la dinámica centroidal en la planeación de caminata convierte el problema de optimización en no lineal y esto a su vez no ofrece una garantía sobre la convergencia del

optimizador. En el presente trabajo el momento angular se linealiza siguiendo una metodología similar a Bledt *et al.* (2017), por lo tanto el momento angular se simplifica a:

$$\dot{k} = \sum_j s_j \times \lambda_j \tag{5.2}$$

donde $s_j \in \mathbb{R}^3$ es una constante que se obtiene especificando la posición del centro de masas y la posición de los puntos de contacto a priori mediante heurísticas. En particular las posiciones de los puntos de contacto se determinan mediante la heurística de Raibert (1986), esto es:

$$c_z = r_z + \frac{1}{2} v_z \Delta t_s \tag{5.3}$$

donde $c_z \in \mathbb{R}$ es la posición final del punto de contacto, $v_z \in \mathbb{R}$ es la velocidad del centro de masas y $\Delta t_s \in \mathbb{R}$ es el tiempo que el punto de contacto permanece en el suelo durante la fase de soporte. Mientras la posición y velocidad del centro de masas en el punto de planeación k se obtiene mediante interpolación lineal, esto es:

$$r[k] = r_i + k\Delta t C\left(r_f - r_i\right) \tag{5.4a}$$

$$\dot{r}[k] = \dot{r}_i + k\Delta t C\left(\dot{r}_f - \dot{r}_i\right) \tag{5.4b}$$

donde $r_i \in \mathbb{R}^3$ y $r_f \in \mathbb{R}^3$ son la posición inicial y final del centro de masas, respectivamente; $\dot{r}_i \in \mathbb{R}^3$ y $\dot{r}_f \in \mathbb{R}^3$ son la velocidad inicial y final del centro de masas, respectivamente; $C = 1/N\Delta t$ es una constante positiva; N es el horizonte de planeación; y Δt es el tiempo de integración.

Las fuerzas de contacto están restringidas a ser físicamente consistentes empleando el cono de fricción de Coulomb, esto es:

$$\mathcal{K} = \{\lambda \in \mathbb{R}^3 \mid n^T\lambda \geq 0, \|\lambda - (n^T\lambda)n\| \leq \mu n^T\lambda\} \tag{5.5}$$

donde $n \in \mathbb{R}^3$ es un vector tangente a la superficie de contacto y $\mu \in \mathbb{R}$ es el coeficiente de fricción. En el presente trabajo el cono $\mathcal{K}$ es remplazado por una pirámide de fricción $\mathcal{P}$ más conservativa $\mathcal{P} \subset \mathcal{K}$, esto es:

$$\mathcal{P} = \left\{ \sum_{j=1}^{N_d} \beta_j w_j \mid \beta_j \geq 0 \right\} \tag{5.6}$$

donde $w_j = n + \mu d_j$ es uno de los vectores base, $d_j \in \mathbb{R}^3$ es un vector tangente a la superficie de contacto y N_d es el número de vectores base empleados para aproximar $\mathcal{K}$. Utilizar (5.6) permite formular el problema de planeación como un programa cuadrático ya que (5.5) resultaría en un programa cónico de segundo orden. Para mantener la optimización de planeación como un problema convexo, el patrón de locomoción se especifica a priori ya que el uso de restricciones complementarias introducen no linealidades y la programación entera posee un gran costo computacional.

La dinámica centroidal linealizada se discretiza mediante el integrador de Euler simpléctico dado que este integrador en particular tiende a preservar la energía del sistema; a diferencia del integrador de Euler explícito e implícito que introducen o remueven energía del sistema (Stam, 2009). Con estas restricciones se formula el problema de planeación que se muestra en (P1). El conjunto de variables a optimizar en (P1) es:

$$x = \left\{ h[k], \dot{h}[k], \lambda[k], \beta[k], r[k], \dot{r}[k], \ddot{r}[k] \;\middle|\; k = 1, \dots, N \right\} \tag{5.7}$$

donde $\lambda[k] = [\lambda_1^T[k] \quad \lambda_2^T[k]]^T \in \mathbb{R}^6$ y $\beta[k] = [\beta_1^T[k] \quad \beta_2^T[k]]^T \in \mathbb{R}^8$.

El número de variables a optimizar en (P1) por punto de planeación es 29, entonces el número de variables a optimizar en total dado un horizonte de planeación N es $29N$.

$$
\begin{aligned}
\min_{x} \quad & \frac{1}{2}\sum_{j=1}^{N}\left(\left\|h[j]-\bar{h}[j]\right\|^2+\|r[j]-\bar{r}[j]\|^2+\sum_{i=1}^{2}\|(r[j]+s_i[j])-\bar{c}_i[j]\|^2\right) && \text{(P1)}\\
s.t. \quad & \dot{r}[k+1]=\dot{r}[k]+\ddot{r}[k+1]\Delta t \\
& r[k+1]=r[k]+\dot{r}[k+1]\Delta t \\
& k[k+1]=k[k]+\dot{k}[k+1]\Delta t && \left.\right\} \ \forall\, k \in \{1,..,N-1\}\\
& m\ddot{r}[k]=mg+\sum_{j=1}^{2}\lambda_j[k] \\
& \dot{k}[k]=\sum_{j=1}^{2}s_j[k]\times\lambda_j[k] \\
& \forall j \quad \lambda_j[k]=W_j\beta_j[k] \\
& \forall j \quad \beta_j[k]\geq 0 \\
& \forall j \quad n^T\lambda_j^{swing}[k]=0 && \left.\right\} \ \forall\, k \in \{1,..,N\}\\
& r[1]=r_i, \qquad \dot{r}[1]=\dot{r}_i \\
& r[N]=r_f, \qquad \dot{r}[N]=\dot{r}_f
\end{aligned}
$$

Los resultados de la planeación cinemática se almacenan en los arreglos del momento angular $\bar{h}\in\mathbb{R}^{3N}$ y posición del centro de masas $\bar{r}\in\mathbb{R}^{3N}$. En (P1) las posiciones de los puntos de contacto no se optimizan y se determinan a priori mediante heurísticas, los resultados se almacenan en los arreglos $\bar{c}_1\in\mathbb{R}^{3N}$ y $\bar{c}_2\in\mathbb{R}^{3N}$. Además, en (P1) las fuerzas de contacto se igualan a cero durante la fase aérea para la pierna en cuestión con la restricción $n^T\lambda_j^{swing}[k]=0$. Minimizando el término $(r[j]+s_i[j])-\bar{c}_i[j]$ es una manera de introducir los límites cinemáticos del robot.

La función objetivo de (P1) minimiza una posible desviación de la solución encontrada por la planeación cinemática. Posteriormente los resultados de la planeación dinámica se utilizan como valores deseados en la planeación cinemática, este proceso se itera hasta la convergencia. El problema (P1) se resuelve con el método del punto interior empleando regularización dual, tal como en Rojas *et al.* (2019).

5.3 Planeación cinemática

La configuración del cuerpo completo $q \in \mathbb{R}^{6+n_J}$ del robot es:

$$q = \begin{bmatrix} \theta_B \\ p_B \\ \theta \end{bmatrix} \tag{5.8}$$

donde $\theta_B \in \mathbb{R}^3$ es la orientación de la base en ángulos de Euler, $p_B \in \mathbb{R}^3$ es la posición de la base, $\theta \in \mathbb{R}^n$ son las coordenadas articulares y n_J es el número de articulaciones. Cuando un robot es redundante (como es el caso típico con los robots con patas) es posible ejecutar tareas secundarias en el espacio nulo de las tareas de mayor prioridad. Formulaciones previamente propuestas en la literata suponen que todas las tareas tienen la misma prioridad, lo cual puede causar problemas de convergencia cuando existan conflictos entre las tareas. Por lo tanto, en el presente trabajo la planeación cinemática se resuelve empleando programación cuadrática jerárquica. La planeación cinemática consiste en resolver la cinemática inversa con múltiples tareas en cada punto de planeación. Para garantizar la convergencia del problema de cinemática inversa se impone una dinámica exponencial en el error de las tareas (Rocchi *et al.*, 2014), esto es:

$$\dot{e} + \lambda e = 0 \tag{5.9a}$$

$$\frac{\partial e}{\partial q}\dot{q} + \lambda e = J\dot{q} + \lambda e = 0 \tag{5.9b}$$

donde $\lambda \in \mathbb{R}$ es una constante positiva que controla la convergencia del error, $J \in \mathbb{R}^{3\times(6+n_J)}$ es el Jacobiano de la función de error y $\dot{q} \in \mathbb{R}^{6+n_J}$ son las velocidades articulares y velocidad lineal y angular de la base. En el presente trabajo se consideran 4 tareas donde la tarea de mayor prioridad consiste en el cómputo de las coordenadas articulares dadas las posiciones deseadas de los puntos de contacto, esto es:

$$J_{FK_j}(q)\dot{q} = -\lambda\left(FK_j(q) - c_j^d\right) \tag{5.10}$$

donde $c_j^d \in \mathbb{R}^3$ es la posición deseada, $FK_j(q)\colon \mathbb{R}^{6+n_J} \to \mathbb{R}^3$ es la función cinemática directa y $J_{FK_j}(q)$ es el Jacobiano de la función cinemática directa del punto de contacto j respectivamente.

La segunda tarea controla la posición del centro de masas, esto es:

$$J_{COM}(q)\dot{q} = -\lambda(COM(q) - r^d) \tag{5.11}$$

donde $r^d \in \mathbb{R}^3$ es la posición deseada, $COM(q)\colon \mathbb{R}^{6+n_J} \to \mathbb{R}^3$ es la función cinemática directa y $J_{COM}(q)$ es el Jacobiano del centro de masas respectivamente.

La siguiente tarea consiste en la regulación del momento angular, esto es:

$$A_G(q)\dot{q} = k^d \tag{5.12}$$

donde $k^d \in \mathbb{R}^3$ es la momento angular deseado obtenido de la planeación dinámica y $A(q) \in \mathbb{R}^{3\times(6+n_J)}$ son las primeras 3 filas de la matriz del momento centroidal. Por último, se regula la postura del cuerpo completo durante la locomoción con la siguiente tarea:

$$I\dot{q} = -\lambda(q - q^d) \tag{5.13}$$

donde $I \in \mathbb{R}^{(6+n_J)\times(6+n_J)}$ es la matriz identidad y $q^d \in \mathbb{R}^{6+n_J}$ es una referencia deseada. En la tabla 1 se muestran todas las tareas empleadas en esta propuesta junto con sus prioridades.

El problema de optimización a resolver en la jerarquía de tareas es:

$$\dot{q}_j^* = \min_{\dot{q}} \frac{1}{2}\left\|J_j\dot{q} + \lambda e_j\right\|^2 + \frac{\varepsilon}{2}(\dot{q}^T\dot{q} + \lambda^T\lambda) \quad (5.14)$$
$$s.t \quad J_1\dot{q} = J_A\dot{q}_1^*$$
$$\vdots$$
$$J_{j-1}\dot{q} = J_{j-1}\dot{q}_{j-1}^*$$

donde $\dot{q}_j^*$ es la solución de la tarea j. Una vez obtenida la solución $\dot{q}_4^* \in \mathbb{R}^{6+n_J}$ a la jerarquía de tareas, ésta se integra para obtener las posiciones articulares $q = q + \dot{q}_4^*\delta t$. Para incrementar la robustez ante configuraciones singulares e inconsistencias en los programas cuadráticos de la jerarquía de tareas, se agrega regulación a las variables duales y de optimización.

Tabla 5.1. Jerarquía de tareas.

Prioridad	Tarea (Descripción)	No. Ecuación
1	Posición de los puntos de contacto	5.10
2	Posición del centro de masas	5.11
3	Regulación del momento angular	5.12
4	Postura de referencia	5.13

En la figura 5.2 se muestra un pseudo código del algoritmo propuesto para la planeación de cuerpo completo que se itera hasta un número máximo de iteraciones k_{max}.

```
x = x_init,  q = q_init
for k = 1 to k_max
    x = CD_Planner(x)
    [q, x] = HQP_Planner(q, x)
end
```

Figura 5.2. Algoritmo propuesto para planeación de cuerpo completo.

En el algoritmo de la figura 5.2 la función $CD\_Planner()$ realiza la planeación con la dinámica centroidal con (P1). Por último, la función $HQP\_Planner(q, x)$ realiza la planeación cinemática con programación cuadrática jerárquica.

5.4 Resultados

Se consideran 4 escenarios de planeación con un horizonte de $N = 80$ puntos. Por lo tanto, el número de variables de optimización son 1160. El patrón de locomoción se especifica a priori activando y desactivando las fuerzas de contacto de las piernas izquierda y derecha cada 10 puntos de planeación, también se normalizan las fuerzas de contacto por el peso del robot para mejorar el condicionamiento del problema de optimización. La matriz $A_G(q)$ se obtiene empleando el algoritmo de Orin *et al.* (2013). Por último, los primeros dos escenarios consisten en la locomoción en el eje z con el modelo simplificado de la figura 5.1 (b) con una masa de $m = 67.5kg$ y un paso de integración de $\Delta t = 1/80$. Mientras los escenarios tres y cuatro consisten en la locomoción en el eje z con el modelo de la figura 5.1 (a) con una masa de $m = 77.5kg$ y un paso de integración de $\Delta t = 1/100$.

Las condiciones iniciales y finales en posición y velocidad del centro de masas en el primer escenario son: $r_i = [0 \quad 0 \quad 0]^T m, r_f = [0 \quad 0 \quad 2]^T m, \dot{r}_i = [0 \quad 0 \quad 2]^T\, m/s$ y $\dot{r}_f = [0 \quad 0 \quad 2]^T\, m/s$. Los resultados se muestran en las figuras 5.3 y 5.4.

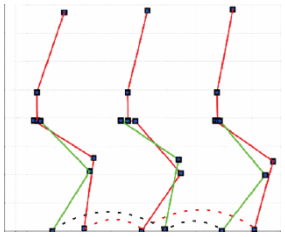


Figura 5.3. Locomoción en el escenario 1.

