Proyecto Fin de Carrera
Ingeniería Industrial

Cálculo de fuerzas de contacto pie-suelo por esferas mediante el método de Hunt-Crossley en la marcha humana

Autor: Francisco de Asís Borrego Díaz

Tutor: Dra. Juana María Mayo Núñez

Dpto. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

PROYECTO FIN DE CARRERA

INGENIERÍA INDUSTRIAL

Cálculo de fuerzas de contacto pie-suelo por esferas mediante el método de Hunt-Crossley en la marcha humana

Autor:
Francisco de Asís Borrego Díaz

Directora de proyecto:
Dra. Juana María Mayo Núñez

Dpto. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera: Cálculo de fuerzas de contacto pie-suelo por esferas mediante el método de Hunt-Crossley en la marcha humana

Autor: Francisco de Asís Borrego Díaz
Tutor: Dra. Juana María Mayo Núñez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Resumen

En este proyecto se trata el ajuste y cálculo de las fuerzas de contacto en la marcha humana mediante esferas siguiendo el modelo de Hunt-Crossley para su uso en cálculos de Optimización Estática y Control Muscular Computerizado del programa OpenSim. Para ello un programa de cinemática multi-cuerpo fue programado en Matlab para ser usado junto con una modificación de un programa hecho con la API de OpenSim. También se determinó mediante optimización la posición de las esferas en el pie.

Agradecimientos

Agradezco a Juana Mayo su atención y consejos durante la elaboración del proyecto y a Pedro Moreira y Ayman Habib la ayuda prestada desinteresadamente.

Índice general

Índice de figuras

Índice de tablas

Lista de Abreviaturas

CMC	**C**ontrol **M**uscular **C**omputerizado. **(CMC)** en inglés.
OE	**O**ptimización **E**stática. **(SO)** en inglés.
CI	**C**inemática **I**nversa. **IK** en inglés.
ARR	**A**lgoritmo de **R**educción de **R**esiduales. **(RRA)** en inglés.
IGU	**I**nterfaz **G**ráfica de **U**suario. **(GUI)** en inglés.
IPA	**I**nterfza de **P**rogramación de **A**plicaciones. **(API)** en inglés.
PFC	**P**royecto de **F**in de **C**arrera.

Para mis padres...

Capítulo 1

Introducción

1.1. Objetivo del proyecto

El objetivo principal de este Proyecto de Fin de Carrera es el cálculo de las fuerzas de contacto que se producen en la interacción pie-suelo en la marcha humana y su posterior empleo en simulaciones mediante el programa de dinámica multicuerpo centrado en la biomecánica OpenSim, con el objetivo de averiguar si se puede sustituir la medición experimental de las fuerzas de contacto mediante plataformas de fuerza u otro métodos por la aproximación de estas fuerzas según la posición y velocidad de los pies del modelo.
Para ello se adoptará como referencia un modelo virtual humano del cual se conocen su valores cinemáticos a lo largo de una marcha medida experimentalmente, a ese modelo virtual se le acoplarán unas esferas en los pies siguiendo un patrón de esferas concreto tanto en el número, posición, radio y otros parámetros referentes a estas esferas y se medirá la penetración de esas esferas con el suelo durante la marcha. Una ecuación relacionará la penetración de esas esferas con la fuerza de contacto producida.
Con este modelo de esferas se realizarán varias simulaciones empleando los métodos disponibles del programa OpenSim, y se compararán los resultados con aquellas simulaciones donde no se empleó el modelo de contacto de esferas sino los valores medidos en laboratorio mediante métodos experimentales. Respecto a OpenSim se prestará especial atención en los métodos de optimización estática y control muscular computarizado, que permiten obtener las fuerzas y activaciones musculares necesarias para permitir la cinemática indicada, y en el caso del segundo método, poder predecir una cinemática alternativa.

Contenido de cada capítulo

El capítulo actual alberga la declaración de objetivos del proyecto, una exploración de la vanguardia de la técnica en el campo del estudio de la marcha humana y una leve explicación sobre la marcha humana.
En el segundo capítulo se tratará en detalle el programa OpenSim, tanto las características generales de sus modelos y sus componentes como sus métodos, indicando la base teórica de estos métodos y se detallará finalmente el modelo esqueleto-muscular empleado en este proyecto.
El lugar de las esferas del modelo de contacto será estudiado en el tercer capítulo, concretando el método de optimización directo empleado para obtenerlo como también la función a optimizar y las variables de ésta.
En el cuarto capítulo se describen los dos programas desarrollados para llevar a cabo los cálculos deseados siendo el primero de ellos de ellos una variación de las aplicaciones de consola de OpenSim modificado para interactuar con el segundo programa, una aplicación compilada en Matlab que calcula las fuerzas de contacto a partir de las coordenadas generalizadas de un modelo en marcha.

Los resultados obtenidos se muestran en el quinto capítulo, tanto de los métodos empleados en el tercer como los mostrados en el cuarto capítulo.
Por último, en el sexto capítulo se muestran las conclusiones.

1.2. La biomecánica, orígenes y usos

1.2.1. Biomecánica

Uno de los campos de la mecánica con más potencial de desarrollo de las últimas décadas es el campo de la biomecánica. Una posible definición de la biomecánica es:

> El estudio de los modelos, fenómenos y leyes que sean relevantes en el movimiento de un sistema biológico a través de los métodos de la mecánica [1] [12].

. Tres aspectos deben ser considerados para estudiar el movimiento:

- el *control* del movimiento, relacionado con el ámbito neurofisiológico. En este proyecto no se profundizará en este aspecto.
- la *estructura* del cuerpo que se mueve, un sistema complejo formado por músculos, huesos y tendones en el caso del cuerpo humano, del cual se observarán sus características mecánicas.
- las *fuerzas* que producen ese movimiento, las cuales pueden ser *externas* o *no actuadas*, como son la gravedad, el viento o el contacto con otros cuerpos. También pueden ser *internas* o *actuadas*, como son las fuerzas musculares propias del ser vivo.

1.2.2. Resumen histórico de la biomecánica

Uno de los pioneros en estudiar el campo fue Aristóteles de Estagira (384-322 a. C.), alumno de Platón. Fue el primero que describió la estructura corporal y el movimiento de pequeños animales, además de establecer leyes y clases de movimiento, clasificando a los animales según ese movimiento. [3].
Mucho después, el napolitano Giovanni Alfonso Borelli (1608-1679 d. C.), con un enfoque actualizado, describió los movimientos anatómicos, la influencia de las fuerzas, el mecanismo de palanca y la influencia del centro de masas y la gravedad en los seres vivos, a través de las leyes de la mecánica [4].

A finales del siglo XIX, Étienne-Jules Marey empleó técnicas cinematográficas para investigar científicamente el movimiento. Consiguió pefeccionar una cámara inventada antes por Jules Jannsen capaz de tomar doce fotos por segundo, que empleó para analizar el movimiento de los seres vivos, creando lo que él llamó *cronofotografía*. También investigó, en lo que concierne a la biomecánica, la marcha mediante muchos artilugios que inventó, midiendo desde el balanceo al andar al movimiento de los centros de masa. [17]. En la década de los 60 la biomecánica cobró importancia a partir de diferentes congresos en Leipzig y Zurich. La revista *Journal of Biomechanics* nació en 1968, desde entonces el conocimiento del campo ha crecido constantemente, ayudado por las mejoras en la tecnología que permiten novedosas maneras de estudio, a la vez que la informática permite hacer cálculos más complejos.

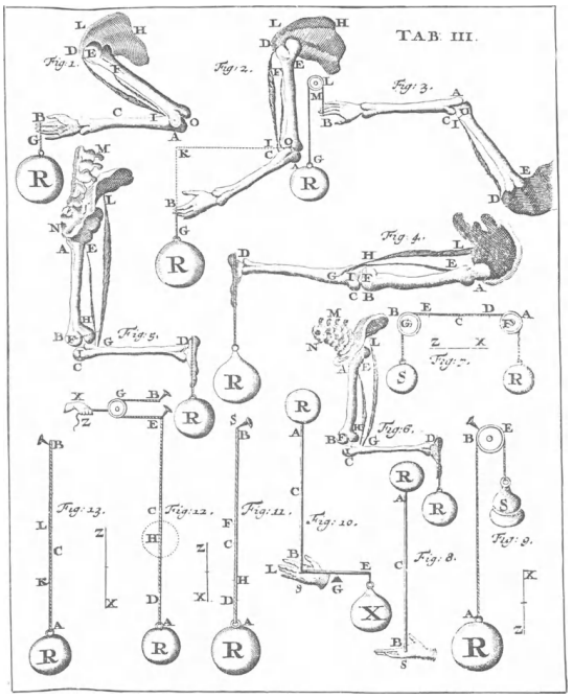

FIGURA 1.1: Ejemplo de una de las tablas de figuras de la obra *De Motu Animalium* de Borelli

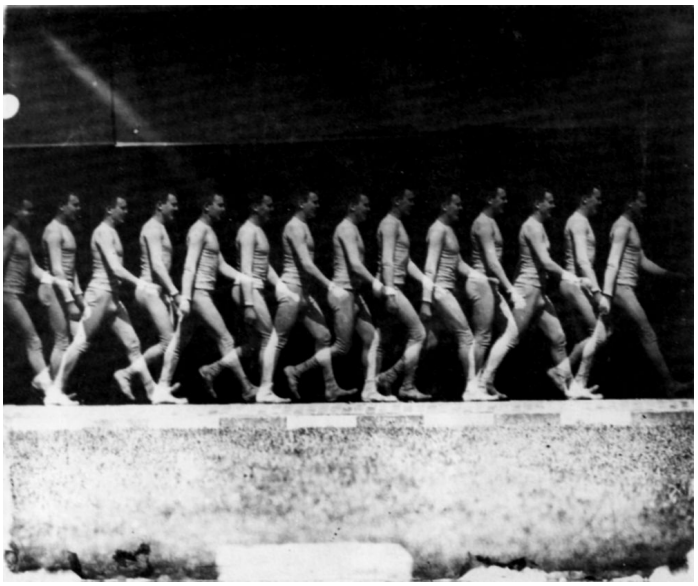

FIGURA 1.2: Estudio gráfico de la marcha

1.2.3. Aplicaciones

Las dos principales áreas de estudio de la biomecánica son la medicina y el deporte. Por el lado de la medicina, la biomecánica se emplea para analizar los problemas y defectos corporales relacionados con la locomoción corporal y puede encontrar soluciones a estos problemas en forma de prótesis y otros dispositivos, además de ayudar a prevenir daños. En el campo deportivo la biomecánica busca incrementar los límites del rendimiento humano, especialmente en aquellos aspectos donde la técnica sea más importante que la potencia o la resistencia, ayudando a los deportistas a encontrar maneras de mejorar sus resultados modificando sus movimientos una vez han sido estudiados por otros optimizados dentro de sus capacidades, a la vez que evitan posibles lesiones. También ha servido la biomecánica como base para el desarrollo de equipamiento deportivo diseñado para obtener mejores resultados, como patines de hielo, zapatillas de deporte o jabalinas. Por último cabe añadir lo importante que el estudio de la marcha humana es para el campo de la robótica y la animación por ordenador. El análisis de la cinemática y dinámica de la marcha permite comprender sus causas y replicarlas, permitiendo así crear robots y modelos más semejantes a nosotros.

1.3. Estudio de la vanguardia del arte

1.3.1. Modelos de pie

Los diferentes modelos biomecánicos consideran el pie humano según muy diferentes aproximaciones. Las más simples omiten el pie en sí y crean una unión directa del suelo y el tobillo. Otros consideran el pie como un sólido rígido y desprecian las deformaciones y consecuencias de éstas que se produjeran durante la marcha. Estos modelos sólo suelen considerar la fase de apoyo de la marcha.
Otras aproximaciones crean una restricción que une al pie cinemáticamente con el suelo durante la parte relevante de la marcha.

Uno de los primeros modelos que no reducían el pie a una restricción cinemática fue presentado por Meglan [18], que desarrolló la interacción con el suelo por elementos viscoelásticos, basándose como referencia en datos medidos con plantillas con sensores colocadas en el talón, aunque no consiguió resultados suficientemente satisfactorios debido a errores de medición y a tener un modelo con no la suficiente holgura en sus ecuaciones.

Gilchrist y Winter [11] describieron un modelo de pie tridimensional de dos partes mediante elementos viscoelásticos, que emplearon para describir el el ciclo completo. El modelo poseía nueve elementos muelle-amortiguador verticales colocados por la línea media del pie que medían la fuerza de contacto, más un elemento muelle-amortiguador torsional para la flexión de la parte meta-tarsial, obteniendo resultados satisfactorios.

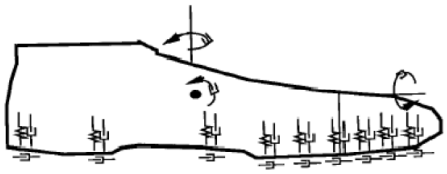

FIGURA 1.3: Representación del modelo de Gilchrist y Winter.

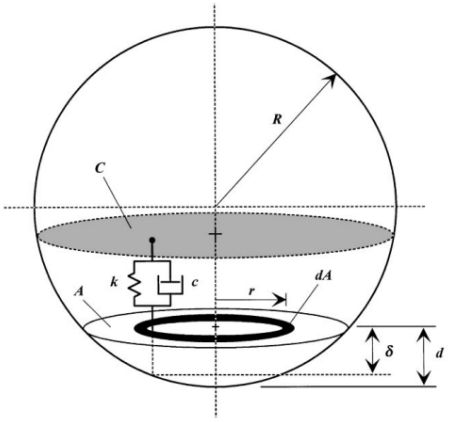

FIGURA 1.4: Esfera viscoelástica empleada en el modelo de Güller y colaboradores.

Una esfera viscoelástica fue el elemento que Güller *et al.* [7] emplearon para modelar el pie, determinando que era posible modelar la superficie del pie durante la locomoción mediante ella para obtener las fuerzas de contacto.

Millard *et al.* [19] en 2009, partiendo del modelo biomecánico humano de Peasgood *et al.* [22], probaron diferentes modelos de pie, indicando las fuerzas de contacto según la teoría de Hunt-Crossley [14], con el objetivo de realizar simulaciones de dinámica directa de marcha de varios pasos. El primer modelo mostraba el pie como un único sólido rígido con dos esferas de contacto, despreciando la presión que pudiera realizar el suelo en los metatarsos. Sus resultados resultaron ser mejores en simulaciones de marcha lenta, ya que la presión de la parte metatarsial es menos relevante en esa marcha.
El segundo modelo mostraba el pie formados por dos sólidos rígidos, el anterior con dos esferas de contacto y el posterior con una sólo, añadiendo un grado de libertad procedente de los dedos del pie al modelo. Sus resultados empeoraron para simulaciones de marcha lenta, pero mejoraron considerablemente en marcha rápida. También experimentaron con fuerzas de contacto tangenciales diferentes a las de Coulomb, probando un modelo de fuerzas de fricción de cepillo (*bristle*).

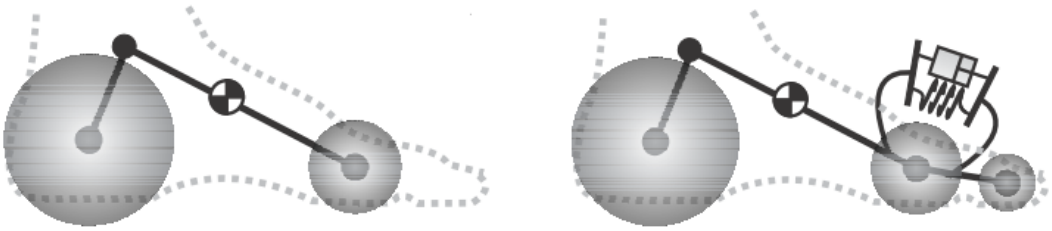

FIGURA 1.5: Los dos modelos de pie probados por Millard *et al.*, el primero tiene el pie como un sólido rígido y el segundo divide en pie en dos partes rígidas.

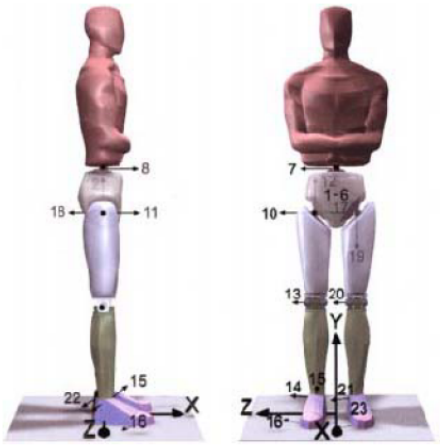

FIGURA 1.6: Modelo humano HAT de Anderson y Pandy

1.3.2. Modelos biomecánicos del cuerpo

Los modelos de marcha humana más sencillos modelan la marcha como una variedad del mecanismo de péndulo doble invertido [10], siendo éstos modelos pasivos.

Los modelos más similares a la mecánica humana pueden clasificarse de varias maneras. Una forma de clasificar distingue modelos parciales y los de cuerpo completo. Los modelos parciales estudian una parte concreta del cuerpo, sea una extremidad como un brazo o una articulación como la rodilla [28]. Los modelos completos adaptan el cuerpo completo, aunque pueden reducir o simplificar su fisionomía o agrupar partes de él para facilitar los cálculos hecho en ellos. Los modelos tratados en este proyecto pertenecen en ambos casos a la segunda clase.

Anderson y Pandy en 2001 [2] crearon un modelo corporal neuromusculoesqueletal formado por 10 sólidos con 23 grados de libertad. La cabeza, brazos y torso (*Head*, *Arms* y *Torso*, *HAT*) fueron unidos en un simple sólido rígido, unido a la pelvis mediante una articulación de rótula, dada su relativa poca importancia para la dinámica de la marcha.

Wojtyra [31] ideó otro modelo humano centrado en la marcha para el uso de sistemas de simulación multicuerpo (ADAMS en su caso concreto), formado por ocho cuerpos rígidos y 21 grados de libertad. El modelo era capaz de resolver el problema de dinámica directa y siguiendo un bucle cerrado sencillo de realimentación estabilizaba su marcha. Modelaba los pies como un sólido rígido con 5 vectores de contacto con el suelo distribuidos en la planta del pie, los cuales reportaban una fuerza según penetraban el suelo. Además, el modelo de Wojtyra permitía tener una masa tambaleante o *wobbling mass* asociada que reflejaba las inercias de los tejidos orgánicos, lo que hace a su modelo destacar frente a otros que consideraban los segmentos corporales únicamente como sólidos rígidos.

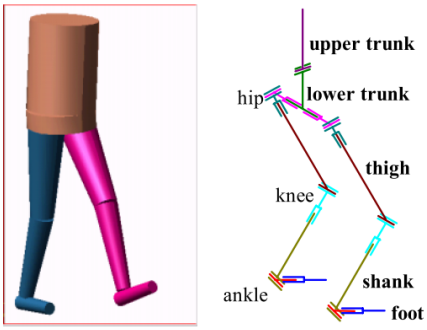

FIGURA 1.7: Modelo de Wojtyra y su representación esquemática.

