Estudio de la Marcha Humana y su Validación en un Ambiente Virtual para un Robot Bípedo

Monografía presentada como requisito parcial
para optar el título de Magíster en Electrónica

Diego Alfonso Aguilar Cardona

Tutor:
PhD. Andrés Vivas Albán

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Maestría en Ingeniería Área Electrónica y Telecomunicaciones
Popayán

Noviembre 2009

Doy las gracias a todas las personas que de una u otra forma colaboraron en el desarrollo de este trabajo, especialmente a mi director PhD. Andrés Vivas por sus valiosos consejos, aportes, paciencia y ayuda, a todos los profesores de la Maestría en Ingeniería, al Instituto de Posgrados en Ingeniería Electrónica y Telecomunicaciones, a la Facultad de Electrónica y Telecomunicaciones, a mi esposa Eliana por escucharme y permitirme mejorar como persona y como profesional, a mi madre, donde quiera que estés, por todos los consejos de vida, nunca los olvidaré, a mi padre por siempre estar ahí para mi como amigo, apoyo y ejemplo, y por último a mi hermana que siempre ha sido una extensión de mi ser.

Índice general

Índice de figuras

Índice de cuadros

RESUMEN

Los robots bípedos presentan altas semejanzas a la estructura humana, mostrando ventajas significativas con respecto a robots con otro tipo de locomoción; esto se debe a su facilidad para desplazarse en ambientes diseñados para seres humanos, su caminar erguido y estructura física hacen de ellos los mejores candidatos para desarrollar tareas junto o en remplazo de humanos. Otra de las ventajas del estudio de la configuración de robots bípedos es el aporte al desarrollo de prótesis activas de miembros inferiores y exoesqueletos.

El Grupo de Investigación en Automática Industrial (GAI) de la Universidad del Cauca tiene un firme interés en el diseño y desarrollo de prótesis de pierna, pero se vislumbra la dificultad de medir su impacto en las personas, ya que al tratar de obtener la información directamente del paciente se obtienen resultados subjetivos (que dependen del punto de vista de cada usuario). Una posibilidad alternativa consiste en aproximar el caminar humano por medio de modelado de robots bípedos, los cuales simulan el comportamiento de prótesis de pierna y de esta forma verificar la marcha.

Este trabajo busca determinar el impacto de las prótesis de pierna en los discapacitados, ya que la calidad y aceptabilidad de dichas prótesis depende de su incidencia en los ciclos de caminado reflejado en los esfuerzos y movi-

mientos realizados por articulaciones restantes (miembros no amputados).

El modelado y simulación de un robot bípedo permite comparar la marcha con la obtenida de un ciclo normal de caminado humano e impulsa el diseño e implantación de futuras prótesis de pierna. Es necesario construir una base de conocimiento propia que permita la utilización de los resultados obtenidos en esta investigación a aplicaciones y diseños futuros propios de nuestro entorno.

Con lo anterior expuesto el trabajo consiste en modelar un robot bípedo y construir un ambiente gráfico en tres dimensiones para el estudio de la incidencia de prótesis de pierna en la marcha humana.

El trabajo se divide en:

- Diseñar un robot bípedo teniendo en cuenta los patrones de movimiento humanos y obtener sus respectivos modelos geométrico, cinemático y dinámico.
- Diseñar e implementar una estrategia de control por par calculado para el robot bípedo.
- Simular el sistema en un ambiente Matlab-Simulink®.
- Construir un ambiente gráfico en tres dimensiones del robot bípedo.
- Estudiar la marcha humana y compararla con la generada por el robot bípedo, con el fin de proponer una prótesis de pierna que proporcione una marcha lo más natural posible.

INTRODUCCIÓN

Los animales en nuestro planeta utilizan extremidades para su locomoción. No existe ningún ser vivo multicelular que utilice ruedas para su movilidad; esto se debe a la dificultad de éstas para adaptarse a terrenos irregulares presentes en nuestro entorno [1]. Los seres humanos son bípedos que poseen cierta destreza comparados con seres de mayor número de miembros inferiores, esto se debe a su habilidad de caminar sobre sus pies, lo que permite la liberación de sus manos para realizar cualquier otra función [2]. Dicha ventaja evolutiva se ve entorpecida por la pérdida parcial o total de extremidades inferiores, que se refleja en la dificultad del individuo para movilizarse en un ambiente que ha sido creado por el ser humano para la locomoción bípeda, además el uso de sillas de ruedas o muletas limita y entorpece el uso de los miembros restantes. Todos los integrantes de una sociedad tienen derecho a la igualdad, esto implica tener las mismas opciones laborales, de educación y desarrollo como ser integral; nos encontramos en un instante en el que para nuestra sociedad es primordial velar por la calidad de vida de todos los seres humanos; se presenta un interés global alrededor de los derechos de las personas con discapacidad. La promulgación del año internacional para las personas con discapacidad en 1981 y la aprobación del programa de acción mundial para los impedidos en 1982 demuestran un trabajo inicial sobre estos temas [3]. Trabajos más recientes en nuestro país como la inclusión de preguntas referentes a discapacidad en el censo del año 2005 demuestran un interés creciente sobre estos temas.

Según este censo el 6.3 % de la población en nuestro país presenta algún tipo de discapacidad o limitación; de los cuales el 29.3 % posee limitaciones para moverse o caminar [4]; estas personas requieren un mejoramiento de su calidad de vida, por medio del desarrollo de prótesis de pierna o algún artefacto similar. En la actualidad estos dispositivos presentan deficiencias debido a su pasividad, ya que obligan a esforzar los miembros restantes de una manera exagerada [5], limitando la movilidad de la persona discapacitada ya que se hace más dificil adaptarse a diferentes ritmos de caminar que exigen los terrenos existentes (escaleras, andenes, terrenos inclinados, etc). Se hace necesario entonces estudiar la estructura y marcha humana con el fin de mejorar el desempeño de las prótesis actuales, haciéndolas activas y adaptables.

Existen diversas formas de estudiar el modo de caminar humano: una de ellas es midiendo los ángulos de las articulaciones y fuerzas de reacción del suelo de la marcha normal de las personas. Otra es diseñando y probando máquinas que caminen de una forma similar a la humana (robots bípedos) que luego se comparen en términos de morfología, modo de andar, uso de energía y control [6].

En Colombia se han realizado varios esfuerzos tendientes a la obtención de una prótesis de pierna funcional. Tal es el caso de un sistema protésico para amputación transfermoral basado en el desarrollo de una rodilla policéntrica que aporta estabilidad y buenos desempeños en amputaciones bilaterales [7]; el diseño y modelamiento de pié para prótesis transfemoral que consiste en un sistema de amortiguación semiactivo a partir de materiales con memoria [8]; o el desarrollo de sistemas inteligentes para el control mecatrónico de prótesis bioeléctricas [9]. En [9] se presentan otros casos de desarrollo de prótesis en Colombia.

Con el fin de tener un punto de comparación es necesario obtener un esquema próximo a la realidad de la marcha humana. Para ello es preciso determinar las posiciones articulares de una manera confiable; una de las técnicas

de medición indirecta es la utilización de sistemas basados en visión artificial [10].

Los resultados de las investigaciones en el desarrollo de robots bípedos impulsa la fabricación de prótesis de pierna y robots ayudantes o de asistencia en general [11].

Todo esto lleva al modelado, control y representación gráfica de un robot bípedo, con el fin de estudiar su marcha y compararla con esquemas de caminado humano, obtenidos con técnicas de medición basadas en visión artificial, como base de estudio para futuras prótesis de pierna.

Capítulo 1

ANTECEDENTES

1.1. Antecedentes de prótesis de pierna

Los seres humanos han sufrido de accidentes, efectos bélicos, malformaciones genéticas, entre otras, que han provocado la pérdida total o parcial de miembros inferiores. Esto ha puesto a prueba el ingenio humano para la solución de dicha discapacidad.

La primera evidencia escrita de prótesis de pierna está da al historiador Heródoto quien cuenta la historia de un soldado persa el cual escapó de su cautiverio cortando parte de su pierna y reemplazándola posteriormente por una prótesis de madera [12]. Durante la época del renacimiento el médico francés Ambrosio Paré inventó las prótesis de extremidades superiores e inferiores que son la base de las prótesis actuales [13]. Otros eventos históricos de relevancia son [14]: en 1696, Pieter Verduyn, un cirujano Holandés introdujo la primera prótesis con rodilla sin bloqueo, en el siglo XIX Dubois Parmlee inventó una prótesis avanzada que tenía un acople de succión al muñón, una rodilla policéntrica y un pie multiarticulado; Gustav Hermann sugirió usar aluminio en reemplazo de acero y Heather Bigg escribió un libro detallando de la ubicación de las articulaciones para un correcto alineamiento.

En el siglo XX y XXI los estudios se han dirigido al desarrollo de prótesis activas que permitan mejor adecuación con el usuario y una mayor recuperación de la movilidad perdida [5], [15], [16].

1.2. Antecedentes de robots bípedos

Los robots bípedos se diferencian de otros robots con piernas en que su estructura presenta un comportamiento inherentemente inestable, sin importar la forma en que han sido diseñados [17]. Ellos requieren un balance activo para moverse ya que su centro de masa se encuentra fuera del área de soporte en un ciclo de caminado [18]. Se ha recorrido mucho camino para llegar a lo que se tiene en la actualidad, es por eso que se procede a mencionar los eventos más significativos en el desarrollo de los robots bípedos.

1.2.1. Máquinas ”Hopping”(saltadoras)

El trabajo realizado por Raibert [18] en la implementación de una máquina saltadora fue de suma importancia, aunque los diseños de robots bipedos no se asemejan en nada con una máquina saltadora de una pierna, los principios de control pierna-cuerpo son los mismos. También se realizaron versiones de máquinas saltadoras de dos piernas. Para éstas se realizaron algoritmos de control muy similares a los utilizados en los anteriores [19].

1.2.2. Robot experimental Harvard

El factor mejorado en este robot fue la suavidad en el caminado. Este problema fue propuesto y resuelto por Dunn y Howe [20]. Ellos propusieron la solución de cambiar la longitud de las piernas para reducir el movimiento vertical del centro de masa; de esta forma se reduce el impulso y se ayuda a reducir

el cambio instantáneo de aceleración cuando el pie cae.

1.2.3. Robot BR-1

En la búsqueda de un mejor control, balance y funcionamiento de los robots bípedos, se analizó la dinámica del robot formulando las ecuaciones cinemáticas y dinámicas del mismo. Un ejemplo de este tipo de trabajo fue el realizado por Shih [21], el cual desarrolló una técnica para mantener el punto de momento cero (ZMP) entre los pies del robot [22], de esta forma se puede realizar un ciclo de caminado más simple. El robot BR-1 se utilizó en este experimento y fue necesario considerar la cinemática directa e inversa de todo el robot para poder tener un ciclo de caminado dinámicamente estable. Las ecuaciones obtenidas fueron simples, se realizó una aproximación de la masa del robot en cuatro masas distribuidas y con esto se obtuvo una estimación del centro de masa. Después se modeló la dinámica del sistema utilizando la forma del lagrangiano, obteniendo las funciones de la energía cinética y potencial en el espacio cartesiano y de las variables manipuladas.

1.2.4. Robot SD-2

La mayoría de los diseños de robots bípedos poseen actuadores en los tobillos. Tener actuadores de par considerable en los tobillos hace que el control del caminado sea mucho más sencillo. Yi y Zheng [23] consideraron el caso en donde el par en los actuadores de los tobillos es pequeño, de esta forma se disminuye la masa de las piernas, reduciendo el momento generado y el consumo de potencia. Su solución consistió en doblar la cadera para ajustar el centro de masa y de esta forma reducir el torque requerido por los tobillos. El robot utilizado para este experimento fue el SD-2.

1.3. Antecedentes de simulación bípeda

En el área de simulación se han efectuado avances significativos. Específicamente en simulación física existe una serie de procedimientos y algoritmos que permiten emular el comportamiento natural de cuerpos rígidos [24]. Existe una serie de librerías y programas libres [25], [26], gratuitos [27] y no gratuitos [28] que permiten tratar el comportamiento físico como una caja negra, en donde se configura cada uno de los cuerpos a simular y la relación entre si. Es de esa forma que se puede simular de una manera cercana a la realidad el comportamiento de cuerpos sujetos a colisiones, ligaduras, pares y fuerzas. En el ámbito de la animación se han realizado avances en la simulación de sistemas bípedos en dos dimensiones [29] y en tres dimensiones [30], [31], en donde se utilizan consignas adquiridas previamente, utilizando técnicas de captura de movimiento [10], para controlar el comportamiento del bípedo.

1.4. Caminado estático y dinámico de un robot bípedo

Balance estático hace referencia a un sistema que permanece balanceado manteniendo siempre su centro de masa verticalmente proyectado sobre el polígono de soporte formado por sus bases de soporte [18]; es por esto que mientras un objeto permanezca en balance estático no se caerá. Tal es el caso del caminado estático de un robot bípedo en el cual cuando un pie se mueve, el centro de masa no debe estar fuera del área de soporte formada por el pie que continúa en contacto con la tierra.

De otro lado el balance dinámico no funciona de esta manera. En este caso la proyección vertical del centro de masa puede encontrarse fuera del área de soporte formada por las bases del robot bípedo en periodos de tiempo. Esto

permite que el sistema presente cortos instantes de caída; sin embargo estos periodos de tiempo deben ser pequeños y bien controlados para que el sistema no se torne inestable [32], [33].

Al comparar los dos métodos de balance, es claro que el método estático es altamente restrictivo y genera movimientos poco eficientes y lentos. Al contrario aunque el método dinámico es mucho más efectivo y rápido, éste presenta un comportamiento inestable, lo que obliga a tener algoritmos de control altamente eficientes y complejos.

1.5. Representación tridimensional de la estructural de un robot bípedo

La configuración básica de un robot bípedo suele tener dos piernas como extremidades bajas, cada una tiene cuatro grados de libertad; tres de éstos son rotoides con su eje de giro paralelo a la base (cadera, rodilla, tobillo) y el cuarto es también rotoide pero con su eje de giro perpendicular a la base. El tronco también tiene dos grados de libertad, sin embargo uno de ellos se comparte con la cadera. Estos dos grados de libertad se utilizan para estabilizar el robot durante el ciclo de caminado. A esta configuración se le denomina como de nueve grados de libertad [34].

Aunque esta es la configuración más utilizada existen algunas variantes en las que se utilizan más o menos grados de libertad. Se destacan entre las más comunes otra de siete grados de libertad [21], en donde se eliminan las articulaciones con eje de rotación perpendicular a la base.

Capítulo 2

MODELADO DEL ROBOT BÍPEDO

La Figura 2.1 muestra la arquitectura general del robot bípedo seleccionado. El modelo del robot fue concebido como una estructura tipo arborescente

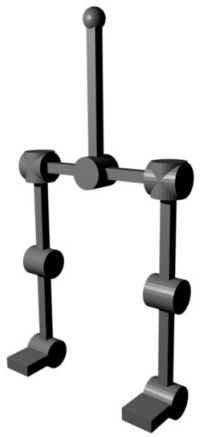


Figura 2.1: Esquema general del robot bípedo

[35] con nueve grados de libertad tipo rotoide como se muestra en la Figura 2.2.

2.1. Modelo geométrico

Como se puede ver en la Figura 2.2:

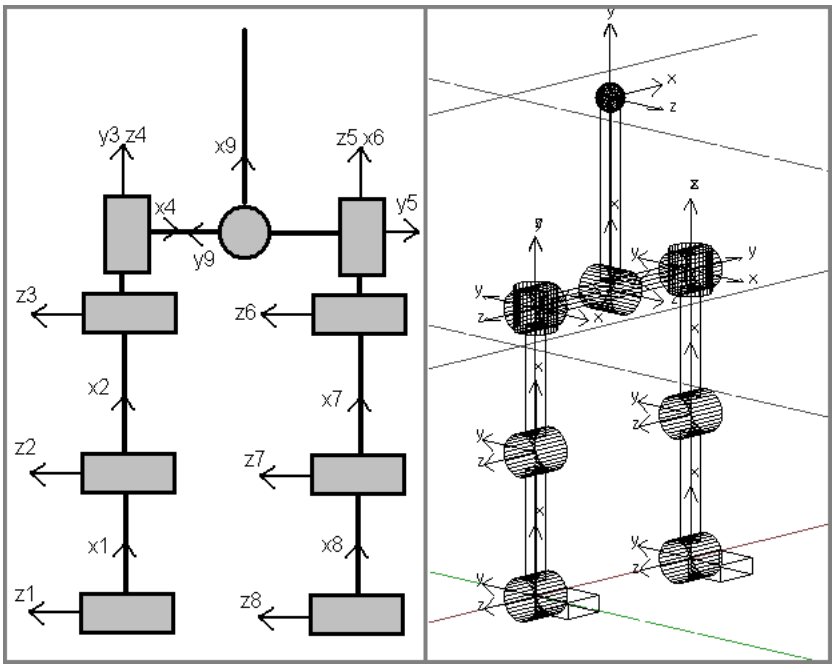


Figura 2.2: Estructura del robot bípedo

- z_j es el eje de la articulación j,
- x_j es perpendicular a z_j y z_{j+1}

La tabla de parámetros geométricos se muestra en el cuadro 2.1, en donde:

ant: articulación anterior

σ_j: tipo de articulación (en este caso rotoide σ=0)

γ_j: ángulo entre ejes de ramificación

b_j: distancia entre los ejes de cada juntura

α_j: ángulo entre ejes z_j

d_j: longitud de cada unión

θ_j: variable articular de cada rotoide

r_j: distancia entre ejes x_j

Cabe anotar que la articulación que antecede a la 9 es la 4, al igual que para la articulación 5, debido a la configuración arborescente que presenta el

j	ant	σ_j	γ_j	b_j	α_j	d_j	θ_j	r_j
1	0	0	0	0	0	0	θ_1	0
2	1	0	0	0	0	D_2	θ_2	0
3	2	0	0	0	0	D_3	$-90+\theta_3$	0
4	3	0	0	0	-90	0	$90+\theta_4$	0
5	4	0	0	0	0	D_5	$-90+\theta_5$	0
6	5	0	0	0	90	0	$90+\theta_6$	0
7	6	0	0	0	0	$-D7$	θ_7	0
8	7	0	0	0	0	$-D_8$	θ_8	0
9	4	0	0	0	90	$D_5/2$	$90+\theta_9$	0

Cuadro 2.1: Parametros geométricos

robot.