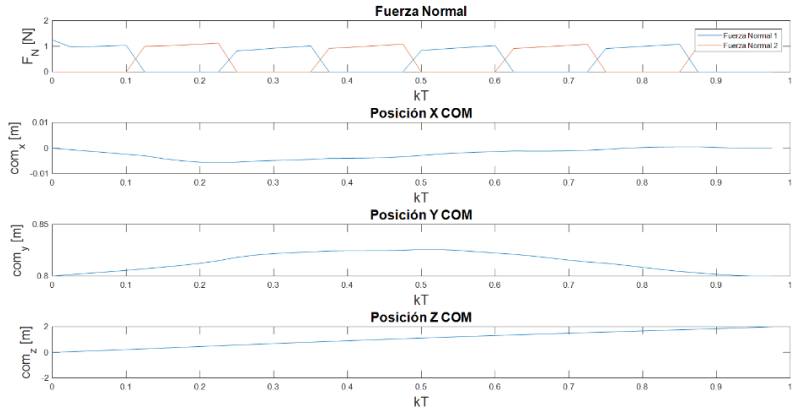


Figura 5.4. Gráficas de la fuerza normal y posición del centro de masas del escenario 1.

Las condiciones iniciales y finales en posición y velocidad del centro de masas del escenario 2 son: $r_i = [0 \quad 0 \quad 0]^T m$, $r_f = [0 \quad 0 \quad 3]^T m$, $\dot{r}_i = [0 \quad 0 \quad 3]^T m/s$ y $\dot{r}_f = [0 \quad 0 \quad 3]^T m/s$. Los resultados se muestran en la figuras 5.5 y 5.6.

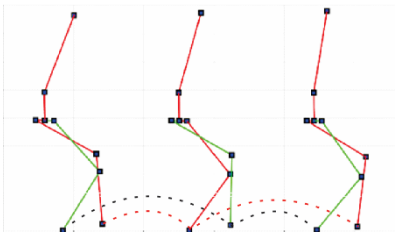


Figura 5.5. Locomoción en el escenario 2.

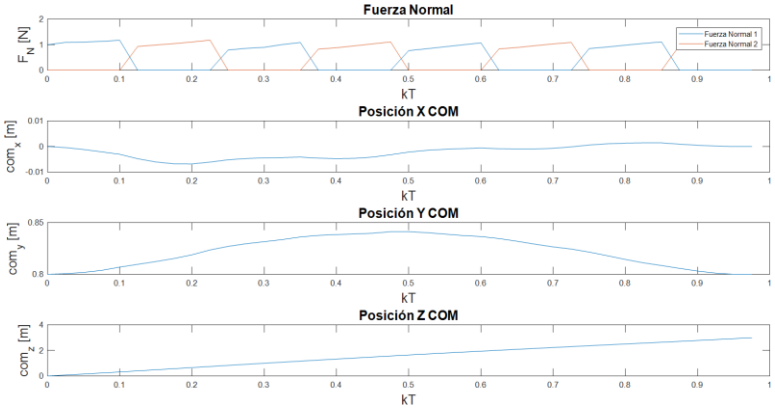


Figura 5.6. Gráficas de la fuerza normal y posición del centro de masas del escenario 2.

Las condiciones iniciales y finales en posición y velocidad del centro de masas en el tercer escenario son: $r_i = [0 \quad 0 \quad 0]^T m$, $r_f = [0 \quad 0 \quad 1.2]^T m$, $\dot{r}_i = [0 \quad 0 \quad 1.5]^T \, m/s$ y $\dot{r}_f = [0 \quad 0 \quad 1.5]^T \, m/s$. Los resultados se muestran en la figuras 5.7 y 5.8.

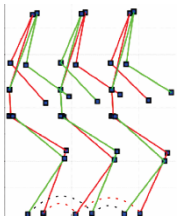


Figura 5.7. Locomoción en el escenario 3.

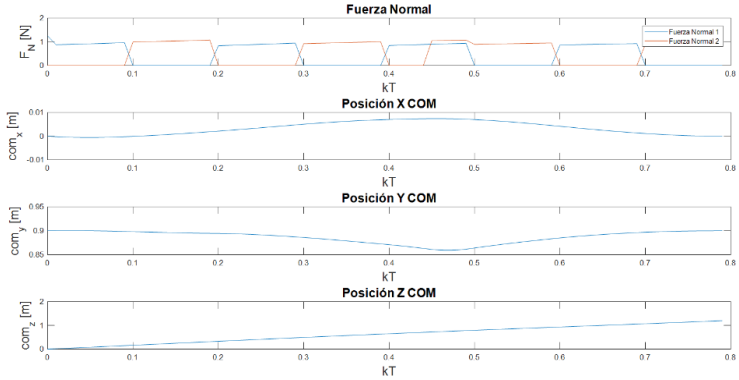


Figura 5.8. Gráficas de la fuerza normal y posición del centro de masas del escenario 3.

En las figuras 5.9 y 5.10 se ilustran los ciclos límites de las piernas izquierda y derecha durante la locomoción en el escenario 3.

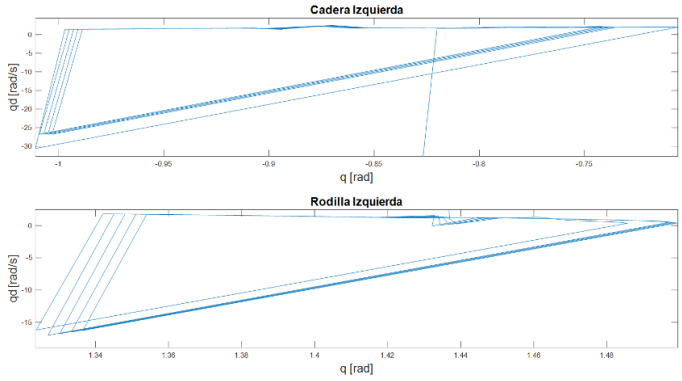


Figura 5.9. Ciclo límite de la cadera y la rodilla de la pierna izquierda en el escenario 3.

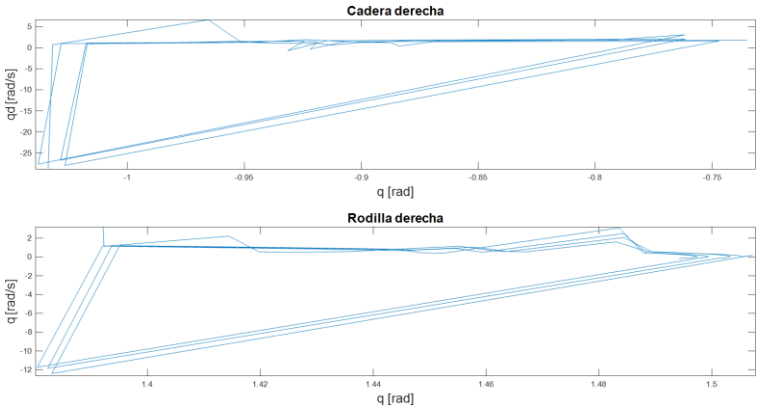


Figura 5.10. Ciclo límite de la cadera y la rodilla de la pierna derecha en el escenario 3.

Por último, las condiciones iniciales y finales en posición y velocidad del centro de masas en el cuarto escenario son: $r_i = [0 \quad 0 \quad 0]^T m$, $r_f = [0 \quad 0 \quad 2]^T m$, $\dot{r}_i = [0 \quad 0 \quad 2.5]^T \, m/s$ y $\dot{r}_f = [0 \quad 0 \quad 2.5]^T \, m/s$. Los resultados se ilustran en la figuras 5.11 y 5.12.

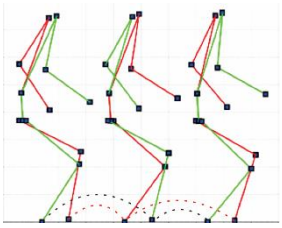


Figura 5.11. Locomoción del escenario 4.

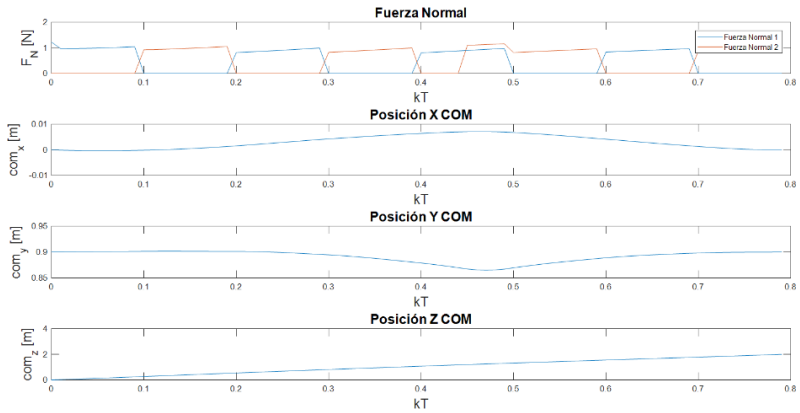


Figura 5.12. Gráficas de la fuerza normal y posición del centro de masas del escenario 4.

En las figuras 5.13 y 5.14 se ilustran los ciclos límites de las piernas izquierda y derecha durante la locomoción en el escenario 4.

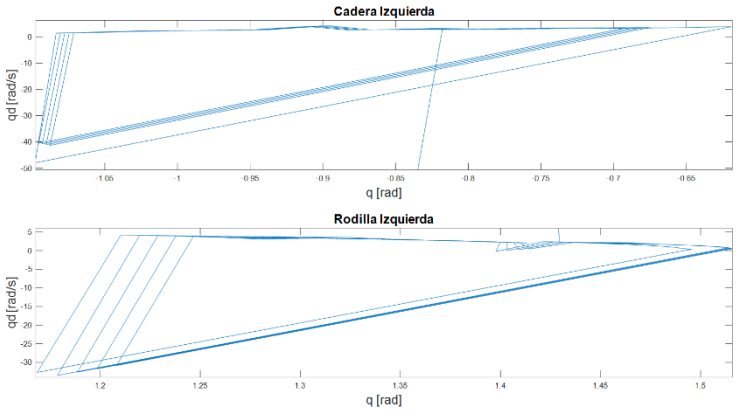


Figura 5.13. Ciclo límite de la cadera y la rodilla de la pierna izquierda en el escenario 4.

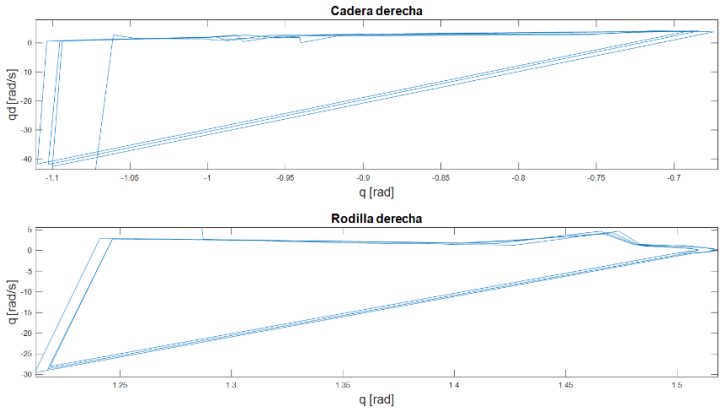


Figura 5.14. Ciclo límite de la cadera y la rodilla de la pierna derecha en el escenario 4.

En los 4 escenarios se observa que la fuerza normal de las dos piernas están desfasadas con una magnitud unitaria lo cual es un resultado esperado ya que durante la locomoción cuando sólo hay una pierna en el suelo ésta debe soportar el peso del robot. De igual manera se observa que el sistema se desplaza con una velocidad constante hasta llegar a su destino.

5.5 Conclusiones

Se ha introducido un algoritmo eficiente y robusto para la planeación de trayectorias de cuerpo completo capaz de lidiar con múltiples tareas y posibles conflictos entre ellas mediante PCJ. Sin embargo, debe hacerse notar que la solución encontrada por el algoritmo propuesto se degrada entre más grande sea la longitud del horizonte de planeación.

Emplear optimización convexa en la planeación cinemática y dinámica ofrece una garantía sobre la convergencia del algoritmo, esto a su vez incrementa la robustez del problema de planeación de cuerpo completo (a diferencia de las formulaciones con programación no lineal previamente propuestos en la literatura los cuales son dependientes del punto inicial de búsqueda para converger).

Durante este proyecto de tesis se han desarrollado e implementado algoritmos de optimización no lineal es para su aplicación en la planeación de caminata; en particular el método del Lagrangiano aumentado, método del punto interior no lineal y programación cuadrática secuencial. En los experimentos numéricos se observa una gran sensibilidad a los diversos parámetros del problema y del optimizador, en donde la programación cuadrática secuencial muestra la mayor eficiencia y robustez.

Los escenarios presentados en este capítulo muestran el potencial del algoritmo propuesto para sintetizar la locomoción con diferentes velocidades de desplazamiento. El algoritmo propuesto es aplicable a cualquier tipo de morfología del robot con patas, sólo es necesario especificar los tiempos de contacto de las patas y su modelo cinemático.

CAPÍTULO 6

CONCLUSIONES

En este capítulo se describen las conclusiones generales como resultado del presente trabajo de tesis, se mencionan las principales contribuciones y los problemas para trabajo futuro.

6.1 Conclusiones generales

Dado que la síntesis de trayectorias mediante la formulación del problema de planeación de caminata como una optimización no lineal presenta una gran dificultad en la práctica, en el presente trabajo de tesis se propone un nuevo algoritmo jerárquico capaz de lidiar con múltiples tareas en la planeación de caminata incrementando de esta manera la robustez del algoritmo de manera significativa en comparación con la formulación no lineal.

De igual manera se han implementado diversos algoritmos de optimización convexa para abordar el problema de balance de postura; en particular el método de multiplicadores, el método del conjunto activo y el método del punto interior. En donde se destaca que el método del punto interior muestra la mayor eficiencia y robustez. Además, se ha mostrado el potencial de los algoritmos propuestos para el balance en superficies parejas y disparejas.

Finalmente se han propuesto e implementado algoritmos recursivos para el cómputo de la dinámica y cinemática de robots de base flotante y se han empleado en el balance de postura con control de cuerpo completo y la planeación de caminata.

6.2 Principales contribuciones

- Se ha propuesto un algoritmo del punto interior con regulación dual capaz de lidiar con múltiples tareas y configuraciones singulares de manera eficiente y robusta para el balance de postura con control de cuerpo completo.

- Se ha introducido un algoritmo $O(N)$ para el cómputo del momento centroidal y su derivada temporal de manera conjunta empleando álgebra espacial y de Lie. El algoritmo resultante se utiliza en el balance de postura con control de cuerpo completo mediante programación cuadrática jerárquica.

- Finalmente, se realiza una reformulación del problema de planeación de trayectorias de cuerpo completo con el modelo de la dinámica centroidal y la cinemática de cuerpo completo, que consiste de una descomposición en 2 partes que se alternan hasta la convergencia. La primera parte es la planeación con la dinámica centroidal que se linealiza con heurísticas y la segunda parte es la planeación cinemática que se resuelve con programación cuadrática jerárquica de forma tal que es posible resolver conflictos entre las múltiples tareas deseadas.