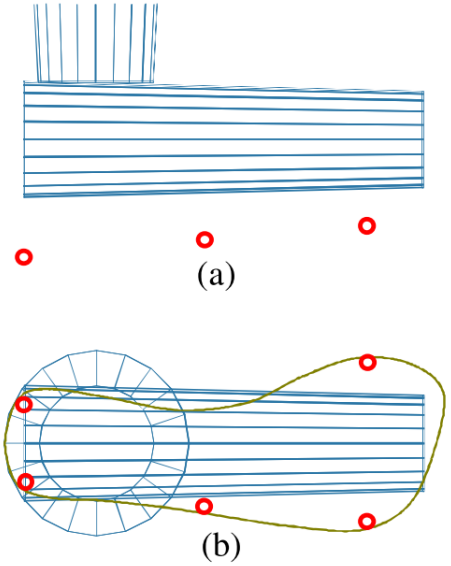

FIGURA 1.8: Vistas del pie del modelo de Wojtyra, con sus 5 puntos de contacto.

1.4. La marcha humana

El ciclo de marcha o zancada

Para este proyecto es importante identificar cada una de las fases de la marcha humana para una mejor identificación de ésta. El ciclo comenzará con el sujeto ya en movimiento, con las extremidades abiertas de manera similar a un compás y la extremidad derecha a punto de hacer contacto en el suelo con el retropié. Un ciclo completo devolverá a la extremidad en la misma posición inicial, con la otra extremidad realizando el mismo movimiento con medio ciclo de desfase. El ciclo de zancada de marcha humana se divide en 2 fases [26], la fase de apoyo y la fase de oscilación, que toman por referencia la posición de una pierna respecto al suelo y el torso, la pierna es en el caso de este proyecto, la derecha. Una manera alternativa de observar las fases de la marcha es mediante el apoyo de ambos pies, distinguiendo así 4 fases, 2 de apoyo bipodal y una de apoyo monopodal para cada pierna.

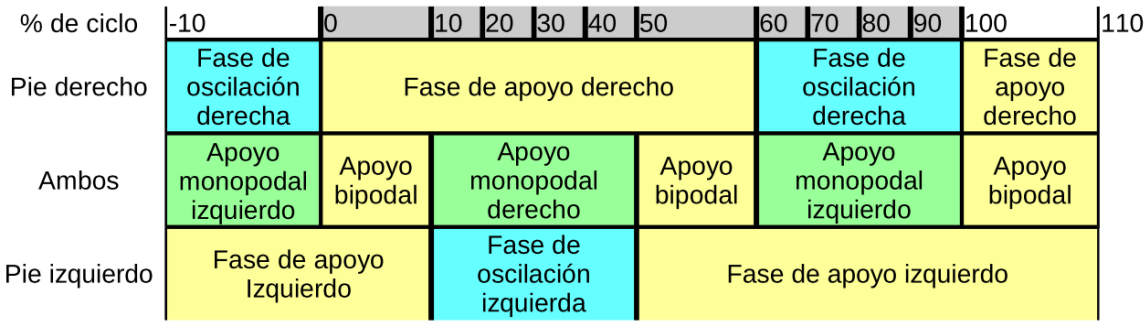

FIGURA 1.9: Diagrama de fases de la marcha

La fase de apoyo

Durante esta fase el pie se encontrará transmitiendo las fuerzas del cuerpo al suelo, sirviendo de soporte al cuerpo, sin deslizar y retrasándose respecto al cuerpo mientras éste avanza. Representa un aproximadamente un 60 % del tiempo del ciclo.

Etapa de contacto inicial Representa el momento en que comienza el contacto del talón con el suelo, mientras los metatarsianos del otro pie aún hacen contacto. Se considerará en este proyecto el inicio del ciclo de andadura. Ocupa del 0 % al 2 % del ciclo de marcha.

Etapa de respuesta de apoyo En esta etapa el talón toca el suelo, recibiendo fuerza en dirección delantera con sentido hacia atrás. Entre el 0 % y el 12 % del ciclo transcurre en esta etapa.

Etapa de apoyo medio Aquí tanto el talón como los metatarsianos tocan el suelo, con el pie paralelo a la superficie. La otra extremidad ya ha entrado en la fase de oscilación. La parte entre el 12 % y el 31 % del ciclo transcurre aquí.

Etapa final de apoyo El pie comienza a abandonar el contacto con el suelo, flexionándose y ejerciendo contacto con los metatarsianos. Las fuerzas de contacto impulsan al sujeto hacia delante. Esta etapa comprende del 31 % al 50 % del tiempo del ciclo.

Etapa previa a la oscilación Sólo la parte más delantera del pie toca el suelo mientras el otro pie inicia la etapa de contacto inicial. Esta etapa ocupa del 50 % al 62 % del ciclo.

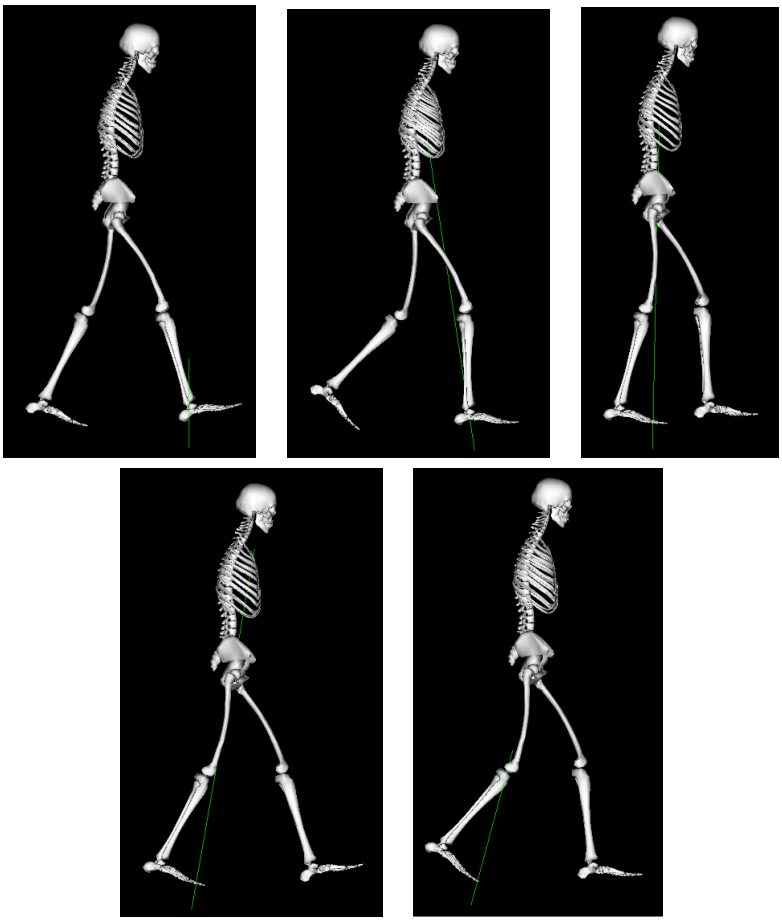

FIGURA 1.10: Las cinco etapas de la fase de apoyo, en orden.

La fase de oscilación

Durante esta fase el pie se encuentra suspendido, adelantando al cuerpo mientras el otro pie realiza el apoyo. El pie se balanceará mientras avanza para poder adoptar la inclinación necesaria para contactar posteriormente el suelo con el talón. Representa el 40 % del tiempo del ciclo restante.

Etapa inicial de oscilación El pie acaba de abandonar el contacto con el suelo. Esta etapa ocupa del 62 % al 75 % del ciclo.

Etapa media de oscilación Parte intermedia del proceso, que ocupa del 75 % al 87 %.

Etapa final de oscilación El pie acaba su balanceo y se prepara a descender. Esta parte final ocupa del 87 % al 100 % del ciclo de marcha.

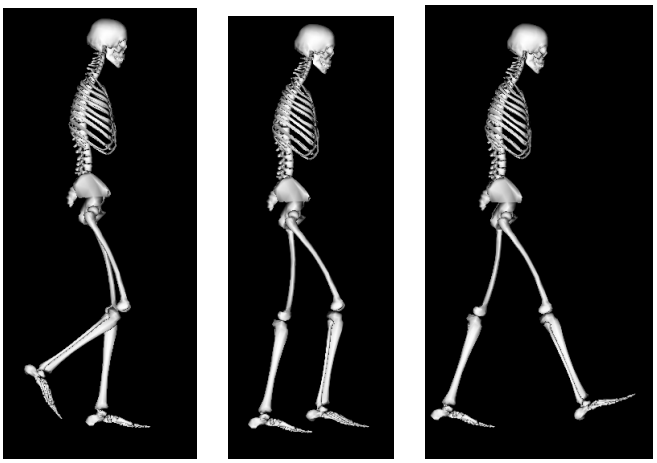

FIGURA 1.11: Las tres etapas de la fase de oscilación, en orden.

Capítulo 2

OpenSim

2.1. OpenSim

OpenSim es una aplicación de código abierto desarrollada por la universidad de Standford en los E.U.A. que emplea los modelos físicos del programa Simbody para calcular la cinemática y dinámica de modelos músculo-esqueletales del cuerpo humano, incluyendo también las ecuaciones musculares. Está construido sobre el motor físico SimTK.

Posee una I.P.A. (Interfaz de Programación de Aplicaciones) que permite programar diferentes simulaciones multi-cuerpo en C++, además de permitir su expansión mediante extensiones (plug-ins) y una interfaz gráfica de usuario (I.G.U.) que permite observar simulaciones de movimiento, además de crear gráficas con los resultados de las simulaciones y permitir su visualización en modelos gráficos de tres dimensiones.

2.1.1. La IPA

Una Interfaz de Programación de Aplicaciones (IPA) es un conjunto de código y especificaciones que permiten la comunicación entre programas de *software*. La IPA de OpenSim, gracias a su estructura abierta y disponible, permite, en diferentes lenguajes de programación acceder al código fuente de OpenSim sin la necesidad de modificar éste, pudiendo emplear sus capacidades e incluso ampliarlas.

La IPA de OpenSim está escrito mediante programación orientada a objetos, en el lenguaje de programación C++. Ordenando el programa según sus capas de funcionamiento, en su nivel más bajo se encuentra el calculador de dinámica multicuerpo Simbody.

Un nivel por encima, están los modelos y sus componentes (cuerpos, fuerzas, controladores, etc). Un nivel más arriba se encuentra la capa de análisis, que posee los solucionadores y optimizadores. En el nivel más alto se encuentra la aplicación en sí, que bien puede ser manejada desde la IGU o mediante programas ejecutables independientes.

No está en el alcance de este proyecto definir cada aspecto del funcionamiento interno de OpenSim, así que no se detallará el funcionamiento de cada capa y sus dependencias.

2.1.2. La IGU

OpenSim también dispone de una interfaz gráfica de usuario (I.G.U.), que facilita el empleo del programa al no requerir del usuario la necesidad de programar. Es capaz de realizar las tareas habituales de OpenSim, explicadas ellas en la sección 2.2, permitiendo editar archivos de configuración directamente, indicar rutas de ficheros y acceder directamente a los resultados producidos.

Dispone de un visualizador de modelos en tres dimensiones con la posibilidad de observar el comportamiento cinemático y dinámico de los modelos, y que permite incluso la edición de algunos parámetros de localización de los modelos, tales como la longitud y

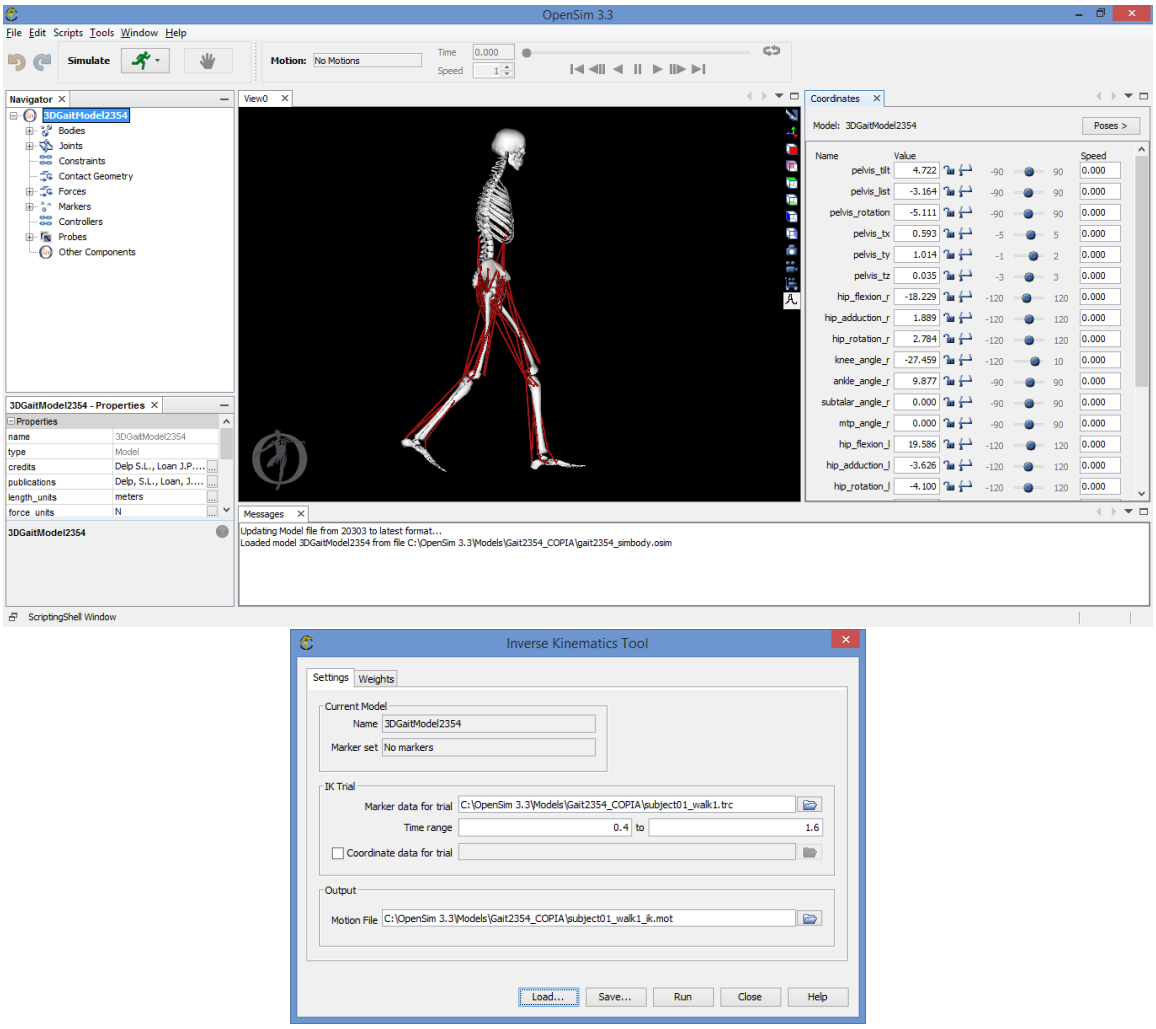

FIGURA 2.1: La IGU de OpenSim y un ejemplo de un cuadro de diálogo de una herramienta.

puntos de agarre de los músculos.
La IGU acepta la inserción de *plugins* para incorporar nuevas funcionalidades al programa, con su correspondiente gestión de éstos. Además incluye también una función para dibujar gráficas a partir de datos de una simulación o datos externos, permitiendo exportar éstas a otros formatos, a la vez que ayuda a visualizar los diferentes resultados de las simulaciones de una forma rápida y precisa.

2.2. Métodos de OpenSim

La información de esta sección puede encontrarse con más detalle en la documentación de OpenSim [9], pero se ha optado por incluir una referencia de algunos métodos de OpenSim, con especial hincapié en los método de Optimzación estática y Control muscular computarizado.

2.2.1. Escalado

Antes de realizar cualquier trabajo sobre un modelo en OpenSim, es necesario ajustar el modelo con el que se quiere trabajar al tamaño de la persona sobre la que se han hecho las pruebas experimentales. Esto se hace con la función de escalado.

Explicación Será necesario partir de un modelo no escalado con marcadores virtuales añadidos (véase el apartado 2.3.5) y realizar mediciones en laboratorio del sujeto con marcadores experimentales. El escalado se basa en la combinación de la distancias medidas entre las localizaciones de los marcadores y factores de escala especificados manualmente. Una vez las dimensiones de los segmentos en el modelo se escalan para que coincidan con las del sujeto, la herramienta de escalado puede mover algunos o todos de estos marcadores virtuales en el modelo para que coincidan con las localizaciones de los marcadores experimentales.

Factores de escala Sea un par de marcadores virtuales en el modelo $\{mv_1, mv_2\}$ coincidentes con los marcadores experimentales del sujeto $\{me_1, me_2\}$, sea nm el total de marcadores en el sujeto y también en el modelo y sea una trayectoria definida por los datos experimentales de marcadores medidos en una captura de movimiento; la distancia entre ambos marcadores en un instante t_i en el sujeto experimental será de_i, existiendo una distancia media $dm_{1,2}$ entre esos dos marcadores para cada uno de los instantes medidos. El factor de escala para esa pareja de marcadores $s_{1,2}$ será el cociente entre la distancia experimental media y la distancia entre los marcadores virtuales en el modelo cuando todos los parámetros generalizados del modelo están en su valor por defecto $dv_{1,2}$.

$$s_{1,2} = \frac{\sum_{i=1}^{n} de_{1,2}(t_i)/n}{dv_{1,2}} \tag{2.1}$$

Este cálculo se repetirá para cada una de las parejas posibles de marcadores. El factor de escala final será la media de todos los factores de escala teniendo en cuenta los pesos p_i que defina el usuario para dar más importancia a unos marcadores que a otros.

$$s = \frac{\sum_{i \in marcadores} \sum_{j \in marcadores} p_i \cdot p_j \cdot s_{i,j} \quad \forall \quad i \neq j}{\sum_{i \in marcadores} \sum_{j \in marcadores} p_i \cdot p_j \quad \forall \quad i \neq j} \tag{2.2}$$

Este factor también se puede indicar manualmente.

Modificación de la geometría del modelo A continuación la herramienta de escalado modifica todos los parámetros geométricos del modelo en función del factor de escala aplicado, incluyendo longitudes de fibras musculares, puntos de ruta muscular y cuerpos. Con ello las masas e inercias de los cuerpos son también modificados, observando también que la masa total del cuerpo equivalga a la masa total del sujeto. Por último los marcadores virtuales se modificarán para que concuerden con aquellos del sujeto en pose estática. Una vez hecho esto se creará un nuevo archivo de modelo ajustado a la escala hecha.

2.2.2. Cinemática Inversa

La cinemática inversa (CI, *inverse kynematics* o IK en inglés) para cada instante de tiempo determinado intenta encontrar una combinación de valores para las coordenadas generalizadas que mejor se ajuste a los valores de los marcadores experimentales. Esta aproximación se realiza empleando el método de los mínimos cuadrados ponderados, que busca

minimizar los errores de marcador y coordenada. El resultado de este método será un conjunto de coordenadas generalizadas y velocidades generalizadas que permiten obtener los movimientos de las uniones articuladas.