Para el modelo utilizado se definieron dimensiones tomadas de bases de datos humanas promediadas [36], medidas en metros:

$D2$ = 0.407

$D3$ = 0.383

$D5$ = 0.35

$D7$ = 0.383

$D8$ = 0.407

La forma general para las matrices de transformación es [37]:

$$^{i}T_j = Rot(x, a_j)Trans(x, d_j)Rot(z, \theta_j)Trans(z, r_j),$$

que de forma matricial se expresa como:

$$ {}^{i}T_j = \begin{pmatrix} cos(\theta_j) & -sin(\theta_j) & 0 & d_j \\ cos(\alpha_j)sin(\theta_j) & cos(\alpha_j)cos(\theta_j) & -sin(\alpha_j) & -r_j sin(\alpha_j) \\ sin(\alpha_j)sin(\theta_j) & sin(\alpha_j)cos(\theta_j) & cos(\alpha_j) & r_j sin(\alpha_j) \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (2.1) $$

Las matrices de transformación teniendo en cuenta la anterior fórmula son:

$$ {}^{0}T_1 = \begin{pmatrix} cos(\theta_1) & -sin(\theta_1) & 0 & 0 \\ sin(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, $$

$$ {}^{1}T_2 = \begin{pmatrix} cos(\theta_2) & -sin(\theta_2) & 0 & D2 \\ sin(\theta_2) & cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, $$

$$ {}^{2}T_3 = \begin{pmatrix} sin(\theta_3) & cos(\theta_3) & 0 & D3 \\ -cos(\theta_3) & sin(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, $$

$$ {}^{3}T_4 = \begin{pmatrix} -sin(\theta_4) & -cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -cos(\theta_4) & sin(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, $$

$$
{}^{4}T_{5} = \begin{pmatrix} sin(\theta_5) & cos(\theta_5) & 0 & D5 \\ -cos(\theta_5) & sin(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^{5}T_{6} = \begin{pmatrix} -sin(\theta_6) & -cos(\theta_6) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ cos(\theta_6) & -sin(\theta_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^{6}T_{7} = \begin{pmatrix} cos(\theta_7) & -sin(\theta_7) & 0 & -D7 \\ sin(\theta_7) & cos(\theta_7) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^{7}T_{8} = \begin{pmatrix} cos(\theta_8) & -sin(\theta_8) & 0 & -D8 \\ sin(\theta_8) & cos(\theta_8) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^{4}T_{9} = \begin{pmatrix} -sin(\theta_9) & -cos(\theta_9) & 0 & 1/2 * D5 \\ 0 & 0 & -1 & 0 \\ cos(\theta_9) & -sin(\theta_9) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
$$

De lo anterior se puede deducir que para ${}^{0}T_{8}$ (Modelo geométrico directo para una pierna o rama):

$${}^0T_811 = cos(\theta_1+\theta_2+\theta_3)*cos(\theta_6+\theta_7+\theta_8) - cos(\theta_4+\theta_5)*sin(\theta_1+\theta_2+\theta_3)*sin(\theta_6+\theta_7+\theta_8)$$

$${}^0T_821 = cos(\theta_6+\theta_7+\theta_8)*sin(\theta_1+\theta_2+\theta_3) + cos(\theta_1+\theta_2+\theta_3)*cos(\theta_4+\theta_5)*sin(\theta_6+\theta_7+\theta_8)$$

$${}^0T_831 = sin(\theta_4+\theta_5)*sin(\theta_6+\theta_7+\theta_8)$$

$${}^0T_812 = -(cos(\theta_4+\theta_5)*cos(\theta_6+\theta_7+\theta_8)*sin(\theta_1+\theta_2+\theta_3)) - cos(\theta_1+\theta_2+\theta_3)*sin(\theta_6+\theta_7+\theta_8)$$

$${}^0T_822 = cos(\theta_1+\theta_2+\theta_3)*cos(\theta_4+\theta_5)*cos(\theta_6+\theta_7+\theta_8) - sin(\theta_1+\theta_2+\theta_3)*sin(\theta_6+\theta_7+\theta_8)$$

$${}^0T_832 = cos(\theta_6+\theta_7+\theta_8)*sin(\theta_4+\theta_5)$$

$${}^0T_813 = sin(\theta_1+\theta_2+\theta_3)*sin(\theta_4+\theta_5)$$

$${}^0T_823 = -(cos(\theta_1+\theta_2+\theta_3)*sin(\theta_4+\theta_5))$$

$${}^0T_833 = cos(\theta_4+\theta_5)$$

$${}^0T_814 = D2*cos(\theta_1) + D3*cos(\theta_1+\theta_2) - D7*cos(\theta_1+\theta_2+\theta_3)*cos(\theta_6) - D8*cos(\theta_1+\theta_2+\theta_3)*cos(\theta_6+\theta_7) - D5*sin(\theta_1+\theta_2+\theta_3)*sin(\theta_4) + D7*cos(\theta_4+\theta_5)*sin(\theta_1+\theta_2+\theta_3)*sin(\theta_6) + D8*cos(\theta_4+\theta_5)*sin(\theta_1+\theta_2+\theta_3)*sin(\theta_6+\theta_7)$$

$${}^0T_824 = D2*sin(\theta_1) + D3*sin(\theta_1+\theta_2) - D7*cos(\theta_6)*sin(\theta_1+\theta_2+\theta_3) - D8*cos(\theta_6+\theta_7)*sin(\theta_1+\theta_2+\theta_3) + D5*cos(\theta_1+\theta_2+$$

$\theta_3) * sin(\theta_4) - D7 * cos(\theta_1 + \theta_2 + \theta_3) * cos(\theta_4 + \theta_5) * sin(\theta_6) - D8 * cos(\theta_1 + \theta_2 + \theta_3) * cos(\theta_4 + \theta_5) * sin(\theta_6 + \theta_7)$

$^0T_834 = -(D5 * cos(\theta_4)) - D7 * sin(\theta_4 + \theta_5) * sin(\theta_6) - D8 * sin(\theta_4 + \theta_5) * sin(\theta_6 + \theta_7)$

Y para $^0T_9$ (Modelo geométrico directo para la otra rama):

$^0T_911 = cos(\theta_1+\theta_2+\theta_3)*cos(\theta_9)+sin(\theta_1+\theta_2+\theta_3)*sin(\theta_4)*sin(\theta_9)$

$^0T_921 = cos(\theta_9)*sin(\theta_1+\theta_2+\theta_3)-cos(\theta_1+\theta_2+\theta_3)*sin(\theta_4)*sin(\theta_9)$

$^0T_931 = cos(\theta_4) * sin(\theta_9)$

$^0T_912 = cos(\theta_9)*sin(\theta_1+\theta_2+\theta_3)*sin(\theta_4)-cos(\theta_1+\theta_2+\theta_3)*sin(\theta_9)$

$^0T_922 = -(cos(\theta_1+\theta_2+\theta_3)*cos(\theta_9)*sin(\theta_4))-sin(\theta_1+\theta_2+\theta_3)*sin(\theta_9)$

$^0T_932 = cos(\theta_4) * cos(\theta_9)$

$^0T_913 = cos(\theta_4) * sin(\theta_1 + \theta_2 + \theta_3)$

$^0T_923 = -(cos(\theta_1 + \theta_2 + \theta_3) * cos(\theta_4))$

$^0T_933 = -sin(\theta_4)$

$^0T_914 = D2 * cos(\theta_1) + D3 * cos(\theta_1 + \theta_2) - (D5 * sin(\theta_1 + \theta_2 + \theta_3) * sin(\theta_4))/2$

$${}^0T_924 = D2 * sin(\theta_1) + D3 * sin(\theta_1 + \theta_2) + (D5 * cos(\theta_1 + \theta_2 + \theta_3) * sin(\theta_4))/2$$

$${}^0T_934 = -(D5 * cos(\theta_4))/2$$

2.2. Modelo dinámico

La mayoría de los robots bípedos presentan configuraciones generales similares entre ellos. Un robot desplazado por piernas se modela en dos fases, para cada ciclo de caminado (un paso), que se repiten para conformar la marcha del robot [38], [39], [40]:

A. Como un manipulador libre el cual no tiene un punto de amarre pero tiene interacción con la tierra.

B. Como un modelo de impacto impulsivo en el momento de contacto.

A continuación se explica cada una de las fases:

2.2.1. Manipulador libre

Un manipulador libre se modela introduciendo las variables que representan la posición y la orientación con respecto a la base [18]. Se consideran las coordenadas generalizadas q, las velocidades generalizadas $\dot{q}$, las aceleraciones generalizadas $\ddot{q}$ y las fuerzas generalizadas Γ [35]. El modelo dinámico

del robot modelado como manipulador libre es representado por las siguientes ecuaciones basadas en el método del lagrangiano:

$$A_e\ddot{q}_e + C_e\dot{q}_e + Q_e = B_e\Gamma + {J_R}^T R, \tag{2.2}$$

donde:

$q_e := [q^T\ x_t\ y_t\ z_t]^T$ vector formado por las variables articulares y las coordenadas cartesianas del centro de masa

A_e: matriz de inercia

C_e: matriz de fuerzas centrífugas y Coriolis

Q_e: vector de gravedad

B_e: matriz de selección de par

J_R : matriz jacobiana que convierte las fuerzas externas en pares articulares

Γ: par de los actuadores

R: vector que representa las fuerzas externas actuando en el punto de contacto

Se puede asumir que la pierna de apoyo actúa como pivote y que la pierna de avance no se desliza ni rebota, lo que hace que el vector que representa las fuerzas externas actuando en el punto de contacto R sea despreciable y que ya no se consideren las coordenadas cartesianas del centro de masa; por lo que (2.2) se puede escribir como:

$$A\ddot{q} + C\dot{q} + Q = B\Gamma, \tag{2.3}$$

donde:

A: matriz de inercia

C: matriz de fuerzas centrífugas y Coriolis

Q: vector de gravedad
B: matriz de selección de par

La ecuación (2.3) se puede escribir en forma de espacio de estados no lineales:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} A^{-1}(-C\dot{q} - Q + B\Gamma) \\ \dot{q} \end{bmatrix} \\ &= f(x) + g(x)\Gamma, \end{aligned} \tag{2.4}$$

donde x es el vector de variables de estado formado por las velocidades y posiciones articulares ($x = [\dot{q}^T, q^T]^T$), $f(\cdot)$ y $g(\cdot)$ son funciones no lineales.

2.2.2. Modelo de impacto

El modelo de impacto es aplicable en el instante en que la pierna de avance toca el suelo (impacto), y permite reiniciar el sistema para un siguiente ciclo de caminado. Para el modelo de impacto propuesto las siguientes suposiciones son hechas:

- Se asume que el impacto es instantáneo.
- Las fuerzas impulsivas generan discontinuidad en las velocidades, pero no en las posiciones.
- El contacto de la pierna de avance con el suelo no produce ni rebotes ni deslizamientos de dicha pierna.
- La pierna de apoyo abandona el suelo sin interacciones.

De esta forma el momento de doble apoyo es instantáneo y el problema de actualización de las variables de estado se reduce a encontrar las velocidades angulares $\dot{q}^+$, en el instante posterior al impacto, ya que las posiciones articulares q no varían ($q^- = q^+$). La ecuación (2.2) puede ser utilizada para este fin partiendo de las velocidades anteriores al impacto $\dot{q}^-$. Con esto se puede reiniciar el sistema para un siguiente ciclo de caminado, con los nuevos valores iniciales de las variables de estado x^+ formados por $\dot{q}^+$ como velocidades iniciales y q^- como las posiciones iniciales con un cambio de coordenadas [1]. Una manera de representar matemáticamente lo anterior es por medio de una ecuación de cambio:

$$x^+ = \Delta(x^-), \tag{2.5}$$

donde:

$\Delta(\cdot)$: es una función de cambio.

Las dos fases anteriores representan la forma de modelar un ciclo de caminado completo (un paso), el modelo de manipulador libre se utiliza en los instantes en donde la pierna de avance se encuentra levantada y el modelo de impacto únicamente cuando la pierna de avance interactúa con el suelo (instante impulsivo) y se convierte en la pierna de apoyo. Es por esto que la fase correspondiente al modelo de impacto es simplemente un proceso de actualización de variables de estado que se encarga de reiniciar el sistema y de esta forma empezar un nuevo ciclo de caminado. Por lo anterior la unión de las dos fases que conforman el ciclo de caminado se presenta a continuación.

[1] El cambio de coordenadas es necesario debido a que la pierna de avance después del impacto se convierte en la pierna de apoyo y viceversa

2.2.3. Modelo no lineal para toda la marcha del robot bípedo

Las ecuaciones (2.4) y (2.5) permiten representar la marcha del robot bípedo, por lo que el modelo total puede ser expresado de la siguiente forma:

$$\begin{aligned} \dot{x} = &\quad f(x) + g(x)\Gamma, \quad x^- \notin S, \\ x^+ = &\quad \Delta(x^-), \qquad x^- \in S, \end{aligned} \tag{2.6}$$

donde S es el conjunto de instantes donde la pierna de avance toca el suelo.

La ecuación (2.6) está compuesta por dos expresiones las cuales no actúan de manera simultánea, son seleccionadas dependiendo del estado de la pierna de avance, la primera para todos los instantes en los cuales la pierna de avance *no toca el suelo* y la segunda para los instantes impulsivos en que la pierna de avance *toca el suelo* convirtiéndose en la pierna de apoyo. Por lo anterior cada vez que el robot avanza un paso la primera parte del modelo es utilizada, hasta que la pierna de avance se convierte en la de apoyo, lo que hace que la segunda parte del modelo se utilice para reiniciar el sistema y de esta forma proseguir con un siguiente paso, volviendo a la primera parte del modelo. Este proceso se repite de manera cíclica en toda la marcha.

2.2.4. Modelo dinámico directo para el robot propuesto

Con el fin de poder realizar la simulación de la primera parte de la ecuación (2.6), se debe utilizar el modelo dinámico directo general de un robot manipulador, este es:

$$\ddot{q} = A^{-1}(-C\dot{q} - Q + B\Gamma) \tag{2.7}$$

Como se explicó anteriormente para obtener el modelo dinámico es necesario partir de los parámetros dinámicos del robot y las fuerzas externas. Los parámetros dinámicos de un robot son once, seis elementos del tensor de inercia $(XX_j, XY_j, XZ_j, YY_j, YZ_j, ZZ_j)$, tres del primer momento de inercia

(MX_j, MY_j, MZ_j), uno de la masa (M_j) y uno de la inercia del accionador (Ia_j). En el caso del robot bípedo propuesto se tienen parámetros dinámicos para nueve junturas como se muestra a continuación:

	Parámetro										
j	**XX**	**XY**	**XZ**	**YY**	**YZ**	**ZZ**	**MX**	**MY**	**MZ**	**M**	**Ia**
1	XX1	XY1	XZ1	YY1	YZ1	ZZ1	MX1	MY1	MZ1	M1	IA1
2	XX2	XY2	XZ2	YY2	YZ2	ZZ2	MX2	MY2	MZ2	M2	IA2
3	XX3	XY3	XZ3	YY3	YZ3	ZZ3	MX3	MY3	MZ3	M3	IA3
4	XX4	XY4	XZ4	YY4	YZ4	ZZ4	MX4	MY4	MZ4	M4	IA4
5	XX5	XY5	XZ5	YY5	YZ5	ZZ5	MX5	MY5	MZ5	M5	IA5
6	XX6	XY6	XZ6	YY6	YZ6	ZZ6	MX6	MY6	MZ6	M6	IA6
7	XX7	XY7	XZ7	YY7	YZ7	ZZ7	MX7	MY7	MZ7	M7	IA7
8	XX8	XY8	XZ8	YY8	YZ8	ZZ8	MX8	MY8	MZ8	M8	IA8
9	XX9	XY9	XZ9	YY9	YZ9	ZZ9	MX9	MY9	MZ9	M9	IA9

Cuadro 2.2: Parámetros dinámicos

Partiendo de medidas tomadas de [36] se tiene:

	Parámetro										
j	**XX**	**XY**	**XZ**	**YY**	**YZ**	**ZZ**	**MX**	**MY**	**MZ**	**M**	**Ia**
1	0.0022	0	0	0.1535	0	0.1535	0.5617	0	0	2.76	0
2	0.0123	0	0	0.3416	0	0.3416	1.3137	0	0	6.86	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0.0019	0	0	0.0977	0	0.0977	0.4147	0	0	2.37	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0.0123	0	0	0.3416	0	0.3416	1.3137	0	0	6.86	0
7	0.0022	0	0	0.1535	0	0.1535	0.5617	0	0	2.76	0
8	0.0009	0	0	0.0198	0	0.0198	0.132	0	0	1.2	0
9	0.0581	0	0	9.9448	0	9.9448	18.592	0	0	46.48	0

Cuadro 2.3: Parámetros dinámicos numéricos

Debido a la complejidad de los cálculos en el modelo de un robot cualquiera, un paso casi necesario es la simplificación de operaciones con el uso

de los llamados *parámetros dinámicos de base* del robot [35]. Los parámetros reducidos se muestran a continuación:

	Parámetro										
j	**XX**	**XY**	**XZ**	**YY**	**YZ**	**ZZ**	**MX**	**MY**	**MZ**	**M**	**Ia**
1	0	0	0	0	0	ZZ1R	MX1R	MY1	0	0	0
2	0	0	0	0	0	ZZ2R	MX2R	MY2	0	0	IA2
3	0	0	0	0	0	ZZ3R	MX3	MY3R	0	0	IA3
4	XX4R	XY4R	XZ4R	0	YZ4	ZZ4R	MX4R	MY4R	0	0	IA4
5	XX5R	XY5	XZ5	0	YZ5	ZZ5R	MX5	MY5R	0	0	IA5
6	XX6R	XY6	XZ6R	0	YZ6	ZZ6R	MX6R	MY6	0	0	IA6
7	XX7R	XY7	XZ7R	0	YZ7	ZZ7R	MX7R	MY7	0	0	IA7
8	XX8R	XY8	XZ8	0	YZ8	ZZ8	MX8	MY8	0	0	IA8
9	XX9R	XY9	XZ9	0	YZ9	ZZ9	MX9	MY9	0	0	IA9