6.3 Problemas abiertos

En base a las contribuciones realizadas en el presente trabajo de tesis a continuación se describen problemas abiertos y posibles trabajos futuros.

- Introducir límites en los pares articulares durante la planeación de trayectorias de cuerpo completo con el modelo de la dinámica centroidal y la cinemática de cuerpo completo.

- Utilizar soluciones previas para acelerar la convergencia del método del punto interior suavizando las restricciones de no negatividad del algoritmo en combinación con la regularización dual para su aplicación en el balance de postura con control de cuerpo completo y la planeación de trayectorias.

- Utilizar optimización mixta entera cuadrática en la descomposición de planeación de trayectorias de cuerpo completo empleada en el presente trabajo de tesis para la solución de la planeación dinámica; esto permitiría reducir el número de variables binarias de la formulación original de planeación con programación mixta entera. Ya que sólo es necesario aproximar el momento angular centroidal mientras la orientación de la base se manejaría con la planeación cinemática de cuerpo completo empleando programación cuadrática jerárquica.

- La implementación de los algoritmos propuestos en el presente trabajo de tesis para el balance de postura y la planeación de caminata en una plataforma experimental actuada por par.

LITERATURA CITADA

Abe, Y., Da Silva, M., & Popović, J. (2007). Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, pp. 249-258.

Aceituno-Cabezas, B., Cappelletto, J., Grieco, J. C., & Fernandez-Lopez, G. (2016). A Generalized Mixed-Integer Convex Program for Multilegged Footstep Planning on Uneven Terrain. *arXiv preprint arXiv:1612.02109*.

Aceituno-Cabezas, B., Dai, H., Cappelletto, J., Grieco, J. C., & Fernández-López, G. (2017a). A Mixed-Integer Convex Optimization Framework for Robust Multilegged Robot Locomotion Planning over Challenging Terrain. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4467-4472.

Aceituno-Cabezas, B., Mastalli, C., Dai, H., Focchi, M., Radulescu, A., Caldwell, D. G., & Semini, C. (2017b). Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, *3*(3), pp. 2531-2538.

Andrei, N. (2017). *Continuous Nonlinear Optimization for Engineering Applications in GAMS Technology*. Springer International Publishing.

Antoniou, A., & Lu, W. S. (2007). *Practical optimization: algorithms and engineering applications*. Springer Science & Business Media

Apgar, T., Clary, P., Green, K., Fern, A., & Hurst, J. W. (2018). Fast Online Trajectory Optimization for the Bipedal Robot Cassie. In *Robotics: Science and Systems*.

Barthélemy, S., & Bidaud, P. (2008). Stability measure of postural dynamic equilibrium based on residual radius. In *Advances in Robot Kinematics: Analysis and Design*, Springer, pp. 399-407.

Baumgarte, J. (1972). Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, *1*(1), pp. 1-16.

LITERATURA CITADA (Continuación)

Bledt, G., Powell, M. J., Katz, B., Di Carlo, J., Wensing, P. M., & Kim, S. (2018). MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2245-2252.

Bledt, G., Wensing, P. M., & Kim, S. (2017). Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4102-4109.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, *3*(1), pp. 1-122.

Buchli, J., Kalakrishnan, M., Mistry, M., Pastor, P., & Schaal, S. (2009). Compliant quadruped locomotion over rough terrain. In *2009 IEEE/RSJ international conference on Intelligent robots and systems*, pp. 814-820.

Carius, J., Ranftl, R., Koltun, V., & Hutter, M. (2018). Trajectory Optimization with Implicit Hard Contacts. *IEEE Robotics and Automation Letters*, *3*(4), pp. 3316-3323.

Carius, J., Ranftl, R., Koltun, V., & Hutter, M. (2019). Trajectory Optimization for Legged Robots with Slipping Motions. *IEEE Robotics and Automation Letters*, *4*(3), pp. 3013-3020.

Chatterjee, A. (1999). On the realism of complementarity conditions in rigid body collisions. *Nonlinear Dynamics*, *20*(2), pp. 159-168.

Coros, S., Beaudoin, P., & Van de Panne, M. (2010). Generalized Biped Walking Control. *ACM Transactions on Graphics (TOG)*, *29*(4), pp. 1-9.

Coros, S., Karpathy, A., Jones, B., Reveret, L., & Van De Panne, M. (2011). Locomotion Skills for Simulated Quadrupeds. *ACM Transactions on Graphics (TOG)*, *30*(4), pp. 1-12.

Dai, H. (2016). *Robust multi-contact dynamical motion planning using contact wrench set* (Tesis de Doctorado, Massachusetts Institute of Technology).

LITERATURA CITADA (Continuación)

Dai, H., & Tedrake, R. (2016). Planning robust walking motion on uneven terrain via convex optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids*, pp. 579-586.

Dai, H., Izatt, G., & Tedrake, R. (2019). Global inverse kinematics via mixed-integer convex optimization. *The International Journal of Robotics Research*, *38*(12-13), pp. 1420-1441.

Dai, H., Valenzuela, A., & Tedrake, R. (2014). Whole-body Motion Planning with Centroidal Dynamics and Full Kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295-302.

De Lasa, M., Mordatch, I., & Hertzmann, A. (2010). Feature-based locomotion controllers. ACM Transactions on Graphics (TOG), 29(4), p. 131.

Deits, R., & Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 279-286.

Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., & Kim, S. (2018). Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1-9.

Drumwright, E., Hsu, J., Koenig, N., & Shell, D. (2010). Extending Open Dynamics Engine for Robotics Simulation. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Springer, pp. 38-50.

Duff, I. S. (2004). MA57---a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, *30*(2), pp. 118-144.

Escande, A., Mansard, N., & Wieber, P. B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, *33*(7), pp. 1006-1028.

Fang, A. C., & Pollard, N. S. (2003). Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)*, *22*(3), pp. 417-426.

LITERATURA CITADA (Continuación)

Faraji, S., Pouya, S., Atkeson, C. G., & Ijspeert, A. J. (2014). Versatile and robust 3d walking with a simulated humanoid robot (atlas): A model predictive control approach. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1943-1950.

Featherstone, R. (1984). Robot dynamics algorithms. Kluwer Academic Publishers.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Springer.

Feng, S. (2016). *Online Hierarchical Optimization for Humanoid Control* (Tesis de Doctorado, Carnegie Mellon University).

Feng, S., Whitman, E., Xinjilefu, X., & Atkeson, C. G. (2014). Optimization based full body control for the atlas robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 120-127.

Focchi, M., Del Prete, A., Havoutis, I., Featherstone, R., Caldwell, D. G., & Semini, C. (2017). High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, *41*(1), pp. 259-272.

Gehring, C., Bellicoso, C. D., Coros, S., Bloesch, M., Fankhauser, P., Hutter, M., & Siegwart, R. (2015). Dynamic trotting on slopes for quadrupedal robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5129-5135.

Gehring, C., Diethelm, R., Siegwart, R., Nützi, G., & Leine, R. I. (2014). An Evaluation of Moreau's time-stepping scheme for the simulation of a legged robot. In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection.

Geijtenbeek, T., Van De Panne, M., & Van Der Stappen, A. F. (2013). Flexible Muscle-Based Locomotion for Bipedal Creatures. *ACM Transactions on Graphics (TOG)*, *32*(6), pp. 1-11.

Gertz, E. M., & Wright, S. (2002). *OOQP user guide* (No. ANL/MCS-TM-252). Argonne National Lab., IL (US).

LITERATURA CITADA (Continuación)

Geyer, H., & Herr, H. (2010). A Muscle-Reflex Model That Encodes Principles of Legged Mechanics Produces Human Walking Dynamics and Muscle Activities. *IEEE Transactions on neural systems and rehabilitation engineering*, *18*(3), pp. 263-273.

Gill, P. E., Murray, W., & Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, *47*(1), pp. 99-131.

Hendel, G. (2018). Adaptive large neighborhood search for mixed integer programming. ZIB Report 18-60, Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Dec. 2018, https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/7116.

Hereid, A., Cousineau, E. A., Hubicki, C. M., & Ames, A. D. (2016). 3D Dynamic Walking with Underactuated Humanoid Robots: A Direct Collocation Framework for Optimizing Hybrid Zero Dynamics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1447-1454.

Herr, H. M., & McMahon, T. A. (2000). A trotting horse model. *The International Journal of Robotics Research*, *19*(6), pp. 566-581.

Herr, H. M., & McMahon, T. A. (2001). A galloping horse model. *The International Journal of Robotics Research*, *20*(1), pp. 26-37.

Herzog, A., Righetti, L., Grimminger, F., Pastor, P., & Schaal, S. (2014). Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 981-988.

Herzog, A., Schaal, S., & Righetti, L. (2016). Structured contact force optimization for kino-dynamic motion generation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2703-2710.

Hirukawa, H., Hattori, S., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., & Morisawa, M. (2006). A Universal Stability Criterion of the Foot Contact of Legged Robots-Adios ZMP. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA* 2006, pp. 1976-1983.

LITERATURA CITADA (Continuación)

Hoffman, E. M., Rocchi, A., Tsagarakis, N. G., &, D. G. (2018). Robot Dynamics Constraint for Inverse Kinematics. In *Advances in Robot Kinematics 2016*, Springer, pp. 275-283.

Hu, J. J., Pratt, J. E., Chew, C. M., Herr, H. M., & Pratt, G. A. (1999). Virtual model based adaptive dynamic control of a biped walking robot. *International Journal on Artificial Intelligence Tools*, *8*(03), pp. 337-348.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation,* 2(1), pp. 1620-1626.

Kajita S., & Espiau B. (2008). Legged Robots. In Siciliano B., Khatib O. Springer Handbook of Robotics, pp. 361-389.

Katz, B. G. (2018). *A low cost modular actuator for dynamic robots* (Tesis de Maestría, Massachusetts Institute of Technology).

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, *3*(1), pp. 43-53.

Kim, D., Di Carlo, J., Katz, B., Bledt, G., & Kim, S. (2019). Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control. *arXiv preprint arXiv:1909.06586*.

Kim, J., Baek, J., & Park, F. C. (1995). Newton-type algorithms for robot motion optimization. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients*, 3(1), pp. 1842-1847.

Kokkevis, E., & Metaxas, D. (1998). Efficient Dynamic Constraints for Animating Articulated Figures. *Multibody system dynamics*, *2*(2), pp. 89-114.

LITERATURA CITADA (Continuación)

Kolter, J. Z., Rodgers, M. P., & Ng, A. Y. (2008). A control architecture for quadruped locomotion over rough terrain. In *2008 IEEE International Conference on Robotics and Automation*, pp. 811-818.

Koolen, F. A. (2019). *Balance control and locomotion planning for humanoid robots using nonlinear centroidal models* (Tesis de Doctorado, Massachusetts Institute of Technology).

Koolen, T., Bertrand, S., Thomas, G., De Boer, T., Wu, T., Smith, J., & Pratt, J. (2016). Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, *13*(01), p. 1650007.

Krasny, D. P., & Orin, D. E. (2010). Evolution of a 3D gallop in a Quadrupedal Model with Biological Characteristics. *Journal of Intelligent & Robotic Systems*, *60*(1), pp. 59-82.

Kuindersma, S., Permenter, F., & Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2589-2594.

Lee, S. H., & Goswami, A. (2010). Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3157-3162.

Lee, S. H., Kim, J., Park, F. C., Kim, M., & Bobrow, J. E. (2005). Newton-type algorithms for dynamics-based robot movement optimization. *IEEE Transactions on robotics*, *21*(4), pp. 657-667.

Lilly, K. (1993). *Efficient dynamic simulation of robotic mechanisms*. Springer Science & Business Media.

Liu, M., Qu, D., Xu, F., Zou, F., Di, P., & Tang, C. (2019). Quadrupedal Robots Whole-Body Motion Control Based on Centroidal Momentum Dynamics. *Applied Sciences*, *9*(7), p. 1335.

LITERATURA CITADA (Continuación)

Lo, J., Huang, G., & Metaxas, D. (2002). Human motion planning based on recursive dynamics and optimal control techniques. *Multibody System Dynamics*, *8*(4), pp. 433-458.

Macchietto, A., Zordan, V., & Shelton, C. R. (2009). Momentum control for balance. In *ACM Transactions on graphics (TOG)*, 28(3), p. 80.

Maes, C. M. (2011). *A regularized active-set method for sparse convex quadratic programming* (Tesis de Doctorado, Stanford University).

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., & Semini, C. (2016). Hierarchical Planning of Dynamic Movements without Scheduled Contact Sequences. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4636-4641.

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D., & Semini, C. (2018). Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control.

Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, *2*(4), pp. 575-601.

Mirtich, B. V. (1996). *Impulse-based dynamic simulation of rigid body systems* (Tesis de Doctorado, University of California Berkeley).

Mistry, M., Buchli, J., & Schaal, S. (2010). Inverse dynamics control of floating base systems using orthogonal decomposition. In *2010 IEEE international conference on robotics and automation*, pp. 3406-3412.

Mistry, M., Nakanishi, J., & Schaal, S. (2007). Task space control with prioritization for balance and locomotion. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 331-338.

Mistry, M., Nakanishi, J., Cheng, G., & Schaal, S. (2008). Inverse kinematics with floating base and constraints for full body humanoid robot control. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pp. 22-27.

Mordatch, I., De Lasa, M., & Hertzmann, A. (2010). Robust physics-based locomotion using low-dimensional planning. In *ACM Transactions on Graphics (TOG)*, 29(4), p. 71.

LITERATURA CITADA (Continuación)

Mordatch, I., Todorov, E., & Popović, Z. (2012). Discovery of Complex Behaviors through Contact-Invariant Optimization. *ACM Transactions on Graphics (TOG)*, *31*(4), pp. 1-8.

Murphy, K. N., & Raibert, M. H. (1985). Trotting and bounding in a planar two-legged model. In *Theory and Practice of Robots and Manipulators*, Springer, pp. 411-420.

Neunert, M., Farshidian, F., & Buchli, J. (2016). Efficient whole-body trajectory optimization using contact constraint relaxation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 43-48.

Neunert, M., Stäuble, M., Giftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., & Buchli, J. (2018). Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters*, *3*(3), pp. 1458-1465.

Nguyen, Q., Powell, M. J., Katz, B., Di Carlo, J., & Kim, S. (2019). Optimized Jumping on the MIT Cheetah 3 Robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7448-7454.