Error de marcadores

Un error de marcador es la distancia entre un marcador experimental y su marcador virtual correspondiente en el modelo tiene unas coordenadas generalizadas concretas calculadas mediante CI. A cada marcador se le asocia un peso, dando mayores pesos relativos a aquellos marcadores localizados en regiones con poco deslizamiento entre piel y hueso.

Error de coordenadas

Un error de coordenadas es la diferencia entre el valor de una coordenada obtenida por CI y una coordenada experimental, en el caso de que ésta exista. Las coordenadas experimentales son aquellas que se han medido en laboratorio y que se desea que las obtenidas por CI se parezcan tanto como sea posible a ellas. Por esto se habla de dos tipos de coordenadas, las *coordenadas bloqueadas* son aquellas cuyos valores se fijan para que coincidan con las coordenadas experimentales en la CI; las *coordenadas libres* se comparan a las coordenadas experimentales, aunque no tienen por qué coincidir y a su error se le asigna también un peso.

Ecuación de mínimos cuadrados ponderados

La ecuación a resolver por CI es:

$$min \left[\sum_{i \in marcadores} p_i \| x_i^{exp} - x_i(q) \|^2 + \sum_{j \in coord.libres} p_j (q_j^{exp} - q_j)^2 \right]$$
$$q_j = q_j^{exp} \quad \text{para todas las coordenadas bloqueadas} \tag{2.3}$$

Donde q es un vector de coordenadas generalizadas, x_i^{exp} representa un vector con la localización de un marcador experimental, $x_i(q)$ es la posición de un marcador virtual que depende de las coordenadas generalizadas y p es el peso que se indica para cada marcador o actuador en archivos complementarios. Las coordenadas experimentales son valores que pudieran haberse obtenido en laboratorio a través de métodos que directamente obtuvieran un valor numérico, como podría ser empleando goniómetros. No se usaron las coordenadas experimentales para este proyecto.

Salida de datos de coordenadas

Los datos de trayectoria de coordenadas generalizadas se escribirán en un archivo de almacenamiento de formato *.mot* o *.sto* y tendrán a no ser que se especifique lo contrario la misma frecuencia de aparición en lista que los datos de trayectoria de marcadores. Estos valores son los empleados en métodos subsiguientes que necesiten tomar valores de trayectoria y/o velocidad.

```
Coordinates
version=1
nRows=73
nColumns=24
inDegrees=yes

Units are S.I. units (second, meters, Newtons, ...)
Angles are in degrees.

endheader
time    pelvis_tilt pelvis_list pelvis_rotation pelvis_tx   pelvis_ty   pelvis_tz
      0.40000000   4.84455933   -1.84559258   -0.77772078
      0.41666667   4.82888855   -1.77594552   -0.63778542
      0.43333333   4.73081144   -1.47942768   -0.69109473
      0.45000000   4.84714107   -1.05132429   -0.57441588
      0.46666667   4.70285510   -0.36595029   -0.52555635
      0.48333333   4.64180567   -0.08939722   -0.08196891
      0.50000000   4.51937273    0.34971979   -0.07387465
      0.51666667   4.38978395    0.92470470    0.03629531
      0.53333333   4.25696141    1.51694381    0.36124148
      0.55000000   4.18445821    2.12386543    0.27691401
      0.56666667   4.12353820    2.59671508    0.27526862
      0.58333333   4.00009327    2.93885700    0.91880355
      0.60000000   3.99967947    2.98923541    0.96880997
      0.61666667   3.91026815    2.76716335    1.22848709
      0.63333333   3.87344012    2.76119266    1.74513442
      0.65000000   3.75270076    2.65032546    2.20626710
      0.66666667   3.60942619    2.64169664    2.57127716
      0.68333333   3.67356751    2.66996233    2.66045738
      0.70000000   3.93861893    2.48912340    2.93461066
```

FIGURA 2.2: Muestra de archivo de almacenamiento de coordenadas empleado en este PFC. Es importante notar que los ángulos van en grados.

2.2.3. Dinámica directa

Otro método de OpenSim, como programa de simulación dinámica multicuerpo, es el cálculo de dinámica directa (*forward dynamic* o FD en inglés). La función de la dinámica directa es calcular la cinemática de un modelo a partir de sus fuerzas, teniendo en cuenta los controles de aquellas fuerzas que dispongan de ellos. A diferencia del CMC, el método de dinámica directa es de bucle abierto, es decir, no intenta ajustar sus resultados en función de una entrada deseada.

A partir de la segunda ley de Newton, las aceleraciones del modelo son:

$$\ddot{q} = [M(q)]^{-1} \{\tau + C(q, \dot{q}) + G(q) + F\} \tag{2.4}$$

donde $\ddot{q}$ son las aceleraciones mostradas para coordenadas generalizadas, τ son los pares inducidos en las articulaciones , $C(q, \dot{q})$ son las fuerzas de Coriolis y las centrífugas, $G(q)$ las fuerzas gravitatorias, F representa el resto de fuerzas del modelo, como las musculares y de contacto, $[M(q)]^{-1}$ es la inversa de la matriz de masa, q las coordenadas generalizadas y $\dot{q}$ sus velocidades.

2.2.4. Dinámica inversa

La función de la Dinámica inversa (DI, Inverse Dynamics en inglés o ID) es encontrar qué fuerzas impulsan el movimiento del cuerpo dada una cinemática. Partiendo de la ecuación de equilibro de fuerzas para un sistema con N grados de libertad:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau \tag{2.5}$$

donde $q, \dot{q}, \ddot{q}$ son los vectores de coordenadas generalizadas, velocidades y aceleraciones, respectivamente, $M(q)$ es la matriz de masa, $C(q, \dot{q})$ el vector de fuerzas de Coriolis y centrífugas, $G(q)$ el vector de fuerzas gravitacionales y τ el vector fuerzas generalizadas.

Como todos los miembros de la izquierda son conocidos, se pueden averiguar directamente las fuerzas generalizadas.

2.2.5. Optimización Estática

La optimización estática (OE, *Static Optimization* o SO en inglés) es una ampliación de la dinámica inversa (véase el apartado 2.2.4) que añade a su procedimiento el cálculo de activaciones musculares. Es un método que calcula las fuerzas en cada instante de la marcha sin considerar instantes anteriores ni posteriores, buscando obtener la combinación de esfuerzos musculares más eficientes según algún criterio definido, pues un mismo movimiento puede realizarse con diferentes patrones de activaciones musculares.
El método CMC emplea la optimización estática como una parte de su proceder.

Base teórica

Conocida la cinemática y fuerzas externas del modelo, se pueden plantear las ecuaciones de equilibro dinámico de éste como un sistema de ecuaciones lineales donde las incógnitas son las fuerzas musculares. Dado que existen más incógnitas que ecuaciones (el número de músculos es superior a los g.d.l. del sistema), se trata de un sistema sobredeterminado y un criterio debe de concretarse para decidir qué músculos actúan.
Existen diferentes criterios de optimización posibles que pueden aplicarse para este problema, pudiendo buscar éstos minimizar el coste metabólico, la fatiga muscular, las fuerzas musculares o las fuerzas muscualares normalizadas (donde la fuerza muscular partida por la máxima fuerza muscular posible en el músculo concreto), con estas fuerzas elevadas a un exponente lineal, cuadrático o superior. El trabajo de Pedotti *et al.* [23] mostró que un criterio de minimización de fuerzas normalizadas cuadráticas encontraba una distribución de fuerzas similar a la distribución encontrada experimentalmente mediante electromiografía *(EMG)*. Kaufman *et al.* [15] sugirieron el uso de las activaciones musculares en vez de las fuerzas como variable a minimizar. Combinando ambas ideas, se minimiza la suma de los cuadrados de las activaciones.

$$J = \sum_{m=1}^{n} (a_m)^2 \tag{2.6}$$

En esta función a minimizar, n es el número de músculos del modelo, a_m es la activación del músculo m en un instante concreto de tiempo.
Las activaciones musculares, variables entre cero y uno, definen la fuerza activa definida por un músculo, una vez conocida la longitud de las fibras de este y su velocidad de contracción (más en 2.3.3). El momento provocado por estas fuerzas musculares se puede calcular de dos maneras, una simplificada en función de la constante muscular de fuerza y otra basada en las propiedades internas del modelo muscular.

$$\sum_{m=1}^{n} (a_m F_m^0) r_{m,j} = \tau_j \tag{2.7}$$

Aquí τ_j es un momento generalizado para un músculo, F_m^0 es una fuerza isométrica máxima ideal definida para un músculo m y $r_{m,j}$ es el brazo de la fuerza del músculo m para la articulación j.
Si en vez de plantearse un músculo ideal se plantean las ecuaciones que tienen en cuenta la elongación l_m y velocidad $\dot{l}_m$ de elongación del músculo con las propiedades del músculo (véase el apartado 2.3.3), se tiene:

$$\sum_{m=1}^{n} \left[a_m f \left(F_m^0, l_m, \dot{l}_m \right) \right] r_{m,j} = \tau_j \tag{2.8}$$

Los errores en la detección de los marcadores durante la marcha, los errores en la colocación de los marcadores en el modelo, las imprecisiones en el modelado de cuerpo, sea de inercia o masa, los errores en las mediciones de las fuerzas de contacto y otros errores posibles generan imprecisiones que podrían provocar que las ecuaciones de equilibrio dinámico del modelo no se ajustaran. Para corregirlo, se añaden al modelo fuerzas residuales para cada uno de los 6 grados de libertad que indican la posición y orientación de la pelvis y un par de reserva para cada una de las coordenadas generalizadas asociadas a las articulaciones.
La aparición de estas fuerzas en los resultados finales no es deseable, pues indican que el modelo, por lo motivos indicados previamente no puede seguir la cinemática indicada con sus propios actuadores y necesita fuerzas y pares extra para cumplir sus ecuaciones. Por ello, su capacidad de activación no se limita de 1 a 0 como con los músculos normales, si no que no tiene límite, lo que permite alcanzar unas fuerzas equivalentes $F_{m,eq} = a_m F_m^0$ con valores elevados.

Con la función a minimizar, las fuerzas y las restricciones definidas, el programa resolverá la optimización cuadrática. Una vez se obtienen los resultados del método es conveniente comparar los valores de las fuerzas residuales con los de las fuerzas reales, si su orden es mayor puede indicar que algún aspecto de la simulación está mal planteado o modelado.

Archivos necesarios

Para realizar una optimización estática son necesarios los siguientes datos:

Archivo de configuración Un archivo de configuración que especifique la dirección del resto de archivos, el instante inicial y final de cálculo, el integrador a usar, la carpeta donde escribir los resultados y si se filtran o no las datos de entrada de coordenadas generalizadas.

Modelo *.osim* El modelo sobre el que se va a realizar el cálculo. Es conveniente que sobre ese modelo se haya realizado un escalado previamente.

Actuadores adicionales Un archivo que especifica los actuadores residuales que se añadirán al modelo en memoria.

Archivo de coordenadas Un archivo de almacenamiento en formato *.mot* o *.sto* con los valores de las coordenadas generalizadas durante el tiempo de marcha. Suele ser un modelo previamente obtenido mediante el método de cinemática inversa.

Además, si se quiere calcular las fuerzas de contacto, hará falta este archivo.

Archivo de descripción de esferas de contacto Con el nombre *esferas.txt*, contendrá la información sobre la geometría y fuerzas de contacto a calcular.

Si por otra parte, se desea que las fuerzas de contacto sean leídas y no calculadas, este otro archivo hará falta.

Archivo de fuerzas de contacto externas Un archivo que contiene la lista de fuerzas de contacto, su punto de aplicación y momento asociado para cada instante.

Archivos de salida

Una vez la optimización estática se ha realizado, se escribirán los siguientes archivos en la carpeta de resultados:

Valores de fuerzas Un archivo en formato *.sto* con los valores de fuerza de cada músculo a lo largo de la marcha, incluyendo además las fuerzas residuales y también las fuerzas de contacto, en el caso de que hayan sido calculadas.

Valores de activación Un archivo de formato *.sto* con los valores de activación de cada músculo y cada fuerza residual.

Control de los músculos Un archivo en formato *.xml* con los valores de los controles del modelo a lo largo de la marcha.

Consideraciones adicionales

Para producir resultados con menos ruido, se hace conveniente aplicar un filtro de 6 Hz a las coordenadas generalizadas de entrada. Si el integrador anunciara en el registro que no se ha encontrado solución en algunos pasos, será conveniente aumentar la fuerza de los actuadores residuales.

2.2.6. Reducción de residuales

El propósito del algoritmo de reducción de residuales (*ARR*, Residual Reduction Algorithm o *RRA* en inglés), es el de minimizar los efectos que los errores provocados por el deslizamiento piel-hueso del marcador en el sujeto más otros errores de modelado. Este método permite modificar los valores de masa del modelo y sus valores cinemáticos con el objetivo de producir un modelo más consistente con los valores de fuerza de contacto con el suelo.

Base teórica Como se explica en la sección 2.3.3, para asegurar el equilibro de las ecuaciones dinámicas del modelo, es necesario añadir una actuador de reserva para cada uno de los grados de libertad del modelo, capaz de producir una fuerza o par residual que asegure la igualdad en la ecuación de equilibro dinámico.

$$F + F_{residual} = ma \tag{2.9}$$

El ARR, partiendo de la posición inicial de los valores calculados previamente por cinemática inversa, progresa por el tiempo indicado en pequeños pasos. En cada paso se computarán los valores de fuerza de los actuadores del modelo. Una vez calculados los actuadores para cada paso, el algoritmo, según mandado podrá hacer algunas o todas las siguientes tareas:

Ajustar el lugar del centro de masa Tomando como base los valores medio del residual M_x y del residual M_z, se ajustará la posición del centro de masas.

Ajustar la cantidad de masa El valor medio de F_y es empleado para buscar reducir la masa de los segmentos del cuerpo, logrando una reducción total de F_y/g kilos en el modelo.

Ajustar la cinemática Se realizará otra simulación con los residuales limitados en capacidad de actuación, el ajuste de la masa y el centro del modelo realizado y se buscará que el modelo siga su propia cinemática con debido al movimiento de sus propios actuadores, limitando en lo posible el empleo de los residuales. Si la cinemática resultante es similar a la deseada, se podrá emplear posteriormente en el CMC por el hecho de que ha necesitado menos residuales para funcionar.

Este método debió de haberse empleado en los cálculos realizados en este proyecto, pero no fue así. Su inclusión en esta sección es debida a la intención de mantener el rigor.

2.2.7. Control Muscular Computarizado

El Control Muscular Computarizado (Computer Muscle Control ó CMC) es un método de control por re-alimentación que estima los controles de actuadores de un modelo músculo-esqueletal tal que las fuerzas generadas por dichos actuadores provoquen un movimiento que se ajuste en lo posible a una trayectoria deseada previamente definida. En esta subsección se resumirá el funcionamiento de este método en función de lo expuesto por Thelen y Anderson [30].

Base teórica

En base a intervalos de tiempos definidos por el usuario (generalmente de 0,01 segundos), la herramienta de CMC calcula las excitaciones musculares que conseguirán que las coordenadas generalizadas de un modelo siga una trayectoria cinemática deseada. Esto lo consigue mediante una combinación de control proporcional-derivativo (PD) , optimización estática y dinámica directa. La herramienta primeramente calcula unos valores iniciales para el modelo, siendo estos valores una combinación de coordenadas, velocidades y aceleraciones generalizadas y los estados musculares de activación y longitud y velocidad de alargamiento $(a, l, \dot{l})$.
En un instante t de la simulación, existirá una aceleración deseada $\ddot{q}^{des}$ que convertirá unas activaciones u mediante optimización estática 2.2.5 y estas activaciones en posiciones y velocidades $(q, \dot{q})$ a través de dinámica directa 2.2.3. También habrán unas coordenadas de posición , velocidad y aceleración experimentales $(q^{exp}, \dot{q}^{exp}, \ddot{q}^{exp})$ que se tomarán como referencia en el modelo de control.
El error de posición y velocidad $\left\{e_j(t) = q_j^{exp}(t) - q_j(t), \quad \dot{e}_j(t) = \dot{q}_j{}^{exp}(t) - \dot{q}_j(t)\right\}$ servirán para controlar las aceleraciones deseadas para una coordenada j. Una definición del sistema de cálculo para el sistema en el momento t tras un intervalo T, para una coordenada j es:

$$\ddot{q}_j^{des}(t+T) = \ddot{q}_j^{exp}(t+T) + k_v \cdot \dot{e}_j(t) + k_p \cdot e_j(t) \tag{2.10}$$

donde k_p y k_v son los parámetros de control de re-alimentación para posición y velocidad. Para reducir los errores de posición y velocidad en la menor cantidad de cálculos posible para cada instante t, se establece que el parámetro de control de posición sea $k_p = \sqrt{2}k_v$. Cuando la aceleración deseada $(\ddot{q}^{des})$ coincide con la aceleración experimental $(\ddot{q}^{exp})$ se pasa al siguiente instante $t + T$.
En el siguiente instante, partiendo de la aceleración deseada se obtendrá mediante optimización estática las activaciones necesarias para equilibrar las fuerzas del modelo en ese instante. Ello se hará minimizando la siguiente función de mínimos cuadrados, si la opción *fast target* está activada, que intenta por un lado minimizar las activaciones musculares x y por otro acercar las aceleraciones experimentales a las deseadas :

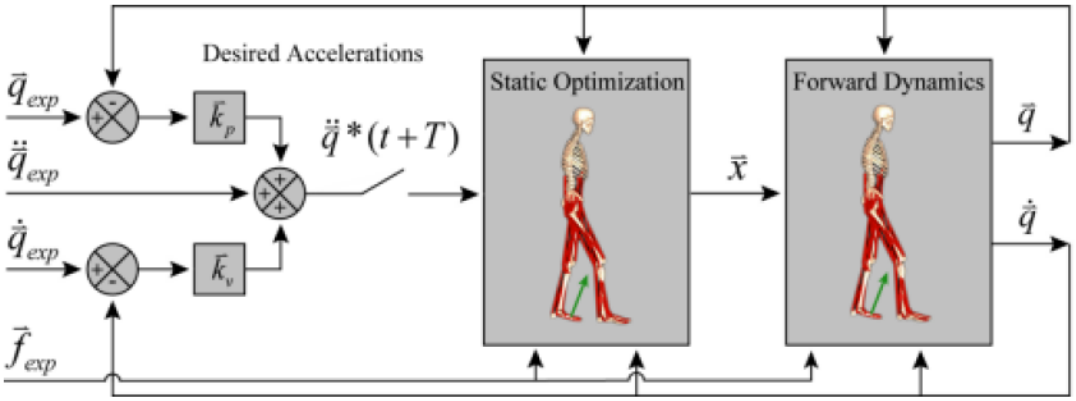

FIGURA 2.3: Esquema de funcionamiento del CMC. Las fuerzas de contacto se muestran aquí añadidas aparte.

$$J = \sum_{i=1}^{n_x} x_i^2 + \sum_{j=1}^{n_q} \omega_j (\ddot{q}_j^{des} - \ddot{q}_j)^2 \tag{2.11}$$

Si *fast target* está desactivada la función a minimizar será ésta:

$$J = \sum_{i=1}^{n_x} x_i^2 \tag{2.12}$$

$$C_j = \ddot{q}_j^{des} - \ddot{q}_j \quad \forall j \tag{2.13}$$

Si la optimización por *fast target* no es capaz de satisfacer la restricciones indicadas el programa fallará. Para solucionar eso, se añaden actuadores de reserva que necesitan altos valores de activación para satisfacer las ecuaciones del modelo, y que por tanto son muy penalizados en la optimización. Un valor alto de estos actuadores suele significar un modelo débil o mal definido.
Como parte final, con los valores de los activaciones de los músculos obtenidos en la optimización estática se hará un cálculo de dinámica directa con el objetivo de obtener la cinemática experimental a controlar en la ecuación (2.10).