Cuadro 2.4: Parámetros dinámicos de base

Donde:

ZZ1R = IA1 + D2^2*(M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9) + ZZ1

MX1R = D2*(M2 + M3 + M4 + M5 + M6 + M7 + M8 + M9) + MX1

ZZ2R = D3^2*(M3 + M4 + M5 + M6 + M7 + M8 + M9) + ZZ2

MX2R = D3*(M3 + M4 + M5 + M6 + M7 + M8 + M9) + MX2

ZZ3R = D5^2*(M5 + M6 + M7 + M8) + (D5^2*M9)/4 + YY4 + YY5 + ZZ3

MY3R = MY3 + MZ4 + MZ5

XX4R = -(D5^2*(M5 + M6 + M7 + M8)) - (D5^2*M9)/4 + XX4 - YY4 + YY9

XY4R = (D5*MZ9)/2 + XY4

XZ4R = -(D5*MZ5) + XZ4

ZZ4R = D5^2*(M5 + M6 + M7 + M8) + (D5^2*M9)/4 + YY9 + ZZ4

MX4R = D5*(M5 + M6 + M7 + M8) + (D5*M9)/2 + MX4

MY4R = MY4 - MZ9

XX5R = D8^2*M8 + D7^2*(M7 + M8) + XX5 - YY5 + YY6 + YY7 + YY8

ZZ5R = D8^2*M8 + D7^2*(M7 + M8) + YY6 + YY7 + YY8 + ZZ5

MY5R = MY5 - MZ6 - MZ7 - MZ8

XX6R = -(D7^2*(M7 + M8)) + XX6 - YY6

XZ6R = D7*(MZ7 + MZ8) + XZ6

ZZ6R = D7^2*(M7 + M8) + ZZ6

MX6R = -(D7*(M7 + M8)) + MX6

XX7R = -(D8^2*M8) + XX7 - YY7

XZ7R = D8*MZ8 + XZ7

ZZ7R = D8^2*M8 + ZZ7

MX7R = -(D8*M8) + MX7

XX8R = XX8 - YY8

XX9R = XX9 - YY9

Con lo que se obtienen los siguientes parámetros de base numéricos:

	Parámetro										
j	**XX**	**XY**	**XZ**	**YY**	**YZ**	**ZZ**	**MX**	**MY**	**MZ**	**M**	**Ia**
1	0	0	0	0	0	11.1741	27.6394	0	0	0	0
2	0	0	0	0	0	9.0945	24.1673	0	0	0	0
3	0	0	0	0	0	2.8466	0	0	0	0	0
4	7.1001	0	0	0	0	12.7914	12.3357	0	0	0	0
5	1.2946	0	0	0	0	1.2946	0	0	0	0	0
6	-0.9101	0	0	0	0	0.9225	-0.2030	0	0	0	0
7	-0.3501	0	0	0	0	0.3523	0.0733	0	0	0	0
8	-0.0189	0	0	0	0	0.0198	0.1320	0	0	0	0
9	-9.8867	0	0	0	0	9.9448	18.592	0	0	0	0

Cuadro 2.5: Parámetros dinámicos de base numéricos

Con estos parámetros fue posible obtener el modelo dinámico del robot por medio de la herramienta SYMORO [41].

2.2.5. Modelo dinámico inverso para el robot propuesto

El modelo dinámico inverso obtiene el par a partir de las variables articulares:

$$\Gamma = A\ddot{q} + C\dot{q} + Q, \tag{2.8}$$

en donde se ha supuesto la matriz B de selección de par como la identidad. Es utilizado en el control por par calculado con el fin de linealizar el sistema. Las ecuaciones que representan el modelo también fueron generadas por medio de la herramienta SYMORO.

Capítulo 3

CONTROL DEL ROBOT

Para el control del robot bípedo se utilizó el algoritmo de control por par calculado [35],[42]. Este tipo de control asegura el desacople y la linealización de las ecuaciones del modelo no lineal del robot. Este método exige el cálculo del modelo dinámico inverso en línea a partir de los valores numéricos de los parámetros inerciales y sus correspondientes parámetros de base.

3.1. Control por par calculado

La idea central del control por par calculado es transformar la dinámica no-lineal del robot a una lineal que permita que técnicas lineales sean aplicadas [43]. En este trabajo la técnica lineal aplicada es el control proporcional derivativo o PD.

Para la implementación de este tipo de control se consideró un sistema no lineal descrito por $x_n = f(x) + b(x)u$, con x como la salida, y u como la entrada, el vector $x = [x, \dot{x}....x^{n-1}]^T$ es el vector de estado y $f(x)$ una función no lineal. Se puede apreciar que no existen derivadas de la entrada y el sistema puede describirse en la forma compañera:

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ . \\ . \\ . \\ x^n \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ . \\ . \\ . \\ f(x) + b(x)u \end{pmatrix}, \tag{3.1}$$

Si se define un sistema inverso:

$$u = \frac{1}{b(x)}(\omega - f(x)), \tag{3.2}$$

con ω como una nueva entrada equivalente, se tiene:

$$x^n = f(x) + b(x)u = f(x) + b(x)\frac{1}{b(x)}(\omega - f(x)), \tag{3.3}$$

y la $n - esima$ derivada de x es:

$$x^n = \omega, \tag{3.4}$$

con lo cual se tiene un sistema equivalente linealizado al cual se puede aplicar una técnica de control lineal.

Para el robot bípedo (Ecuación (2.3)) se tiene que la n-ésima derivada es (con $n = 2$):

$$\ddot{q} = A^{-1}(\Gamma - Q) = A^{-1}\Gamma - A^{-1}Q, \tag{3.5}$$

donde: $\ddot{q} = x^n$, $\Gamma = u$, $A^{-1} = b(x)$ y $-A^{-1}Q = f(x)$, teniendo en cuenta que la matriz de selección de par B es la identidad y que las fuerzas centrífugas y Coriolis son despreciables $C = 0$.
Aplicando lo anterior al robot se tiene:

$$\ddot{q} = \omega, \tag{3.6}$$

donde ω es el nuevo vector de control y $\ddot{q}$ es la aceleración articular.

Por otro lado el sistema inverso de la ecuación (3.2) se puede escribir como:

$$\tau = A(\omega - (-A^{-1}Q)), \tag{3.7}$$

que genera:

$$\tau = A\omega + Q, \tag{3.8}$$

que al hacer la sustitución de $\ddot{q} = \omega$ es simplemente la ecuación del modelo inverso del robot expuesta anteriormente.

Debido a que este sistema (ecuación (3.8)) busca linealizar el comportamiento del robot a partir de suponer sus parámetros dinámicos es más adecuado hablar de funciones estimadas, por lo que se debe reescribir de la siguiente forma:

$$\tau = \hat{A}\omega + \hat{Q}, \tag{3.9}$$

con:

$\hat{A}$: estimado de la matriz de inercia
$\hat{Q}$: estimado del vector de gravedad

Al aplicar una técnica de control lineal como la PD es posible escribir $\omega(t)$ como:

$$\omega = K_p(q^d - q) - K_v\dot{q}, \tag{3.10}$$

donde q^d es la posición articular deseada, q y $\dot{q}$ son la posición y velocidad articulares medidas y K_p y K_v son las ganancias proporcional y derivativa respectivamente.

El esquema del control por par calculado es indicado por (3.10) y se muestra en la figura 3.1.

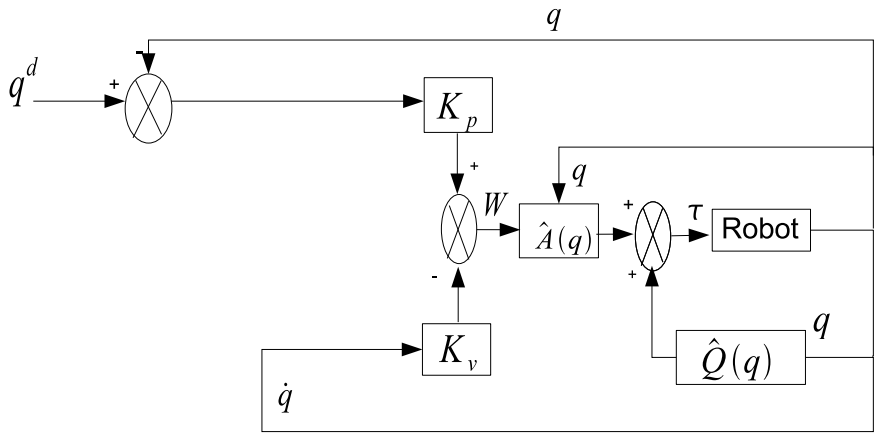


Figura 3.1: Control por par calculado

Su implementación en Matlab-Simulink® se muestra en la figura 3.2.

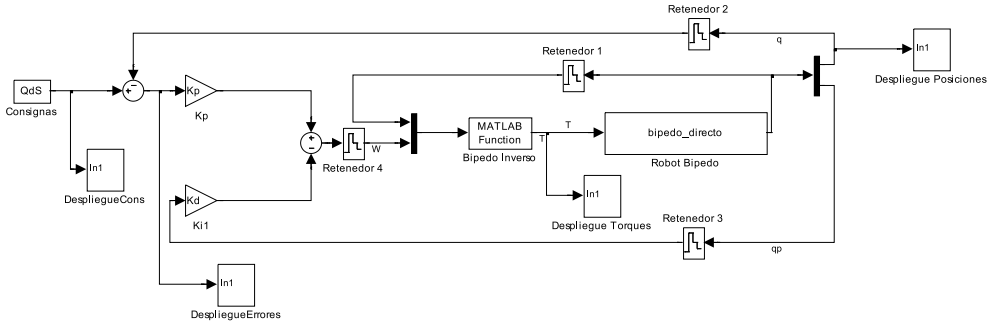


Figura 3.2: Control por par calculado Matlab-Simulink®

Capítulo 4

AMBIENTE GRÁFICO DE SIMULACIÓN 3D

Para realizar el ambiente gráfico de simulación 3D es necesario considerar las características propias del sistema a desarrollar. La aplicación debe permitir modificar de una manera rápida las condiciones de simulación y de construcción del robot, ya que la arquitectura de éste puede variar en el proceso de investigación. Teniendo en cuenta las consideraciones expuestas anteriormente trabajar con un lenguaje de programación basado en *scripts* fue considerada como la mejor opción. Después de analizar las diferentes posibilidades de desarrollo se optó por un lenguaje basado en *scripts* denominado Python [44], debido a que permite modificar lineas de código sin la necesidad de compilar y genera archivos binarios intermedios; esto permite que la velocidad de ejecución se vea afectada en poca medida.

4.1. Simulación física

Los sistemas dinámicos pueden ser modelados utilizando ecuaciones diferenciales que describan la relación entre una función desconocida y sus de-

rivadas. Resolver una ecuación diferencial permite encontrar una función que cumpla con la relación expuesta.
Un sistema se puede expresar de forma canónica [45] como una ecuación diferencial ordinaria (ODE por sus siglas en inglés):

$$\dot{x} = f(x, t), \tag{4.1}$$

donde f es una función conocida y x es el estado del sistema y $\dot{x}$ es la derivada de los estados en el tiempo.

Para obtener una simulación de la ecuación (4.1) se debe partir de un valor inicial $x(t_0) = x_0$ y generar el estado x para cualquier instante de tiempo mayor a t_0.

Una forma de deducir el resultado de la ecuación (4.1) es reducir el problema a pequeños intervalos de tiempo en donde se soluciona el valor de $\dot{x}$ y se integra el resultado. Este método presenta una serie de inestabilidades debido a la discretización del tiempo, pero genera resultados favorables para intervalos de tiempo pequeños [24]. Existen una serie de librerías software que se encargan de realizar este tipo de simulación permitiendo trabajar con sistemas medianamente complejos.

Con el fin de efectuar la simulación física del robot bípedo se optó por trabajar con la librería *Open Dynamics Engine* [25] (OpenDE) la cual es de distribución libre y abierta lo que permite su modificación y verificación. OpenDE está diseñada para efectuar simulación física de objetos rígidos expuestos a fuerzas, pares, ligaduras y colisiones [1].

En el caso particular del robot bípedo se puede decir que es un sistema que contiene un conjunto de objetos rígidos (junturas) expuestos a diferentes tipos de ligaduras:

- Colisiones y contactos con la superficie del suelo

[1] Se pueden considerar las colisiones como un caso especial de ligaduras

- Articulaciones rotoides motorizadas y no motorizadas

Las junturas fueron realizadas a partir de cilindros de las dimensiones y condiciones físicas descritas anteriormente (Cuadro 2.3) como se puede observar en la figura 4.1

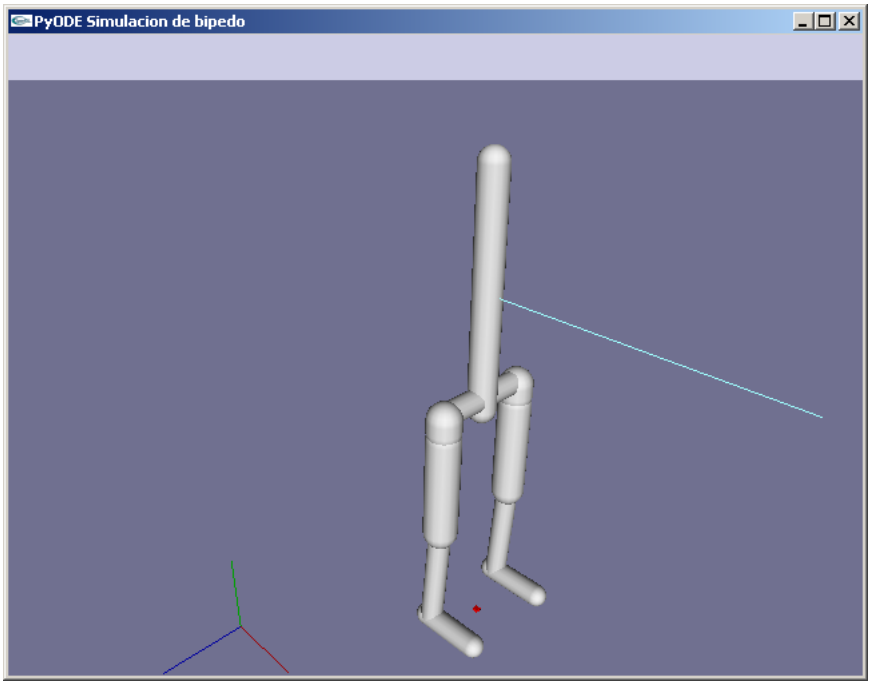


Figura 4.1: Ambiente gráfico de simulación 3D

4.2. Modelado del robot para el ambiente de simulación

La forma de modelar un robot bípedo fue expuesta en la sección 2.2, pero para poder simular un robot bípedo que interactúa con su ambiente (colisiones con la superficie del suelo y con objetos en general) es necesario partir de su expresión más básica y con ella generar ligaduras que permitan simular condiciones cercanas a las reales en el entorno del robot. Es así que se debe

retomar la ecuación general de un robot (ecuación (2.2)) y renombrarla como:

$$A_e\ddot{q}_e + C_e\dot{q}_e + Q_e = B_e\Gamma + J_R^T R, \tag{4.2}$$

en este caso se toma $q_e = [q^T u_t\ v_t\ x_t\ y_t\ z_t]^T$ como el vector formado por las variables articulares q^T, la orientación u_t y v_t y las coordenadas cartesianas del centro de masa del robot $x_t\ y_t\ z_t$. Es relevante aclarar que el escalar u_t y el vector de tres dimensiones v_t conforman el cuaternión unitario que representa la orientación total del robot [2].

En el caso de la matriz B_e de selección de par mostrada en la ecuación (4.3) se puede decir que está formada por unos y ceros que permiten representar la acción de cada uno de los pares aplicados al robot, que en este caso en particular está formada por una serie de unos en diagonal (para seleccionar una variable articular por par aplicado) y solo ceros en las últimas siete columnas con el fin de representar la *no influencia* del par aplicado en la orientación y posición del robot (u_t, v_t, x_t, y_t, z_t)

$$B_e = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix}. \tag{4.3}$$

Por otro lado el conjunto $J_R^T R$ representa el par generado por las fuerzas externas (como contactos con la tierra y/o interacciones con objetos externos), donde J_R^T representa la traspuesta de la matriz Jacobiana de los puntos de contacto de las fuerzas externas:

$$J_R = \frac{\partial \phi(q)^T}{\partial q}, \tag{4.4}$$

$\phi(q)$ que representa las coordenadas cartesianas de los puntos de aplicación de las fuerzas externas a partir de las coordenadas articulares del robot q.

[2] u_t y v_t son normalmente tomados con respecto al torso del robot

R: es un vector que representa la magnitud de las fuerzas cartesianas aplicadas.

En los bípedos las condiciones cambian muy seguido debido a la interacción del robot con el piso, que genera una serie de fuerzas actuando en el pie de contacto (y algunas veces en el pie de avance). Estas fuerzas se pueden dividir en:

$$J_R^T R = J_{Rn}(q)^T R_n(q, \dot{q}) + J_{Rt}(q)^T R_t(q, \dot{q}) \tag{4.5}$$

Se puede asumir que no hay penetración en el punto de contacto, lo que genera fuerzas normales y aceleraciones semipositivas [46]:

$$R_n^T(q, \dot{q})\ddot{\phi}_n(q) = 0,\ R_n(q, \dot{q}) \geq 0,\ \ddot{\phi}_n(q) \geq 0, \tag{4.6}$$

suponiendo que no hay deslizamiento se tiene:

$$\ddot{\phi}_t(q) = 0. \tag{4.7}$$

Cuando el bípedo camina interactúa con la tierra generando instantes discontinuos (sección 2.2.2) llamados momentos de impacto en donde la velocidad cambia instantáneamente, partiendo de la ecuación (4.2) se puede decir:

$$A_e(q)(\dot{q}^+ - \dot{q}^-) = J_{Rn}(q)^T R_{nIMPACTO}(q, \dot{q}) + J_{Rt}(q)^T R_{tIMPACTO}(q, \dot{q}), \tag{4.8}$$

Con $R_{nIMPACTO}$ y $R_{tIMPACTO}$ las fuerzas impulsivas actuando en el momento del impacto.