Nguyen, Q., Powell, M. J., Katz, B., Di Carlo, J., & Kim, S. (2019). Optimized jumping on the MIT Cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7448-7454.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Orin, D. E., Goswami, A., & Lee, S. H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous Robots*, *35*(2-3), pp. 161-176.

Pardo, D., Neunert, M., Winkler, A. W., Grandia, R., & Buchli, J. (2017). Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion. In *Robotics: Science and Systems*.

Park, F. C., Bobrow, J. E., & Ploen, S. R. (1995). A Lie group formulation of robot dynamics. *The International journal of robotics research*, *14*(6), pp. 609-618.

LITERATURA CITADA (Continuación)

Ponton, B., Herzog, A., Schaal, S., & Righetti, L. (2016). A convex model of humanoid momentum dynamics for multi-contact motion generation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 842-849.

Popovic, M., Hofmann, A., & Herr, H. (2004). Angular momentum regulation during human walking: biomechanics and control. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. (ICRA)*, 3(1), pp. 2405-2411.

Posa, M., & Tedrake, R. (2013). Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, Springer, pp. 527-542.

Posa, M., Kuindersma, S., & Tedrake, R. (2016). Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1366-1373.

Pratt, J. E. (1995). *Virtual model control of a biped walking robot*. (Tesis de Maestría, Massachusetts Institute of Technology)

Pratt, J., Chew, C. M., Torres, A., Dilworth, P., & Pratt, G. (2001). Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, *20*(2), pp. 129-143.

Pratt, J. E., & Tedrake, R. (2006). Velocity-Based Stability Margins for Fast Bipedal Walking. In *Fast Motions in Biomechanics and Robotics*, Springer, pp. 299-324.

Raibert, M. H. (1986). *Legged robots that balance*. MIT press.

Raibert, M., Chepponis, M., & Brown, H. B. J. R. (1986). Running on four legs as though they were one. *IEEE Journal on Robotics and Automation*, *2*(2), pp. 70-82.

Raibert, M. H., & Hodgins, J. K. (1991). Animation of dynamic legged locomotion. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pp. 349-358.

Rebula, J. R., Neuhaus, P. D., Bonnlander, B. V., Johnson, M. J., & Pratt, J. E. (2007). A Controller for the LittleDog Quadruped Walking on Rough Terrain. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1467-1473.

LITERATURA CITADA (Continuación)

Redon, S., Kheddar, A., & Coquillart, S. (2002). Gauss' least constraints principle and rigid body simulations. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, 1(1), pp. 517-522.

Remy, C. D. (2011). *Optimal Exploitation of Natural Dynamics in Legged Locomotion* (Tesis de Doctorado, ETH Zurich).

Rocchi, A., Hoffman, E. M., Farnioli, E., & Tsagarakis, N. G. (2015). A Whole-Body Stack-of-Tasks compliant control for the Humanoid Robot COMAN. In *IEEE/RSJ international conference on intelligent robots and systems (IROS 2015)*, 28(1).

Rojas, J., Aguilar, A., & Bugarin E. (2019). Sobre la regularización dual en el balance de postura de un robot humanoide empleando control de cuerpo completo. Tecnificación Mecatrónica, Capítulo 5, pp. 60-74.

Selig, J. M. (2005). *Geometric Fundamentals of Robotics*. Springer Science & Business Media.

Sentis, L., & Khatib, O. (2005). Control of free-floating humanoid robots through task prioritization. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 1718-1723.

Slotine, S. B. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *proceeding of 5th International Conference on Advanced Robotics*, 2(1), pp. 1211-1216.

Smith, R. (2005). Open dynamics engine v0.5 user guide.

Stellato, B. (2017). *Mixed-integer optimal control of fast dynamical systems* (Tesis de Doctorado, University of Oxford).

Stephens, B. J., & Atkeson, C. G. (2010). Dynamic balance force control for compliant humanoid robots. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pp. 1248-1255.

LITERATURA CITADA (Continuación)

Stępień, J. (2012, November). A unified constraint framework for physical animation of articulated rigid bodies. In *International Conference on Motion in Games*, Springer, pp. 90-101.

Stewart, D. E., & Trinkle, J. C. (1996). An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, *39*(15), pp. 2673-2691.

Studer, C. (2009). Numerics of Unilateral Contacts and Friction. Springer.

Tedrake, R. L. (2004). *Applied optimal control for dynamically stable legged locomotion* (Tesis de Doctorado, Massachusetts Institute of Technology).

Thatte, N., & Geyer, H. (2015). Toward balance recovery with leg prostheses using neuromuscular model control. *IEEE Transactions on Biomedical Engineering*, *63*(5), pp. 904-913.

Todorov, E. (2014). Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6054-6061.

Valenzuela, A. K. (2016). *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain* (Tesis de Doctorado, Massachusetts Institute of Technology).

Vanderbei, R. J. (1999). LOQO: An interior point code for quadratic programming. *Optimization methods and software*, *11*(1-4), pp. 451-484.

Vukobratović, M., & Juričić, D. (1968). Contribution to the synthesis of biped gait. *IFAC Proceedings Volumes*, *2*(4), pp. 469-478.

Wächter, A. (2002). *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering* (Tesis de Doctorado, Carnegie Mellon University).

Wadayama, T. (2008). Interior point decoding for linear vector channels. In *Journal of Physics: Conference Series*, IOP Publishing, 95(1), p. 012009.

LITERATURA CITADA (Continuación)

Waltz, R. A., Morales, J. L., Nocedal, J., & Orban, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical programming*, *107*(3), pp. 391-408.

Wampler, K., & Popović, Z. (2009). Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)*, *28*(3), pp. 1-8.

Wang, J. M., Fleet, D. J., & Hertzmann, A. (2009). Optimizing Walking Controllers. In *ACM SIGGRAPH Asia 2009 papers*, pp. 1-8.

Wang, Y., & Boyd, S. (2009). Fast Model Predictive Control Using Online Optimization. *IEEE Transactions on control systems technology*, *18*(2), pp. 267-278.

Wensing, P. M., & Orin, D. E. (2013a). Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *2013 IEEE International Conference on Robotics and Automation*, pp. 3103-3109.

Wensing, P. M., & Orin, D. E. (2013b). High-speed humanoid running through control with a 3D-SLIP model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5134-5140.

Wensing, P. M., & Orin, D. E. (2016). Improved Computation of the Humanoid Centroidal Dynamics and Application for Whole-Body Control. *International Journal of Humanoid Robotics*, *13*(01), p. 1550039.

Wieber, P. B. (2006). Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pp. 137-142.

Winkler, A. W., Bellicoso, C. D., Hutter, M., & Buchli, J. (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, *3*(3), pp. 1560-1567.

Winkler, A. W., Farshidian, F., Pardo, D., Neunert, M., & Buchli, J. (2017). Fast Trajectory Optimization for Legged Robots using Vertex-based ZMP Constraints. *IEEE Robotics and Automation Letters*, *2*(4), pp. 2201-2208.

LITERATURA CITADA (Continuación)

Yin, K., Loken, K., & Van de Panne, M. (2007). Simbicon: Simple Biped Locomotion Control. *ACM Transactions on Graphics (TOG), 26*(3), p. 105.

Zhao, H., Hereid, A., Ambrose, E., & Ames, A. D. (2016). 3D multi-contact gait design for prostheses: Hybrid system models, virtual constraints and two-step direct collocation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 3668-3674.

ANEXOS

A Código de Matlab de algoritmos de dinámica y cinemática de cuerpos rígidos articulados

A.1 Algoritmo del cuerpo compuesto

```
%{
Esta función realiza el cómputo de la matriz de inercia del robot. La función “CompositeRigidBody”
toma como argumentos el modelo del robot (“model”) y las posiciones articulares (“q”).
%}

function [H, F, I0c] = CompositeRigidBody(model, q)


Nb = numel(model);

dof = zeros(Nb, 1);
for i = 2:Nb
    dof(i) = size(q(i).q, 1);
end
systemSize = sum(dof);

ine = struct('I', zeros(6, 6));
H = spalloc(systemSize, systemSize, 0);
F = zeros(6, systemSize);
Ic = repmat(ine, Nb, 1);

for i=1:Nb
    Ic(i).I = model(i).I;
end

for i=Nb:-1:2

    R = rotate(q(i).q);
    r = skew(model(i).offset);
    pXc = childToParent(R, r);
    cXp = parentToChild(R, r);

    Ic(model(i).parent).I = Ic(model(i).parent).I + pXc*Ic(i).I*cXp;
    Fa = Ic(i).I*model(i).S;

    Hii = transpose(model(i).S)*Fa;
    [a, b, c, d] = mapIndices(i, i, Hii);
    H(a:b, c:d) = Hii;

    j = i;
    while(model(j).parent ~= 1)
        R1 = rotate(q(j).q);
        r1 = skew(model(j).offset);
        pXc = childToParent(R1, r1);

        Fa = pXc*Fa;
        j = model(j).parent;

        Hji = transpose(model(j).S)*Fa;
        [a, b, c, d] = mapIndices(j, i, Hji);
        H(a:b, c:d) = Hji;
```

```
        [a, b, c, d] = mapIndices(i, j, transpose(Hji));
        H(a:b, c:d) = transpose(Hji);
    end

    R2 = rotate(q(j).q);
    r2 = skew(model(j).offset);
    pXc = childToParent(R2, r2);

    [~, ~, m, n] = mapIndices([], i, Fa);
    F(:, m:n) = pXc*Fa;
end

I0c = Ic(1).I;

end
```

A.2 Jacobiano aumentado

```
%{
Esta función realiza el cómputo del Jacobiano aumentado de todos los puntos de contacto activos. La función “ComputeContactJacobians” toma como argumentos el modelo del robot (“model”), la orientación de la base (“Rb”), las posiciones y velocidades articulares (“q” y “qd”) y las posiciones de los puntos de contacto respecto a un marco coordenado atado al centro de masas de los pies del robot.
%}

function [J, b] = ComputeContactJacobians(model, Rb, q, qd, geom)

global cR0;
WorldTransform(model, Rb, q);

[jac1, bias1] = BodyJacobian(model, q, qd, 4);
[jac2, bias2] = BodyJacobian(model, q, qd, 7);

J1 = transpose(cR0(4).R)*[-skew(geom(1).v), eye(3)]*jac1;
J2 = transpose(cR0(4).R)*[-skew(geom(2).v), eye(3)]*jac1;
J3 = transpose(cR0(4).R)*[-skew(geom(3).v), eye(3)]*jac1;
J4 = transpose(cR0(4).R)*[-skew(geom(4).v), eye(3)]*jac1;

J5 = transpose(cR0(7).R)*[-skew(geom(1).v), eye(3)]*jac2;
J6 = transpose(cR0(7).R)*[-skew(geom(2).v), eye(3)]*jac2;
J7 = transpose(cR0(7).R)*[-skew(geom(3).v), eye(3)]*jac2;
J8 = transpose(cR0(7).R)*[-skew(geom(4).v), eye(3)]*jac2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b1 = transpose(cR0(4).R)*[-skew(geom(1).v), eye(3)]*bias1;
b2 = transpose(cR0(4).R)*[-skew(geom(2).v), eye(3)]*bias1;
b3 = transpose(cR0(4).R)*[-skew(geom(3).v), eye(3)]*bias1;
b4 = transpose(cR0(4).R)*[-skew(geom(4).v), eye(3)]*bias1;

b5 = transpose(cR0(7).R)*[-skew(geom(1).v), eye(3)]*bias2;
b6 = transpose(cR0(7).R)*[-skew(geom(2).v), eye(3)]*bias2;
b7 = transpose(cR0(7).R)*[-skew(geom(3).v), eye(3)]*bias2;
b8 = transpose(cR0(7).R)*[-skew(geom(4).v), eye(3)]*bias2;

J = [J1; J2; J3; J4; J5; J6; J7; J8];
b = [b1; b2; b3; b4; b5; b6; b7; b8];
end
```

A.3 Jacobiano de eslabón

```
%{
Esta función realiza el cómputo del Jacobiano del centro de masas de un eslabón con índice “index”. La función “BodyJacobian” toma como argumentos el modelo del robot (“model”), las posiciones y velocidades articulares (“q” y “qd”) y el índice del eslabón deseado (“index”).
%}

function [J, b] = BodyJacobian(model, q, qd, index)

global v;
global a;
global dof;
sys_size = sum(dof);
J = spalloc(6, sys_size, 0);

j = index;
[m, n] = mapJacobianIdx(j, model(j).S);
J(:, m:n) = model(j).S;
X = eye(6);

while(model(j).parent ~= -1)
    R = rotate(q(j).q);
    r = skew(model(j).offset);
    cXp = parentToChild(R, r);
    X = X*cXp;

    j = model(j).parent;
    [m, n] = mapJacobianIdx(j, model(j).S);
    J(:, m:n) = X*model(j).S;
end

Nb = numel(model);
v(:, 1) = qd(1).q;
a(:, 1) = zeros(6, 1);
for i=2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    v(:, i) = cXp*v(:, model(i).parent) + model(i).S*qd(i).q;
    a(:, i) = cXp*a(:, model(i).parent) + scross(v(:, i), model(i).S*qd(i).q);
end
b = a(:, index);

end %end function
```