Archivos necesarios

Para llevar a cabo una simulación de CMC serán necesarios los siguientes archivos, algunos más que con la optimización estática:

Archivo de configuración Un archivo de configuración que especifique la dirección del resto de archivos, el instante inicial y final de cálculo, el integrador a usar, la carpeta donde escribir los resultados y si se filtran o no las datos de entrada de coordenadas generalizadas.

Modelo *.osim* El modelo sobre el que se va a realizar el cálculo. Es conveniente que sobre ese modelo se haya realizado un escalado previamente.

Actuadores adicionales Un archivo que especifica los actuadores residuales que se añadirán al modelo en memoria.

Archivo de coordenadas Un archivo de almacenamiento en formato *.mot* o *.sto* con los valores de las coordenadas generalizadas durante el tiempo de marcha. Suele ser un modelo previamente obtenido mediante el método de cinemática inversa.

Archivo de tareas para el controlador Lista de coordenadas generalizadas del modelo que serán controladas por CMC incluyendo los valores de los parámetros de control $\{K_p, K_v\}$ para estas coordenadas.

Archivo de restricciones en los actuadores Para cada actuador músculo-tendón que tiene el modelo, especifica los valores permitidos para su activación en su salida en la optimización estática.

Además, si se quiere calcular las fuerzas de contacto, hará falta este archivo.

Archivo de descripción de esferas de contacto Con el nombre *esferas.txt*, contendrá la información sobre la geometría y fuerzas de contacto a calcular.

Si por otra parte, se desea que las fuerzas de contacto sean leídas y no calculadas, este otro archivo hará falta.

Archivo de fuerzas de contacto externas Un archivo que contiene la lista de fuerzas de contacto, su punto de aplicación y momento asociado para cada instante.

Archivos de salida

Valores de fuerzas Un archivo en formato *.sto* con los valores de fuerza de cada músculo a lo largo de la marcha, incluyendo además las fuerzas residuales y también las fuerzas de contacto, en el caso de que hayan sido calculadas.

Valores de activación Un archivo de formato *.sto* con los valores de activación de cada músculo y cada fuerza residual.

Control de los músculos Un archivo en formato *.xml* con los valores de los controles del modelo a lo largo de la marcha.

El proceso habitual de trabajo con el control muscular computarizado requiere partir de un modelo inicial, datos de trayectoria de marcadores y datos de fuerza de contacto con el suelo. Con esos procesos, se escala el modelo, con el modelo escalado se obtiene mediante cinemática inversa los valores de posición y velocidad de las coordenadas generalizadas del modelo. Luego se somete el modelo escalado al algoritmo de reducción de residuales, obteniendo un modelo ligeramente corregido y por último con ese modelo corregido se realiza el CMC. En este proyecto en particular se ha omitido el proceso de reducción de algoritmos.

2.3. Modelos de Opensim

OpenSim para su funcionamiento emplea modelos compuestos de diferentes tipos de elementos con diferentes funciones y características. El mínimo para ejercer la simulación cinemática más sencilla será tener cuerpos con sus pares y coordenadas correspondientes. A partir de eso pueden agregarse elementos fuerzas activas y pasivas, marcadores de posición, geometrías de contacto y controladores.
Estos modelos se guardan en archivos de formato *xml* con extensión *.osim*. Pueden editarse desde la IGU, crearse a través del propio programa o manualmente con cualquier editor de texto.
En esta sección se detallarán cada uno de los tipos de elementos posibles de un modelo de OpenSim, con atención en aquellos empleados en este proyecto.

2.3.1. Cuerpos

Al ser OpenSim un programa de cálculo de sistemas multi-cuerpo, el cuerpo *(Body)* es la base de su funcionamiento. Cada cuerpo (excepto el cuerpo de referencia) posee una articulación que lo une con su cuerpo padre. La articulación define las coordenadas y transformaciones cinemáticas que gobiernan el movimiento del cuerpo respecto a su cuerpo padre. En el modelo todos los cuerpo están contenido dentro del conjunto *Bodyset*.

Cuerpo de referencia

Todo modelo de OpenSim posee un cuerpo inicial llamado *Ground* que es tomado como referencia las simulaciones. El cuerpo *Ground* será padre en todas las articulaciones de las que forme parte.
Tendrá masa e inercia nula y ninguna geometría de contacto o visual propia.

```
<Body name="ground">
    <mass>0</mass>
    <mass_center> 0 0 0</mass_center>
    <inertia_xx>0</inertia_xx>
    <inertia_yy>0</inertia_yy>
    <inertia_zz>0</inertia_zz>
    <inertia_xy>0</inertia_xy>
    <inertia_xz>0</inertia_xz>
    <inertia_yz>0</inertia_yz>
    <!--Joint that connects this body with the parent body.-->
    <Joint />
    <VisibleObject>
    <WrapObjectSet>
</Body>
```

FIGURA 2.4: Definición del cuerpo de referencia dentro de un archivo de modelo *.osim*.

Cuerpos comunes

Son los diferentes sólidos rígidos de los que se componen los modelos. Poseen propiedades de masa e inercia diferentes de cero, y sus dimensiones no están definidas. Todos ellos están conectados a otros cuerpos mediante articulaciones. Pueden tener geometrías visuales asociadas. Cada cuerpo posee un centro propio, un centro de gravedad (que no tiene por qué coincidir con el centro propio), y un grupo de coordenadas de orientación propias que serán definidas por su propia articulación. Si ésta es una articulación fija heredará la orientación de su cuerpo padre.

Cuerpo plataforma

En muchos modelos de marcha humana, dado que las fuerzas externas de contacto con el suelo son suministrados externamente, no es necesario modelar el contacto con el suelo. Por ello, esos modelos no suelen tener un cuerpo que simbolice el suelo en sí, o usan el cuerpo de referencia.
En este proyecto, al ser necesaria una geometría de contacto que representara el suelo y cuya altura fuera variable según se necesitara, se optó por incluir un cuerpo extra sin masa unido a la referencia por una articulación estática o *Welded Joint* el cual tuviera asociado un geometría de contacto con forma de semi-plano infinito. Ese cuerpo se llamó *Platform*.

```
<Body name="tibia_r">
    <mass>3.58100089669871</mass>
    <mass_center> 0 -0.184557 0</mass_center>
    <inertia_xx>0.0486803628303749</inertia_xx>
    <inertia_yy>0.0049259890959308</inertia_yy>
    <inertia_zz>0.0493564789807968</inertia_zz>
    <inertia_xy>0</inertia_xy>
    <inertia_xz>0</inertia_xz>
    <inertia_yz>0</inertia_yz>
    <!--Joint that connects this body with the parent body.-->
    <Joint>
        <CustomJoint name="knee_r">
    </Joint>
```

FIGURA 2.5: Ejemplo de un cuerpo común

Geometría visible

Para una visualización en la IGU más vistosa, es posible añadir a cada cuerpo una geometría visible. Ésta se define en el archivo *XML* indicando la dirección de un archivo de mallado tridimensional *.vtp*, *.stl* ó *.obj* que puede referirse a algunas de las geometrías ya incluidas en OpenSim (esferas, cilindros, huesos del cuerpo humano concretos) o a archivos creados por el usuario con algún programa externo.

```
<VisibleObject>
    <!--Set of geometry files and associated attributes, allow .vtp, .stl, .obj-->
    <GeometrySet>
        <objects>
            <DisplayGeometry>
            <DisplayGeometry>
                <!--Name of geometry file .vtp, .stl, .obj-->
                <geometry_file>fibula.vtp</geometry_file>
                <!--Color used to display the geometry when visible-->
                <color> 1 1 1</color>
                <!--Name of texture file .jpg, .bmp-->
                <texture_file />
                <!--in body transform specified as 3 rotations (rad) followed by 3 translations rX rY rZ tx ty tz-->
                <transform> -0 0 -0 0 0 0</transform>
                <!--Three scale factors for display purposes: scaleX scaleY scaleZ-->
                <scale_factors> 1 1 1</scale_factors>
                <!--Display Pref. 0:Hide 1:Wire 3:Flat 4:Shaded-->
                <display_preference>4</display_preference>
                <!--Display opacity between 0.0 and 1.0-->
                <opacity>1</opacity>
            </DisplayGeometry>
        </objects>
        <groups />
    </GeometrySet>
    <!--Three scale factors for display purposes: scaleX scaleY scaleZ-->
    <scale_factors> 0.988523 0.988523 0.988523</scale_factors>
    <!--transform relative to owner specified as 3 rotations (rad) followed by 3 translations rX rY rZ tx ty tz-->
    <transform> -0 0 -0 0 0 0</transform>
    <!--Whether to show a coordinate frame-->
    <show_axes>false</show_axes>
    <!--Display Pref. 0:Hide 1:Wire 3:Flat 4:Shaded Can be overriden for individual geometries-->
    <display_preference>4</display_preference>
</VisibleObject>
```

FIGURA 2.6: Contenido dentro de la llave *VisibleObject* para el cuerpo *tibia_r*.

2.3.2. Articulaciones

Las articulaciones en OpenSim relacionan la posición y orientación de cada cuerpo, tomando como punto de referencia para ello el punto de articulación de cada cuerpo. Así, en una articulación existirán relacionados dos cuerpos, el padre y el hijo. Cada uno tendrá un punto de articulación propio que se encontrará a una distancia y orientación fija del centro de su cuerpo. Estos puntos de articulación estarán luego relacionado entre sí

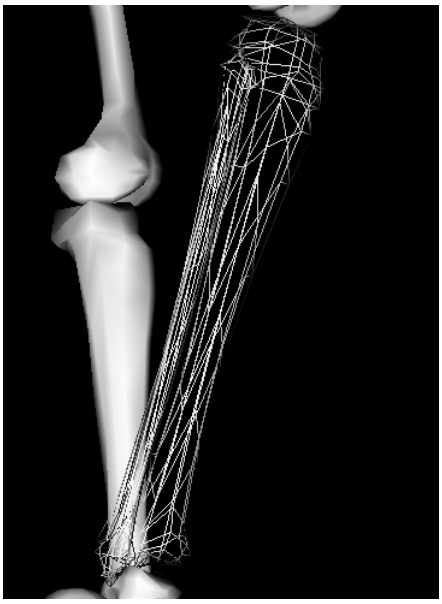

FIGURA 2.7: Ejemplo de representación de la malla de la geometría *fibula.vtp* perteneciente al cuerpo *tibia_r*. El resto de cuerpos se muestran opacos.

según las restricciones que interponga cada tipo de articulación y a su vez los parámetros generalizados que puedan gobernar esa articulación.

Descripción de una articulación

Sea un cuerpo padre con un centro $\overrightarrow{OC^p}$ y cuyo sistema de ejes cartesianos ortonormal local parta de ese centro y sea $\{\overrightarrow{u}^p_1, \overrightarrow{u}^p_2, \overrightarrow{u}^p_3\}$. Ese cuerpo poseerá un punto de articulación fijo en el sistema local de ejes $\overrightarrow{C^pC^{ap}}$ y un sistema de ejes cartesianos ortonormal propio definido por las coordenadas $\{\overrightarrow{u}^{ap}_1, \overrightarrow{u}^{ap}_2, \overrightarrow{u}^{ap}_3\}$. Sea también un cuerpo hijo con un centro $\overrightarrow{OC^h}$, también con un sistema de ejes cartesianos ortonormal local $\{\overrightarrow{u}^h_1, \overrightarrow{u}^h_2, \overrightarrow{u}^h_3\}$ y un punto de articulación en el sistema local $\overrightarrow{C^hC^{ah}}$, también con su propia base cartesiana ortonormal propia $\{\overrightarrow{u}^{ah}_1, \overrightarrow{u}^{ah}_2, \overrightarrow{u}^{ah}_3\}$.
Así, puede considerarse un vector $\overrightarrow{C^{ap}C^{ah}}(q_n)$ desde el punto A de la articulación padre al punto A de la articulación hijo que pudiera estar en función de n parámetros generalizados, y a la vez que existe la matriz de rotación $R(q_m)$ basada en m parámetros generalizados que transformara la base cartesiana basada de la parte de la articulación del sólido padre C^{ap} a la base cartesiana de la misma articulación en el sólido hijo C^{ah}. Una articulación es el conjunto de restricciones que basadas en un conjunto de parámetros, en el espacio de la articulación padre, relaciona el vector $\overrightarrow{C^{ap}C^{ah}}$ y la matriz de giro R que relaciona el cuerpo padre con el cuerpo hijo.

Tipos de articulación

OpenSim posee varios tipos de articulaciones, a los que es posible añadir más, debido a su naturaleza abierta. Sólo dos tipos se emplean en este proyecto.

WeldedJoint Es una articulación soldada. Impide cualquier tipo de movimiento o giro relativo. Es empleada para unir el cuerpo *Platform* al cuerpo de referencia *Ground*.

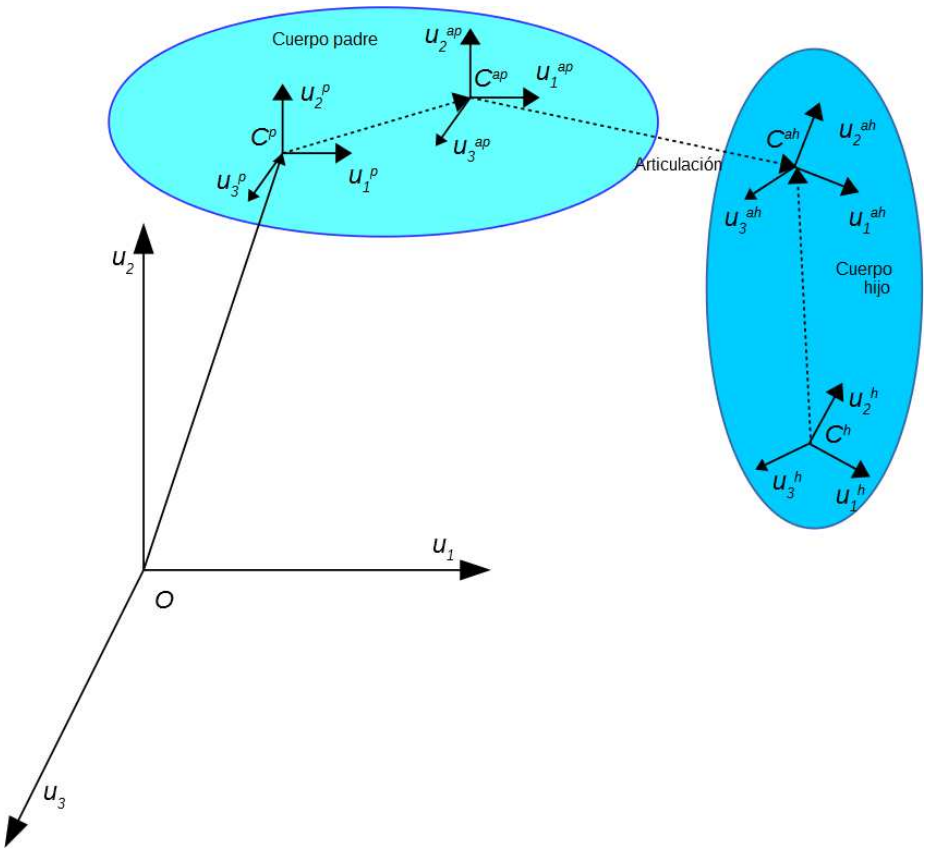

FIGURA 2.8: Relación entre sólidos en una articulación.

CustomJoint Es una articulación personalizable. Permite definir cualquier tipo de relación entre un cuerpo y otro. Es aquí donde se definen los parámetros generalizados que caracterizarán el movimiento del modelo. Una vez estos parámetros, la posición y orientación de los puntos de articulación y qué cuerpo es el padre, se define el movimiento de la articulación mediante una relación espacial.

Relación espacial de la *CustomJoint* La transformación que ocurre en una *CustomJoint* se compone de seis transformaciones. Las tres primeras son rotaciones definidas alrededor de un eje diferente cada una y las tres siguientes son traslaciones a lo largo de un eje diferente cada una, mientras que los ejes de rotación y desplazamiento son fijos (dentro del espacio cartesiano de la articulación padre), los valores de rotación y desplazamiento a lo largo de estos dependerán de los valores de las coordenadas generalizadas (q). En tres dimensiones, las rotaciones se distinguen por tres parámetros empleando el sistema de Euler.

2.3.3. Fuerzas

Estos modelos tienen tanto fuerzas no actuadas (la gravedad, el contacto con el suelo) como fuerzas actuadas.

Fuerzas no actuadas

Existen diferentes fuerzas que se aplican al modelo, pero que no son controladas.

```
<Joint>
    <CustomJoint name="knee_r">
        <parent_body>femur_r</parent_body>
        <location_in_parent>0 0 0</location_in_parent>
        <orientation_in_parent>0 0 0</orientation_in_parent>
        <location>0 0 0</location>
        <orientation>0 0 0</orientation>
        <CoordinateSet>
            <objects>
                <Coordinate name="knee_angle_r">
                    <motion_type>rotational</motion_type>
                    <default_value>0</default_value>
                    <default_speed_value>0</default_speed_value>
                    <range>-2.0943951 0.17453293</range>
                    <clamped>false</clamped>
                    <locked>false</locked>
                    <prescribed_function />
                    <prescribed>false</prescribed>
                </Coordinate>
            </objects>
            <groups />
        </CoordinateSet>
        <reverse>false</reverse>
        <SpatialTransform>
    </CustomJoint>
</Joint>
```

FIGURA 2.9: Contenido dentro de la llave *CustomJoint* para el cuerpo *tibia_r*.

Gravedad Una de ellas es la gravedad, que actúa en el centro de todos los sólidos con el vector que se especifica al comienzo del archivo de modelo.

```
<!--Acceleration due to gravity.-->
<gravity> 0 -9.80665 0</gravity>
```

FIGURA 2.11: Línea del modelo OpenSim que indica el valor y la dirección de las fuerzas de gravedad, en newtons.

Fuerzas de contacto Estas fuerzas modelan el contacto de cuerpos con otros y requieren la especificación posterior de geometrías de contacto. En este trabajo se emplean las fuerzas de Hunt-Crossley [14] (aunque sólo como referencia para el programa de Matlab) y la *FzaContactoMatlab*.