Suponiendo que el pie de contacto nunca se desliza ni rebota, se tiene:

$$\dot{\phi}_n(q) = J_{Rn}(q)\dot{q}^+ = 0, \tag{4.9}$$

y

$$\dot{\phi}_t(q) = J_{Rt}(q)\dot{q}^+ = 0, \tag{4.10}$$

por lo cual se puede deducir que el robot bípedo en su forma más general cumple con la siguiente grupo de ecuaciones:

$$continuo \left\{ \begin{array}{c} A_e\ddot{q}_e + C_e\dot{q}_e + Q_e = B_e\Gamma + J_R^T R, \\ J_{Rn}(q)\ddot{q} + \dot{J}_{Rn}(q)\dot{q} \geq 0, \\ J_{Rt}(q)\ddot{q} + \dot{J}_{Rt}(q)\dot{q} = 0, \\ R_n(q,\dot{q}) \geq 0, \\ R_n^T(q,\dot{q})(J_{Rn}(q)\ddot{q} + \dot{J}_{Rn}(q)\dot{q}) = 0 \end{array} \right\}$$

$$impacto \left\{ \begin{array}{c} \dot{q}^+ = \dot{q}^- + A_e^{-1} J_R^T R_{IMPACTO} \\ R_{IMPACTO} = -(J_R(q)A_e(q)^{-1}J_R^T(q))^{-1} J_R(q)\dot{q}^- \end{array} \right\} \quad (4.11)$$

4.3. Control del robot para el ambiente de simulación

En el caso del control del robot se utilizó el control propuesto en el capítulo 3. La implementación consistió en generar una clase en Python la cual contiene el control proporcional-derivativo, los módulos de suma y multiplicación y el módulo del modelo dinámico inverso que fue tomado del código generado por la aplicación Symoro.

4.4. Generación de consignas articulares

Las consignas articulares se propusieron como resultado de la captura de movimiento humano real utilizando técnicas de visión artificial, sin embargo para tener resultados confiables es necesario el uso de equipos de visión de alta calidad que son sumamente costosos, con el objetivo de contar con señales reales se optó por utilizar muestras adquiridas en laboratorios especializados.

Para tal fin se decidió recurrir a una base de datos libre dedicada a la captura de movimientos humanos [47], en esta base de datos las capturas están clasificadas por sujetos, tipo de movimiento y tipo de archivo deseado. Para este desarrollo en específico se utilizaron archivos de captura en formato ASF/ASM [48].

4.4.1. Transformación de datos

Los datos usados son el resultado de la captura de marcha de un sujeto real e involucra 56 grados de libertad, basándose en la simplicidad de la estructura de robot propuesta, se hace necesario un procedimiento de transformación para poder convertir los datos provenientes de los 56 grados de libertad a tan sólo 9 grados de libertad.
El procedimiento para la conversión consiste en:

- Realizar un remuestreo de los datos con el fin de generar nuevas consignas a la frecuencia de muestreo deseada
- Partir de las muestras generadas en el punto anterior y construir unas nuevas que dependan directamente de la configuración geométrica específica del robot

Para la primera tarea se puede trabajar con técnicas de remuestreo basadas en *splines* o *bsplines* de jerarquía [49], por otro lado con el fin de generar las consignas para los 9 grados de libertad se pueden utilizar mapas de desplazamiento acompañados de cinemática inversa sujeta a ligaduras [29], [49]. En el caso particular de este proyecto se trabajó con esta técnica para contar con consignas articulares en 2D y 3D.

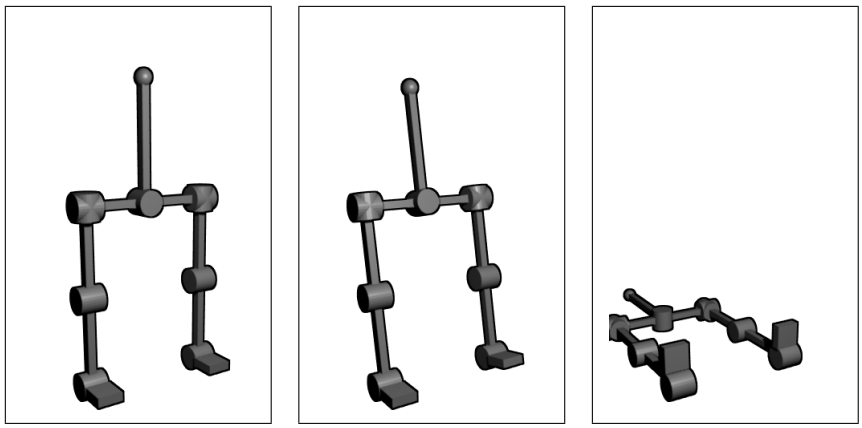
4.5. Balance del robot

El robot bípedo expuesto en la sección 2.2 es capaz de seguir fielmente consignas articulares con un error considerablemente pequeño; pero al efectuar su simulación en un ambiente físico como OpenDE éste simplemente cae después de un pequeño instante de tiempo debido a que el modelo no contiene información de su orientación y las consignas articulares son adquiridas de personas que no tienen las mismas características físicas del robot.
El problema consiste en procurar mantener la consigna articular, pero modificarla un poco o por completo para evitar caer, para lo cual se consideran los siguientes casos:

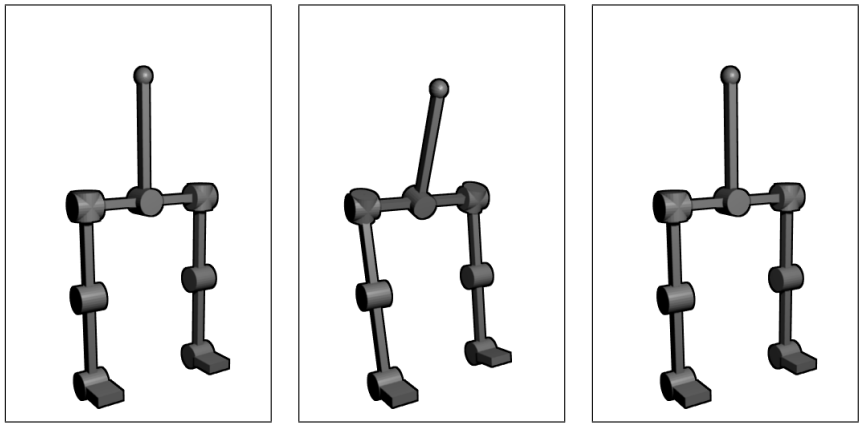
1. *Existen problemas de balance dinámico que son corregibles con pequeñas alteraciones de las consignas articulares deseadas*. Esto se ilustra en la figura 4.2, en donde se observa que el robot en equilibrio es expuesto a una pequeña fuerza que lo hace caer; pero si éste realiza un pequeño movimiento correctivo no caerá.

2. *Existen problemas de balance dinámico que no son corregibles manteniendo la misma consigna deseada*. En éste caso el robot caerá si pretende corregir o modificar su postura; es necesario modificar las consignas con el fin de evitar la caída y luego regresar al estado inicial. Esto se ilustra en la figura 4.3.

Por lo anterior es necesario encontrar una forma de solucionar cada uno de los problemas de balance expuestos.

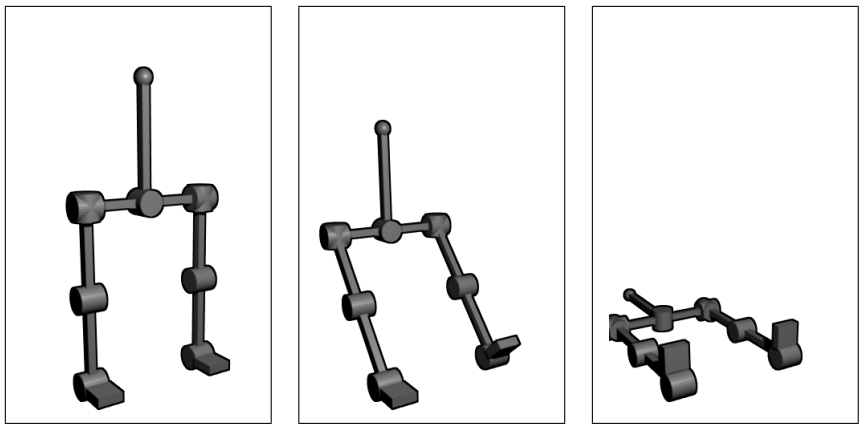


(a) Sistema cae debido a la no corrección de movimiento

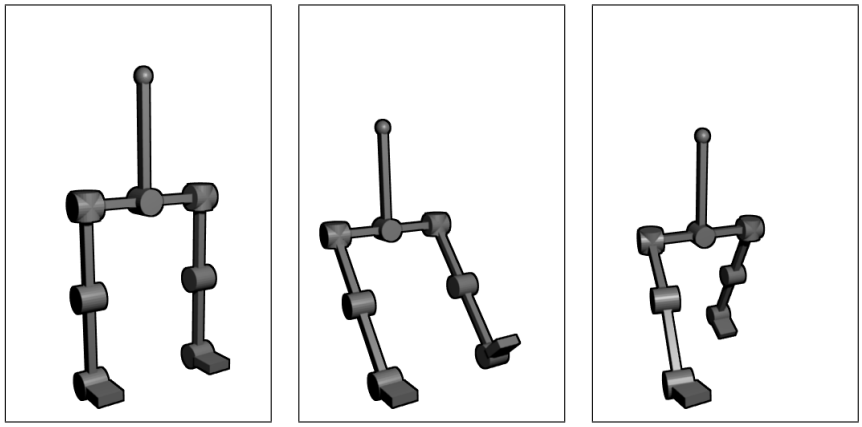


(b) Movimiento para no caer

Figura 4.2: Balance para no caer con pequeña modificación de consignas



(a) Sistema cae debido a una corrección de movimiento que no es suficiente



(b) Movimiento para no caer

Figura 4.3: Balance para no caer con selección de nuevas consignas

4.5.1. Corrección de balance dinámico con pequeñas modificaciones de consignas

En este caso lo que se busca es continuar con las consignas deseadas buscando mantener el balance mediante una pequeña modificación de las consignas articulares. Para conseguir este fin el método más utilizado es el de modificar un poco el ángulo de la pierna de avance (en el caso de caminado) dependiendo de la posición del centro de masa con respecto al pie de apoyo [30]. De esta forma se dice que el ángulo de inclinación de la pierna de avance es:

$$q_d = q_{d0} + c_d d + c_v v, \tag{4.12}$$

donde q_d es el ángulo de la pierna de avance modificado, q_{d0} es el ángulo deseado por la consigna, d es la distancia horizontal entre el centro de masa y el pie de apoyo y v es la velocidad del centro de masa. Por otro lado c_d y c_v son ganancias de realimentación que permiten aumentar o disminuir el efecto del control de balance. En el caso particular del robot bípedo propuesto los valores utilizados fueron de 0.3 y 0.1 respectivamente.

Posteriormente es necesario efectuar una corrección en el par entregado al actuador de la pierna de apoyo ya que este debe compensar el efecto del par entregado a la pierna de avance y al torso como se muestra a continuación,

$$\tau_{P_{AP}} = -\tau_{P_{AV}} - \tau_{Torso} \tag{4.13}$$

de esta forma se controla el ángulo de inclinación del torso y al mismo tiempo el ángulo de inclinación de la pierna de avance con respecto al origen y no con respecto al robot. Es necesario anotar que existe un caso particular cuando los dos pies se encuentran en apoyo, si esto sucede se modifican las consignas de las dos piernas para mantener el balance.

4.5.2. Corrección de balance dinámico con selección de nuevas consignas

Cuando no es posible mantener el balance con pequeñas modificaciones es necesario generar nuevas consignas que eviten que el robot caiga. Aunque existen diferentes métodos [29],[30],[46], todos concuerdan en que existen una serie de *espacios de consignas* en los espacios de estados. Es decir el problema consiste en partir de una configuración de espacio de estados y de las posibles consignas para mantener el equilibrio. Se puede hablar de un espacio de consignas en donde la solución del numeral anterior es suficiente (Sección 4.5.1); pero si por algún motivo se alcanza un estado que no se encuentra dentro del espacio de consignas utilizado en el momento se debe tomar la decisión de iniciar un nuevo espacio de consignas hasta poder regresar al estado inicial. Esto se ilustra en la figura 4.4.

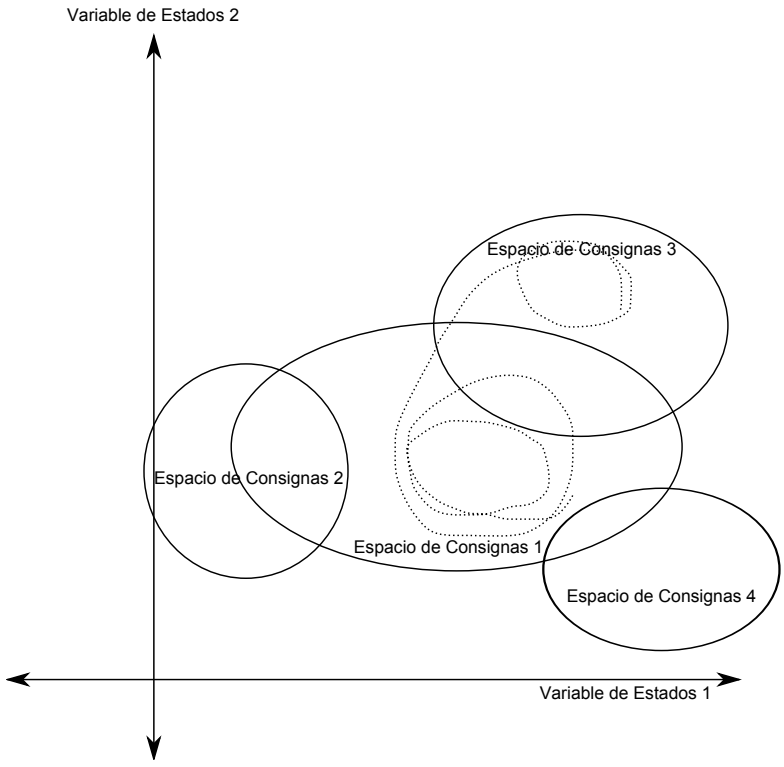


Figura 4.4: Espacio de consignas

Como se puede observar el ejemplo presenta un sistema que se encuentra en el espacio de consignas 1 (la línea punteada representa el movimiento de los estados en el tiempo), como se ve los estados pueden pasar cerca a el espacio de consignas 3 o pueden ingresar en una parte compartida, pero eso no genera un cambio de consignas deseadas, la única forma de decidir cambiar de consignas es pasar por completo a una zona en la que el espacio de consignas 1 no se encuentre. En ese momento se debe tomar la decisión de cambiar de consigna y mantener el equilibrio en el espacio de consignas 3, hasta que sea posible regresar al espacio 1 (si es lo que se desea).

4.5.3. Corrección de balance utilizada

Para el caso particular de la simulación propuesta se ha trabajado con corrección de balance dinámico con pequeñas modificaciones de consignas; método que es suficiente para evaluar el comportamiento de diferentes configuraciones de robot bajo condiciones de marcha humana.

4.6. Interfaz de usuario

Debido a que se trabajó con el lenguaje de programación basado en *scripts* Python, la aplicación desarrollada es capaz de visualizar el comportamiento del robot en un ambiente 3D de una manera intuitiva y amigable (Figura 4.1), pero a su vez permite modificar la configuración y el control del robot de una manera rápida.

Por otro lado la aplicación es capaz de exportar los estados del robot o cualquier tipo de señal que haga parte del sistema robot-control a un archivo plano para su posterior análisis en herramientas especializadas como por ejemplo Matlab-Simulink®.

Capítulo 5

ANÁLISIS DE LOS RESULTADOS

5.1. Simulación del modelo geométrico

Con el fin de verificar el correcto funcionamiento del modelo geométrico se realizó una comparación entre las posiciones obtenidas de la implementación del modelo geométrico en un ambiente Matlab/Simulink® y los datos arrojados por el movimiento de las mismas articulaciones de un esquema tridimensional construido con la herramienta Blender. La prueba consistió en utilizar el modelo geométrico directo (matrices de transformación para una rama) con variables articulares preestablecidas para obtener las coordenadas cartesianas del último enlace calculado con Matlab®, para luego compararlas con las coordenadas cartesianas del pie de avance que se obtienen en Blender3D al efectuar las mismas modificaciones en las variables articulares.

Los resultados se entregan en el cuadro 5.1.