A.4 Algoritmo $O(N)$ propuesto

```
%{
Esta función implementa el algoritmo propuesto del presente trabajo de tesis para el cómputo del momento centroidal, la matriz del momento centroidal y su derivada temporal multiplicada por las velocidades articulares. La función “CentroidalMomentum2” toma como argumentos el modelo del robot (“model”), la posición de la base (“pb”), la orientación de la base (“Rb”), las posiciones y velocidades articulares (“q” y “qd”).
%}

function [AG, bG, hG] = CentroidalMomentum2(model, pb, Rb, q, qd)
Nb = numel(model);
global dof;
numDof = sum(dof(2:Nb));
```

```
spa = struct('X', zeros(6, 6));
rota = struct('R', eye(3));
gXc = repmat(spa, Nb, 1);
Rg = repmat(rota, Nb, 1);
pg = zeros(3, Nb);
rg = zeros(3, Nb);
massT = 0;
for j=1:Nb
    massT = massT + model(j).m;
end

Rg(1).R = Rb;
pg(:, 1) = pb;
rg(:, 1) = model(1).m*pb;
%center of mass 1
for j=2:Nb
    R = rotate(q(j).q);
    Rg(j).R = Rg(model(j).parent).R*R;
    pg(:, j) = pg(:, model(j).parent) + Rg(model(j).parent).R*model(j).offset;
    rg(:, j) = model(j).m*(pg(:, j) + Rg(j).R*model(j).c);
end

com_pos = zeros(3, 1);
for j=1:Nb
    com_pos = com_pos + rg(:, j);
end
com_pos = (1/massT)*com_pos;

r = com_pos - pg(:, 1);
gXc(1).X = [Rb, -skew(r)*Rb; zeros(3, 3), Rb];
for j=2:Nb
    r = com_pos - pg(:, j);
    R = Rg(j).R;
    gXc(j).X = [R, skew(r)*R; zeros(3, 3), R];
end

ine = struct('I', eye(6));
ined = struct('I', zeros(6, 6));
AG = spalloc(6, numDof + 6, 6*(6+numDof));
Ic = repmat(ine, Nb, 1);
Id = repmat(ined, Nb, 1); %inertia derivative
for i=1:Nb
    Ic(i).I = model(i).I;
    Id(i).I = zeros(6, 6);
end

for i=Nb:-1:2
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    pXc = childToParent(R, r);
    cXp = parentToChild(R, r);

    Ic(model(i).parent).I = Ic(model(i).parent).I + pXc*Ic(i).I*cXp;
    Ia = op_dcross(model(i).S*qd(i).q)*Ic(i).I - Ic(i).I*op_scross(model(i).S*qd(i).q);
    Id(model(i).parent).I = Id(model(i).parent).I + pXc*Ia*cXp;
end

bG = zeros(6, 1);
hG = zeros(6, 1);
a = 0;
for i=1:Nb
    b = a + size(model(i).S, 2);
    AG(:, a+1:b) = gXc(i).X*Ic(i).I*model(i).S;
    hG = hG + AG(:, a+1:b)*qd(i).q;
```

```
    a = b;

    Ad = gXc(i).X*(Id(i).I*model(i).S + op_dcross(model(i).S*qd(i).q)*Ic(i).I*model(i).S);
    bG = bG + Ad*qd(i).q;
end

end
```

A.5 Algoritmo Recursivo de Newton Euler

```
%{
Esta función implementa el algoritmo Recursivo de Newton Euler para el cómputo de los términos de
Coriolis, gravedad y fuerzas centrífugas, la matriz del momento centroidal y su derivada temporal
multiplicada por las velocidades articulares. La función "CentroidalMomentum2" toma como argumentos
el modelo del robot ("model"), la posición de la base ("pb"), la orientación de la base ("Rb"), las
posiciones y velocidades articulares ("q" y "qd").
%}

function [C, p0c] = NewtonEuler(model, Rb, q, qd)

Nb = numel(model);
dof = zeros(Nb, 1);
for i = 2:Nb
    dof(i) = size(q(i).q, 1);
end
systemSize = sum(dof);

global v;
global a;
f = zeros(6, Nb);
C = zeros(systemSize, 1);

g = [0; 0; 0; 0; 9.81; 0];
v(:, 1) = qd(1).q;
a(:, 1) = diag_spa(transpose(Rb))*g;

for i=2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    v(:, i) = cXp*v(:, model(i).parent) + model(i).S*qd(i).q;
    a(:, i) = cXp*a(:, model(i).parent) + scross(v(:, i), model(i).S*qd(i).q);
    f(:, i) = model(i).I*a(:, i) + dual_scross(v(:, i), model(i).I*v(:, i));
end

f(:, 1) = model(1).I*a(:, 1) + dual_scross(v(:, 1), model(1).I*v(:, 1));
for i=Nb:-1:2
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    pXc = childToParent(R, r);

    Ca = transpose(model(i).S)*f(:, i);
    [i1, i2, ~, ~] = mapIndices(i, [], Ca);
    C(i1:i2) = Ca;

    f(:, model(i).parent) = f(:, model(i).parent) + pXc*f(:, i);
end

p0c = f(:, 1);

end
```

A.6 Pirámide de fricción

```
%{
Esta función calcula las bases vectoriales de la pirámide de fricción. La función
“computeFrictionBasis” toma como argumento el número de puntos de contacto activos (“numCon”).
%}

function basis = computeFrictionBasis(numCon)

theta = 20;
mu = 0.8;
Rf1 = rotateX((theta/180)*pi);
Rf2 = rotateX(-(theta/180)*pi);

basis = spalloc(3*numCon, 4*numCon, 0);
for i=1:numCon
    r = 3*(i - 1);
    c = 4*(i - 1);

    if( i > 4)
        n = Rf1*[0; 1; 0];
        d1 = Rf1*[1; 0; 0];
        d2 = Rf1*[-1; 0; 0];
        d3 = Rf1*[0; 0; 1];
        d4 = Rf1*[0; 0; -1];
        w1 = n + mu*d1;
        w2 = n + mu*d2;
        w3 = n + mu*d3;
        w4 = n + mu*d4;
        W = [w1, w2, w3, w4];
        basis(r+1:r+3, c+1:c+4) = W;
    else
        n = Rf2*[0; 1; 0];
        d1 = Rf2*[1; 0; 0];
        d2 = Rf2*[-1; 0; 0];
        d3 = Rf2*[0; 0; 1];
        d4 = Rf2*[0; 0; -1];
        w1 = n + mu*d1;
        w2 = n + mu*d2;
        w3 = n + mu*d3;
        w4 = n + mu*d4;
        W = [w1, w2, w3, w4];
        basis(r+1:r+3, c+1:c+4) = W;
    end
end

end
```

A.7 Cómputo de la posición del centro de masas

```
%{
Esta función calcula la posición del centro de masas del robot. La función “ComputeCOM” toma como
argumentos el modelo del robot (“model”), la posición de la base (“pb”), la orientación de la base
(“Rb”), las posiciones y velocidades articulares (“q” y “qd”).
%}

function [com_pos, com_vel] = ComputeCOM(model, pb, Rb, q, qd)

Nb = numel(model);
rota = struct('R', eye(3));
Rg = repmat(rota, Nb, 1);
pg = zeros(3, Nb);
```

```
rg = zeros(3, Nb);
vcm = zeros(3, Nb);
global v;

%forward kinematics
massT = 0;
for j=1:Nb
    massT = massT + model(j).m;
end

Rg(1).R = Rb;
pg(:, 1) = pb;
rg(:, 1) = model(1).m*pb;
%center of mass 1
for j=2:Nb
    R = rotate(q(j).q);
    Rg(j).R = Rg(model(j).parent).R*R;
    pg(:, j) = pg(:, model(j).parent) + Rg(model(j).parent).R*model(j).offset;
    rg(:, j) = model(j).m*(pg(:, j) + Rg(j).R*model(j).c);
end

com_pos = zeros(3, 1);
for j=1:Nb
    com_pos = com_pos + rg(:, j);
end
com_pos = (1/massT)*com_pos;

vb = q(1).q;
vcm(:, 1) = model(1).m*vb(4:6);
for i = 2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    v(:, i) = cXp*v(:, model(i).parent) + model(i).S*qd(i).q;

    vel = v(:, i);
    Rf = Rg(i).R;
    vcm(:, i) = model(i).m*Rf*( vel(4:6) + cross(vel(1:3), model(i).c) );
end

com_vel = zeros(3, 1);
for i=1:Nb
    com_vel = com_vel + vcm(:, i);
end

com_vel = (1/massT)*com_vel;

end %end function
```

A.8 Aceleración por defecto de los puntos de contacto

```
%{
Esta función calcula las aceleraciones de los puntos de contacto considerando el controlador de cuerpo
completo y las fuerzas externas. La función “DefaultAcc” toma como argumentos el modelo del robot
(“model”), la orientación de la base (“Rb”), las posiciones y velocidades articulares (“q” y “qd”),
los pares articulares (“tau”), los puntos de contacto (“contact_data”) y las fuerzas externas
(“e_force”).
%}

function acc = DefaultAcc(model, Rb, q, qd, tau, contact_data, e_force)
Nb = numel(model);
foot = contact_data(:, 1);
```

```
nx = contact_data(:, 3);
ny = contact_data(:, 4);
nz = contact_data(:, 5);

trans = struct('X', zeros(6, 6));
rota = struct('R', eye(3, 3));
ine = struct('I', zeros(6, 6));
bias = struct('p', zeros(6, 1));
qdd = q;

global v;
global a;
global c;

global IA;
global cX0;
global cR0;
IA = repmat(ine, Nb, 1);
cX0 = repmat(trans, Nb, 1);
cR0 = repmat(rota, Nb, 1);
pA = repmat(bias, Nb, 1);

gb = transpose(Rb)*[0; -9.81; 0];
g = [0; 0; 0; gb];

v(:, 1)= qd(1).q;
cR0(1).R = transpose(Rb);
cX0(1).X = [transpose(Rb), zeros(3);
            zeros(3), transpose(Rb)];
for i = 2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);
    cX0(i).X = cXp*cX0(model(i).parent).X;
    cR0(i).R = transpose(R)*cR0(model(i).parent).R;

    v(:, i) = cXp*v(:, model(i).parent) + model(i).S*qd(i).q;
    c(:, i) = scross(v(:, i), model(i).S*qd(i).q);
    IA(i).I = model(i).I;

    if(i == 8)
        f = cR0(i).R*e_force;
        pA(i).p = dual_scross(v(:, i), model(i).I*v(:, i)) - [zeros(3, 1); f];
    else
        pA(i).p = dual_scross(v(:, i), model(i).I*v(:, i));
    end
end

IA(1).I = model(1).I;
pA(1).p = dual_scross(v(:, 1), model(1).I*v(:, 1));
for i=Nb:-1:2
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    pXc = childToParent(R, r);
    cXp = parentToChild(R, r);

    U = IA(i).I*model(i).S;
    D = transpose(model(i).S)*U;
    u = tau(i).q - transpose(model(i).S)*pA(i).p - 1e-9*qd(i).q;
    Ia = IA(i).I - U*(D\transpose(U));
    pa = pA(i).p + Ia*c(:, i) + U*(D\u);
    IA(model(i).parent).I = IA(model(i).parent).I + pXc*Ia*cXp;
    pA(model(i).parent).p = pA(model(i).parent).p + pXc*pa;
end
```

```
numContacts = numel(foot);
acc = zeros(3*numContacts, 1);
a(:, 1) = -IA(1).I\pA(1).p + g;
for i=2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    U = IA(i).I*model(i).S;
    D = transpose(model(i).S)*U;
    u = tau(i).q - transpose(model(i).S)*pA(i).p;

    ap = cXp*a(:, model(i).parent) + c(:, i);
    qdd(i).q = D\(u - transpose(U)*ap);
    a(:, i) = ap + model(i).S*qdd(i).q;

    if(i == 4)
        %foot 1
        s1_f1 = foot(1).v;
        s2_f1 = foot(2).v;
        s3_f1 = foot(3).v;
        s4_f1 = foot(4).v;

        a1_f1 = parentToChild(eye(3), skew(s1_f1))*a(:, i);
        a2_f1 = parentToChild(eye(3), skew(s2_f1))*a(:, i);
        a3_f1 = parentToChild(eye(3), skew(s3_f1))*a(:, i);
        a4_f1 = parentToChild(eye(3), skew(s4_f1))*a(:, i);

        Rot = cX0(i).X(1:3, 1:3);
        a1 = transpose(Rot)*a1_f1(4:6);
        a2 = transpose(Rot)*a2_f1(4:6);
        a3 = transpose(Rot)*a3_f1(4:6);
        a4 = transpose(Rot)*a4_f1(4:6);
        acc(1:3) = [dot(nx(1).v, a1); dot(ny(1).v, a1); dot(nz(1).v, a1)];
        acc(4:6) = [dot(nx(2).v, a2); dot(ny(2).v, a2); dot(nz(2).v, a2)];
        acc(7:9) = [dot(nx(3).v, a3); dot(ny(3).v, a3); dot(nz(3).v, a3)];
        acc(10:12) = [dot(nx(4).v, a4); dot(ny(4).v, a4); dot(nz(4).v, a4)];
    end %end if
    if(i == 7)
        %foot 2
        s1_f2 = foot(5).v;
        s2_f2 = foot(6).v;
        s3_f2 = foot(7).v;
        s4_f2 = foot(8).v;

        a1_f2 = parentToChild(eye(3), skew(s1_f2))*a(:, i);
        a2_f2 = parentToChild(eye(3), skew(s2_f2))*a(:, i);
        a3_f2 = parentToChild(eye(3), skew(s3_f2))*a(:, i);
        a4_f2 = parentToChild(eye(3), skew(s4_f2))*a(:, i);

        Rot = cX0(i).X(1:3, 1:3);
        a1 = transpose(Rot)*a1_f2(4:6);
        a2 = transpose(Rot)*a2_f2(4:6);
        a3 = transpose(Rot)*a3_f2(4:6);
        a4 = transpose(Rot)*a4_f2(4:6);
        acc(13:15) = [dot(nx(5).v, a1); dot(ny(5).v, a1); dot(nz(5).v, a1)];
        acc(16:18) = [dot(nx(6).v, a2); dot(ny(6).v, a2); dot(nz(6).v, a2)];
        acc(19:21) = [dot(nx(7).v, a3); dot(ny(7).v, a3); dot(nz(7).v, a3)];
        acc(22:24) = [dot(nx(8).v, a4); dot(ny(8).v, a4); dot(nz(8).v, a4)];
    end %end if
end %end for

end
```