Fuerzas de Hunt-Crossley Modelan la fuerza como la reacción de una esfera formada por elementos muelle-amortiguador con disposición radial, que reacciona al penetrar ésta la geometría ajena de contacto. Además el modelo de OpenSim añade las fuerzas tangenciales a la superficie en el cálculo. Para definir una es necesario indicar los parámetros siguientes:

geometry Geometría: Indica a OpenSim entre qué dos geometrías de contacto se da la fuerza. Se mide en m.

stiffness Rigidez: Depende de la rigidez y radio de ambas superficies. Para este trabajo ha sido estimada. Este valor k_{eq} se detalla en la siguiente ecuación, con $\frac{N}{m^{\frac{3}{2}}}$ como unidades:

```
<SpatialTransform>
    <TransformAxis name="rotation1">
        <coordinates>knee_angle_r</coordinates>
        <axis>0 0 1</axis>
        <function>
            <LinearFunction>
                <coefficients> 1 0</coefficients>
            </LinearFunction>
        </function>
    </TransformAxis>
    <TransformAxis name="rotation2">
        <coordinates></coordinates>
        <axis>0 1 0</axis>
        <function>
            <Constant>
                <value>0</value>
            </Constant>
        </function>
    </TransformAxis>
    <TransformAxis name="rotation3">
    <TransformAxis name="translation1">
        <coordinates>knee_angle_r</coordinates>
        <axis>1 0 0</axis>
        <function>
            <MultiplierFunction>
                <function>
                    <SimmSpline>
                        <x> -2.0944 -1.74533 -1.39626 -1.0472 -0.698132 -0.349066 -0.174533 0.197344 0.337395 0.490178 1.
                        <y> -0.0032 0.00179 0.00411 0.0041 0.00212 -0.001 -0.0031 -0.005227 -0.005435 -0.005574 -0.005435
                    </SimmSpline>
                </function>
                <scale>1.14724</scale>
            </MultiplierFunction>
        </function>
    </TransformAxis>
    <TransformAxis name="translation2">
    <TransformAxis name="translation3">
</SpatialTransform>
```

FIGURA 2.10: Contenido dentro de la llave *SpatialTransform* dentro del *CustomJoint* para el cuerpo *tibia_r*.

$$k_{eq} = \frac{4}{3} \cdot \left(\frac{R_1 + R_2}{R_1 \cdot R_2} \right)^{\frac{1}{2}} \cdot \left(\frac{E_1^{\frac{2}{3}} + E_2^{\frac{2}{3}}}{E_1^{\frac{2}{3}} \cdot E_2^{\frac{2}{3}}} \right)^{\frac{3}{2}} \tag{2.14}$$

dissipation Disipación. Parámetro que depende de ambas superficies de contacto, y que al igual que la rigidez se estima numéricamente en este proyecto. Este valor c_{eq} se muestra a continuación:

$$c_{eq} = c_1 \cdot \left(\frac{E_1^{\frac{2}{3}} + E_2^{\frac{2}{3}}}{E_1^{\frac{2}{3}} \cdot E_2^{\frac{2}{3}}} \right)^{\frac{3}{2}} + c_2 \cdot \left(1 - \left(\frac{E_1^{\frac{2}{3}} + E_2^{\frac{2}{3}}}{E_1^{\frac{2}{3}} \cdot E_2^{\frac{2}{3}}} \right)^{\frac{3}{2}} \right) \tag{2.15}$$

static friction Fricción estática. Determina el valor de μ_{est} para velocidades de deslizamiento inferiores a la velocidad de transición. Es adimensional.

dynamic friction Fricción dinámica. Determina el valor de μ_{din} para velocidades de deslizamiento superiores a la velocidad de transición. Es adimensional.

viscous friction Fricción viscosa (μ_{vis}). Se mide en $\frac{s}{m}$.

transition velocity Velocidad umbral (v_{um}). Determina a partir de qué velocidad de deslizamiento el modelo de fricción comienza a aplicar la fricción dinámica en vez de la estática.

La fórmula de fuerza normal, siendo δ la penetración y $\dot{\delta}$ la velocidad de penetración de la esfera en el plano será:

$$F_{normal} = k_{eq} \cdot \left(1 + \frac{3}{2} \cdot c_{eq} \cdot \dot{\delta} \right) \cdot \delta^{1,5} \tag{2.16}$$

La ecuación de fuerza tangencial será, tomando v como la velocidad de deslizamiento:

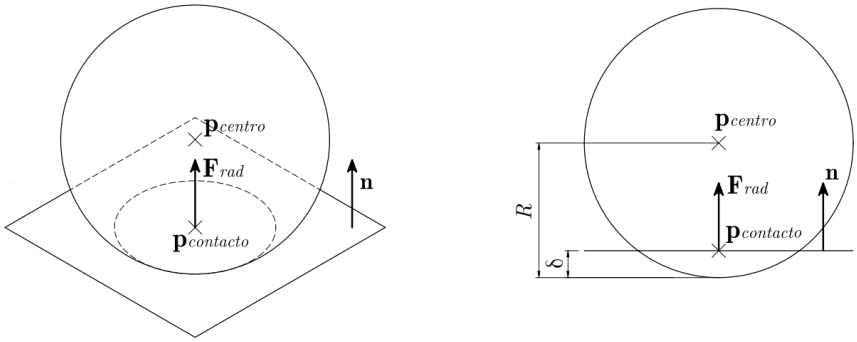

FIGURA 2.12: Vistas frontales e isométricas de un contacto entre esfera y plano

$$F_{tangencial} = F_{normal} \cdot \left(\mu_{din} + \frac{2\left(\mu_{est} - \mu_{din}\right)}{1 + \left(\frac{v}{v_{um}}\right)^2} + \mu_{vis} \cdot v \right) \tag{2.17}$$

Es importante tener en cuenta que el modelo de fuerzas de Hunt-Crossley que se implementa en este proyecto no es exactamente la que implementa OpenSim.

```
<HuntCrossleyForce name="esfera_r_2">
    <isDisabled>true</isDisabled>
    <!--Material properties.-->
    <HuntCrossleyForce::ContactParametersSet name="contact_parameters">
        <objects>
            <HuntCrossleyForce::ContactParameters>
                <geometry>plane_geo esfera_geo_r_2</geometry>
                <stiffness>46473.4</stiffness>
                <dissipation>19.6284</dissipation>
                <static_friction>1</static_friction>
                <dynamic_friction>0.2</dynamic_friction>
                <viscous_friction>3</viscous_friction>
            </HuntCrossleyForce::ContactParameters>
        </objects>
        <groups />
    </HuntCrossleyForce::ContactParametersSet>
    <!--Slip velocity (creep) at which peak static friction occurs.-->
    <transition_velocity>0.001</transition_velocity>
</HuntCrossleyForce>
```

FIGURA 2.13: Definición de los parámetros de contacto y geometría de una esfera de contacto.

Fuerzas actuadas

Músculos Los músculos dirigen el movimiento humano en base a su contracción, pudiendo causar diferentes direcciones en base a qué grupos de músculo actúan y se contraen. En este proyecto se empleó el modelo *Thelen2003Muscle*, que ya viene incluido en los modelos de OpenSim, aunque este modelo de músculo no es el único que posee OpenSim. Un músculo está unido a los cuerpos mediante puntos de unión y tienen definida una ruta mediante puntos de ruta para contraerse. Reaccionan a excitaciones, que van de 0 a 1, las cuales en el modelo muscular se transforman en activaciones y posteriormente en fuerzas.

Thelen2003Muscle El modelo *Thelen2003Muscle*, músculo-tendón de tipo Hill, está basado en el modelo diseñado por Thelen, [29], a su vez inspirado por el trabajo de Zajac [33]. Está compuesto de dos elementos en serie; el primer elemento, el músculo es un elemento elástico paralelo a un elemento contractor. El segundo elemento, el tendón, tiene un elemento elástico.

Activación Una señal de excitación ideal (u), tras un pequeño retraso definido por la constante de tiempo τ_a inicia la activación del músculo (a).

$$\frac{da}{dt} = \frac{u-a}{\tau_a(u,a)} \tag{2.18}$$

Constante de activación τ_a depende de la activación y tiene diferentes valores según el músculo esté activándose o desactivándose, siendo τ_{act} y τ_{desact} constantes definibles:

$$\tau_a(u,a) = \begin{cases} \tau_{act}(0,5+1,5a) & \text{si } u > a \\ \tau_{desact}/(0,5+1,5a) & \text{si } u \leq a \end{cases} \tag{2.19}$$

Equilibrio de fuerzas en el músculo La siguiente ecuación describe la fuerza isométrica del músculo como F_{iso}, teniendo en cuenta el ángulo penniforme α^M.

$$F_{iso}\left(a\left(t\right)f_{AL}\left(l^M\right)F_v\left(\dot{l}^M\right)+F_{EP}\left(l^M\right)\right)\cos\alpha^M - F_{iso}F_{ES}\left(l^T\right)=0 \tag{2.20}$$

Fuerza elástica del elemento paralelo del músculo Se representa por una fuerza exponencial, donde k^{EP} es un factor de forma exponencial, ϵ_0^M es la deformación del elemento pasivo del músculo cuando la fuerza isométrica es máxima; $\overline{L^M}$ es la longitud normalizada de la fibra muscular, tomando como cociente la longitud de fibra cuando la fuerza es máxima.

$$F^{EP} = \frac{e^{k^{EP}(\overline{L^M}-1)/\epsilon_0^M}}{e^{k^{EP}}-1} \tag{2.21}$$

Factor de relación fuerza activa-longitud La fuerza activa del músculo se dispone como una distribución gaussiana, donde $\overline{L^M}$ es la longitud normalizada del músculo y γ es un factor de forma.

$$f_{AL} = e^{-\left(\overline{L_M}-1\right)^2/\gamma} \tag{2.22}$$

Fuerza-deformación del tendón Es una función exponencial durante una región inicial y posteriormente lineal. La deformación del tendón es ϵ^T, $\epsilon^T_{inicial}$ es el límite de deformación de la región inicial, $\overline{F^T}_{inicial}$ es la fuerza producida por el tendón en cuando su comportamiento pasa de exponencial a lineal. Las constantes $k_{inicial}$ y k_{lin} aseguran la continuidad de la función.

$$F^T = \begin{cases} \frac{F^T_{inicial}}{e^{k_{inicial}}-1}\left(e^{\left(k_{inicial}\epsilon^T/\epsilon^T_{inicial}\right)}-1\right) & \text{si } \epsilon^T \leq \epsilon^T_{inicial} \\ k_{lin}\left(\epsilon^T-\epsilon^T_{inicial}\right)+F^T_{inicial} & \text{si } \epsilon^T > \epsilon^T_{inicial} \end{cases} \tag{2.23}$$

Fuerza-velocidad de la fibra La velocidad de la fibra depende de la activación a, la longitud de la fibra muscular L^M y el resto de fuerzas F^M. Así, mediante equilibro de fuerzas en el músculo, se deduce el valor de la fuerza amortiguadora muscular.

$$F_v\left(\dot{L}^M\right) = \frac{\frac{f_{ES}(L^T)}{\cos\alpha^M} - f_{EP}(L^M)}{a(t)f_{AL}(L^M)} \tag{2.24}$$

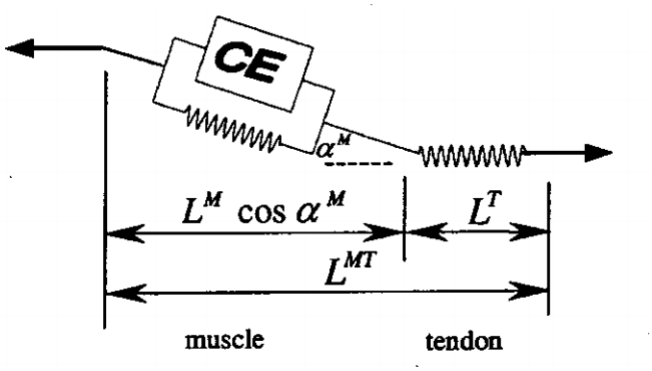

FIGURA 2.14: Representación del modelo muscular.

La figura 2.15 contiene las llaves necesarias para definir un músculo en un archivo *xml*.

```
<Thelen2003Muscle name="psoas_r">
    <isDisabled>false</isDisabled>
    <!--Minimum allowed value for control signal. Used primarily when solving for control values.-->
    <min_control>0</min_control>
    <max_control>1</max_control>
    <!--The set of points defining the path of the muscle.-->
    <GeometryPath>
    <!--The maximum force this actuator can produce.-->
    <optimal_force>1</optimal_force>
    <!--Maximum isometric force that the fibers can generate-->
    <max_isometric_force>1113</max_isometric_force>
    <!--Optimal length of the muscle fibers-->
    <optimal_fiber_length>0.105172289885285</optimal_fiber_length>
    <!--Resting length of the tendon-->
    <tendon_slack_length>0.168275663816457</tendon_slack_length>
    <!--Angle between tendon and fibers at optimal fiber length expressed in radians-->
    <pennation_angle_at_optimal>0.13962634</pennation_angle_at_optimal>
    <!--Maximum contraction velocity of the fibers, in optimal fiberlengths/second-->
    <max_contraction_velocity>10</max_contraction_velocity>
    <!--time constant for ramping up muscle activation-->
    <activation_time_constant>0.01</activation_time_constant>
    <!--time constant for ramping down of muscle activation-->
    <deactivation_time_constant>0.04</deactivation_time_constant>
    <!--tendon strain at maximum isometric muscle force-->
    <FmaxTendonStrain>0.033</FmaxTendonStrain>
    <!--passive muscle strain at maximum isometric muscle force-->
    <FmaxMuscleStrain>0.6</FmaxMuscleStrain>
    <!--shape factor for Gaussian active muscle force-length relationship-->
    <KshapeActive>0.5</KshapeActive>
    <!--exponential shape factor for passive force-length relationship-->
    <KshapePassive>4</KshapePassive>
    <!--force-velocity shape factor-->
    <Af>0.3</Af>
    <!--maximum normalized lengthening force-->
    <Flen>1.8</Flen>
</Thelen2003Muscle>
```

FIGURA 2.15: Código xml necesario para definir un músculo Thelen2003.
Los puntos de ruta han sido ocultados.

Fuerzas residuales Estas se añaden a los modelos en el SO y el CMC. Pueden ser actuadores puntuales o de par. Su importancia está explicada en el apartado 2.2. En el sólido *pelvis* se añadirán 6 actuadores para los grados de libertad del cuerpo en el espacio, tres de par (*MX*, *MY* y *MZ*) y tres vectoriales (*FX*, *FY* y *FZ*), para el resto de grados de libertad del modelo se añadirá un actuador de coordenada residual extra. En los archivo de configuración de actuadores, para cada uno de ellos se definirá su fuerza óptima (a mayor fuerza óptima menos se penalizará el uso de ese actuador), lugar o coordenada generalizada de aplicación, y sus valores límite de activación (siempre infinitos en este PFC).

2.3.4. Geometrías de contacto

En caso de existir fuerzas de contacto, estas geometrías definen las regiones que delimitan estas fuerzas. Un modelo puede tener asociadas geometrías de contacto y estas pueden tener diferentes formas y/o archivos asociados. Para definir una geometría de contacto es necesario definir qué cuerpo la contiene (*body_name*), su localización respecto al centro del cuerpo que la contiene (*location*), su orientación (*orientation*) y datos adicionales según el tipo.

Esferas de contacto

Es el tipo de geometría que he usado para las fuerzas de contacto de este trabajo. Necesitan que se especifiquen su radio (*radius*).

```
<ContactSphere name="esfera_geo_r_1">
    <!--Body name to connect the contact geometry to-->
    <body_name>calcn_r</body_name>
    <!--Location of geometry center in the body frame-->
    <location>0.0413777 -0.00633021 0</location>
    <!--Orientation of geometry in the body frame-->
    <orientation>0 0 0</orientation>
    <radius>0.0291622</radius>
</ContactSphere>
```

FIGURA 2.16: Código xml necesario para definir un esfera de contacto.

Semi-planos de contacto

Son semi-planos infinitos. No necesitan que se les de información adicional. En este proyecto se ha empleado para definir el contacto con el cuerpo *Platform*.

Mallas de contacto

Otra forma es crear mallas tridimensionales formadas por triángulos que se asemejen a la geometría del pie. Esa geometría se define mediante archivos de formato *.obj*, *.stl* o *.vtp*. No se han usado en este proyecto, pero sí representan una posible dirección a estudiar al profundizar en el tema de cálculo de fuerzas de contacto en la marcha humana.

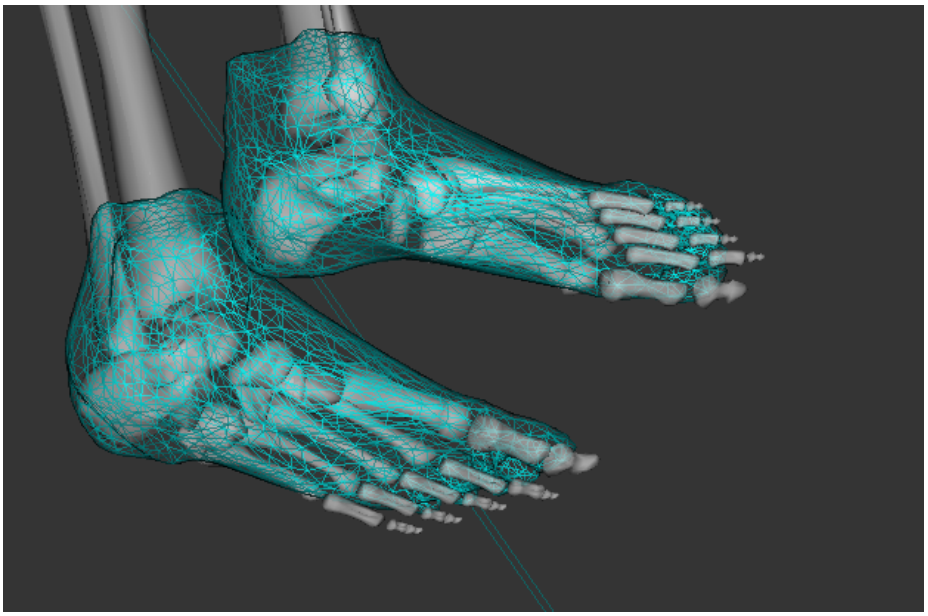

FIGURA 2.17: Posible geometría de contacto de una malla de triángulos que si se usara debería ser simplificada.

2.3.5. Marcadores

La función de los marcadores es múltiple. Por una parte sirven para ajustar la escala de los modelos para que se ajusten a los datos de mediciones tomados en laboratorio mediante captura de movimientos (véase 2.2.1). Otro uso es, mediante la herramienta de cinemática inversa (*IK*), obtener los valores de las coordenadas y velocidades generalizadas a partir de los datos de movimiento de marcadores suministrados por una medición realizada en laboratorio, en un archivo de formato *.trc*.