Articulación									Blender			Matlab®		
θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	x	y	z	x	y	z
0	40^o	0	0	0	0	0	0	0	4.8	2.8	-2	4.8	2.8	-2
0	0	30^o	0	0	0	0	0	0	5.7	1	-2	5.7	1	-2
0	0	0	60^o	0	0	0	0	0	6	1.7	-1	6	1.7	-1
0	0	0	0	0	0	0	0	-30^o	5.7	0	-3	5.7	0	-3
0	-30^o	15^o	30^o	0	0	0	0	-30^o	5.8	-0.8	-1.3	5.8	-0.8	-1.3

Cuadro 5.1: Simulacion modelo geométrico

5.2. Resultados de la simulación del control por par calculado en Matlab®

Con el fin de efectuar una simulación acorde al movimiento humano se implementaron consignas articulares que se acercan a un movimiento básico de inicio de paso [50]; dado que dichas consignas no pueden tener cambios bruscos [51] se propone un patrón de trayectoria polinomial como se muestra en la figura 5.1. Los valores finales de las consignas son mostrados en el cuadro 5.2, asumiendo que los valores iniciales son todos cero.

q_1^d	q_2^d	q_3^d	q_4^d	q_5^d	q_6^d	q_7^d	q_8^d	q_9^d
6^o	0^o	0^o	0^o	0^o	35^o	-15^o	0^o	10^o

Cuadro 5.2: Consignas articulares

En el cuadro 5.3 se muestran las ganancias implementadas para la simulación. Posteriormente se muestran los resultados obtenidos en la simulación

j	1	2	3	4	5	6	7	8	9
K_p	3.0e5	9.0e4	6.0e4	3.0e5	3.0e5	6.0e4	7.0e4	3.0e5	1.0e5
K_v	400	100	70	400	400	70	100	400	120

Cuadro 5.3: Ganancias del controlador

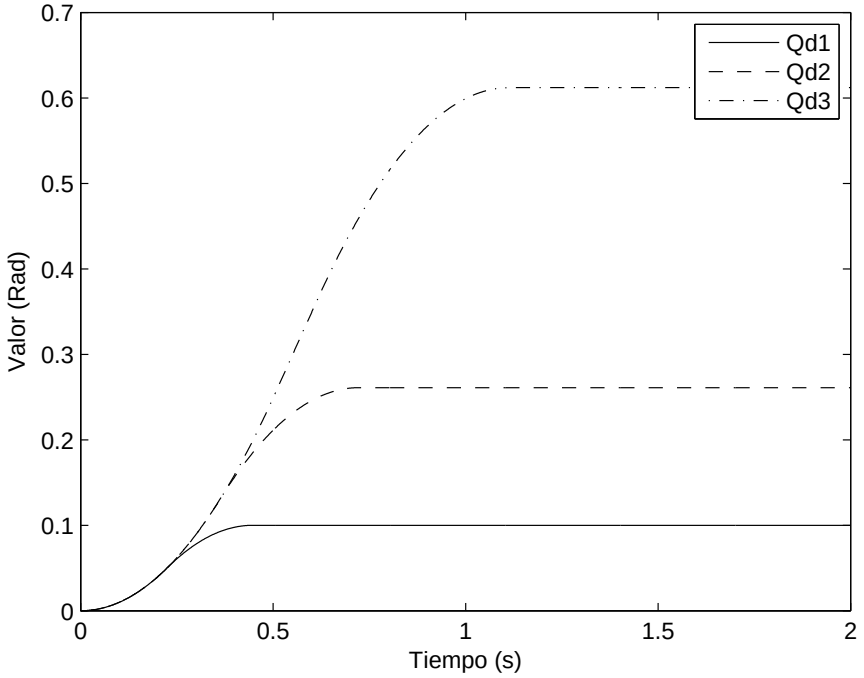


Figura 5.1: Consignas articulares

(Figuras 5.2, 5.3) donde se puede observar que el error articular es del orden de $10^{-4}rad$ lo que es suficiente para el caso de un robot bípedo.

Estos valores muestran el correcto funcionamiento del control para un ambiente de simulación inicial (Matlab/Simulink®). Aunque numéricamente el error es grande hay que tener en cuenta que en los robots bípedos el error articular no es lo más determinante, ya que es mucho más importante realizar la tarea de locomoción, manteniendo el equilibrio, que conseguir un error articular o cartesiano ideal.

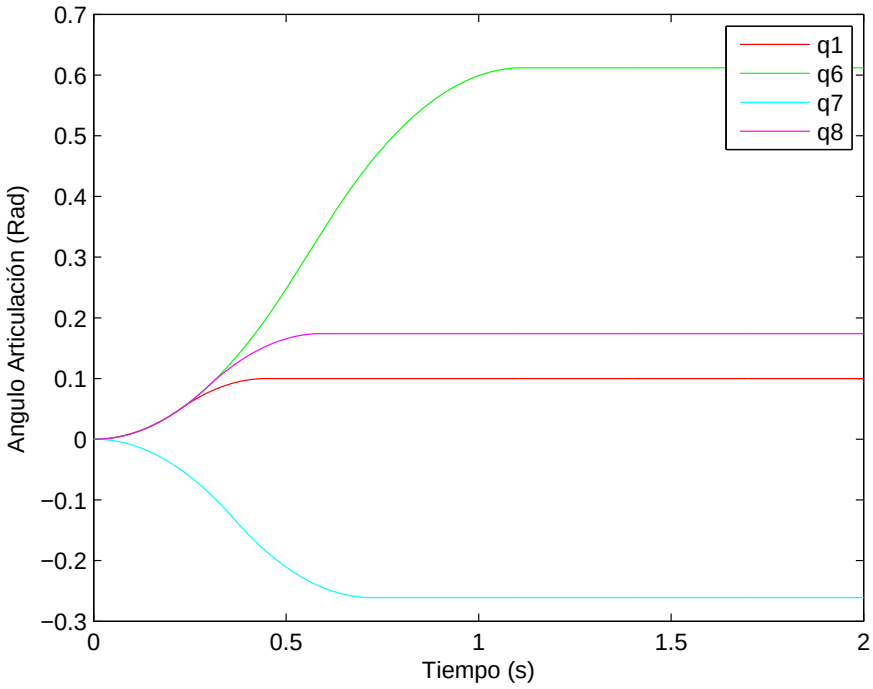
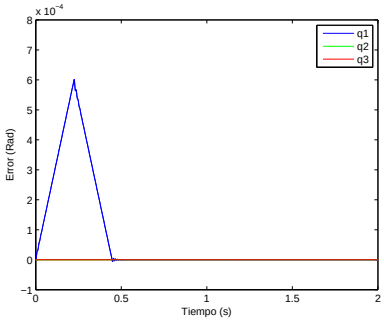
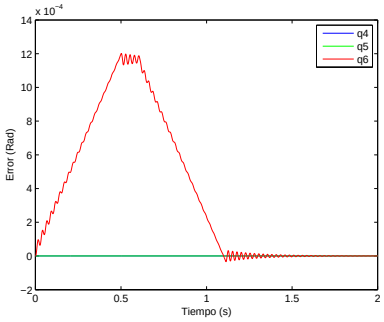


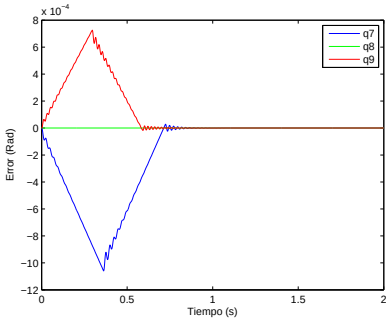
Figura 5.2: Salida variables articulares del robot



(a) Articulaciones 1 2 3



(b) Articulaciones 4 5 6



(c) Articulaciones 7 8 9

Figura 5.3: Errores articulares

5.3. Resultados de la simulación del robot bípedo en entorno 3D propio

Al realizar el montaje del robot bípedo en el ambiente de simulación 3D se pudo verificar su correcto funcionamiento, esta primera prueba consistió en dejar caer el robot y observar que su estructura no tomará posiciones indeseables, para esto no se aplicó Par a sus articulaciones y el robot demostró tener una configuración correcta, aunque es de anotar que se fijaron límites geométricos en sus articulaciones (Figura 5.4) con el fin de hacer que la estructura del robot tomará posiciones cercanas a las que tendría un humano que se desvanece y cae al suelo.

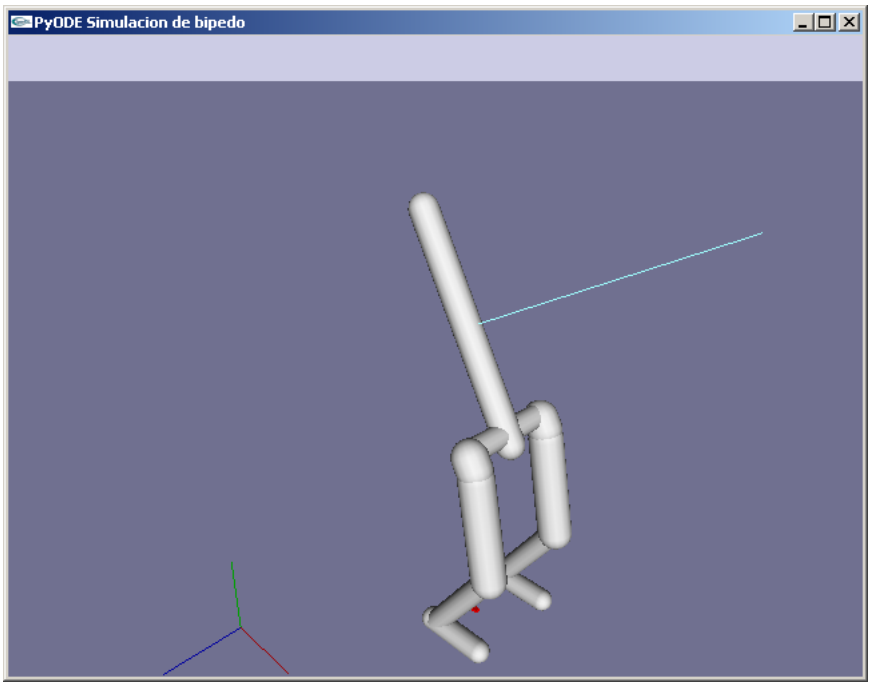


Figura 5.4: Simulación 3D del robot sin pares aplicados

Al realizar el montaje del sistema de control por par calculado, en el entorno de simulación física 3D propio, el robot se comportó de una manera poco estable. Al realizar pruebas de verificación se pudo observar un crecimiento del

par aplicado tendiente a infinito, lo cual resulta poco deseable.

Después de estudiar el comportamiento del robot se identificó que el error producido se debía a la suposición asumida sobre el pie de apoyo (se supuso que actuaba como pivote). Al efectuar una modificación en la configuración del robot se observó un comportamiento adecuado; en la figura 5.5 se puede apreciar el robot realizando un desplazamiento con la característica de la pierna de apoyo actuando como pivote[1] y en la figura 5.6 se presenta el valor del ángulo de la articulación de la pierna de avance en donde se puede apreciar que el comportamiento es correcto y estable.

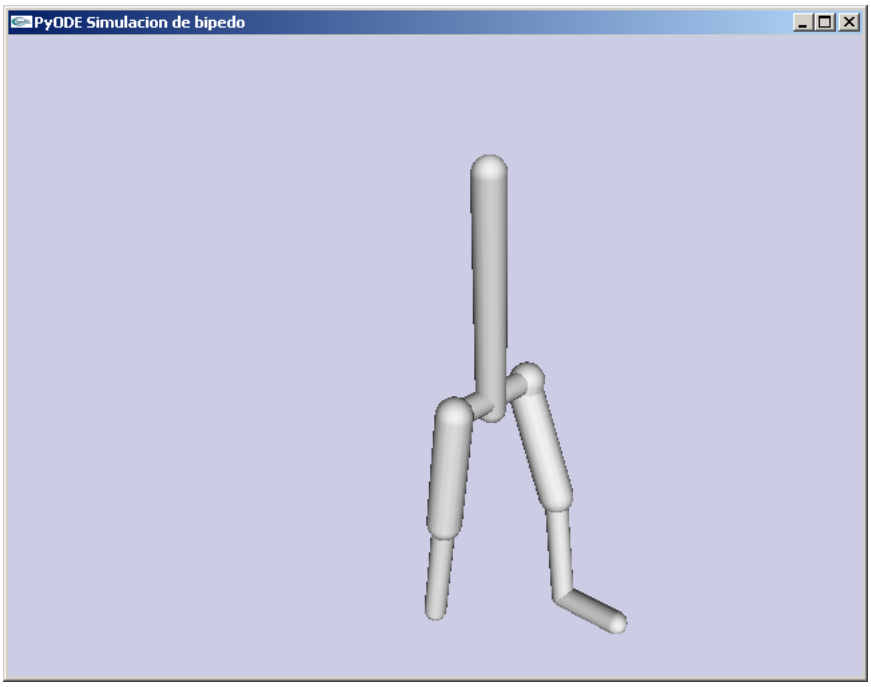


Figura 5.5: Simulación 3D del robot con control por par calculado

El control por par calculado tiene la debilidad de depender demasiado de la correcta generación del modelo inverso, es decir las diferencias entre el modelo simulado y el real afectan de manera considerable el comportamiento, generando posibles inestabilidades. Por tal motivo este tipo de control no es

[1] En la figura la pierna de apoyo parece no tener pie debido a que es un pivote

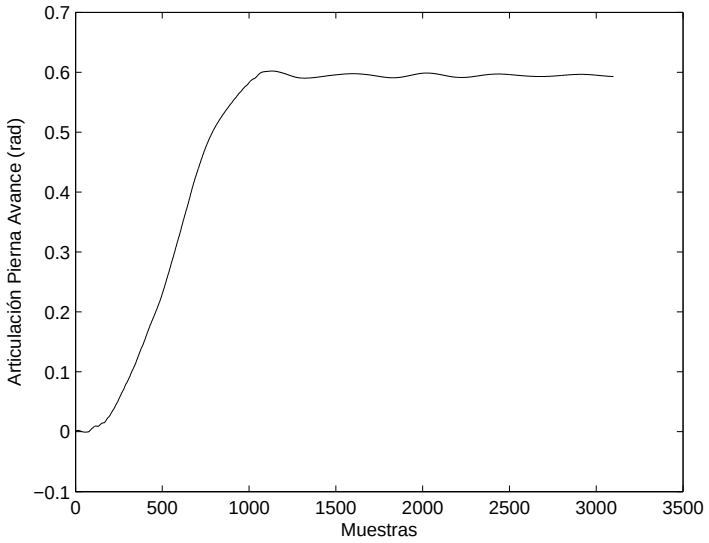


Figura 5.6: Simulación 3D control por par calculado pierna avance

recomendable para una aplicación de prótesis ya que ésta se encuentra expuesta a condiciones externas variables y condiciones de dinámica impulsiva, es recomendable de utilizar técnicas basadas en control robusto u óptimo [52], [53].

En el caso del mantenimiento de equilibrio se implementó el algoritmo de corrección de balance dinámico con pequeñas modificaciones de consignas utilizando control PD, el cual consiguió un buen resultado, aunque genera discontinuidades en la dinámica, debido al cambio del pie de apoyo. En la figura 5.7 se presenta el robot bípedo manteniendo el equilibrio con el pie derecho como apoyo en donde se puede verificar que la proyección del centro de masa en el plano del suelo (triángulo rojo cerca del pie derecho) se encuentra fuera del rectángulo que forma el apoyo del robot, lo que demuestra un balance dinámico. En la figura 5.8 se muestra cómo se comporta el par aplicado en la pierna derecha al tratar de mantener el equilibrio, en ésta figura se pueden observar discontinuidades al cambiar de pie de apoyo, así como picos de pares relativamente altos debido a la necesidad instantánea de mantener el

equilibrio.

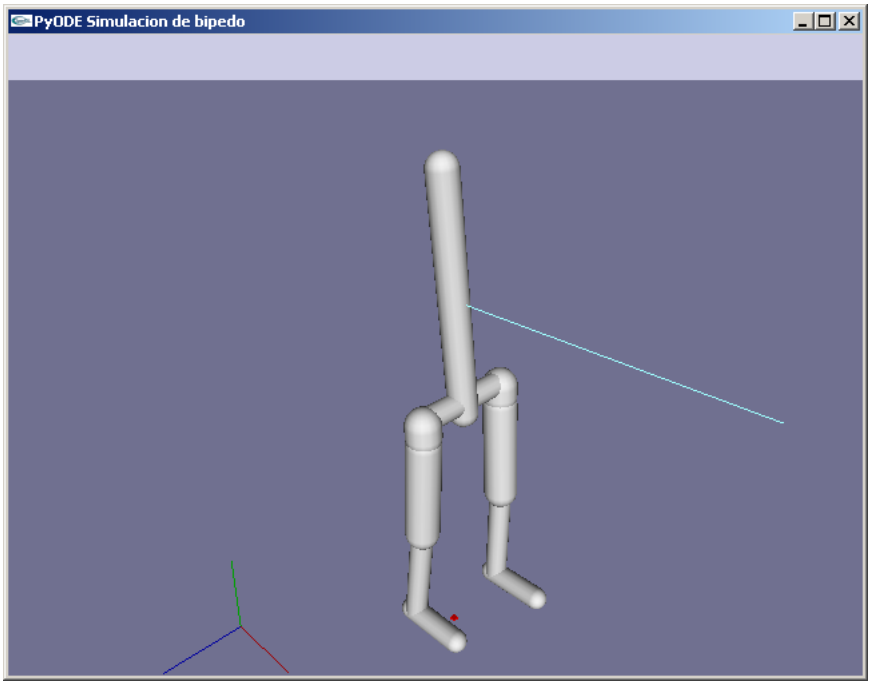


Figura 5.7: Simulación 3D del robot manteniendo su balance

5.4. Resultados del análisis de la marcha del robot para diferentes configuraciones

En el caso del análisis de la marcha del robot comparada con la marcha humana se inició por definir indicadores de la calidad de la marcha del robot con el fin de establecer la mejor configuración de robot bípedo que permita sugerir futuras prótesis de pierna.

El primer indicador que se debe considerar es el de establecer si el robot es capaz de mantenerse en pie y realizar al menos un ciclo de marcha sin caer. Por otro lado existen indicadores que permiten medir la calidad de la marcha y el esfuerzo que esta implica para una persona con prótesis de pierna [54],[55],

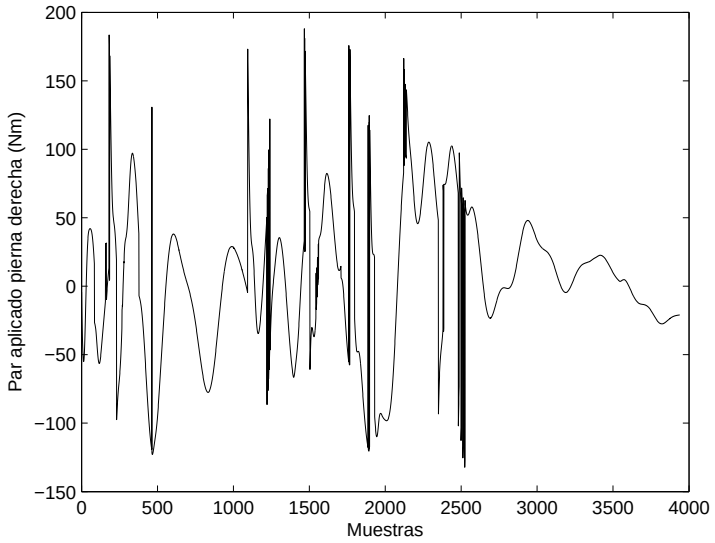


Figura 5.8: Simulación 3D par aplicado pierna derecha con control de balance

para el caso particular del robot bípedo propuesto se definieron los siguientes indicadores:

- Velocidad de la marcha
- Cadencia (pasos por minuto)
- Longitud de los pasos
- Desplazamiento transversal del centro de masa
- Desplazamiento angular del torso

Con estos indicadores establecidos se inició con el análisis de la configuración del robot con el fin de determinar si con los grados de libertad propuestos (9) es posible realizar la marcha humana.