A.9 Posición y velocidad de los puntos de contacto

```
%{
Esta función realiza el cómputo de la posición y velocidad de los puntos de contacto de los pies del
robot. La función “ComputePosVel” toma como argumentos el modelo del robot (“model”), la posición de
la base (“pb”), la orientación de la base (“Rb”), las posición y velocidades articulares (“q” y “qd”)
y las posiciones de los puntos de contacto respecto a los marcos coordenados de los pies del robot
(“contact_data”).
%}

function [pos, vel] = ComputePosVel(model, pb, Rb, q, qd, contact_data)

global cR0;
Nb = numel(model);
pos = zeros(3, Nb);
vel = zeros(3, Nb);
geom = contact_data(:, 1);
nx = contact_data(:, 3);
ny = contact_data(:, 4);
nz = contact_data(:, 5);

WorldTransform(model, Rb, q);
[body1_pos, body1_vel] = BodyPosVel(model, q, qd, 4);
[body2_pos, body2_vel] = BodyPosVel(model, q, qd, 7);

%foot 1
pos(:, 1) = pb + transpose(cR0(4).R)*(body1_pos + geom(1).v);
pos(:, 2) = pb + transpose(cR0(4).R)*(body1_pos + geom(2).v);
pos(:, 3) = pb + transpose(cR0(4).R)*(body1_pos + geom(3).v);
pos(:, 4) = pb + transpose(cR0(4).R)*(body1_pos + geom(4).v);
%foot 2
pos(:, 5) = pb + transpose(cR0(7).R)*(body2_pos + geom(1).v);
pos(:, 6) = pb + transpose(cR0(7).R)*(body2_pos + geom(2).v);
pos(:, 7) = pb + transpose(cR0(7).R)*(body2_pos + geom(3).v);
pos(:, 8) = pb + transpose(cR0(7).R)*(body2_pos + geom(4).v);

%foot 1
v1 = transpose(cR0(4).R)*[-skew(geom(1).v), eye(3)]*body1_vel;
v2 = transpose(cR0(4).R)*[-skew(geom(2).v), eye(3)]*body1_vel;
v3 = transpose(cR0(4).R)*[-skew(geom(3).v), eye(3)]*body1_vel;
v4 = transpose(cR0(4).R)*[-skew(geom(4).v), eye(3)]*body1_vel;
%foot 2
v5 = transpose(cR0(7).R)*[-skew(geom(1).v), eye(3)]*body2_vel;
v6 = transpose(cR0(7).R)*[-skew(geom(2).v), eye(3)]*body2_vel;
v7 = transpose(cR0(7).R)*[-skew(geom(3).v), eye(3)]*body2_vel;
v8 = transpose(cR0(7).R)*[-skew(geom(4).v), eye(3)]*body2_vel;

%foot 1
vel(:, 1) = [dot(nx(1).v, v1); dot(ny(1).v, v1); dot(nz(1).v, v1)];
vel(:, 2) = [dot(nx(2).v, v2); dot(ny(2).v, v2); dot(nz(2).v, v2)];
vel(:, 3) = [dot(nx(3).v, v3); dot(ny(3).v, v3); dot(nz(3).v, v3)];
vel(:, 4) = [dot(nx(4).v, v4); dot(ny(4).v, v4); dot(nz(4).v, v4)];
%foot 2
vel(:, 5) = [dot(nx(5).v, v5); dot(ny(5).v, v5); dot(nz(5).v, v5)];
vel(:, 6) = [dot(nx(6).v, v6); dot(ny(6).v, v6); dot(nz(6).v, v6)];
vel(:, 7) = [dot(nx(7).v, v7); dot(ny(7).v, v7); dot(nz(7).v, v7)];
vel(:, 8) = [dot(nx(8).v, v8); dot(ny(8).v, v8); dot(nz(8).v, v8)];
end
```

A.10 Posición y velocidad de eslabón

```
%{
Esta función realiza el cómputo de la posición y velocidad del centro de masas de un eslabón con índice “idx”. La función “BodyPosVel” toma como argumentos el modelo del robot (“model”), las posición y velocidades articulares (“q” y “qd”) y el índice del eslabón (“idx”).
%}

function [pos, vel] = BodyPosVel(model, q, qd, idx)

Nb = numel(model);
global v;

v(:, 1) = qd(1).q;
p = zeros(3, Nb);
for i = 2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    p(:, i) = transpose(R)*(model(i).offset + p(:, model(i).parent));
    v(:, i) = cXp*v(:, model(i).parent) + model(i).S*qd(i).q;
end
    pos = p(:, idx);
    vel = v(:, idx);
end
```

A.11 Aceleraciones de los puntos de contacto

```
%{
Esta función realiza el cómputo de las aceleraciones de los puntos de contacto después de aplicar una fuerza unitaria. La función “ConstraintAcc” toma como argumentos el modelo del robot (“model”), las posiciones articulares (“q”), la fuerza unitaria (“force_data”) y los puntos de contacto (“contact_data”).
%}

function acc = ConstraintAcc(model, q, force_data, contact_data)
Nb = numel(model);
foot = contact_data(:, 1);
nx = contact_data(:, 3);
ny = contact_data(:, 4);
nz = contact_data(:, 5);
qdd = q;

global a;
global IA;
global cR0;
pA = zeros(6, Nb);

for i = 2:Nb
    if(i == 4)
        if(force_data.contact_index <= 4)
            s = foot(force_data.contact_index).v;
            f = cR0(i).R*force_data.f;
            pA(:, i) = -[cross(s, f); f];
        end
    elseif(i == 7)
        if(force_data.contact_index > 4)
            s = foot(force_data.contact_index).v;
            f = cR0(i).R*force_data.f;
            pA(:, i) = -[cross(s, f); f];
        end
```

```
    end
end

for i=Nb:-1:2
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    pXc = childToParent(R, r);

    U = IA(i).I*model(i).S;
    D = transpose(model(i).S)*U;
    u = -transpose(model(i).S)*pA(:, i);

    pa = pA(:, i) + U*(D\u);
    pA(:, model(i).parent) = pA(:, model(i).parent) + pXc*pa;
end

acc = zeros(24, 1);
a(:, 1) = -IA(1).I\pA(:, 1);
for i=2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    U = IA(i).I*model(i).S;
    D = transpose(model(i).S)*U;
    u = -transpose(model(i).S)*pA(:, i);

    ap = cXp*a(:, model(i).parent);
    qdd(i).q = D\(u - transpose(U)*ap);
    a(:, i) = ap + model(i).S*qdd(i).q;

    if(i == 4)
        %foot 1
        a1_f1 = [-skew(foot(1).v), eye(3)]*a(:, i);
        a2_f1 = [-skew(foot(2).v), eye(3)]*a(:, i);
        a3_f1 = [-skew(foot(3).v), eye(3)]*a(:, i);
        a4_f1 = [-skew(foot(4).v), eye(3)]*a(:, i);

        Rot = cR0(i).R;
        a1 = transpose(Rot)*a1_f1;
        a2 = transpose(Rot)*a2_f1;
        a3 = transpose(Rot)*a3_f1;
        a4 = transpose(Rot)*a4_f1;
        acc(1:3) = [dot(nx(1).v, a1); dot(ny(1).v, a1); dot(nz(1).v, a1)];
        acc(4:6) = [dot(nx(2).v, a2); dot(ny(2).v, a2); dot(nz(2).v, a2)];
        acc(7:9) = [dot(nx(3).v, a3); dot(ny(3).v, a3); dot(nz(3).v, a3)];
        acc(10:12) = [dot(nx(4).v, a4); dot(ny(4).v, a4); dot(nz(4).v, a4)];
    end %end if
    if(i == 7)
        %foot 2
        a1_f2 = [-skew(foot(5).v), eye(3)]*a(:, i);
        a2_f2 = [-skew(foot(6).v), eye(3)]*a(:, i);
        a3_f2 = [-skew(foot(7).v), eye(3)]*a(:, i);
        a4_f2 = [-skew(foot(8).v), eye(3)]*a(:, i);

        Rot = cR0(i).R;
        a1 = transpose(Rot)*a1_f2;
        a2 = transpose(Rot)*a2_f2;
        a3 = transpose(Rot)*a3_f2;
        a4 = transpose(Rot)*a4_f2;
        acc(13:15) = [dot(nx(5).v, a1); dot(ny(5).v, a1); dot(nz(5).v, a1)];
        acc(16:18) = [dot(nx(6).v, a2); dot(ny(6).v, a2); dot(nz(6).v, a2)];
        acc(19:21) = [dot(nx(7).v, a3); dot(ny(7).v, a3); dot(nz(7).v, a3)];
        acc(22:24) = [dot(nx(8).v, a4); dot(ny(8).v, a4); dot(nz(8).v, a4)];
```

```
    end %end if
end %end for

end
```

A.12 Algoritmo del cuerpo articulado

```
%{
Esta función realiza el cómputo de las aceleraciones articulares dadas las fuerzas externas y los pares articulares. La función “ArticulatedBody” toma como argumentos el modelo del robot (“model”), la orientación de la base (“Rb”), las posiciones y velocidades articulares (“q” y “qd”), los pares articulares (“tau”), las fuerzas de los puntos de contacto (“force”), las posiciones relativas de los puntos de contacto y la fuerza externa (“e_force”).
%}

function qdd = ArticulatedBody(model, Rb, q, qd, tau, force, contact_data, e_force)
Nb = numel(model);
foot = contact_data(:, 1);
bias = struct('p', zeros(6, 1));
qdd = q;

global v;
global a;
global c;

global IA;
global cR0;
pA = repmat(bias, Nb, 1);

gb = transpose(Rb)*[0; -9.81; 0];
g = [0; 0; 0; gb];

cR0(1).R = transpose(Rb);
v(:, 1)= qd(1).q;
for i = 2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);
    cR0(i).R = transpose(R)*cR0(model(i).parent).R;

    v(:, i) = cXp*v(:, model(i).parent) + model(i).S*qd(i).q;
    c(:, i) = scross(v(:, i), model(i).S*qd(i).q);
    IA(i).I = model(i).I;

    if(i == 4)
        s1 = foot(1).v;
        s2 = foot(2).v;
        s3 = foot(3).v;
        s4 = foot(4).v;
        f1 = cR0(i).R*force(1:3);
        f2 = cR0(i).R*force(4:6);
        f3 = cR0(i).R*force(7:9);
        f4 = cR0(i).R*force(10:12);
        f = f1 + f2 + f3 + f4;
        t1 = cross(s1, f1) + cross(s2, f2) + cross(s3, f3) + cross(s4, f4);
        pA(i).p = dual_scross(v(:, i), model(i).I*v(:, i)) - [t1; f];
    elseif(i == 7)
        s1 = foot(5).v;
        s2 = foot(6).v;
        s3 = foot(7).v;
        s4 = foot(8).v;
        f1 = cR0(i).R*force(13:15);
        f2 = cR0(i).R*force(16:18);
```

```
        f3 = cR0(i).R*force(19:21);
        f4 = cR0(i).R*force(22:24);
        f = f1 + f2 + f3 + f4;
        t1 = cross(s1, f1) + cross(s2, f2) + cross(s3, f3) + cross(s4, f4);
        pA(i).p = dual_scross(v(:, i), model(i).I*v(:, i)) - [t1; f];
    elseif(i == 8)
        f = cR0(i).R*e_force;
        pA(i).p = dual_scross(v(:, i), model(i).I*v(:, i)) - [zeros(3, 1); f];
    else
        pA(i).p = dual_scross(v(:, i), model(i).I*v(:, i));
    end
end

IA(1).I = model(1).I;
pA(1).p = dual_scross(v(:, 1), model(1).I*v(:, 1));
for i=Nb:-1:2
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    pXc = childToParent(R, r);
    cXp = parentToChild(R, r);

    U = IA(i).I*model(i).S;
    D = transpose(model(i).S)*U;
    u = tau(i).q - transpose(model(i).S)*pA(i).p - 1e-9*qd(i).q;
    Ia = IA(i).I - U*(D\transpose(U));
    pa = pA(i).p + Ia*c(:, i) + U*(D\u);
    IA(model(i).parent).I = IA(model(i).parent).I + pXc*Ia*cXp;
    pA(model(i).parent).p = pA(model(i).parent).p + pXc*pa;
end

a(:, 1) = -IA(1).I\pA(1).p + g;

for i=2:Nb
    R = rotate(q(i).q);
    r = skew(model(i).offset);
    cXp = parentToChild(R, r);

    U = IA(i).I*model(i).S;
    D = transpose(model(i).S)*U;
    u = tau(i).q - transpose(model(i).S)*pA(i).p;

    ap = cXp*a(:, model(i).parent) + c(:, i);
    qdd(i).q = D\(u - transpose(U)*ap);
    a(:, i) = ap + model(i).S*qdd(i).q;
end

qdd(1).q = a(:, 1);

end
```

B Código de Matlab de funciones elementales

B.1 Transformaciones espaciales

```
%{
Esta función realiza una transformación espacial entre los marcos coordenados de un eslabón hijo a un
eslabón padre. La función “childToParent” toma como argumentos una matriz de rotación (“R”) y un
vector que define una traslación (“r”).
%}
```

```
function X = childToParent(R, r)

X = [R, r*R;
     zeros(3), R];

end


%{
Esta función realiza una transformación espacial entre los marcos coordenados de un eslabón padre a
un eslabón hijo. La función “parentToChild” toma como argumentos una matriz de rotación (“R”) y un
vector que define una traslación (“r”).
%}

function X = parentToChild(R, r)

X = [transpose(R), zeros(3);
     -transpose(R)*r, transpose(R)];
end


%Matriz 6x6 con “R” en la diagonal.

function X = diag_spa(R)
    X = [R, zeros(3, 3); zeros(3, 3), R];
end
```

B.2 Producto cruz espacial

```
%Representación matricial del producto cruz espacial.

function M = op_scross(v)

a = v(1:3);
b = v(4:6);
M =  [skew(a), zeros(3);
      skew(b), skew(a)];
end


%Representación matricial del producto cruz espacial dual.

function M = op_dcross(v)

a = v(1:3);
b = v(4:6);
M = [skew(a), skew(b);
     zeros(3, 3), skew(a)];
end


%Producto cruz espacial de “u” y “v”.

function r = scross(u, v)

a = u(1:3);
b = u(4:6);
c = v(1:3);
d = v(4:6);

r = [cross(a, c); cross(b, c) + cross(a, d)];
```

```
end


%Producto cruz espacial dual de “u” y “v”.

function r = dual_scross(u, v)

a = u(1:3);
b = u(4:6);
c = v(1:3);
d = v(4:6);

r = [cross(a, c) + cross(b, d); cross(a, d)];

end


%Matriz anti simétrica.

function S = skew(a)

S = [    0, -a(3),  a(2);
      a(3),     0, -a(1);
     -a(2),  a(1),    0];

end
```

B.3 Matrices de rotación

```
%Matriz rotación alrededor del eje “X”.

function R = rotateX(q)


R = [1,      0,       0;
     0, cos(q), -sin(q);
     0, sin(q),  cos(q)];
end


%Matriz rotación alrededor del eje “Z”.

function R = rotateZ(q)

R = [cos(q), -sin(q), 0;
     sin(q),  cos(q), 0;
          0,       0, 1];
end


function R = rotate(q)

sys = numel(q);
if sys == 2
    R = rotateX(q(1))*rotateZ(q(2));
else
    R = rotateX(q);
end

end
```

```
%Esta función calcula matrices de rotación para todos los eslabones hacia el marco global.

function WorldTransform(model, Rb, q)

Nb = numel(model);
global cR0;
cR0(1).R = transpose(Rb);
for i=2:Nb
    R = rotate(q(i).q);
    cR0(i).R = transpose(R)*cR0(model(i).parent).R;
end

end


%Esta función trasforma una matriz de rotación a cuaternión.

function R = quaternion2matrix(q)
s = q(1);
v = q(2:4);

r11 = 1 - 2*v(2)*v(2) -2*v(3)*v(3);
r21 = 2*v(1)*v(2) + 2*s*v(3);
r31 = 2*v(1)*v(3) - 2*s*v(2);

r12 = 2*v(1)*v(2) - 2*s*v(3);
r22 = 1 - 2*v(1)*v(1) -2*v(3)*v(3);
r32 = 2*v(2)*v(3) + 2*s*v(1);

r13 = 2*v(1)*v(3) + 2*s*v(2);
r23 = 2*v(2)*v(3) - 2*s*v(1);
r33 =1 - 2*v(1)*v(1) -2*v(2)*v(2);

R = [r11, r12, r13;
     r21, r22, r23;
     r31, r32, r33];

end
```