Marcadores en el modelo de OpenSim

Los marcadores, figuran en los archivos de modelos dentro de la categoría *MarkerSet*. Allí se especifican los modelos, donde para cada uno es necesario indicar a qué cuerpo pertenece y su localización.

Archivos de registro de trayectoria de marcadores

Los archivos *.trc*, (*tracker row column*) son un formato de archivos cuya función es indicar, a lo largos de diferentes filas ordenadas ordenadas por el valor de tiempo de la primera columna, la posición de los marcadores a lo largo de la medición de marcha experimental. Será necesario exportar los datos de la lectura de marcadores a este formato para poder trabajar con ellos en OpenSim.

2.3.6. Controladores

Los controladores permiten modificar en tiempo de ejecución las fuerzas de un modelo, con el objetivo de que estos se comporten a ser posible de una forma deseada. Opensim puede implementar diversos tipos de controladores, así por ejemplo, el método CMC añade al modelo controladores lineales para cada músculo.

Los controladores se añaden mediante conjuntos llamados *ControlSet* que contienen controladores, un ejemplo de los cuales escrito en *.xml* se muestra en la figura 2.20.

```
<Marker name="Sternum">
    <Marker name="R.Acromium">
    <!--Body segment in the model on which the marker resides.-->
    <body> torso </body>
    <!--Location of a marker on the body segment.-->
    <location>-0.03 0.44 0.15</location>
    <!--Flag (true or false) specifying whether or not a marker should be kept
        fixed in the marker placement step.  i.e. If false, the marker is
        allowed to move.-->
    <fixed> false </fixed>
    <!--Used for displaying a marker in the visuals.-->
    <VisibleObject name="">
        <!--Set of geometry files and associated attributes, allow .vtp, .stl,
        <GeometrySet name="">
        <!--Three scale factors for display purposes: scaleX scaleY scaleZ-->
        <scale_factors>1 1 1</scale_factors>
        <!--transform relative to owner specified as 3 rotations (rad) followed by
            3 translations rX rY rZ tx ty tz-->
        <transform>0 0 0 0 0 0</transform>
        <!--Whether to show a coordinate frame-->
        <show_axes> false </show_axes>
        <!--Display Pref. 0:Hide 1:Wire 3:Flat 4:Shaded Can be overriden for
            individual geometries-->
        <display_preference> 4 </display_preference>
    </VisibleObject>
</Marker>
```

FIGURA 2.18: Detalle de la información de un marcador dentro de un modelo *.osim* que no ha sido todavía escalado.

2.4. El modelo Gait2354

En está sección se detallará información sobre el modelo músculo-esqueletal para marcha humana **Gait2354**. Desarrollado por Darryl Thelen, Ajay Seth, Frank C. Anderson y Scott L. Delp, el modelo incluye las articulaciones de extremeidad inferior adoptadas de Delp y col. [8], la articulación de la espalda y su antropometría están adoptadas desde el trabajo de Anderson y Pandy [2] y el modelo de rodilla articulada uni-dimensional plana de Yamaguchi y Zajac [32].

2.4.1. Contenido del modelo

El modelo tiene 23 grados de libertad (véase la tabla 2.1) , 13 cuerpos más la plataforma añadida para calcular fuerzas de contacto (véase la tabla 2.2), que modela las extremidades inferiores humanas a partir de la pelvis, al que se le ha eliminado la rótula con el objetivo de simplifica la articulación de la rodilla y que reúne en un simple sólido los datos de masa e inercia de torso, cabeza y extremidades superiores. El número de actuadores músculo-tendón es de 54, que representan de manera simplificada los grupos de músculos en las extremidades inferiores y torso.

```
PathFileType    4   (X/Y/Z) subject01_walk1.trc
DataRate    CameraRate  NumFrames   NumMarkers  Units   OrigDataRate    OrigDataStartFrame  OrigNumFrames
60.00   60.00           151  41  mm  60.00   1           151
Frame#  Time    R.ASIS          L.ASIS          V.Sacral            R.Thigh.Upper           R.Thigh.Front
        X1  Y1  Z1  X2  Y2  Z2  X3  Y3  Z3  X4  Y4  Z4  X5  Y5  Z5  X6  Y6  Z6  X7  Y7  Z7  X8  Y8  Z8  X9  Y9

1   0.000000    617.247620  1055.275020 170.781980  639.606380  1044.258420 -88.909790  430.869840  1051.264650
2   0.017000    617.998110  1053.217530 168.513170  641.236210  1042.278560 -90.932130  432.340610  1050.237430
3   0.033000    620.292240  1051.771240 165.859380  643.596920  1041.060790 -94.307220  434.099430  1049.341430
4   0.050000    621.540410  1050.552120 163.532500  646.751040  1040.356810 -96.861880  436.279940  1048.707150
5   0.067000    624.588440  1050.928340 161.246140  649.254150  1041.425170 -98.484610  438.827940  1048.451050
6   0.083000    628.158630  1051.420170 158.448990  652.041260  1043.046510 -101.857370 441.572050  1048.661250
7   0.100000    630.807740  1051.996830 155.282730  654.943360  1045.552490 -104.842580 444.306520  1049.387570
8   0.117000    634.357300  1053.598880 151.485310  656.464110  1048.434810 -108.355420 446.830750  1050.622310
9   0.133000    636.586060  1055.256590 148.460540  658.681400  1051.142090 -111.495770 448.952150  1052.288210
10  0.150000    637.739260  1057.854370 144.716320  660.529910  1054.052980 -115.064610 450.499790  1054.273560
11  0.167000    641.120480  1061.533570 141.792310  660.938480  1056.792600 -118.131740 451.373780  1056.487180
12  0.183000    639.209230  1064.013920 139.109220  661.469180  1059.865360 -121.664920 451.591400  1058.870480
13  0.200000    638.712590  1066.867800 136.076390  660.868650  1062.079960 -124.353650 451.267300  1061.385130
14  0.217000    637.317320  1069.549930 132.966080  659.445010  1063.640260 -127.404200 450.541020  1064.004150
15  0.233000    638.418640  1073.621700 129.326050  658.349490  1065.388060 -130.494800 449.530030  1066.674320
16  0.250000    636.331540  1076.927250 126.842520  656.056340  1067.725590 -132.950560 448.297940  1069.277100
17  0.267000    635.992430  1080.683590 123.968220  653.927430  1068.169430 -136.447600 446.821350  1071.651000
18  0.283000    635.015380  1083.935180 121.069540  650.751040  1069.207150 -138.608920 445.037510  1073.644900
19  0.300000    633.406190  1086.531010 118.334630  648.428770  1070.045530 -141.609800 442.938080  1075.151250
20  0.317000    630.107670  1087.621830 116.429790  644.916750  1069.986330 -143.309980 440.562900  1076.133300
```

FIGURA 2.19: Detalle parcial del comienzo de las primeras columnas de un archivo de registro de trayectoria de marcadores.

```
<ControlLinear name="glut_med1_r.excitation">
    <is_model_control> true </is_model_control>
    <extrapolate> true </extrapolate>
    <default_min>       0.02000000 </default_min>
    <default_max>       1.00000000 </default_max>
    <filter_on> false </filter_on>
    <use_steps> true </use_steps>
    <x_nodes/>
    <min_nodes/>
    <max_nodes/>
    <kp>      100.00000000 </kp>
    <kv>       20.00000000 </kv>
</ControlLinear>
```

FIGURA 2.20: Muestra de un controlador lineal dentro de un archivo de controladores empleado por el método CMC.

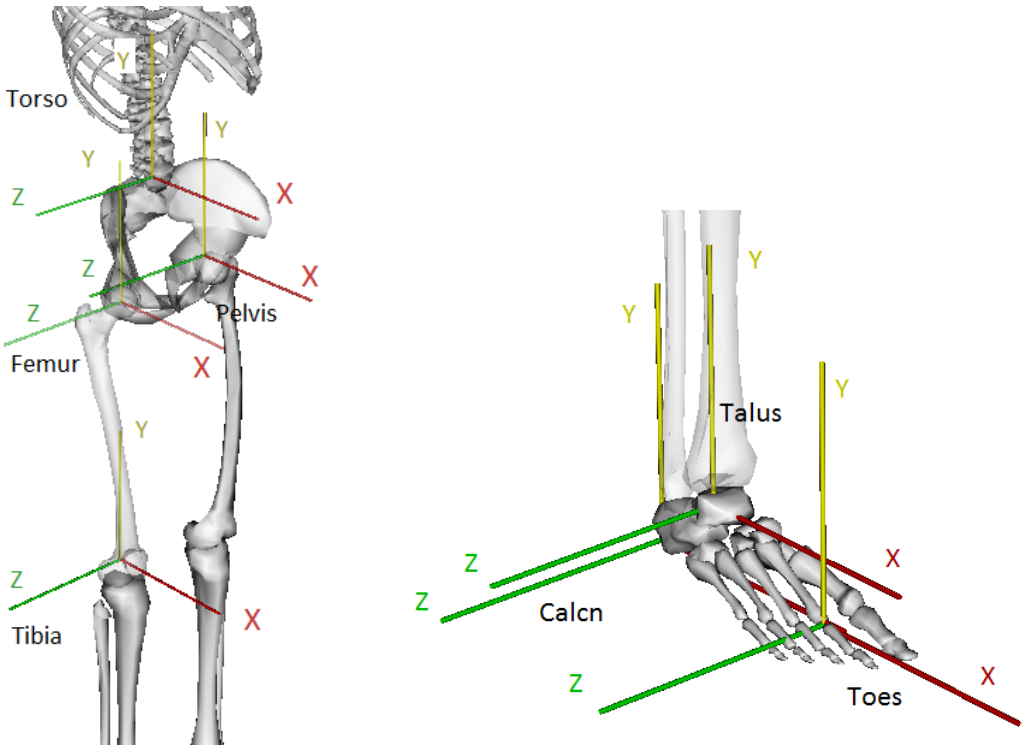

FIGURA 2.21: Centros de cada cuerpo del modelo Gait2354.

2.4.2. Datos de marcadores y escalado

Para el escalado del modelo se adjuntaron al modelo un archivo de posición *subject01_static.trc* con la posición de 49 marcadores a través de 300 imágenes a una frecuencia de muestreo 60 Hz con el sujeto en una posición estática. Para las operaciones de cinemática inversa, al modelo se le adjuntan datos sobre la posición de 39 marcadores medidos en laboratorio, junto con sus trayectorias en una medición de marcha de un sujeto. De ahí se obtuvo el archivo *subject01_walk.trc*, con la información de posición de de 41 marcadores a través de 151 imágenes con una frecuencia de muestreo de 60 Hz. El número de marcadores de los datos experimental es superior a los marcadores virtuales que se añadirán al modelo, así que esos marcadores experimentales sin correspondencia serán ignorados.

El pie como un único sólido rígido Un dato interesante de los datos de trayectoria es que no indicarán ningún movimiento en las coordenadas *mtp_angle* y *subtalar_angle* en ninguno de los dos pies. De este dato y el hecho de que el modelo no tiene músculos que controlen la flexión de los dedos del pie se decidirá tratar para el cálculo posterior de fuerzas de contacto al pie como un sólido rígido.

2.4.3. Datos de fuerzas de contacto con el suelo experimentales

Se ha tomado la hipótesis, dada la ausencia de fuerzas tangenciales significativas que el sujeto caminó por una cinta transportadora a una velocidad constante. La cinta transportadora tenía acoplada placas sensores de presión capaces de medir el punto medio de aplicación de éstas, su valor y dirección y el momento causado por estas fuerzas al punto de aplicación. En la marcha se produjo un archivo de almacenamiento llamado *subject01_walk1_grf.mot* que contenía las fuerzas, posición y par medidas por cada sensor con una frecuencia de muestreo de 600 Hz, en 1501 mediciones. Estos valores se emplearon como referencia en el proceso de optimización de la posición de las esferas en el apartado 3.

FIGURA 2.22: Posible colocación de marcadores en el pie del modelo.

Nombre modelo	Nombre en español	Aclaración
ground	Tierra	Cuerpo de referencia
pelvis	Pelvis	Los demás cuerpos parten de aquí
femur_r	Fémur derecho	
tibia_r	Tibia y peroné derecho	Unión de ambos huesos en un sólo sólido
talus_r	Talo	Conecta el pie con la pierna
calcn_r	Calcáneo y escafoides derechas	Equivale al talón
toes_r	Metatarsos y falanges	Dedos tratados como un mismo cuerpo
femur_l	Fémur izquierdo	
tibia_l	Tibia y peroné izquierdo	Unión de ambos huesos en un sólo sólido
talus_l	Talo	Conecta el pie con la pierna
calcn_l	Calcáneo y escafoides izquierdas	Equivale al talón
toes_l	Metatarsos y falanges	Dedos tratados como un mismo cuerpo
torso	Torso	Parte superior a partir de la pelvis

CUADRO 2.1: Lista de cuerpos en el modelo Gait2354

Coordenada	Sólido padre	Sólido hijo	Tipo
pelvis_tilt	ground	pelvis	Rotación pelvis eje Z.
pelvis_list	ground	pelvis	Rotación pelvis eje X.
pelvis_rotation	ground	pelvis	Rotación pelvis eje Y.
pelvis_tx	ground	pelvis	Posición pelvis eje X.
pelvis_ty	ground	pelvis	Posición pelvis eje Y.
pelvis_tz	ground	pelvis	Posición pelvis eje Z.
hip_flexion_r	pelvis	femur_r	Rotación fémur derecho eje Z.
hip_adduction_r	pelvis	femur_r	Rotación fémur derecho eje X.
hip_rotation_r	pelvis	femur_r	Rotación fémur derecho eje Y.
knee_angle_r	femur_r	tibia_r	Flexión de rodilla derecha. Combina rotación y desplazamiento.
ankle_angle_r	tibia_r	talus_r	Flexión del tobillo derecho.
subtalar_angle_r	talus_r	calcn_r	Desviación del talo derecho frente al tobillo.
mtp_angle_r	calcn_r	toes_r	Flexión dedos del pie derecho. En este trabajo es siempre cero.
hip_flexion_l	pelvis	femur_l	Rotación fémur izquierdo eje Z.
hip_adduction_l	pelvis	femur_l	Rotación fémur izquierdo eje X.
hip_rotation_l	pelvis	femur_l	Rotación fémur izquierdo eje Y.
knee_angle_l	femur_l	tibia_l	Flexión de rodilla izquierda. Combina rotación y desplazamiento.
ankle_angle_l	tibia_l	talus_l	Flexión del tobillo izquierdo.
subtalar_angle_l	talus_l	calcn_l	Desviación del talo izquierdo frente al tobillo.
mtp_angle_l	calcn_l	toes_l	Flexión dedos del pie izquierdo. En este trabajo es siempre cero.
lumbar_extension	pelvis	torso	Rotación del torso eje Z
lumbar_bending	pelvis	torso	Rotación del torso eje X
lumbar_rotation	pelvis	torso	Rotación del torso eje Y

CUADRO 2.2: Grados de libertad del modelo Gait2354

Nombre músculo en el modelo	Nombre del músculo
glut_med1_r	Glúteo medio 1 derecho
glut_med1_2	Glúteo medio 2 derecho
glut_med3_r	Glúteo medio 3 derecho
bifemlh_r	Bíceps crural de cabeza larga derecho
bifemsh_r	Bíceps crural de cabeza corta derecho
sar_r	Sartorio derecho
add_mag2_r	Aductor mayor 2 derecho
tfl_r	Tensor de la fascia derecho
pect_r	Pectíneo derecho
grac_r	Gracius derecho
glut_max1_r	Glúteo mayor 1 derecho
glut_max2_r	Glúteo mayor 2 derecho
glut_max3_r	Glúteo mayor 3 derecho
iliacus_r	Psoas-ilíaco derecho
psoas_r	Psoas mayor derecho
quad_fem_r	Cuadrado crural derecho
gem_r	Músculo no identificado en el glúteo derecho
peri_r	Piramidal de la pelvis derecho
rect_fem_r	Recto anterior derecho
vas_int_r	Vasto interno derecho
med_gas_r	Gastrocnemio medio derecho
soleus_r	Sóleo derecho
tib_post_r	Tibial posterior derecho
tib_ant_r	Tibial anterior derecho
ercspn_r	Erector de la columna derecho
intobl_r	Oblicuo interno abdominal derecho
extobl_r	Oblicuo externo abdominal derecho

CUADRO 2.3: Lista de los 27 actuadores musculares derechos del modelo Gait2354, el lado izquierdo posee los mismos músculos.

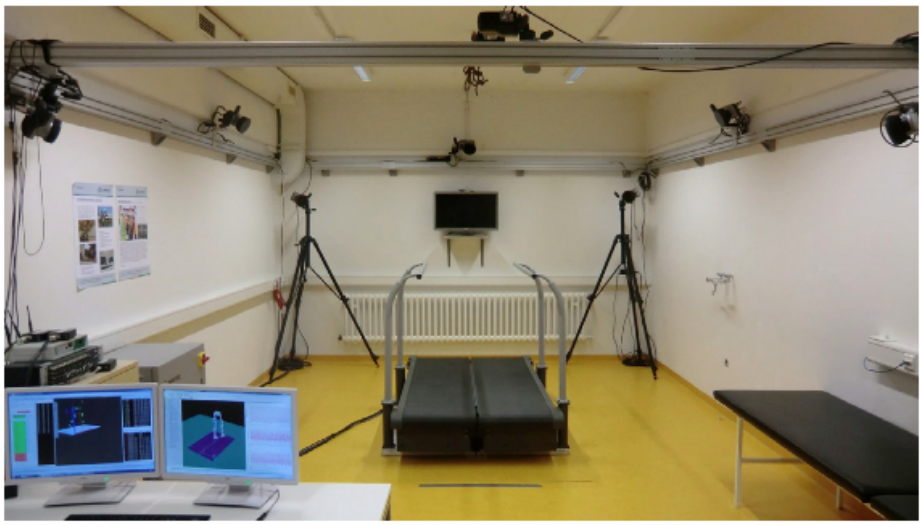

FIGURA 2.23: Posible equipamiento empleado en la captura de movimientos.

Capítulo 3

Optimización del lugar de las esferas de contacto

3.1. Descripción

Antes de comenzar con las simulaciones dinámicas de OS y CMC con OpenSim, el programa CreaEsferas.m se encarga de, mediante un proceso de optimización de una función fuertemente no lineal, de encontrar la mejor aproximación de la colocación de las esferas y sus parámetros de rigidez y amortiguamiento. Esa configuración de esferas luego se escribirá en el archivo *esferas.txt*, que podrá emplear MiCMCplugin.exe para crear el modelo adecuado.

3.2. Elecciones de diseño

Las siguiente hipótesis fueron adoptadas a la hora de definir las esferas. Como referencia inicial se utilizó el modelo creado por [21]. La mayoría de las elecciones vienen derivadas de la necesidad de reducir la cantidad de parámetros a optimizar al definir las esferas.