El resultado de la simulación de la configuración inicial propuesta se ilustra en la figura 5.9 en donde se coloca el avance del tiempo en tonos de grises (de claro a oscuro). En esta imagen se puede apreciar cómo el robot no es capaz de mantener su equilibrio debido a la falta de grados de libertad que permitan

mover las piernas en el sentido transversal, por esta razón el robot solo puede realizar cerca de 2 pasos antes de caer. En este caso no es posible efectuar análisis de indicadores.

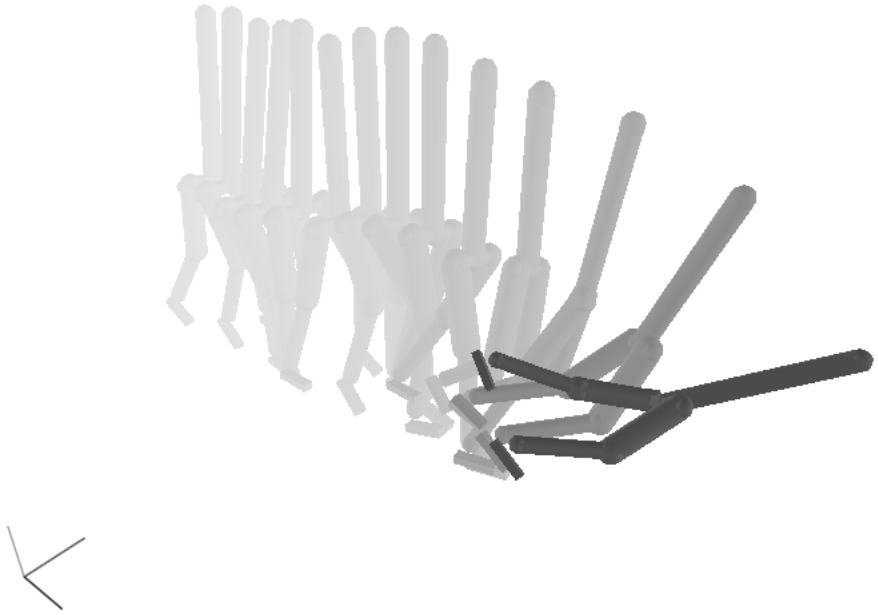


Figura 5.9: Simulación marcha de robot de 9 grados de libertad

Debido al resultado obtenido con la configuración anterior se procedió a agregar 2 grados de libertad en las articulaciones de las piernas (figura 5.10) con el fin de permitir el movimiento lateral que hace falta. Con esta nueva configuración los resultados fueron favorables, como se puede apreciar en la figura 5.11. En este caso la marcha fue estable y continua por lo que fue posible determinar los indicadores.

Se realizó otra configuración partiendo de la inicial de 9 grados de libertad y agregando 2 grados de libertad más a los tobillos (uno en cada tobillo) como se muestra en la figura 5.12. Esta configuración presentó un comportamiento similar al robot inicial propuesto (similar al de la figura 5.9) por lo que no fue posible determinar sus indicadores.

Por último se probó una configuración de 13 grados de libertad en la que se

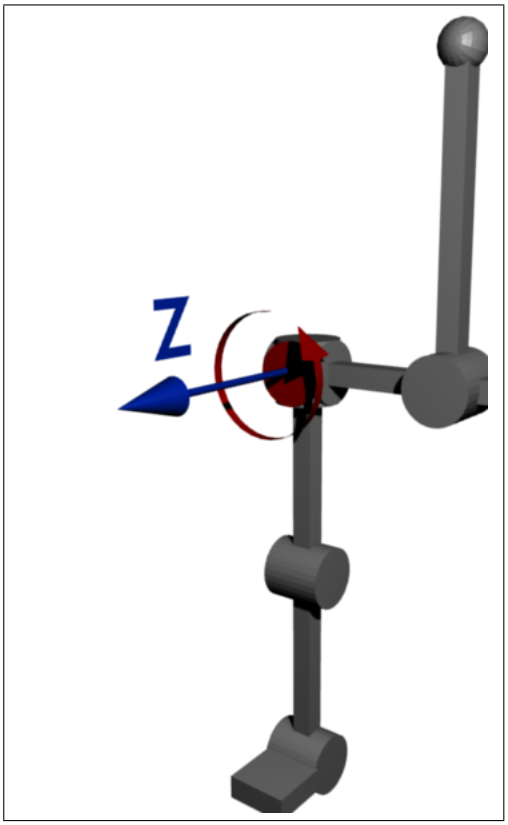


Figura 5.10: Nuevo grado de libertad agregado en piernas

agregaron los grados de libertad anteriores de pierna y tobillo conjuntamente. Esta configuración presentó un comportamiento aparentemente similar al de 11 grados de libertad (figura 5.11) pero al analizar los indicadores se puede observar que presenta un comportamiento considerablemente mejor.

Los resultados de indicadores obtenidos con las diferentes configuraciones se ilustran en el cuadro 5.4, en donde:

- *Camina* indica si el robot es capaz de caminar o no
- *Vel.* indica la velocidad en metros por segundo
- *Cad.* indica la cadencia en pasos por minuto
- *L.Pas.* indica la longitud de paso en metros

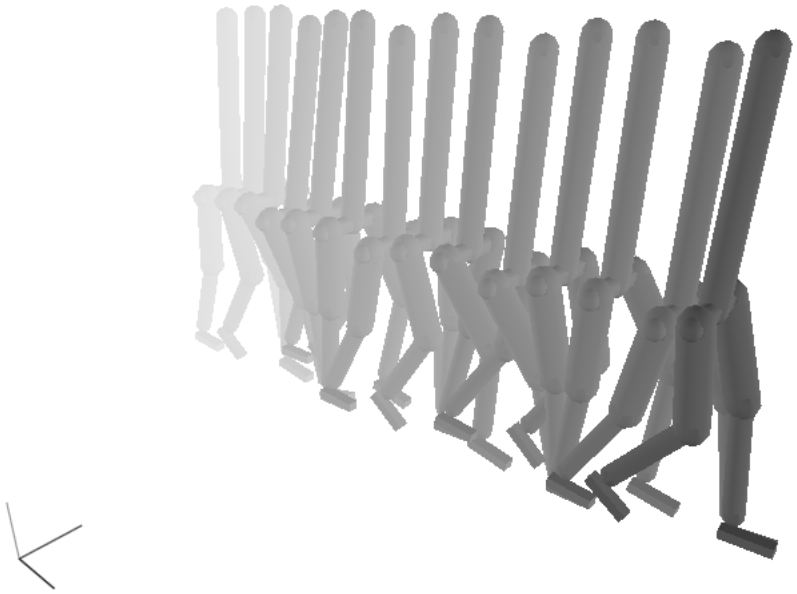


Figura 5.11: Simulación marcha de robot de 11 grados de libertad

- *D.Tr.Max.* indica el desplazamiento transversal máximo del centro de masa en caminado estable
- *D.To.Max.* indica el desplazamiento angular máximo (en grados) del torso en caminado estable

Robot	Camina	Vel.	Cad.	L.Pas.	D.Tr.Max.	D.To.Max
9 gdl	NO	–	–	–	–	–
11 gdl (2+ piernas)	SI	1.1884	83.58	0.5901	6.022	2.83
11 gdl (2+ tobillos)	NO	–	–	–	–	–
13 gdl	SI	1.1950	83.58	0.6014	3.529	1.35
Marcha base	SI	1.2	83.58	0.612	3.752	1.716

Cuadro 5.4: Indicadores de marcha de diferentes configuraciones

Es importante aclarar que en la tabla se ha colocado una última fila llamada *Marcha base* en la cual se analizan los mismos indicadores para los datos

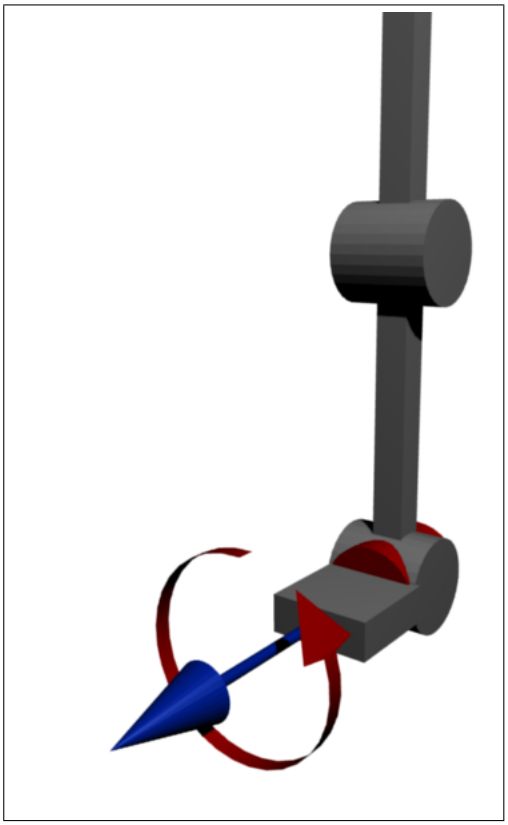


Figura 5.12: Nuevo grado de libertad agregado en tobillos

adquiridos de un humano real. Los movimientos adquiridos de éste han servido como base para la generación de las consignas de movimiento de los robots.

Como se puede observar en el cuadro anterior aunque existen dos configuraciones de robot bípedo que permiten efectuar caminados correctos, el de 11 con los dos grados de libertad adicionales a lado y lado de la cadera y el de 13 grados de libertad, siendo este ultimo el que presenta mejor desempeño debido al grado de libertad agregado en cada uno de los tobillos. Esta modificación hace que el robot presente un balanceo mucho menor (cerca de la mitad) y por lo tanto su caminar sea más eficiente. Se puede observar en la figura 5.13 el desplazamiento transversal de cada uno de los robots a lo largo del tiempo y verificar que el balanceo es mucho mayor en el robot de 11 grados de libertad.

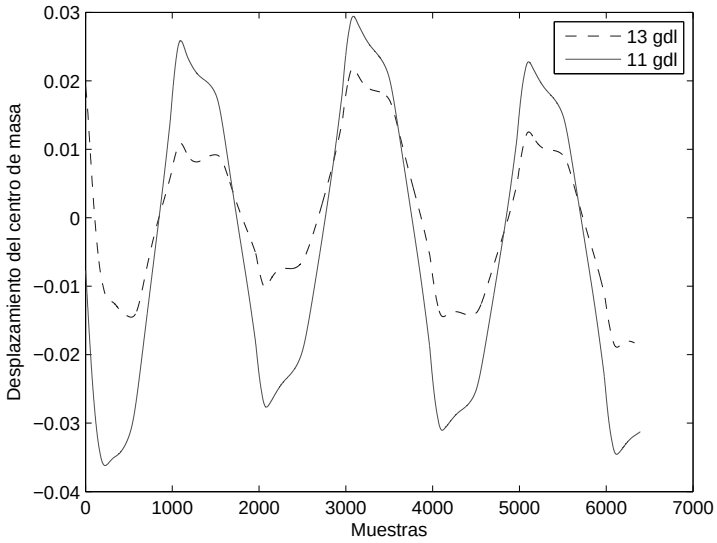


Figura 5.13: Comparación de desplazamiento transversal de los robots de 11 y 13 gdl

El mejor desempeño del robot de 13 grados de libertad se puede verificar de una manera más formal analizando el comportamiento del punto de presión en el pié de apoyo, el cual coincide con el punto de momento cero (ZMP) [56], cuando el punto de presión no se encuentra en los límites geométricos de la cara en contacto con el suelo (en caso contrario se dice que el punto de momento cero no existe o es ficticio y se le llama ZMPF). Cuando no existe el punto de momento cero no se puede garantizar que la orientación y la posición global del robot sean controlables. En las Figuras 5.14 y 5.15 se puede observar que el pie izquierdo en el caso de 11 grados de libertad mantiene su punto de presión cerca a los límites de la cara de contacto (-0.8 a 0.1 en Z y -0.03 a 0.03 en X), lo que lo hace poco controlable en orientación y posición global (ZMPF) cuando el pie izquierdo es el de apoyo (caso en el que la fuerza de reacción del suelo es alta). Por otro lado en el caso de 13 grados de libertad el pie izquierdo mantiene su punto de presión cercano al centro la mayoría del tiempo de apoyo, por lo que el punto de momento cero existe y es posible

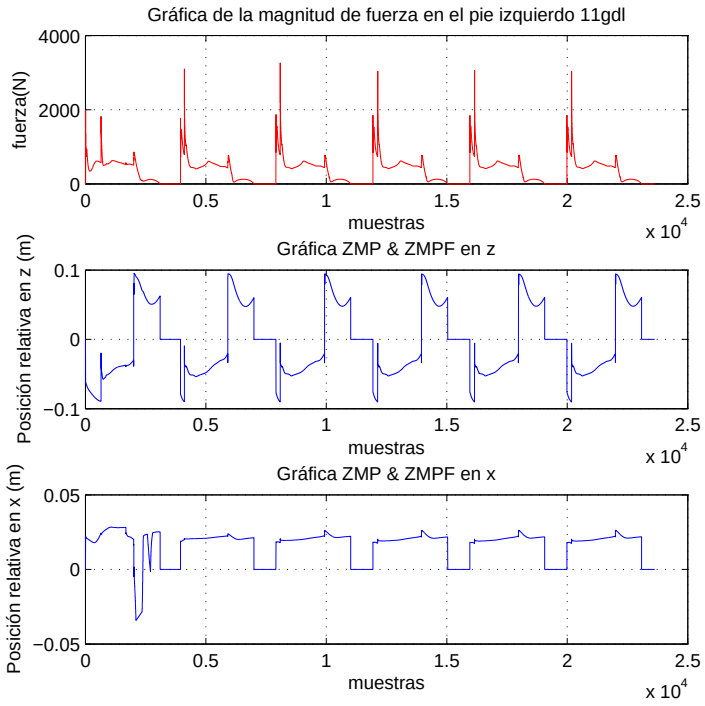


Figura 5.14: ZMP y ZMPF en pie izquierdo robot de 11 gdl

controlar la orientación y posición global de una mejor forma.

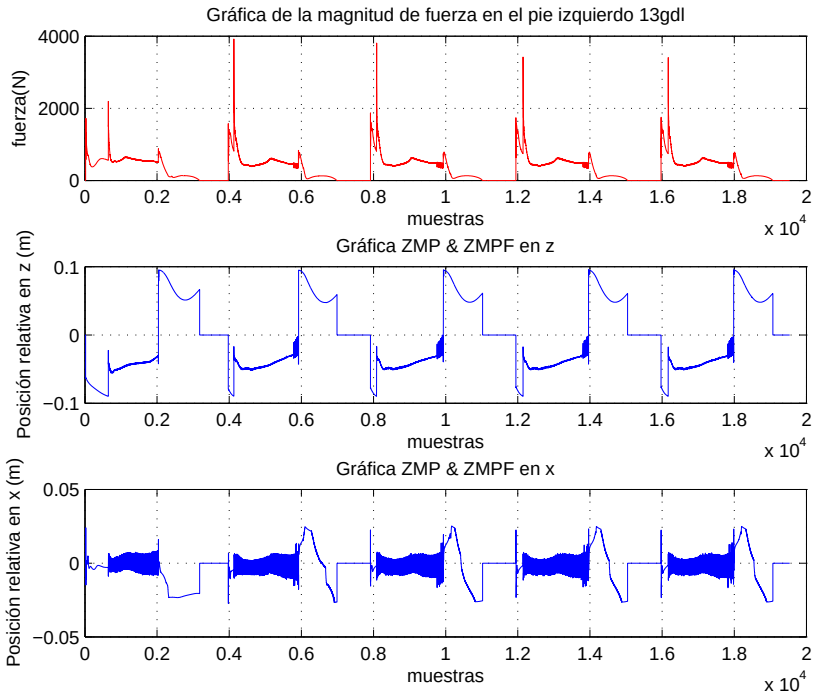


Figura 5.15: ZMP y ZMPF en pie izquierdo robot de 13 gdl

Capítulo 6

CONCLUSIONES Y RECOMENDACIONES

En el presente trabajo se propuso una estructura inicial de robot bípedo conformado por miembros inferiores y torso a partir de articulaciones rotoídes distribuidas en 9 grados de libertad y a partir de la determinación de dicha estructura se obtuvieron los modelos geométricos y dinámicos del robot. En la validación del modelo geométrico del robot (a través de la comparación de las posiciones obtenidas en la implementación del modelo geométrico en un ambiente Matlab/Simulink® y los datos arrojados por el movimiento de las mismas articulaciones en un esquema tridimensional construído en la herramienta Blender) se demostró su correcto funcionamiento y la seguridad de utilizarlo en las fases siguientes del proyecto. Del mismo modo se simuló el modelo dinámico del robot en el ambiente Matlab/Simulink® y se implementó un control articular usando como esquema el control por par calculado, obteniendo, en esta herramienta software, resultados satisfactorios de seguimiento de consignas articulares, lo que permitió concluir que es posible utilizar un modelo de control como el propuesto.

Adicionalmente se implementó una herramienta de simulación física 3D que permitió validar el correcto funcionamiento del modelo propuesto en un ambiente más cercano al real, de lo cual se concluyó que la estructura inicial, con desempeño correcto en el seguimiento de consignas articulares específicas, no presenta un caminar correcto del robot dado que no fue capaz de mantener el equilibrio, lo que determinó la necesidad de adicionar un grado de libertad adicional en cada pierna.

Como conclusión se afirma que es posible simular de una manera adecuada el caminar humano con 11 grados de libertad distribuidos en la forma expuesta en la sección 5.4, pero al ser necesario mejorar la eficiencia de la marcha y el desempeño en general, la configuración de 13 grados de libertad muestra mejores resultados.

Este trabajo aporta el desarrollo de un entorno de simulación 3D apto para estudios científicos, ya que involucra gran parte de las variables del mundo físico real. En este entorno la estructura software permite fácil acceso a modificaciones a nivel de código gracias al entorno de programación del lenguaje Python, facilitando no solo el estudio de la incidencia de las prótesis de pierna en la estructura del robot, óptimo para emular el caminar humano, resultado de este trabajo, sino también la simulación de estructuras de robots bípedos y exoesqueletos y sus respectivos esquemas de control, permitiendo la determinación de la pertinencia y viabilidad de construcción física de los mismos.