B.4 Mapeo de índices

```
%{
Esta función mapea los índices de las articulaciones “m” y “n” a los índices del arreglo donde se
almacenan los resultados del algoritmo empleado.
%}

function [a, b, c, d] = mapIndices(m, n, elem)

global dof;
dof_offset1 = 0;
if(~isempty(m))
    for i=2:m-1
        dof_offset1= dof_offset1 + dof(i);
    end
end

dof_offset2 = 0;
if(~isempty(n))
    for i=2:n-1
        dof_offset2 = dof_offset2 + dof(i);
    end
end
```

```
dof_count = size(elem);
rows = dof_count(1);
cols = dof_count(2);

a = dof_offset1 + 1;
b = dof_offset1 + rows;
c = dof_offset2 + 1;
d = dof_offset2 + cols;

end


%{
Esta función mapea el índice de la articulación “m” al índice del arreglo donde se almacenan el
Jacobiano.
%}

function [a, b] = mapJacobianIdx(m, elem)

global dof;
dof_offset = 0;
if(~isempty(m))
    for i=1:m-1
        dof_offset = dof_offset + dof(i);
    end
end
dof_count = size(elem, 2);

a = dof_offset + 1;
b = dof_offset + dof_count;

end
```

C Simulación del robot humanoide

C.1 Código de Matlab del simulador

```
%{
En esta función se realiza el balance con un programa cuadrático en modo elástico. Además, se realiza
la simulación del robot humanoide
%}

function qdd = bipedSim2(model, dof, numCon, qb, q, qd, tau, A, ad, contact_data, e_force)

    Nb = numel(model);
    numDof = sum(dof(2:Nb));
    qa = zeros(numDof, 1);
    qda = zeros(numDof, 1);
    off = 0;
    for i=2:Nb
        dim = numel(q(i).q);
        qa(off+1: off+dim) = q(i).q;
        qda(off+1: off+dim) = qd(i).q;
        off = off + dim;
    end

    ref = [-0.6435;     0;     1.287;
           -0.6435;     0;   -0.6435;
                 0; 1.287;   -0.6435;
```

```
            0;     0;         0;
            0; -1.5708; 0;
            0;  1.5708; 0];

pb = q(1).q(4:6);
Rb = quaternion2matrix(qb);
massT = getMass();

wfoot = contact_data(:, 2);
sizeSystem = 3*numCon;

%control
[H, F, I0c] = CompositeRigidBody(model, q);
[J, bias] = ComputeContactJacobians(model, Rb, q, qd);
[C, p0c] = NewtonEuler(model, Rb, q, qd);
[AG, bG, hG] = CentroidalMomentum2(model, pb, Rb, q, qd);
M = [I0c, F; transpose(F), H];
S = [spalloc(6, numDof, 0); speye(numDof)];
h = [p0c; C];

w1 = 1000;
w2 = 100;
w3 = 1;
w4 = 1;
dim1 = numDof + 6;    %qdd dim
dim2 = numDof;        %tau dim
dim3 = 3*numCon; %force dim
dim4 = 4*numCon; %beta dim
W = computeFrictionBasis(numCon);
T1 = [M, -S, -transpose(J), spalloc(dim1, dim4, 0);
      spalloc(dim3, dim1, 0), spalloc(dim3, dim2, 0), speye(dim3), -W];
T2 = [J, spalloc(dim3, numDof + 24, 0), spalloc(dim3, dim4, 0)];
T3 = [AG, spalloc(6, numDof + 24, 0), spalloc(6, dim4, 0)];
T4 = [spalloc(dim2, 6, 0), speye(dim2), spalloc(dim2, dim2, 0), spalloc(dim2, dim3, 0),
      spalloc(dim2, dim4, 0)];
A1 = [w1*T1; w2*T2; w3*T3; w4*T4];

[com_pos, ~] = ComputeCOM(model, pb, Rb, q, qd);
kp = 10;
kv = 2*sqrt(kp);
cm_ref = massT*kp*([0; 0.1683; 0.0575] - com_pos) - kv*hG(4:6);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cálculo de las tareas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t1 = [-h; zeros(dim3, 1)];
t2 = -bias;

kp1 = 40;
kv1 = 2*sqrt(kp1);
t4 = kp1*(ref - qa) - kv1*qda;

%kg = 10;
kg = 40;
hG_ref = -kg*hG(1:3);
t3 = [-bG(1:3) + hG_ref; -bG(4:6) + cm_ref];
b1 = [w1*t1; w2*t2; w3*t3; w4*t4];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ensamble del programa cuadrático
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dim = size(A1, 2);
rA = size(A1, 1);

Q = [spalloc(dim1,dim1, 0),spalloc(dim1,dim2, 0),spalloc(dim1,dim3, 0), spalloc(dim1,dim4, 0);
```

```
     spalloc(dim2,dim1, 0), speye(dim2), spalloc(dim2,dim3, 0), spalloc(dim2,dim4, 0);
     spalloc(dim3,dim1, 0), spalloc(dim3,dim2, 0), speye(dim3), spalloc(dim3,dim4, 0);
     spalloc(dim4,dim1, 0), spalloc(dim4,dim2, 0), spalloc(dim4,dim3, 0), spalloc(dim4,dim4,0)];
Q = Q + speye(dim1 + dim2 + dim3 + dim4);

G = w1*[spalloc(dim4, dim1, 0), spalloc(dim4, dim2, 0), spalloc(dim4, dim3, 0), -speye(dim4)];
d = w1*zeros(dim4, 1);
rG = size(G, 1);

gamma = 1e6;
c = zeros(dim, 1);
eA = gamma*ones(rA, 1);
eG = gamma*ones(rG, 1);
bigC = [c; eA; eA; eG; eG];

bigA = [A1, speye(rA), -speye(rA), spalloc(rA, rG, 0), spalloc(rA, rG, 0)];
bigG = [G, spalloc(rG,rA, 0), spalloc(rG, rA, 0), speye(rG), -speye(rG);
        spalloc(rA,dim,0), -speye(rA), spalloc(rA,rA,0), spalloc(rA,rG,0), spalloc(rA,rG,0);
        spalloc(rA,dim,0), spalloc(rA,rA,0), -speye(rA), spalloc(rA,rG,0), spalloc(rA,rG,0);
        spalloc(rG,dim,0), spalloc(rG,rA,0), spalloc(rG,rA,0), -speye(rG), spalloc(rG,rG,0);
        spalloc(rG,dim,0), spalloc(rG,rA,0), spalloc(rG,rA,0), spalloc(rG,rG,0), -speye(rG)];

bigB = b1;
dA = zeros(rA, 1);
dG = zeros(rG, 1);
bigD = [d; dA; dA; dG; dG];

delta = 1e-6;
deltaA = delta*speye(rA);
deltaG = delta*speye(rG);
bigQ = [Q, spalloc(dim, rA, 0), spalloc(dim, rA, 0), spalloc(dim, rG, 0), spalloc(dim, rG, 0);
        spalloc(rA, dim, 0), deltaA, spalloc(rA, rA, 0), spalloc(rA, rG, 0), spalloc(rA, rG, 0);
        spalloc(rA, dim, 0), spalloc(rA, rA, 0), deltaA, spalloc(rA, rG, 0), spalloc(rA, rG, 0);
        spalloc(rG, dim, 0), spalloc(rG, rA, 0), spalloc(rG, rA, 0), deltaG, spalloc(rG, rG, 0);
        spalloc(rG, dim, 0), spalloc(rG, rA, 0), spalloc(rG, rA, 0), spalloc(rG, rG,0), deltaG];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solución de un programa cuadrático en modo elástico para el balance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sol = IPQP(bigQ, bigC, bigG, bigD, bigA, bigB, 20);
x1 = sol(1:dim);

offset1 = numDof + 6 + 1;
offset2 = 6 + (2*numDof);
tau_qp = x1(offset1:offset2);
%map torques
for j=2:Nb
    [idx1, idx2, ~, ~] = mapIndices(j, 0, dof ,tau(j).q);
    tau(j).q = tau_qp(idx1:idx2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cómputo de las fuerzas de contacto
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a0 = DefaultAcc(model, Rb, q, qd, tau, contact_data, e_force);
footFB = ComputePosVel(model, pb, Rb, q, qd, contact_data);

beta = 25;
alpha = beta;
for j=1:numCon
    index1 = 3*(j - 1) + 1;
    index2 = index1 + 2;
    ad(index1:index2) = -2*alpha*footFB(j).V + beta*beta*(wfoot(j).v - footFB(j).P);
end
b = (ad - a0);
```

```
    for j=1:numCon
        for k=1:3
            switch k
                case 1
                    force_data.f = [1; 0; 0];
                    force_data.axis = 1;
                case 2
                    force_data.f = [0; 1; 0];
                    force_data.axis = 2;
                case 3
                    force_data.f = [0; 0; 1];
                    force_data.axis = 3;
                otherwise
                    disp('error');
            end %end switch
            force_data.contact_index = j;
            a1 = ConstraintAcc(model, Rb, q, qd, tau, force_data, contact_data, e_force);
            A(:, 3*(j-1) + k) = a1 - a0;
        end %end for
    end %end for

    [U, S, V] = svd(A + 1e-6*eye(size(A)));
    epsi = 1e-5;
    for j = 1:sizeSystem
        if(S(j, j) < epsi)
            S(j, j) = 0;
        else
            S(j, j) = 1 / S(j, j);
        end
    end
    f = V*S*transpose(U)*b;

    qdd = ArticulatedBody(model, Rb, q, qd, tau, f, contact_data, e_force);
end
```

C.2 Modelo del robot humanoide en Matlab

El modelo del robot humanoide bajo estudio consiste de 12 eslabones conectados por articulaciones. En la figura C.1 se muestra el indexado de los eslabones que conforman al robot y que se emplea en los algoritmos de dinámica y cinemática.