Simetría

Dado que se pretende calcular la cinemática de un ciclo de marcha completo, ninguno de los parámetros de las esferas dependerá de a qué lado del cuerpo del modelo correspondan, así ambos pies presentarán una distribución de esferas simétricas. Una ventaja de esta aproximación es que evita duplicar el número de parámetros a optimizar.

Esferas alineadas en el pie

El parámetro posición lateral local en las esferas (símbolo griego) será siempre cero. Así, las esferas estarán todas en un mismo plano.

Radio constante

Para reducir parámetros, el radio no se optimizará para las esferas. Sólo la posición de éstas.

Esferas tratadas por grupos

Un tratamiento más exhaustivo trataría cada esfera y sus parámetros de manera independiente en la optimización. En este proyecto se trató que las seis esferas que pertenecen al cuerpo talón formas todas parte de un mismo grupo con una misma distancia relativa

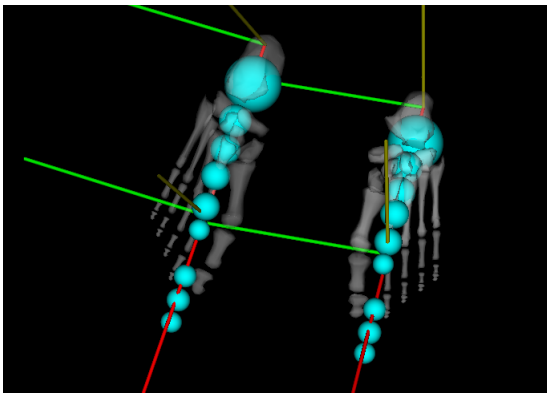

FIGURA 3.1: Se observa que en cada pie cada uno de los dos grupos de esferas sigue una misma dirección.

entre sí y que las tres esferas que pertenecen al cuerpo dedos forman parte de otro grupo con distancias entre sí constantes; la razón de ello es para reducir tiempo de cálculo pues de otro modo sería inabarcable para los recursos de potencia de cálculo que se disponían. Los parámetros que determinan la relación entre las esferas y los radios de éstas vienen en la siguiente tabla 3.1.

Plataforma horizontal

La altura de la plataforma se tratará como un parámetro independiente.

Parámetros de fuerzas tangenciales

No se tratarán las fuerzas tangenciales para la optimización, en ningún sentido.

3.3. Parámetros a optimizar

Con las elecciones de diseño de la sección anterior, quedan unos siete parámetros disponibles para optimizar.

1. Altura h de la plataforma respecto al cuerpo de referencia *ground*
2. Posición horizontal x_1 del grupo talón
3. Posición vertical y_1 del grupo talón
4. Posición horizontal x_2 del grupo dedos
5. Posición vertical y_2 del grupo dedos
6. Ángulo en el plano X-Y θ_1 del grupo talón
7. Ángulo en el plano X-Y θ_2 del grupo dedos

Estos siete parámetros compondrán un vector x que será el argumento de entrada de la función a optimizar. Para los dos grupos relativos a cada cuerpo, los parámetros se explican de la siguiente manera. Sea una esfera de contacto i, en el grupo j; su posición relativa respecto al centro de su cuerpo asociado c es:

Grupo	Esfera	Radio (m)	x_{ci} (m)	y_{ci} (m)
1	1	0,0292	-0,0947	-0,0153
1	2	0,0166	-0,0498	-0,0005
1	3	0,0155	-0,0195	-0,0040
1	4	0,0149	0,0091	-0,0034
1	5	0,0143	0,0412	-0,0086
1	6	0,0108	0,0645	-0,0156
1	1	0,0117	-0,0267	-0,0151
2	2	0,0120	-0,0022	-0,0201
3	3	0,0108	0,0211	-0,0224

CUADRO 3.1: Parámetros de cada grupo de esferas

$$\overrightarrow{r_{ci}} = \overrightarrow{\begin{pmatrix} x_j \\ y_j \\ 0 \end{pmatrix}} + \begin{pmatrix} \cos\theta_j & -\sin\theta_j & 0 \\ \sin\theta_j & \cos\theta_j & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \overrightarrow{\begin{pmatrix} x_{ci} \\ y_{ci} \\ 0 \end{pmatrix}} \tag{3.1}$$

En la anterior tabla, los valores x_{ci} y y_{ci} del vector $(x_{ci}, y_{ci}, 0)^T$ se obtienen de la siguiente tabla, que muestra las posiciones individuales y radios de cada una de las nueve esferas:

3.4. Función a optimizar

El objetivo de esta sección del proyecto es encontrar una geometría de las esferas de contacto que cumpla las siguiente condiciones:

1. Haga contacto con el suelo cuando se produce en la referencia y además tenga esta fuerza de contacto un valor similar.
2. No haga contacto con el suelo cuando no se produce en la referencia.
3. El punto de aplicación de contacto de ambas fuerzas sea el mismo, o estén muy cercanos.

Así, el programa que obtiene la función a optimizar deberá, al recibir unos parámetros de esferas y unas fuerzas de contacto de referencia en un conjunto de momentos dados, ejecutar una simulación de fuerzas de contacto llamando a EstructuraIterPosicion.m que devolverá un conjunto con el sumatorio de las fuerzas de contacto de cada grupo, agrupándolas por pies.

La función J a optimizar es la siguiente:

$$\text{mín } J(x) = \sum_i^2 \sum_{t=t_0}^{t_1} f_i(x,t)$$

donde:

$$f_i(x,t) = \begin{cases} \begin{cases} 0 & si \quad R(t)_i = 0 \\ a_{i1} \cdot \ln\left(R(t)_i\right) & si \quad R(t)_i \neq 0 \end{cases} & si \quad E(x,t)_i = 0 \\ \begin{cases} a_{i2} \cdot \ln\left(E(x,t)_i\right) & si \quad R(t)_i = 0 \\ a_{i3} - a_{i4} \cdot \left|\ln \frac{E(x,t)_i}{R(t)_i}\right| & si \quad R(t)_i \neq 0 \end{cases} & si \quad E(x,t)_i \neq 0 \end{cases} \tag{3.2}$$

Aclaración de los índices usados

i es el índice para cada pie. Vale 1 para el derecho y 2 para el izquierdo.

t es el instante de tiempo. Está comprendido discretamente entre t_0 y t_1.

x es el vector de parámetros dado.

$R(t)_i$ es la fuerza de contacto vertical de referencia para el pie i en el momento t.

$E(x,t)_i$ es la fuerza de contacto calculada para el pie i en el momento t con los parámetros x.

a_{i1} es el peso de la penalización de no penetrar el suelo cuando sí lo hace la referencia para el pie i.

a_{i2} es el peso de la penalización de penetrar el suelo cunado no lo hace la referencia para el pie i.

a_{i3} es el peso de la bonificación por por penetrar cuando lo hace la referencia para el pie i.

a_{i4} es el peso de la penalización por obtener una fuerza de contacto excesiva o demasiado leve para el pie i.

Los últimos 4 parámetros si no están bien pesados pueden llevar una optimzación sesgada.

Logaritmos En la fórmula 3.2 se pesan los resultados usando logaritmos de las fuerzas en vez de el valor actual de éstas; esto es debido a que una penetración excesiva del modelo durante las iteraciones de la optimización podría llevar a arrojar excesivas fuerzas, que comparadas a la referencias podría resultar en una penalización excesiva. Por ello, se optó a comparar las fuerzas según logaritmos.

3.5. Métodos de optimización

La búsqueda de un método adecuado de optimización no es trivial, encontrar un método que sea lo suficiente consistente y rápido es una tarea todavía debatida.

No idoneidad de los métodos por gradiente Los métodos de optimización por gradiente como por ejemplo *fminbnd* en Matlab no eran adecuados en esta optimización debido a que calcular un gradiente para siete parámetros era computacionalmente demasiado caro y porque al ser la función a optimizar fuertemente no lineal y con bastante mínimos locales no globales, el método era muy propenso en obtener un mínimo local como resultado.

Métodos de búsqueda directa Los métodos de búsqueda directa parten del desconocimiento a prior del comportamiento de la función sobre las variaciones de su resultado al variar los parámetros. Tratan a la función como una caja negra y suelen ser más lentos que los métodos de búsqueda por gradiente, pero pueden encontrar resultados óptimos o cercanos al óptimo en funciones donde los métodos por gradiente no podrían funcionar.

Otros métodos potencialmente idóneos no utilizados Existen otro métodos de búsqueda directa que podrían haberse empleado, como los estocásticos, siendo algunos de estos los algoritmos de *simulated annealing* o recocido simulado[16], o los algoritmos genéticos[20]. Sin embargo, por su sencillez conceptual, se empleó el método de búsqueda por patrón.

El método de búsqueda por patrón

Mencionado por Hooke [13], el método de búsqueda por patrón trata de encontrar el valor óptimo evaluando los alrededores de un punto y ampliando o disminuyendo el paso según lo óptimo de los alrededores del punto , según un patrón dado. Para este proyecto se ha empleado el patrón *2N positivo*. Su funcionamiento, junto con el método en sí viene explicada en el siguiente algoritmo.

Explicación Sea un dominio $\Re$ no necesariamente acotado para una cantidad de n variables x con un valor inicial $x^0 = [x_1^0, x_2^0, ..., x_n^0]$, un valor de paso inicial p con un valor mínimo de paso p_l y una función $f(x)$ evaluable en el dominio $\Re$, el algoritmo de búsqueda por patrón funciona así:

1. Se evalúa un punto inicial de búsqueda $f(x^0)$ y se almacena el valor en f_{opt}. Guardamos el punto actual x^0 como x^{act}.

2. Se investiga el entorno de $f(x^{act})$ de la siguiente manera, con el objetivo de encontrar un punto en ese entorno mejor que x^{act}:

   a) Se evaluará primero los puntos del entorno alejados una distancia de paso p en la dirección positiva de cada variable, siempre que estos puntos estén dentro de la región permitida. Por ejemplo, el primer punto a evaluar sería $x^1 = \left[x_1^{act} + p, x_2^{act}, ..., x_n^{act}\right]$, donde $x^1 = x^{act} + p \cdot [1, 0, 0, ..., 0]$.

   b) Si para el punto evaluado $f(x^i) < f_{opt}$, se almacena el valor $f(x^i)$ en f_{opt} y se pasa al paso *4*. Si no, se evaluará la otra dirección hasta haber visto todas las direcciones alejadas una distancia p en una dirección positiva para cada variable. Ejemplo de punto en el entorno de x^{act} alejado dirección negativa respecto a una variable: $x^{n+2} = x^{act} + p \cdot [0, -1, 0, ..., 0] = \left[x_1^{act}, x_2^{act} - p, ..., x_n^{act}\right]$.

   c) Si no se encuentra ningún valor que mejore al óptimo previo después de evaluar en las $2 \cdot n$ direcciones, se pasa al punto *3*.

3. Se reducirá el valor del paso por un factor $0 < c < 1$. Si el nuevo valor de paso p alcanza un valor inferior al paso mínimo p_l, se dará por concluido el algoritmo y con el actual resultado f_{opt} y el vector x^{act}. Si no es así, se repetirá el paso *2* con ese nuevo valor de paso $p = c \cdot p$.

4. Si se encontró un valor mejor $x^i \mid f(x^i) < f_{opt}$, se aumentará el valor del paso en un factor c y se volverá al paso *2* con x^{act} siendo ahora x^i.

El método de búsqueda en el entorno de un punto puede variar, principalmente en la forma de asignar valores a los puntos en el entorno, pero el método descrito es el que se ha aplicado. El criterio de parado puede variar. En este proyecto se empleó el del tamaño del paso junto con un número máximo de evaluaciones de la función. Este método está implementado en Matlab a través de la función *patternsearch*.

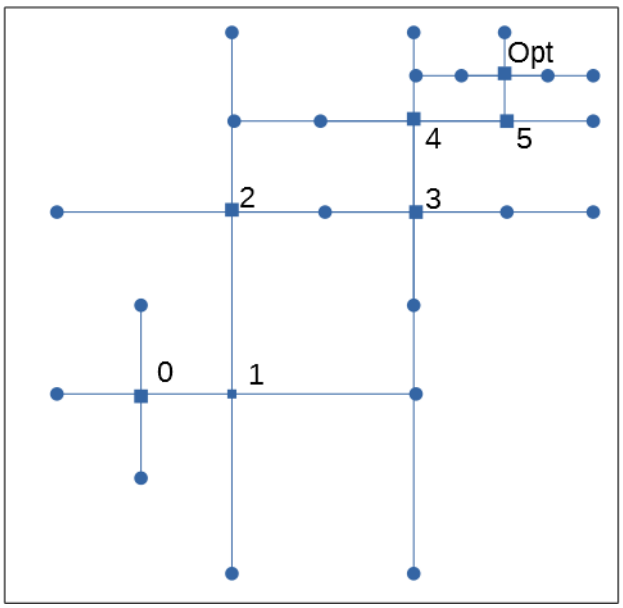

FIGURA 3.2: Ejemplo de búsqueda de óptimo por patrón en una región acotada. Los círculos representan puntos evaluados no favorables. Los cuadrados indican un punto favorable evaluado y el orden en que fue iterado.

Capítulo 4

Aplicaciones informáticas

4.1. El programa MiCMCplugin.exe

Unos de los objetivos de este P.F.C. es el cálculo de una simulación de marcha humana mediante un sistema de control que permita adaptar la cinemáticas a las fuerzas de contacto calculadas mediante esferas de Hunt-Crossley. Para ello, se programó con la API de OpenSim un programa capaz de hacer los cálculos de CMC y a la vez hacer llamadas externas a un segundo programa hecho mediante Matlab, que funcionará de manera simultánea. En este capítulo se tratará el programa tratado para ello.

4.1.1. Requerimientos

El programa *MiCMCplugin.exe* es capaz tanto de hacer cálculos de CMC como SO. Para funcionar necesitará cumplir las siguientes condiciones referidas a continuación.

Sistema operativo

Sólo funcionará en sistemas Windows de 64 bits, debido a una limitación del compilador integrado en la herramienta *Compiler* de Matlab.

Archivos de configuración

Tanto para cálculos de CMC como SO será necesario un archivo de configuración que detalle qué modelo se emplea, los tiempos de simulación, la configuración del integrador interno y otros requerimientos. Estos deberán ir dentro de una carpeta con el nombre *2354*, que esté dentro de la carpeta donde se halle el programa. De otra forma podrían producirse errores en el programa.

Llamada mediante archivos batch

Durante su funcionamiento *MiCMCplugin.exe* generará otros archivos auxiliares (explicados en REFERENCIA) usados principalmente para comunicarse con el segundo programa. Será necesario borrar manualmente o mediante un archivo *batch* antes de ejecutar *MiCMCplugin.exe* para una segunda simulación.

Comunicación entre MiCMCplugin y Programa

Además, *MiCMCplugin.exe* utiliza comunicación mediante la tecnología TCP-IP para comunicarse con Programa.exe, así que será necesario otorgarle privilegios para que pueda superar el cortafuegos de Windows. También se utilizan los puertos de comunicación 8887 y 8888, así que deberán estar libres.

4.1.2. Funcionamiento de MiCMCplugin.exe

Arranque

Habrá que ejecutar programa con el siguiente comando, sea mediante un archivo batch o mediante la consola de Windows, si se busca hacer un cálculo de CMC:

```
CMCplugin.exe -S 2354\nombreArchivoConfiguracionCMC.xml
```

y si se busca efectuar un SO:

```
CMCplugin.exe -SO 2354\nombreArchivoConfiguracionSO.xml
```

Lectura del modelo

El programa leerá el archivo de configuración indicado, cargando en memoria su contenido según código propio de OpenSim. A continuación, el programa leerá el archivo *esferas.txt* generado previamente con el uso del código de Matlab referenciado en el capítulo 3 . Tras la lectura del archivo *esferas.txt*, se añadirán al modelo *.osim* en memoria las esferas de contacto indicadas en el archivo de texto referido.

Arranque del programa auxiliar

Una vez MiCMCplugin.exe tiene en memoria una clase *model* con la información del modelo actualizada, creará un nuevo modelo *.osim* con las esferas y fuerzas de contacto añadidas, de nombre *Expandido_X.osim* donde X es el modelo referido en el archivo de configuración mencionado en el apartado 4.1.2.

Después creará un archivo de texto (modelo.txt) con la dirección del modelo expandido y un número indicando al programa de Matlab un tiempo de espera en mili-segundos (esto será detallado en la sección REF:TiemposDeEspera). Las dos siguiente líneas muestran el contenido de un posible archivo *modelo.txt*.

```
archivoModelo="Expandido_subject01_simbody_adjusted.osim"
tiempoEspera="20"
```

Por último, el programa llamará a un archivo *batch* llamado *Programa.bat* cuya única función será ejecutar el programa de Matlab en una segunda ventana, para así permitir la ejecución de ambos de manera simultánea.

La clase "MiFzaCntcMatlab"

Esta clase, añadida a los modelos según las indicaciones de *esferas.txt*, es la fuerza que permite a OpenSim llamar a mi programa auxiliar de Matlab para calcular las fuerzas de cada pie. Tiene un cuerpo asociado al que ejerce la fuerza, que será *calcn_r* en el caso del pie derecho y *calcn_l* en el caso del pie derecho. Las dos primeras veces que el integrador de OpenSim lea esta fuerza creará un archivo llamado *llamada.txt* con las direcciones de dos archivos de coordenadas de OpenSim, que indican los valores de las coordenadas generalizadas en el momento dado.

```
archivoQ="...Ruta completa...\Qs.sto"
archivoU="...Ruta completa...\Us.sto"
NoBorrar
```

El número de veces que se llama a la clase mediante el integrador es controlado por un archivo de texto dentro de la carpeta *2354* llamado *contador.txt*. El número entero que contiene se verá incrementado en uno cada vez que se llame a una fuerza *MiFzaCntcMatlab*. Su razón de existencia es debido a que la orden *ComputeForce* de OpenSim está limitada por el parámetro de C++ *const*.

Este parámetro impide la modificación de valores internos a las clases una vez estas se encuentran definidas y creadas en el programa, durante el tiempo de ejecución. Su uso permite la programación de programas más complejos y menos propensos a errores, pero para este proyecto obligó a la búsqueda de formas de evitar su limitación, como fue el uso del fichero de texto auxiliar *contador.txt*.

Una vez el programa de Matlab ha calculado la fuerza, éste escribirá un archivo llamado *respuesta.txt*, otro llamado *pre-respuesta.txt* y otro archivo de almacenamiento de fuerzas externas de OpenSim de formato *.xml* junto con su correspondiente archivo *sto* con los datos numéricos de las fuerzas. El objetivo del archivo *respuesta.txt* es indicar la dirección del archivo de almacenamiento. Un ejemplo de un archivo de este tipo contendría:

```
archivo configuracion resultados XML=
"...Ruta completa..\Expandido_NOMBRE_MODELO_fuerzas_por_pie_conjuntas.xml"
archivo datos resultados MOT=
"...Ruta completa...\Expandido_NOMBRE_MODELO_adjusted_fuerzas_por_pie_conjuntas.mot"
```

El archivo *pre-llamada.txt*, que no tiene contenido, tiene como único objetivo avisar al programa de que *respuesta.txt* se ha escrito. Es un remanente de un método anterior que durante el desarrollo del proyecto se usó para evitar problemas de lectura y borrado de archivos y su existencia actual no tiene más necesidad que evitar posibles errores. Será borrado en cuanto el archivo de fuerzas sea leído.