Al utilizar una librería de simulación física como OpenDE junto con el motor de visualización 3D OpenGL bajo el entorno de programación del lenguaje Python se obtienen resultados favorables y se posibilita la realización de modificaciones a la configuración inicial del robot y a los algoritmos de control de una manera rápida y eficiente. Es posible utilizar programas que se ejecutan de manera binaria con librerías precompiladas acompañadas de partes que son

configurables a partir de archivos texto; pero esto genera códigos difíciles de comprender e ineficientes ya que utilizan gran parte de su procesamiento en comprender o interpretar las configuraciones. Es por eso que se recomienda utilizar un lenguaje de programación que se base en *scripts* pero que a su vez implemente librerías compiladas (binarias) para no sacrificar la velocidad.

Referencias Bibliográficas

[1] P. E. Sandin. *Robot Mechanisms and Mechanical Devices Illustrated*. McGraw-Hill, USA, 2003.

[2] D. J. Todd. Mobile robots the lessons from nature. *Journal of Robots and Biological Systems: Towards a New Bionics?*, Vol. 00, pp. 193–206, 1993.

[3] C. I. Gonzales y J. C. Gómez. *Información estadística de la discapacidad*. DANE, 2004.

[4] DANE. *Boletín, Censo General Discapacidad - Colombia*, 2005.

[5] S. Sabolich. Putting your best foot forward. *In Motion Magazine, Amputee Coalition of America*, Vol. 10, No. 6, 2000.

[6] O. Castillo. Lanzan una generación de robots que caminan como humanos. *Periódico El Clarín*, 2005. Disponible en http://www.premionovela.clarin.com/diario/2005/02/19/sociedad/s-04615.htm.

[7] M. Correal, L.J. Palacio, and J.C. Salazar. Desarrollo de un sistema protésico para personas con amputación transfemoral en colombia. *Universidad EAFIT, Medellín*, 2004.

[8] O.J. Ascencio, D.J. Gómez, A.M. Espejo, and P.F. Martín. Diseo y modelamiento de pie para prótesis transfemoral con sistema de amortiguación. *Epsilon, Univesridad de la Salle*, Vol. 9, pp. 7–18, 2007.

[9] A.F. Muoz, A. Pardo, and J.L. Díaz. Investigación y desarrollo de nuevos sistemas inteligentes para el control mecatrónico de una prótesis bioeléctrica. *Revista Colombiana de Tecnologías de Avanzada*, Vol. 1, No. 9, pp. 97–102, 2007.

[10] J. Deutscher, A. Blake, and i. Reid. Articulated body motion capture by annealed particle filtering. In Proceedings *Computer Vision and Pattern Recognition (CVPR)*, pp. 2126–2133, 2000.

[11] E. Smalley. Robots runs like humans. *Technology Research News*, 2005. Disponible en http://www.trnmag.com/Stories/2005/061505/Robot_runs_like_humans_061505.html.

[12] Herodotus. *Herodotus The Histories*. Oxford World's Classics, London, 1998.

[13] Today in science history. www.todayinsci.com/P/PareAmbroise/ParеProstheses.htm. Ultima consulta ao 2007.

[14] M. Sachs, J. Bojunga, and A. Encke. Historical evolution of limb amputation. *World Journal of Surgery*, Vol. 23, No. 10, pp. 1088–1093, 1999.

[15] B. Dupes. What you need to know about knees. *In Motion Magazine, Amputee Coalition of America*, Vol. 14, No. 1, 2004.

[16] Protetización con la c–leg. www.ottobock.com. Ultima consulta ao 2007.

[17] S. O. Anderson, M. Wisse, C. G. Atkeson, J. K. Hodgins, G. J. Zeglin, and B. Moyer. Powered bipeds based on passive dynamic principles. In Proceedings *5th IEEE–RAS International Conference on Humanoid Robots*, pp. 110–116, 2005.

[18] M. H. Raibert. *Legged robots that balance*. MIT press, USA, 1986.

[19] G. N. Boone and J. K. Hodgins. Slipping and tripping reflexes for bipedal robots. *Journal of Autonomous Robots*, Vol. 4, No. 3, pp. 259–271, 1997.

[20] E. R. Dunn and R. D. Howe. Towards smooth bipedal walking. In Proceedings *IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2489–2494, 1994.

[21] C. L. Shih. Analysis of the dynamics of a biped robot with seven degrees of freedom. In Proceedings *IEEE International Conference on Robotics and Automation*, Vol. 5, pp. 3008–3013, 1996.

[22] M. Vukobratovic. Legged locomotion robots. Technical report, Mikkan Kogyo Shinbunsya, Japan, 1975.

[23] K. Y. Yi and Y. F. Zheng. Biped locomotion by reduced ankle power. In Proceedings *IEEE International Conference on Robotics and Automation*, Vol. 5, pp. 584–589, 1996.

[24] D. Baraff and A. Witkin. Physically based modeling: Principles and practice. *Course Notes ACM Transactions on Graphics (SIGGRAPH)*, 1997.

[25] R. Smith. Open Dynamics Engine (ODE). http://www.ode.org. Ultima consulta ao 2009.

[26] Box 2D. http://www.box2d.org. Ultima consulta ao 2009.

[27] Newton Game Dynamics Physics Engine. www.newtondynamics.com. Ultima consulta ao 2009.

[28] NVIDIA Corporation. PhysX. http://www.nvidia.com. Ultima consulta ao 2009.

[29] K.W. Sok, M. Kim, and J. Lee. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics (SIGGRAPH)*, Vol. 26, No. 3, 2007.

[30] K. Yin, K. Loken, and M. van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (SIGGRAPH)*, Vol. 26, No. 3, pp. 105, 2007.

[31] A. Shapiro, P. Faloutsos, and V. Ng-Thow-Hing. Dynamic animation and control environment. In Proceedings *GI '05: Proceedings of the 2005 conference on Graphics interface*, pp. 61–70, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[32] H. Hemami and Y. F. Zheng. Dynamics and control of motion on the ground and in the air with application to biped robots. *Journal of Robotic Systems*, Vol. 1, No. 1, pp. 101–116, 1984.

[33] R. E. Goddard, F. Zheng, and H. Hemami. Control of the heeloff to toeoff motion of a dynamic biped gait. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 1, pp. 92–102, 1992.

[34] A. Takanishi, H. O. Lim, M. Tsuda, and I. Kato. Realisation of dynamic biped walking stabilised by trunk motion on a sagitally uneven surface. In Proceedings *IEEE Int. Workshop on Intelligent Robots and Systems (IROS)*, pp. 323–330, 1990.

[35] W. Khalil and E. Dombre. *Modeling, identification and control of robots*. Hermes Penton Science, London, 2002.

[36] ASIST human body properties database. http://www.sh.aist.go.jp/bodydb. Ultima consulta ao 2009.

[37] R. C. Paul. Robot manipulators: Mathematics, programming and control. *MIT Press*, 1981.

[38] Y. Fujimoto, S Obata, and A. Kawamura. Robust biped walking with active interaction control between foot and ground. In Proceedings *IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 2030–2035, 1998.

[39] V. Lebastartd, Y. Aoustin, and F. Plestan. Observer-based control of a waking biped robot without orientation mesurement. *Journal of Robotica, Cambridge University Press*, Vol. 00, pp. 1–16, 2005.

[40] C. F. Rengifo. Commande et observation pour la marche d'un robot marcheur. Rapport bibliographique, L'Institut de Recherche en Communications et Cybernétique de Nantes, France, 2007.

[41] W. Khalil and D. Creusot. Symoro+: A system for the symbolic modelling of robots. *Robotica,Cambridge University Press*, Vol. 15, No. 2, pp. 153–161, 1997.

[42] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. McGrw-Hill, 1996.

[43] J. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall International, 1991.

[44] Python Programming Language. http://www.python.org. Ultima consulta ao 2009.

[45] C. Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 1999.

[46] C. Azevedo, B. Espiau, B. Amblard, and C. Assaiante. Bipedal locomotion:toward unified concepts in robotics and neuroscience. *Biological Cybernetics*, 2006.

[47] CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/. Ultima consulta ao 2009.

[48] Acclaim ASF/AMC. http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ASF-AMC.html. Ultima consulta ao 2009.

[49] J. Lee and S.Y. Shin. A hierarchical approach to interactive motion editing for human–like figures. In Proceedings *ACM Transactions on Graphics (SIGGRAPH)*, pp. 39–48, 1999.

[50] M. Ogino, K. Hosoda, and M. Asada. Learning energy-efficient walking with ballistic walking. In Proceedings *Adaptive Motion of Animals and Machines*, pp. 155–164, 2006.

[51] F. Lewis, D. Dawson, and C. Abdallah. *Robot Manipulator Control Theory and Practice*. Marcel Dekker, 2004.

[52] M. da Silva, Y. Abe, and J. Popovic. Interactive simulation of sylized human locomotion. *ACM Transactions on Graphics (SIGGRAPH)*, Vol. 27, No. 3, 2008.

[53] U. Muico, Y. Lee, J. Popovic, and Z. Popovic. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics (SIGGRAPH)*, Vol. 28, No. 3, 2009.

[54] P.R.G. Lucareli, M.O. Lima, F.P.S Lima, S.A. Garbelotti, R.O. Gimenes, J.G. Almeida, and J.M.D Greve. Análisis de la marcha y evaluación de la calidad de vida después del entrenamiento de la marcha en pacientes con lesión medular. *Revista de neurología*, Vol. 46, No. 7, pp. 406–410, 2008.

[55] R. Morgenstern, P. Catalán, and R. Gil. Método de análisis biomecánico de la marcha en pacientes portadores de prótesis de rodilla. *Biomecánica: Organo de la Sociedad Ibérica de Biomecánica y Biomateriales*, Vol. 2, No. 2, pp. 54–58, 1994.

[56] M. Vukobratovic and B. Borovac. Zero moment point – thirthy five years of its life. *International Journal of Humanoid Robotics*, Vol. 1, No. 1, pp. 157–173, 2004.

APENDICE

Apéndice A

Código fuente del entorno de simulación 3D

[notoc] Con el fin de efectuar la simulación se ha generado un entorno físico 3D en Python el cual es capaz de emular el comportamiento real de un robot con la configuración propuesta controlado por diferentes tipos de algoritmos.