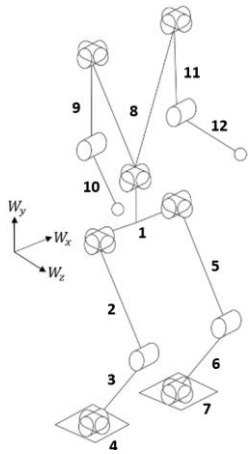


Figura C.1. Indexado de los eslabones del robot humanoide bajo estudio.

```
%{
Esta función de Matlab retorna el modelo del robot humanoide para el balance de postura. El modelo
consiste de todas las propiedades del robot como la masa, inercia, ejes de giro, etc.
%}

function [model, q] = getModel()

scale = 1;
%link 1
r=0.1;
h = 0.4;
link1.m = 13.375*scale;
link1.S = eye(6);
Ixx = (1/12)*link1.m*(3*r*r + h*h);
Iyy = 0.5*link1.m*r*r;
link1.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link1.c = [0;0;0];
link1.I = [link1.Icm, zeros(3);
           zeros(3), link1.m*eye(3)];
link1.parent = -1;
link1.offset = [0; 0; 0];

%link 2
h = 0.5;
link2.m = 8.5*scale;
link2.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link2.m*(3*r*r + h*h);
Iyy = 0.5*link2.m*r*r;
link2.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link2.c = [0; -0.25; 0];
link2.I = [link2.Icm + link2.m*skew(link2.c)*transpose(skew(link2.c)),  link2.m*skew(link2.c);
           link2.m*transpose(skew(link2.c)),  link2.m*eye(3)];
link2.parent = 1;
s1_l1 = [-0.2; 0; 0];
link2.offset = s1_l1;

%link 3
h = 0.5;
link3.m = 3.5*scale;
link3.S = [1;
           0;
           0;
           0;
           0;
           0];
Ixx = (1/12)*link3.m*(3*r*r + h*h);
Iyy = 0.5*link3.m*r*r;
link3.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link3.c = [0; -0.25; 0];
link3.I = [link3.Icm + link3.m*skew(link3.c)*transpose(skew(link3.c)),  link3.m*skew(link3.c);
           link3.m*transpose(skew(link3.c)),  link3.m*eye(3)];
link3.parent = 2;
s2_l1 = [0; -0.5; 0];
```

```
link3.offset = s2_l1;

%link 4
h = 0.2;
link4.m = 1.25*scale;
link4.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link4.m*(3*r*r + h*h);
Iyy = 0.5*link4.m*r*r;
link4.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link4.c = [0;0;0];
link4.I = [link4.Icm, zeros(3);
           zeros(3), link4.m*eye(3)];
link4.parent = 3;
s3_l1 = [0; -0.5; 0];
link4.offset = s3_l1;

%link 5
h = 0.5;
link5.m = 8.5*scale;
link5.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link5.m*(3*r*r + h*h);
Iyy = 0.5*link5.m*r*r;
link5.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link5.c = [0; -0.25; 0];
link5.I = [link5.Icm + link5.m*skew(link5.c)*transpose(skew(link5.c)),  link5.m*skew(link5.c);
           link5.m*transpose(skew(link5.c)),  link5.m*eye(3)];
link5.parent = 1;
s1_l2 = [0.2; 0; 0];
link5.offset = s1_l2;

%link 6
h = 0.5;
link6.m = 3.5*scale;
link6.S = [1;
           0;
           0;
           0;
           0;
           0];
Ixx = (1/12)*link6.m*(3*r*r + h*h);
Iyy = 0.5*link6.m*r*r;
link6.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link6.c = [0; -0.25; 0];
link6.I = [link6.Icm + link6.m*skew(link6.c)*transpose(skew(link6.c)),  link6.m*skew(link6.c);
           link6.m*transpose(skew(link6.c)),  link6.m*eye(3)];
link6.parent = 5;
s2_l2 = [0; -0.5; 0];
link6.offset = s2_l2;
```

```
%link 7
h = 0.2;
link7.m = 1.25*scale;
link7.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link7.m*(3*r*r + h*h);
Iyy = 0.5*link7.m*r*r;
link7.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link7.c = [0; 0; 0];
link7.I = [link7.Icm,  zeros(3);
           zeros(3),  link7.m*eye(3)];
link7.parent = 6;
s3_l2 = [0; -0.5; 0];
link7.offset = s3_l2;

%link 8
h = 0.6;
link8.m = 30.125*scale;
link8.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link8.m*(3*r*r + h*h);
Iyy = 0.5*link8.m*r*r;
link8.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link8.c = [0; 0.3; 0];
link8.I = [link8.Icm + link8.m*skew(link8.c)*transpose(skew(link8.c)),  link8.m*skew(link8.c);
           link8.m*transpose(skew(link8.c)),  link8.m*eye(3)];
link8.parent = 1;
st0 = [0; 0.2; 0];
link8.offset = st0;

%link 9
h = 0.4;
link9.m = 2.5*scale;
link9.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link9.m*(3*r*r + h*h);
Iyy = 0.5*link9.m*r*r;
link9.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link9.c = [0; -0.2; 0];
link9.I = [link9.Icm + link9.m*skew(link9.c)*transpose(skew(link9.c)),  link9.m*skew(link9.c);
           link9.m*transpose(skew(link9.c)),  link9.m*eye(3)];
link9.parent = 8;
st1 = [-0.2; 0.6; 0];
link9.offset = st1;
```

```
%link 10
h = 0.4;
link10.m = 2.5*scale;
link10.S = [1;
            0;
            0;
            0;
            0;
            0];
Ixx = (1/12)*link10.m*(3*r*r + h*h);
Iyy = 0.5*link10.m*r*r;
link10.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link10.c = [0; -0.2; 0];
link10.I = [link10.Icm + link10.m*skew(link10.c)*transpose(skew(link10.c)),
link10.m*skew(link10.c);
            link10.m*transpose(skew(link10.c)),  link10.m*eye(3)];
link10.parent = 9;
sa1 = [0; -0.4; 0];
link10.offset = sa1;

%link 11
h = 0.4;
link11.m = 2.5*scale;
link11.S = [1, 0;
           0, 0;
           0, 1;
           0, 0;
           0, 0;
           0, 0];
Ixx = (1/12)*link11.m*(3*r*r + h*h);
Iyy = 0.5*link11.m*r*r;
link11.Icm = [Ixx, 0, 0;
              0, Iyy, 0;
              0, 0, Ixx];
link11.c = [0; -0.2; 0];
link11.I = [link11.Icm + link11.m*skew(link11.c)*transpose(skew(link11.c)),
link11.m*skew(link11.c);
            link11.m*transpose(skew(link11.c)),  link11.m*eye(3)];
link11.parent = 8;
st2 = [0.2; 0.6; 0];
link11.offset = st2;

%link 12
h = 0.4;
link12.m = 2.5*scale;
link12.S = [1;
            0;
            0;
            0;
            0;
            0];
Ixx = (1/12)*link12.m*(3*r*r + h*h);
Iyy = 0.5*link12.m*r*r;
link12.Icm = [Ixx, 0, 0;
             0, Iyy, 0;
             0, 0, Ixx];
link12.c = [0; -0.2; 0];
link12.I = [link12.Icm + link12.m*skew(link12.c)*transpose(skew(link12.c)),
link12.m*skew(link12.c);
            link12.m*transpose(skew(link12.c)),  link12.m*eye(3)];
link12.parent = 11;
sa2 = [0; -0.4; 0];
```

```
link12.offset = sa2;

model = [link1; link2; link3; link4; link5; link6; link7; link8; link9; link10; link11; link12];
q1 = struct('q', zeros(6, 1));
q2 = struct('q', [-0.6435; 0]);
q3 = struct('q', 1.287);
q4 = struct('q', [-0.6435; 0]);
q5 = struct('q', [-0.6435; 0]);
q6 = struct('q', 1.287);
q7 = struct('q', [-0.6435; 0]);
q8 = struct('q', [0; 0]);
q9 = struct('q', [0; 0]);
q10 = struct('q', -2*0.7853);
q11 = struct('q', [0; 0]);
q12 = struct('q', -2*0.7853);

q = [q1; q2; q3; q4; q5; q6; q7; q8; q9; q10; q11; q12];

end
%{
Esta función retorna la masa total del robot.
%}

function mass = getMass()

model = getModel();
Nb = numel(model);

mass = 0;
for j=1:Nb
    mass = mass + model(j).m;
end

end
```

C.3 Código de Matlab de un escenario de balance con plataformas disparejas

```
%{
Aquí se realiza la programación del escenario 4 del capítulo 3 del presente trabajo de tesis para el
balance con el robot humanoide puesto sobre dos plataformas disparejas que alternan su movimiento.
%}

foot_vec = struct('v', zeros(3, 1));
foot = repmat(foot_vec, 8, 1);

foot(1).v = [-0.1; 0;  0.1];
foot(2).v = [ 0.1; 0;  0.1];
foot(3).v = [ 0.1; 0; -0.1];
foot(4).v = [-0.1; 0; -0.1];
foot(5).v = [-0.1; 0;  0.1];
foot(6).v = [ 0.1; 0;  0.1];
foot(7).v = [ 0.1; 0; -0.1];
foot(8).v = [-0.1; 0; -0.1];

numContacts = 8;
wfoot = repmat(foot_vec, 8, 1);
ground = -0.8;
wfoot(1).v = [-0.3; ground;  0.1];
wfoot(2).v = [-0.1; ground;  0.1];
wfoot(3).v = [-0.1; ground; -0.1];
wfoot(4).v = [-0.3; ground; -0.1];
wfoot(5).v = [ 0.1; ground;  0.1];
wfoot(6).v = [ 0.3; ground;  0.1];
```

```
wfoot(7).v = [ 0.3; ground; -0.1];
wfoot(8).v = [ 0.1; ground; -0.1];

normal = struct('v', zeros(3, 1));
nx = repmat(normal, 8, 1);
ny = repmat(normal, 8, 1);
nz = repmat(normal, 8, 1);
for i=1:numContacts
    nx(i).v = [1; 0; 0];
    ny(i).v = [0; 1; 0];
    nz(i).v = [0; 0; 1];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[model, q] = getModel();
Nb = numel(model); %num bodies
contact_data = [foot, wfoot, nx, ny, nz];

dof = zeros(Nb, 1);
for i = 2:Nb
    dof(i) = size(q(i).q, 1);
end
numDof = sum(dof(2:Nb));
qa = zeros(numDof, 1);

qd = q;
qdd = q;
tau = q;
%extra data
for i=1:Nb
    val = zeros(size(q(i).q));
    qd(i).q = val;
    qdd(i).q = val;
    tau(i).q = val;
end

qb = [1; 0; 0; 0];
pb = [0;0;0];
Rb = eye(3);

force_data = struct('f', zeros(3, 1), 'contact_index', 0, 'axis', 0);
%forward dynamics
numSteps = 6000;
dt = 1e-3;
dt_6 = dt / 6;
sizeSystem = 3*numContacts;
A = zeros(sizeSystem, sizeSystem);
ad = zeros(sizeSystem, 1);

e_force = zeros(3, 1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global v;
global a;
global c;
v = zeros(6, Nb);
a = zeros(6, Nb);
c = zeros(6, Nb);
trans = struct('X', zeros(6, 6));
rota = struct('R', eye(3, 3));
ine = struct('I', zeros(6, 6));
global IA;
global cX0;
global cR0;
IA = repmat(ine, Nb, 1);
cX0 = repmat(trans, Nb, 1);
```

```
cR0 = repmat(rota, Nb, 1);
external_force = [400; 0; 0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%QP Euler
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

surface_vel = 0.4;
deltaBaseFeet = [0; 0; 0.2/500];
backward = 0;
forward = 1;

theta = 20;
q(4).q(1) = q(4).q(1) - (theta/180)*pi;
q(7).q(1) = q(4).q(1) + (theta/180)*pi;
Rf1 = rotateX((theta/180)*pi);
Rf2 = rotateX(-(theta/180)*pi);

for i=1:numContacts
    if( i > 4)
        nx(i).v = Rf1*nx(i).v;
        ny(i).v = Rf1*ny(i).v;
        nz(i).v = Rf1*nz(i).v;
        wfoot(i).v = Rf1*wfoot(i).v;
    else
        nx(i).v = Rf2*nx(i).v;
        ny(i).v = Rf2*ny(i).v;
        nz(i).v = Rf2*nz(i).v;
        wfoot(i).v = Rf2*wfoot(i).v;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Simulación del escenario de balance durante “numSteps”
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:numSteps
    test1 = wfoot(1).v;
    if(test1(3) >= 0.3)
        backward = 1;
        forward = 0;
    elseif(test1(3) <= -0.1)
        forward = 1;
        backward = 0;
    end

    if(forward == 1)
        for j=1:numContacts
            if( j > 4)
                wfoot(j).v = wfoot(j).v - deltaBaseFeet;
            else
                wfoot(j).v = wfoot(j).v + deltaBaseFeet;
            end
        end
    elseif(backward == 1)
        for j=1:numContacts
            if( j > 4 )
                wfoot(j).v = wfoot(j).v + deltaBaseFeet;
            else
                wfoot(j).v = wfoot(j).v - deltaBaseFeet;
            end
        end
    end
    contact_data = [foot, wfoot, nx, ny, nz];

    if( (i >= 1000) && (i <= 1080) )
        e_force = external_force;
```

```
    else
        e_force = [0; 0; 0];
    end

    %aceleraciones articulares
    a2 = bipedSim2(model, dof, numContacts, qb, q, qd, tau, A, ad, contact_data, e_force);

    %integrate Euler
    for j=1:Nb
        qd(j).q = qd(j).q + dt*a2(j).q;
        q(j).q = q(j).q + dt*qd(j).q;
    end

    omega1 = qd(1).q(1:3);
    Q = [-qb(2), -qb(3), -qb(4);
          qb(1), -qb(4),  qb(3);
          qb(4),  qb(1), -qb(2);
          qb(3),  qb(2),  qb(1)];

    qb  = qb + 0.5*Q*omega1*dt;
    qb = qb./norm(qb);

    pb = q(1).q(4:6);
    fprintf('iter: %d\n', i);
end
```

D Optimización convexa

D.1 Código de Matlab de algoritmo del punto interior con regularización dual

```
%{
Esta función implementa el algoritmo propuesto del punto interior con regularización dual para el
balance de postura con control de cuerpo completo.
%}

function [x, y, z] = IPQP(Q, c, G, d, A, b, kmax)

rho = 1e-12;
qpSize = numel(c);
numEqu= numel(b);
numIne = numel(d);

k = 0;
x = zeros(qpSize, 1);
s = ones(numIne, 1);
y = zeros(numEqu, 1);
z = ones(numIne, 1);

di_ni = spalloc(qpSize, numIne, 0);
ne_ni = spalloc(numEqu, numIne, 0);
ni_ne = spalloc(numIne, numEqu, 0);
ideEqu = speye(numEqu);
ideIne = speye(numIne);

S = spalloc(numIne, numIne, numIne);
Z = spalloc(numIne, numIne, numIne);
off2 = qpSize;
off3 = qpSize + numIne;
off4 = qpSize + numIne + numEqu;
```

```
K = [Q + rho*speye(qpSize),        di_ni, transpose(A), transpose(G);
     A,                            ne_ni,   rho*ideEqu,        ne_ni;
     G,                           ideIne,        ni_ne,   rho*ideIne;
     transpose(di_ni),      spdiag(Z, z),        ni_ne, spdiag(S, s)];

rQ = (Q + rho*speye(qpSize))*x + c + transpose(A)*y + transpose(G)*z;
rA = A*x - b + rho*y;
rG = G*x + s - d + rho*z;
rSZ = z.*s;

rhs = [rQ; rA; rG; rSZ];
xaff = -K\rhs;
deltaS = xaff(off2+1 : off2+numIne);
deltaZ = xaff(off4+1 : off4+numIne);

s = max(1, abs(s + deltaS));
z = max(1, abs(z + deltaZ));
e = ones(numIne, 1);

while(k < kmax)

    mu = dot(z,s) / numIne;
    K = [Q + rho*speye(qpSize),        di_ni, transpose(A), transpose(G);
         A,                            ne_ni,   rho*ideEqu,        ne_ni;
         G,                           ideIne,        ni_ne,   rho*ideIne;
         transpose(di_ni),      spdiag(Z, z),        ni_ne, spdiag(S, s)];

    rQ = (Q + rho*speye(qpSize))*x + transpose(A)*y + transpose(G)*z + c;
    rA = A*x - b + rho*y;
    rG = G*x + s - d + rho*z;
    rSZ = z.*s;

    E = [norm(rQ), norm(rA), norm(rG), norm(rSZ)];
    scaling = max([max(max(Q)), norm(c, Inf), max(max(G)), norm(d, Inf),
                   max(max(A)), norm(b, Inf)]);
    if(max(E) < 1e-3*scaling)
        break;
    end
    rhs = [rQ; rA; rG; rSZ];

    sol1 = -K\rhs;
    deltaS = sol1(off2+1 : off2+numIne);
    deltaZ = sol1(off4+1 : off4+numIne);

    alpha_prim = min( 1, ipstep(-s./deltaS) );
    alpha_dual = min( 1, ipstep(-z./deltaZ) );
    alpha = min(alpha_prim, alpha_dual);

    mu_aff = dot(s + alpha*deltaS, z + alpha*deltaZ) / numIne;
    sigma = (mu_aff / mu)^3;

    rSZ = z.*s - sigma*mu*e + deltaS.*deltaZ;
    rhs = [rQ; rA; rG; rSZ];
    sol2 = -K\rhs;
    deltaX = sol2(1:qpSize);
    deltaS = sol2(off2+1:off2+numIne);
    deltaY = sol2(off3+1:off3+numEqu);
    deltaZ = sol2(off4+1:off4+numIne);

    %tau = 1 - 0.1*((0.8)^k);
    tau = max(0.995, 1 - mu);
    alpha_prim = min( 1, ipstep(-tau*s./deltaS) );
    alpha_dual = min( 1, ipstep(-tau*z./deltaZ) );
    alpha = min(alpha_prim, alpha_dual);
```

```
    x = x + alpha*deltaX;
    s = s + alpha*deltaS;
    y = y + alpha*deltaY;
    z = z + alpha*deltaZ;

    k = k + 1;
end

end



%La función “ipstep” forma parte de la búsqueda de línea del algoritmo del punto interior.

function alpha = ipstep(x)

size = numel(x);
alpha = Inf;
for i=1:size
   if x(i) > 0
        alpha = min(alpha, x(i));
   end
end

end
```