A.1. Codigo fuente del robot bípedo

```
import sys, os, random, time
from math import *
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

import ode

from aplicmat import *
from consignas4 import *
```

```
muestra = 0
maxmuestra = len(Qd1)

# Cosas de Opengl
def prepare_GL():
"""Setup basic OpenGL"""

glClearColor(0.8, 0.8, 0.9, 0.0)
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)
glEnable(GL_NORMALIZE)
glShadeModel(GL_SMOOTH)

glMatrixMode(GL_PROJECTION)
glLoadIdentity()
gluPerspective (45.0, 1.3333, 0.2, 20.0)

glViewport(0, 0, 640, 480)

glMatrixMode(GL_MODELVIEW)
glLoadIdentity()

glLightfv(GL_LIGHT0,GL_POSITION,[0, 0, 1, 0])
glLightfv(GL_LIGHT0,GL_DIFFUSE,[1, 1, 1, 1])
glLightfv(GL_LIGHT0,GL_SPECULAR,[1, 1, 1, 1])
glEnable(GL_LIGHT0)

glEnable(GL_COLOR_MATERIAL)
```

```
glColor3f(0.8, 0.8, 0.8)

gluLookAt(cam_x, cam_y, cam_z, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0)

def draw_CM_Box(pos_x, pos_z):
rot = makeOpenGLMatrix((1,0,0, 0,1,0, 0,0,1), (pos_x,-0.1,pos_z))
glPushMatrix()
glMultMatrixd(rot)
glColor3f( 1.0,0.0,0.0)
glutSolidCube(0.02)
glPopMatrix()

def draw_Local_Axis(offset, body):
glPushMatrix()
glBegin(GL_LINES)
glColor3f( 1.0,2.0,3.0)
xp,yp,zp = body.getPosition()
glVertex3f(xp,yp,zp)
v = body.getRotation()
x=v[0+offset]
y=v[3+offset]
z=v[6+offset]
glVertex3f(xp+x,yp+y,zp+z)
glEnd()
glPopMatrix()

def draw_Global_Axes(offset=(0,0,0)):
glPushMatrix()
glBegin(GL_LINES)
```

```
glColor3f(1,0,0)
glVertex3f(offset[0]    ,offset[1]    ,offset[2])
glVertex3f(offset[0]+0.3,offset[1]    ,offset[2])
glColor3f(0,1,0)
glVertex3f(offset[0]    ,offset[1]    ,offset[2])
glVertex3f(offset[0]    ,offset[1]+0.3,offset[2])
glColor3f(0,0,1)
glVertex3f(offset[0]    ,offset[1]    ,offset[2])
glVertex3f(offset[0]    ,offset[1]    ,offset[2]+0.3)
glEnd()
glPopMatrix()

def draw_Plane():
glPushMatrix()
glMultMatrixd((1,0,0,0 ,0,1,0,0, 0,0,1,0, 0,0,0,1))
glBegin(GL_QUADS)
glColor3f( 0.7,0.7,0.9)
glNormal3f(0, 1, 0)
glVertex3f(-100, -0.1, 100)
glVertex3f( 100, -0.1, 100)
glVertex3f( 100, -0.1,-100)
glVertex3f(-100, -0.1,-100)
glEnd()
glPopMatrix()



# Dibujar cilindros como capsulas para mejor visualizacion
CAPSULE_SLICES = 16
CAPSULE_STACKS = 12
```

```
def draw_body(body):
"""Draw an ODE body."""

rot = makeOpenGLMatrix(body.getRotation(), body.getPosition())
glPushMatrix()
glMultMatrixd(rot)
if body.shape == "capsula":
cylHalfHeight = body.length / 2.0
glBegin(GL_QUAD_STRIP)
for i in range(0, CAPSULE_SLICES + 1):
angle = i / float(CAPSULE_SLICES) * 2.0 * pi
ca = cos(angle)
sa = sin(angle)
glNormal3f(ca, sa, 0)
glVertex3f(body.radius * ca, body.radius * sa, cylHalfHeight)
glVertex3f(body.radius * ca, body.radius * sa, -cylHalfHeight)
glEnd()
glTranslated(0, 0, cylHalfHeight)
glutSolidSphere(body.radius, CAPSULE_SLICES, CAPSULE_STACKS)
glTranslated(0, 0, -2.0 * cylHalfHeight)
glutSolidSphere(body.radius, CAPSULE_SLICES, CAPSULE_STACKS)
glPopMatrix()

#clase de definicion del bipedo
class Bipedo():
def __init__(self, world, space, dt):
self.world = world
self.space = space
self.bodies = []
```

```
self.joints = []
self.dt=dt
self.CM_X_ANT = 0
self.CM_Z_ANT = 0
self.TorsoAngle_Z_ANT = 0
self.TorsoAngle_X_ANT = 0

#primer Link D2 (Inicio-Final, radio, masa)
self.linkD2 = self.addLink((0,0,0), (0,0.407,0), 0.04, 2.76,
'pantorrilla Derecha')
#segundo Link D3 (Inicio-Final, radio, masa)
self.linkD3 = self.addLink((0,0.407,0), (0,0.77,0), 0.06, 6.86,
'pierna Derecha')
#joint j2 (rodilla)
self.joint2 = self.addHingeJoint(self.linkD2, self.linkD3,
(0,0.407,0), (0,0,1), -0.8,-0.01)

#quinto Link D5 (Inicio-Final, radio, masa)
self.linkD5 = self.addLink((0,0.83,0), (0,0.83,-0.35), 0.04, 2.37,
'cadera')
#joints j3 y j4
self.linkB3_4 = self.addLink((0,0.79,0), (0,0.83,0), 0.06, 0.1,
'intCaderaDerecho')
self.joint3 = self.addHingeJoint(self.linkD3, self.linkB3_4,
(0,0.79,0), (0,0,1))
self.joint4 = self.addHingeJoint(self.linkB3_4, self.linkD5,
(0,0.83,0), (0,1,0))

#septimo Link D7 (Inicio-Final, radio, masa)
```

```
self.linkD7 = self.addLink((0,0.407,-0.35), (0,0.77,-0.35),
0.06, 6.86, 'pierna Izquierda')
#octavo Link D8 (Inicio-Final, radio, masa)
self.linkD8 = self.addLink((0,0,-0.35), (0,0.407,-0.35),
0.04, 2.76, 'pantorrilla Izquierda')

#joints j5 y j6
self.linkB5_6 = self.addLink((0,0.83,-0.35), (0,0.79,-0.35),
0.06, 0.1, 'intCaderaIzquierdo')

#cambio el anterior por:
self.joint5 = self.addHingeJoint(self.linkB5_6, self.linkD5,
(0,0.83,-0.35), (0,1,0))
self.joint6 = self.addHingeJoint(self.linkD7, self.linkB5_6,
(0,0.79,-0.35), (0,0,1))

#joint j7 (rodilla)
self.joint7 = self.addHingeJoint(self.linkD8, self.linkD7,
(0,0.407,-0.35), (0,0,1), -0.8,-0.01)

#noveno link D9 (Inicio-Final, radio, masa)
self.linkD9 = self.addLink((0,0.79,-0.175), (0,1.59,-0.175), 0.05,
 46.48, 'Torso')
print self.linkD9.getQuaternion()

#joint j9
self.joint9 = self.addHingeJoint(self.linkD9, self.linkD5,
(0,0.79,-0.175), (1,0,0))
```

```
#y los pies...
self.linkPie1 = self.addLink((-0.02,0,0), (0.20,0,0), 0.04, 1.2,
'pieDerecho')
#joint j1
self.joint1 = self.addHingeJoint(self.linkPie1, self.linkD2,
(0,0,0), (0,0,1))
#self.joint1 = self.addHingeJoint(ode.environment, self.linkD2,
(0,0,0), (0,0,1))

self.linkPie2 = self.addLink((-0.02,0,-0.35), (0.20,0,-0.35),
0.04, 1.2, 'pieIzquierdo')
#joint j8
self.joint8 = self.addHingeJoint(self.linkPie2, self.linkD8,
(0,0,-0.35), (0,0,1))



def addLink(self, p1, p2, radio, masa, nombre=''):

# Calculo de longitud de cilindro

tamano = dist3(p1, p2)

body = ode.Body(self.world)
M = ode.Mass()
M.setCylinderTotal(masa, 3, radio, tamano)
body.setMass(M)

# parametros propios para dibujar el cuerpo
body.shape = "capsula"
```

```
body.length = tamano
body.radius = radio

# para colision
geom = ode.GeomCapsule(self.space, radio, tamano)
geom.setBody(body)
geom.nombre=nombre

# definicion de rotacion
za = norm3(sub3(p2, p1))
if (abs(dot3(za, (1.0, 0.0, 0.0))) < 0.7): xa = (1.0, 0.0, 0.0)
else: xa = (0.0, 1.0, 0.0)
ya = cross(za, xa)
xa = norm3(cross(ya, za))
ya = cross(za, xa)
rot = (xa[0], ya[0], za[0], xa[1], ya[1],
za[1], xa[2], ya[2], za[2])

body.setPosition(mul3(add3(p1, p2), 0.5))
body.setRotation(rot)

self.bodies.append(body)
return body



def addHingeJoint(self, body1, body2, anchor, axis,
loStop = -ode.Infinity,hiStop = ode.Infinity):
```

```
joint = ode.HingeJoint(self.world)
joint.attach(body1, body2)
joint.setAnchor(anchor)
joint.setAxis(axis)
joint.setParam(ode.ParamLoStop, loStop)
joint.setParam(ode.ParamHiStop, hiStop)

joint.style = "hinge"
self.joints.append(joint)

return joint

def addUniversalJoint(self, body1, body2, anchor, axis1, axis2,
loStop1 = -ode.Infinity, hiStop1 = ode.Infinity,
loStop2 = -ode.Infinity, hiStop2 = ode.Infinity):

joint = ode.UniversalJoint(self.world)
joint.attach(body1, body2)
joint.setAnchor(anchor)
joint.setAxis1(axis1)
joint.setAxis2(axis2)
joint.setParam(ode.ParamLoStop, loStop1)
joint.setParam(ode.ParamHiStop, hiStop1)
joint.setParam(ode.ParamLoStop2, loStop2)
joint.setParam(ode.ParamHiStop2, hiStop2)

joint.style = "univ"
self.joints.append(joint)
```

```
return joint

def setTorques(self, torque):
"""
Funcion que agrega torques en cada una de las junturas del robot
torque es un vector de 9 elementos
"""
self.joint1.addTorque(torque[0])
self.joint2.addTorque(torque[1])
self.joint3.addTorque(torque[2])
self.joint4.addTorque(torque[3])
self.joint5.addTorque(torque[4])
self.joint6.addTorque(torque[5])
self.joint7.addTorque(torque[6])
self.joint8.addTorque(torque[7])
self.joint9.addTorque(torque[8])

def getAngles(self):
t=[0,0,0,0,0,0,0,0,0]
t[0]=self.joint1.getAngle()
t[1]=self.joint2.getAngle()
t[2]=self.joint3.getAngle()
t[3]=self.joint4.getAngle()
t[4]=self.joint5.getAngle()
t[5]=self.joint6.getAngle()
t[6]=self.joint7.getAngle()
t[7]=self.joint8.getAngle()
t[8]=self.joint9.getAngle()
return t
```

```
def getAnglesRate(self):
t=[0,0,0,0,0,0,0,0,0]
t[0]=self.joint1.getAngleRate()
t[1]=self.joint2.getAngleRate()
t[2]=self.joint3.getAngleRate()
t[3]=self.joint4.getAngleRate()
t[4]=self.joint5.getAngleRate()
t[5]=self.joint6.getAngleRate()
t[6]=self.joint7.getAngleRate()
t[7]=self.joint8.getAngleRate()
t[8]=self.joint9.getAngleRate()
return t

def calcularPosCM(self):
CM_X = 0
CM_Z = 0
MT = 0
for bodi in self.bodies:
pos=bodi.getPosition()
mas=bodi.getMass().mass
CM_X+=pos[0]*mas
CM_Z+=pos[2]*mas
MT+=mas
if MT!=0:
CM_X = CM_X/MT
CM_Z = CM_Z/MT
self.CM_X=CM_X
self.CM_Z=CM_Z
```

```
#print self.linkD9.getQuaternion()
#print self.linkD9.getRotation()

def getPosCM_X(self):
return self.CM_X

def getPosCM_Z(self):
return self.CM_Z

def calcularVelCM(self):
self.velCM_X = (self.CM_X - self.CM_X_ANT)/self.dt
self.velCM_Z = (self.CM_Z - self.CM_Z_ANT)/self.dt
self.CM_X_ANT = self.CM_X
self.CM_Z_ANT = self.CM_Z

def getVelCM_X(self):
return self.velCM_X

def getVelCM_Z(self):
return self.velCM_Z

def calcularTorsoAngles(self):
'''
Funcion que calcula los angulos del torso con respecto al suelo
'''
a=self.linkD9.getRotation()
#para z
vlx=(a[0], a[3], a[6])
vpx=(a[0], 0, a[6])
```

```
vpx=norm3(vpx)
punto=dot3(vlx,vpx)
if punto>1:
punto=1
elif punto<-1:
punto=-1
self.TorsoAngle_Z = sign(a[3])*acos(punto)
#para x
vly=(a[1], a[4], -a[7])
vpy=(a[1], 0, -a[7])
vpy=norm3(vpy)
punto=dot3(vly,vpy)
if punto>1:
punto=1
elif punto<-1:
punto=-1
self.TorsoAngle_X = -sign(a[4])*acos(punto)

def getTorsoAngle_Z(self):
return self.TorsoAngle_Z

def getTorsoAngle_X(self):
return self.TorsoAngle_X

def calcularVelTorsoAngles(self):
self.velTorsoAngle_Z = (self.TorsoAngle_Z
- self.TorsoAngle_Z_ANT)/self.dt
self.velTorsoAngle_X = (self.TorsoAngle_X
- self.TorsoAngle_X_ANT)/self.dt
```

```
self.TorsoAngle_Z_ANT = self.TorsoAngle_Z
self.TorsoAngle_X_ANT = self.TorsoAngle_X

def getVelTorsoAngle_Z(self):
return self.velTorsoAngle_Z

def getVelTorsoAngle_X(self):
return self.velTorsoAngle_X

def getDistanciaPieDer_CM_Front(self):
#centro de masa del bipedo
cmx=self.getPosCM_X()
cmz=self.getPosCM_Z()
#diferencia con respecto al pie (relativo)
cmr=sub3([cmx,0,cmz], self.linkPie1.getPosition())
#eliminar componente en y
cmrv=[cmr[0],0,cmr[2]]
#vector de frente
v = self.linkD9.getRotation()
front=norm3([v[0],0,v[6]])

#distancia entre pie y cm (con respecto al frente)
return dot3(cmrv,front)

def getDistanciaPieIzq_CM_Front(self):
#centro de masa del bipedo
cmx=self.getPosCM_X()
cmz=self.getPosCM_Z()
#diferencia con respecto al pie (relativo)
```

```
cmr=sub3([cmx,0,cmz], self.linkPie2.getPosition())
#eliminar componente en y
cmrv=[cmr[0],0,cmr[2]]
#vector de frente
v = self.linkD9.getRotation()
front=norm3([v[0],0,v[6]])

#distancia entre pie y cm (con respecto al frente)
return dot3(cmrv,front)



def near_callback(args, geom1, geom2):
global soportes

if (ode.areConnected(geom1.getBody(), geom2.getBody())):
return

# Verificar si los dos objetos realmente estan colisionando
contacts = ode.collide(geom1, geom2)

if geom1.nombre=="pieDerecho" and geom2.nombre=="plano"
and len(contacts)>0:
soportes[0]=1
elif geom1.nombre=="pieIzquierdo" and geom2.nombre=="plano"
and len(contacts)>0:
soportes[1]=1
# create contact joints
world, contactgroup = args
```

```
for c in contacts:
c.setBounce(0.3)
c.setMu(500)
# 0-5 = very slippery, 50-500 = normal,
#5000 = very sticky (Recomendado por ODE)
j = ode.ContactJoint(world, contactgroup, c)
j.attach(geom1.getBody(), geom2.getBody())

# funcion de captura de teclado opengl
def onKey(c, x, y):
"""GLUT keyboard callback."""

global SloMo, Paused, cam_z, cam_y, cam_x

# Velocidad de simulacion (solo para despliegue)
if c >= '0' and c <= '9':
SloMo = 4 * int(c) + 1
# pausa simulacion
elif c == 'p' or c == 'P':
Paused = not Paused
# salir
elif c == 'q' or c == 'Q':
sys.exit(0)
elif c == 'd':
cam_z-=0.05
elif c == 'a':
cam_z+=0.05
elif c == 'w':
```

```
cam_y+=0.05
elif c == 's':
cam_y-=0.05
elif c == 'o':
cam_x-=0.05
elif c == 'l':
cam_x+=0.05
elif c == ' ':
#print bipedo.linkD9.getRotation()
a=bipedo.linkD9.getRotation()
vlx=(a[0], a[3], a[6])
vpx=(a[0], 0, a[6])
vpx=norm3(vpx)
print 'res X'
ang=sign(a[3])*acos(dot3(vlx,vpx))
print ang

print 'res Y'
vly=(a[1], a[4], -a[7])
vpy=(a[1], 0, -a[7])
vpy=norm3(vpy)
ang=sign(a[4])*acos(dot3(vly,vpy))
print -ang

def onDraw():
"""GLUT render callback."""

prepare_GL()
```

```
for b in bodies:
draw_body(b)
for b in bipedo.bodies:
draw_body(b)

draw_CM_Box(bipedo.getPosCM_X(), bipedo.getPosCM_Z())
draw_Plane()
draw_Global_Axes((-0.5,-0.09,0.5))
draw_Local_Axis(0,bipedo.linkD9)

glutSwapBuffers()

def modConsignasPieDerecho(consignas):
'''
En esta funcion se hacen las modificaciones de las
consignas con el fin de corregir el balance
'''
#distancia entre pie derecho y cm (con respecto al frente)
distCM_F=bipedo.getDistanciaPieDer_CM_Front()
consignas[5]=consignas[5] + 0.3 * distCM_F +
0.1 * bipedo.getVelCM_X()
if consignas[5]>1:
consignas[5]=1
elif consignas[5]<-1:
consignas[5]=-1
#print 'pieDerecho',distCM_F, consignas[5]
return consignas

def modConsignasPieIzquierdo(consignas):
```

```
,,,
En esta funcion se hacen las modificaciones de las consignas
con el fin de corregir el balance
,,,
#distancia entre pie derecho y cm (con respecto al frente)
distCM_F=bipedo.getDistanciaPieIzq_CM_Front()
consignas[2]=consignas[2] + 0.3 * distCM_F +
0.1 * bipedo.getVelCM_X()
if consignas[2]>1:
consignas[2]=1
elif consignas[2]<-1:
consignas[2]=-1
#print 'pieIzquierdo',distCM_F, consignas[2]
return consignas



def modConsignas2pies(consignas):
,,,
En esta funcion se hacen las modificaciones de las consignas
 con el fin de corregir el balance
,,,
#distancia entre pie derecho y cm (con respecto al frente)
distCM_F=bipedo.getDistanciaPieDer_CM_Front()
consignas[5]=consignas[5] - 1.9 * distCM_F -
0.1 * bipedo.getVelCM_X()
consignas[2]=consignas[2] - 1.9 * distCM_F -
0.1 * bipedo.getVelCM_X()
if consignas[5]>1:
consignas[5]=1
```

```
elif consignas[5]<-1:
consignas[5]=-1
if consignas[2]>1:
consignas[2]=1
elif consignas[2]<-1:
consignas[2]=-1
#print 'dos',distCM_F,consignas[2],consignas[5]
return consignas

def getConsignas():
global muestra
consignas = [Qd1[muestra],Qd2[muestra],Qd3[muestra],
Qd4[muestra],Qd5[muestra],Qd6[muestra],Qd7[muestra],
Qd8[muestra],Qd9[muestra]]
muestra+=4
if (muestra>=maxmuestra-1):
if (repetir==1):
muestra=0
else:
muestra=maxmuestra-2
return consignas

def limitTorques(torques):
for i in range(len(torques)):
if torques[i]>50:
torques[i]=50
return torques

#esta funcion es la encargada de generar la coneccion
```

```
#entre modulos de control y bipedo
def moduloControlTotal():
global consignas_ANT
#controlar los 9 grados de libertad + 1 virtual que es el del torso

consignas = getConsignas()
#consignas = [0.09,-0.07,0,0,0,0,-0.07,0.09,0.05]

#ahora dependiendo de los pies de apoyo
soportes=getSoportes()
if soportes=="dos":
consignas_mod = modConsignas2pies(consignas)
salidaProporcional = micontrolador.proporcional(bipedo.getAngles()+
[bipedo.getTorsoAngle_Z()], consignas_mod+[0],[2000, 2000, 2000,
2000, 2000, 2000, 2000, 2000, 2000, 2000])
salidaDerivativa = micontrolador.derivativo(bipedo.getAnglesRate()+
[bipedo.getVelTorsoAngle_Z()], [6, 6, 6, 6, 6, 6, 6, 6, 6, 6])
salidaControl=micontrolador.restador(salidaProporcional,
salidaDerivativa)
salidaControl[2] = salidaControl[2]-salidaControl[9]
salidaControl[5] = salidaControl[5]-salidaControl[9]
elif soportes=="derecho":
consignas_mod = modConsignasPieDerecho(consignas)
salidaProporcional = micontrolador.proporcional(bipedo.getAngles()+
[bipedo.getTorsoAngle_Z()], consignas_mod+[0],[2000, 2000, 2000,
2000, 2000, 2000, 2000, 2000, 2000, 2000])
salidaDerivativa = micontrolador.derivativo(bipedo.getAnglesRate()+
[bipedo.getVelTorsoAngle_Z()], [6, 6, 6, 6, 6, 6, 6, 6, 6, 6])
salidaControl=micontrolador.restador(salidaProporcional,
```

```
salidaDerivativa)
salidaControl[2] = -salidaControl[5]-salidaControl[9]
elif soportes=="izquierdo":
consignas_mod = modConsignasPieIzquierdo(consignas)
salidaProporcional = micontrolador.proporcional(bipedo.getAngles()+
[bipedo.getTorsoAngle_Z()], consignas_mod+[0],[2000, 2000, 2000,
2000, 2000, 2000, 2000, 2000, 2000, 2000])
salidaDerivativa = micontrolador.derivativo(bipedo.getAnglesRate()+
[bipedo.getVelTorsoAngle_Z()], [6, 6, 6, 6, 6, 6, 6, 6, 6, 6])
salidaControl=micontrolador.restador(salidaProporcional,
salidaDerivativa)
salidaControl[5] = -salidaControl[2]-salidaControl[9]
else:
salidaProporcional = micontrolador.proporcional(bipedo.getAngles()+
[bipedo.getTorsoAngle_Z()], consignas_ANT+[0],[2000, 2000, 2000,
2000,2000, 2000, 2000, 2000, 2000, 2000])
salidaDerivativa = micontrolador.derivativo(bipedo.getAnglesRate()+
[bipedo.getVelTorsoAngle_Z()], [6, 6, 6, 6, 6, 6, 6, 6, 6, 6])
salidaControl=micontrolador.restador(salidaProporcional,
salidaDerivativa)

bipedo.setTorques(limitTorques(salidaControl))
#print salidaControl
consignas_ANT=consignas



def getSoportes():
global soportes
if soportes[0]==1 and soportes[1]==1:
```

```
retorno="dos"
elif soportes[0]==1 and soportes[1]==0:
retorno="derecho"
elif soportes[0]==0 and soportes[1]==1:
retorno="izquierdo"
else:
retorno="ninguno"
soportes=[0, 0]
return retorno

#funcion principal de opengl que permite simulacion ciclica
def onIdle():

global Paused, lasttime, numiter

if Paused:
glutPostRedisplay()
return

t = dt - (time.time() - lasttime)
if (t > 0):
time.sleep(t)

glutPostRedisplay()

for i in range(stepsPerFrame):
# Detectar colisiones y crear grupos de contacto
space.collide((world, contactgroup), near_callback)
```

```
#calcular velocidad y posicion de bipedo
bipedo.calcularPosCM()
bipedo.calcularVelCM()
bipedo.calcularTorsoAngles()
bipedo.calcularVelTorsoAngles()

moduloControlTotal()
# simulacion del bipedo
#world.step(dt / stepsPerFrame / SloMo)
world.step(dt / stepsPerFrame)

numiter += 1

# Eliminar los grupos de contactos (son ligaduras instantaneas)
contactgroup.empty()

lasttime = time.time()

# inicializar GLUT
glutInit([])
glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE)

# create la ventana
x = 0
y = 0
width = 640
height = 480
glutInitWindowPosition(x, y)
glutInitWindowSize(width, height)
```

```
glutCreateWindow("PyODE Simulacion de bipedo")

# crear el mundo de ODE
world = ode.World()
world.setGravity((0.0, -9.81, 0.0))
world.setERP(0.1)
world.setCFM(1E-4)

# crear el espacio para colisiones de ODE
space = ode.Space()

# crear un plano de piso
floor = ode.GeomPlane(space, (0, 1, 0), -0.09)
floor.nombre='plano'
soportes=[0,0]
#lista de cuerpos o junturas
bodies = []

contactgroup = ode.JointGroup()

# parametros iniciales de simulacion
fps = 60
dt = 1.0 / fps
stepsPerFrame = 20
SloMo = 1
Paused = False
lasttime = time.time()
numiter = 0
```

```
# crear el bipdeo
bipedo = Bipedo(world, space, dt)

from controlador3 import *
micontrolador=Controlador()
consignas_ANT=[0,0,0,0,0,0,0,0,0,0]

cam_x=1.5
cam_y=2.0
cam_z=1.5

# fijar GLUT callbacks
glutKeyboardFunc(onKey)
glutDisplayFunc(onDraw)
glutIdleFunc(onIdle)

# GLUT ciclo principal
glutMainLoop()
```

A.2. Codigo fuente del contolador

El código fuente del controlador es simplemente el mismo código entregado por la aplicación Simoro, con pequeñas modificaciones para permitir su interpretación en el lenguaje Python. Además se generó una clase que contiene ese controlador acompañado de controles proporcionales y derivativos.