Como se puede observar, tanta creación y borrado constante de archivos en el disco duro ralentizará el funcionamiento del programa considerablemente, hasta el punto de que cada llamada a cálculo de fuerzas podía tardar hasta 0,8 segundos. Por ello se optó por la solución de crear un servidor local TCP para la comunicación simultánea entre los dos programas.
A partir de la tercera lectura de fuerzas de contacto, ya no se escribirá un archivo de almacenamiento de extensión *sto* ni se escribirá un archivo de llamada. En vez de eso el programa creará un servidor local en el ordenador mediante el protocolo TCP con el que se comunicará con el programa de Matlab.

Cuando es necesario crear el servidor para enviar y recibir los datos de Matlab, es decir, a partir de la tercera llamada, la clase *MiFzaCntcMatlab* a partir de su clase interna *Llamador* enviará los datos de coordenadas generalizadas a la función *LlamaTCPIP*, cuyo funcionamiento se describe gráficamente en la rama derecha del esquema 4.1. La función *LlamaTCPIP* emplea código basado en el tutorial de la página web de Jason Campbell [6]. Una vez Matlab ha enviado el vector con las fuerzas, par y posición deseadas, *LlamaTCPIP* devolverá esos datos a *MiFzaCntcMatlab*, que los aplicará y seguirá adelante.

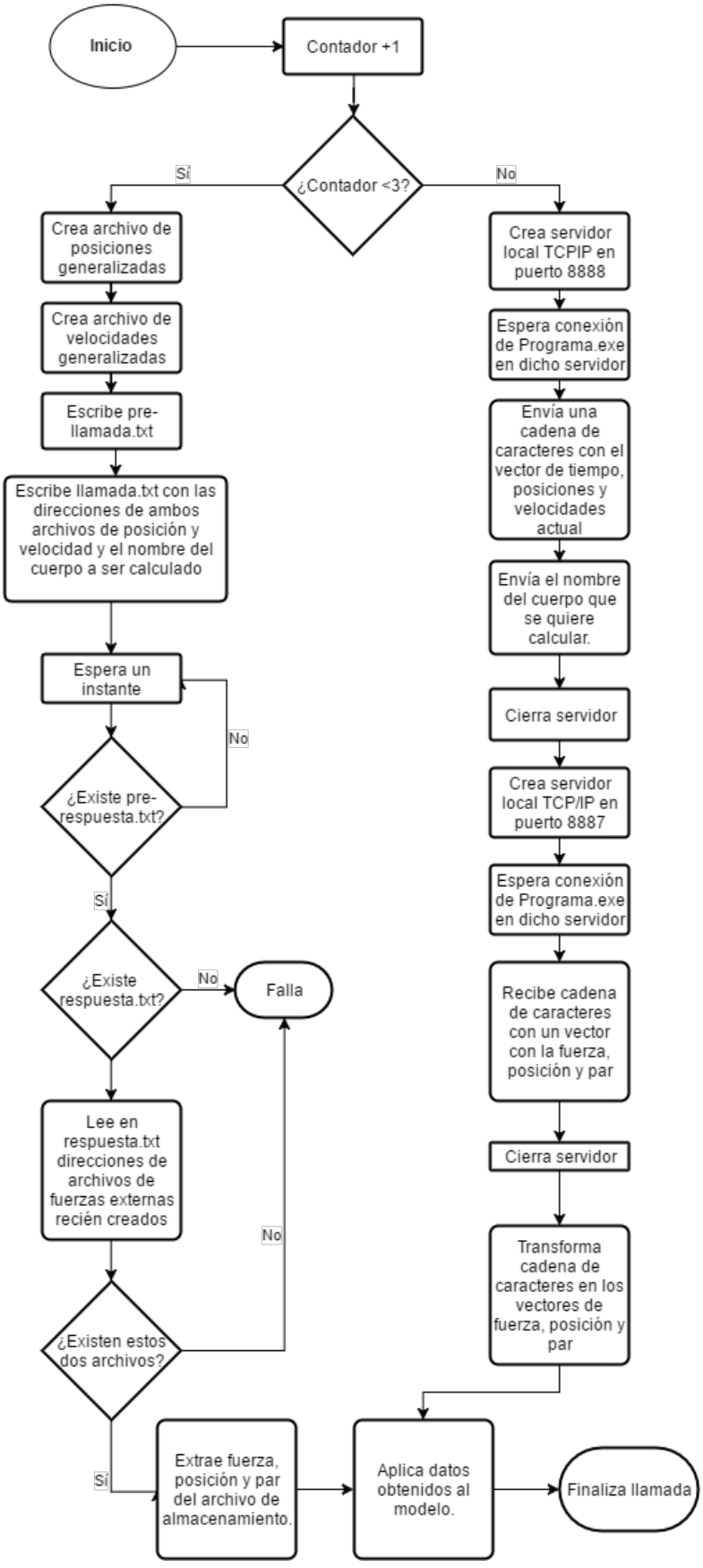

FIGURA 4.1: Diagrama de funcionamiento de la clase MiFzaCntctMatlab una vez su función propia ComputeForce es llamada por OpenSim.

4.2. El programa Programa.exe

4.2.1. Cometido

Esta parte del proyecto consiste en un programa escrito en Matlab que tendrá los siguientes objetivos:

- Leer el modelo de OpenSim expandido con las esferas y replicar su estructura dentro de Matlab.
- Recibir de *MiCMCplugin.exe* las coordenadas y velocidades generalizadas en cada instante de simulación.
- Calcular la posición y fuerza de contacto de cada una de las esferas para luego agruparlas en un vector con la posición, fuerza y momento.
- Enviar ese vector de vuelta al programa *MiCMCplugin.exe*.

Una vez *MiCMCplugin.exe* deje de llamar a *Programa.exe*, pasada una cantidad de tiempo determinada este último se cerrará automáticamente dejando un registro de ejecución llamado *diary.log*.

4.2.2. Estructura

En esta sección se detallará el funcionamiento de *Programa.exe* sin nombrar todas las funciones involucradas en su funcionamiento, de manera simplificada.

Adquisición de modelo y determinación de la jerarquía

Primero, el programa buscará el archivo de texto *modelo.txt* escrito por *MiCMCplugin.exe* para encontrar la dirección del modelo de *.osim* expandido por las esferas. Creará un modelo interno en la memoria del programa de Matlab, con una estrucutra de celdas emulando la estructura de llaves XML en el archivo *.osim*. Para la lectura del modelo se empleó la función *xml_readOSIM.m* del conjunto de herramientas Matlab-OpenSim-Pipeline-Tools [5].
Con el modelo almacenado en una estructura célula en la memoria de Matlab, el siguiente paso es determinar la jerarquía de cuerpos del modelo. Esto se realiza mediante la función *crea_jerarquia*, que en base de leer los valores de las articulaciones de tipo *CustomJoint* indicará las relaciones entre sólidos del modelo de forma explícita para cada sólido. Una vez la jerarquías de sólidos ha sido determinada, se iniciará un contador de lecturas a cero y pasará a la siguiente parte.

Adquisición de coordenadas, método escrito

El método de comunicación del programa de Matlab con el de C++ se efectúa de dos maneras diferentes, las dos primeras se hacen mediante escritura de datos en el disco y todas las siguientes mediante el servidor de TCP. Así, cuando el modelo pasa a esta parte, esperará la existencia de un archivo de texto creado por *MiCMCplugin.exe* llamado *pre-llamada.txt*. Cuando éste exista en la carpeta raíz, leerá entonces en *llamada.txt* la dirección de un archivo de coordenadas *.sto* y el nombre del cuerpo sobre el que *MiCMCplugin.exe* desea saber las fuerzas de contacto, que será *calcn_r* ó *calcn_l*. Ese archivo *.sto* contendrá un vector con los datos de un momento de tiempo y todas las coordenadas y velocidades del modelo. Una vez leídas las coordenadas, *pre-llamada.txt* será borrado, a la espera de que en una siguiente lectura de fuerzas vuelva a ser escrito y *Programa.exe* pasará al cálculo cinemático, dentro de la función *Calculador*.

Cinemática

La función *Calculador*, primero y siguiendo la cadena de jerarquía desde el sólido *ground* hasta el sólido que se le ha indicado como objetivo, tratará de calcular la posición y velocidad del centro virtual de cada cuerpo en esa jerarquía, en la función *proceso_iterativo_pos_y_vel_y_todo.m*, aplicando los valores de las coordenadas generalizadas para calcular el movimiento de las articulaciones.

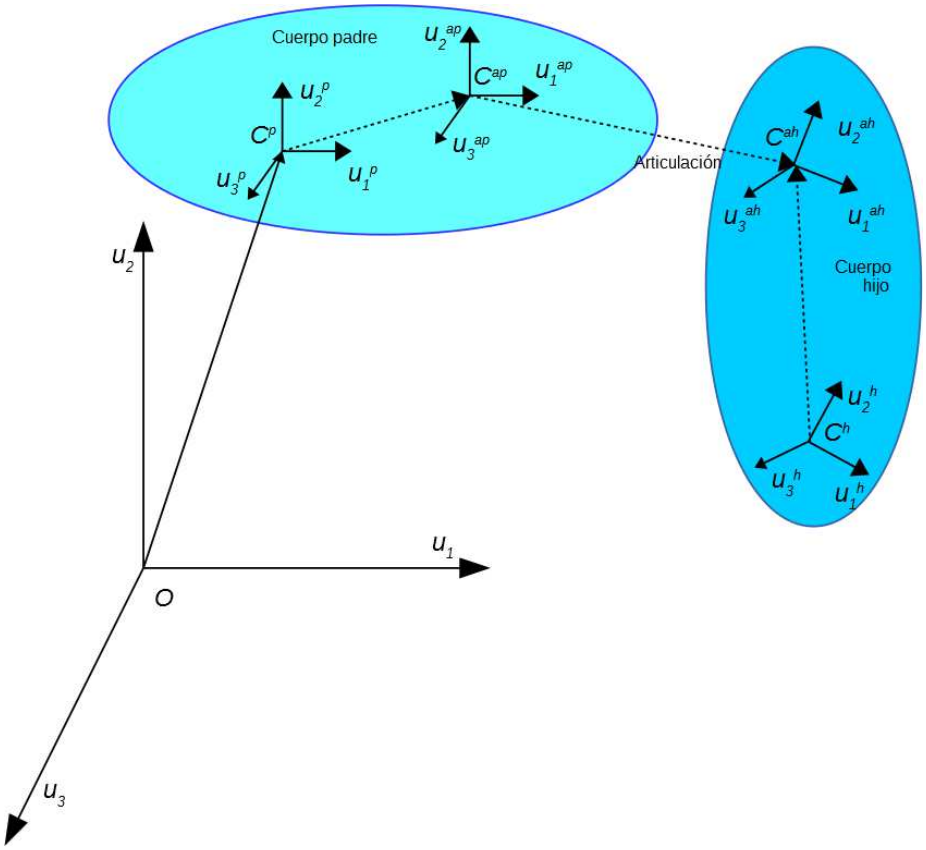

FIGURA 4.2: Relación entre sólidos en una articulación.

Ecuaciones de la articulación Aquí se muestran los cálculos de posición y velocidad realizados, calculados por *proceso_iterativo_pos_y_vel_y_todo.m*, siguiendo el modelo de articulación *CustomJoint* del apartado 2.3.2, basado en el capítulo 2 del texto de Shabana [27]. La ecuación del vector de posición del centro del sólido hijo O^h_C, empleando g para describir el sistema de coordenadas global, p para el sistema de coordenadas del padre y h para el sistemas de coordenadas del hijo.

$$\begin{aligned} \overrightarrow{OC^h}|_g &= \overrightarrow{OC^p}|_g + \overrightarrow{C^pC^{ap}}|_g + \overrightarrow{C^{ap}C^{ah}}|_g + \overrightarrow{C^{ah}C^h}|_g \\ &= \overrightarrow{OC^p}|_g + \overline{M}|_{p-g} \times \left(\overrightarrow{C^pC^{ap}}|_p + \overrightarrow{C^{ap}C^{ah}}|_p + \overline{M}|_{h-p} \times \overrightarrow{C^{ah}C^h}|_h \right) \end{aligned} \tag{4.1}$$

Ecuación de la matriz de giro $\overline{M}|_{ab}$ que transforma las coordenadas del sistema a al b. Tomando como que la matriz de giro está definida por 3 rotaciones diferentes, que conforman todas el producto de ellas la rotación final.

$$\overline{M}|_{ab} = \overline{M}_1 \cdot \overline{M}_2 \cdot \overline{M}_3 \tag{4.2}$$

Donde $\overline{M}_i$ es una matriz de giro concreta basada en los parámetros de alguien.

$$M = \begin{pmatrix} 1-2(\theta_2^2+\theta_3^2) & 2(\theta_1\theta_2-\theta_4\theta_3) & 2(\theta_1\theta_3+\theta_4\theta_2) \\ 2(\theta_1\theta_2+\theta_4\theta_3) & 1-2(\theta_1^2+\theta_3^2) & 2(\theta_2\theta_3-\theta_4\theta_1) \\ 2(\theta_1\theta_3-\theta_4\theta_2) & 2(\theta_2\theta_3+\theta_4\theta_1) & 1-2(\theta_1^2+\theta_2^2) \end{pmatrix} \tag{4.3}$$

Estos parámetros de Euler se pueden obtener del eje de rotación $[v_1, v_2, v_3]$ explicado en la *TransformAxis* y del valor de la coordenada de giro α, en radianes. Se ha cambiado el orden de los parámetros, colocando θ_0 como el cuarto parámetro, en vez de ser el primero y será nombrado θ_4 a partir de aquí.

$$\theta = [\theta_1, \theta_2, \theta_3, \theta_4] = \left[v_1 sen(\frac{\alpha}{2}),\ v_2 sen(\frac{\alpha}{2}),\ v_3 sen(\frac{\alpha}{2}),\ cos(\frac{\alpha}{2})\right] \tag{4.4}$$

Para el cálculo de la velocidad de un punto, tenemos la matriz de velocidad rotacional.

$$\dot{\overline{M}} = \overline{M} \times \overline{\omega} \tag{4.5}$$

$\overline{\omega}$ es el tensor anti simétrico de rotación,

$$\overline{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{4.6}$$

donde los tres términos del vector de velocidad de giro ω están compuesto por la suma de las combinaciones de n variaciones de rotación aplicables, según la articulación aplicable:

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \sum_{i=1}^{n} \dot{\alpha}_i [v] = \sum_{i=1}^{n} \dot{\alpha}_i \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ v_{3,i} \end{bmatrix} \tag{4.7}$$

Siendo $\dot{\alpha}_i$ la velocidad generalizada y $[v]$ el eje de rotación pertinente a esa coordenada. El término de velocidad de traslación en la articulación se obtiene rápidamente, con u como parámetro de velocidad generalizada en un eje vr :

$$\dot{\overrightarrow{C^{ap}C^{ah}}} = \sum_{i=1}^{n} u [vr] = \sum_{i=1}^{n} u \begin{bmatrix} vr_{1,i} \\ vr_{2,i} \\ vr_{3,i} \end{bmatrix} \tag{4.8}$$

Así, la ecuación de la velocidad del centro del sólido hijo en una articulación queda como:

$$\begin{aligned} \dot{\overrightarrow{OC^h}}|_g &= \dot{\overrightarrow{OC^p}}|_g + \dot{\overrightarrow{C^pC^{ap}}}|_g + \dot{\overrightarrow{C^{ap}C^{ah}}}|_g + \dot{\overrightarrow{C^{ah}C^h}}|_g \\ &= \dot{\overrightarrow{OC^p}}|_g + \dot{\overline{M}}|_{p-g} \times \left(\overrightarrow{C^pC^{ap}}|_p + \overrightarrow{C^{ap}C^{ah}}|_p\right) \\ &\quad + \overline{M}|_{p-g} \times \dot{\overrightarrow{C^{ap}C^{ah}}}|_p + \dot{\overline{M}}|_{h-p} \times \overrightarrow{C^{ah}C^h}|_p \end{aligned} \tag{4.9}$$

Cálculo de fuerzas

Después tratará de calcular la fuerza de contacto de cada esfera con la plataforma que pertenezca al sólido objetivo, agrupando las fuerzas de cada esfera para calcular así el punto de aplicación, la fuerza resultante y el momento de las fuerzas respecto al punto

de aplicación. Esto se realizará en las funciones *contacto_esfera_plano_definitivo.m* y *agrupador_fuerzas.m*. Las ecuaciones seguidas para el cálculo de las fuerzas normales se describieron en el apartado de fuerzas del Hunt-Crossley del capítulo 2.3.3.

Las fuerzas tangenciales se calcularon como se ha indicado, pero con una salvedad. Dado que el sujeto humano se encontraba caminando en una cinta transportadora para realizar las mediciones de posición de los marcadores, se modificó la velocidad de deslizamiento relativa entre el pie y el suelo en lo relativo al cálculo de fuerza tangencial, para conseguir que en los resultados las fuerzas tangenciales estuvieran equilibradas en dirección a lo largo de la marcha. Para ello se restó a la velocidad de contacto relativa unos 1,15 m/s en dirección delantera. El valor se averiguó mediante prueba y error.

Envío de valores de las fuerzas

Con las fuerzas ya calculadas, el programa tratará de comunicar con *MiCMCplugin.exe*. Las dos primeras veces escribirá un archivo de almacenamiento de datos de OpenSim con los valores de las fuerzas de contacto a través de la función *escribe_resultados.m*, luego escribirá un archivo de texto llamado *respuesta.txt* con las direcciones de los archivos de almacenamiento de fuerzas, y por último un archivo de texto llamado *pre-respuesta.txt*. La escritura de este último será lo que *MiCMCplugin.exe* tomará como señal para leer la dirección de los archivos de fuerza existente en el archivo *respuesta.txt*. La escritura de estos dos archivos de texto se realizará en la función *CreaRespuesta.m*.

Modo TCP-IP A partir de la tercera llamada la comunicación será por un servidor local TCP-IP, por lo tanto el programa se conectará al servidor local creado en el puerto 8887 por el programa *MiCMCplugin.exe* para enviar un vector con los valores de fuerza, posición y par deseados. Esto se hará mediante la función *ComunicadorTCPIP.m* que se encarga de ser cliente del servidor y enviar los datos. Dicha función emplea el conjunto de herramientas para comunicación TCP-IP para Matlab creadas por Peter Rydesäter [25].

Con el vector enviado, el programa borrará sus datos internos sobre el actual cálculo cinemático y de fuerzas de contacto y reiniciará su bucle de funcionamiento para recibir otro vector de coordenadas de *MiCMCplugin.exe*. Eso será así hasta que trascurra un tiempo prudencial, donde si no recibe ningún valor mediante el servidor en el puerto 8888, se cerrará, entendiendo que ha finalizado su uso.

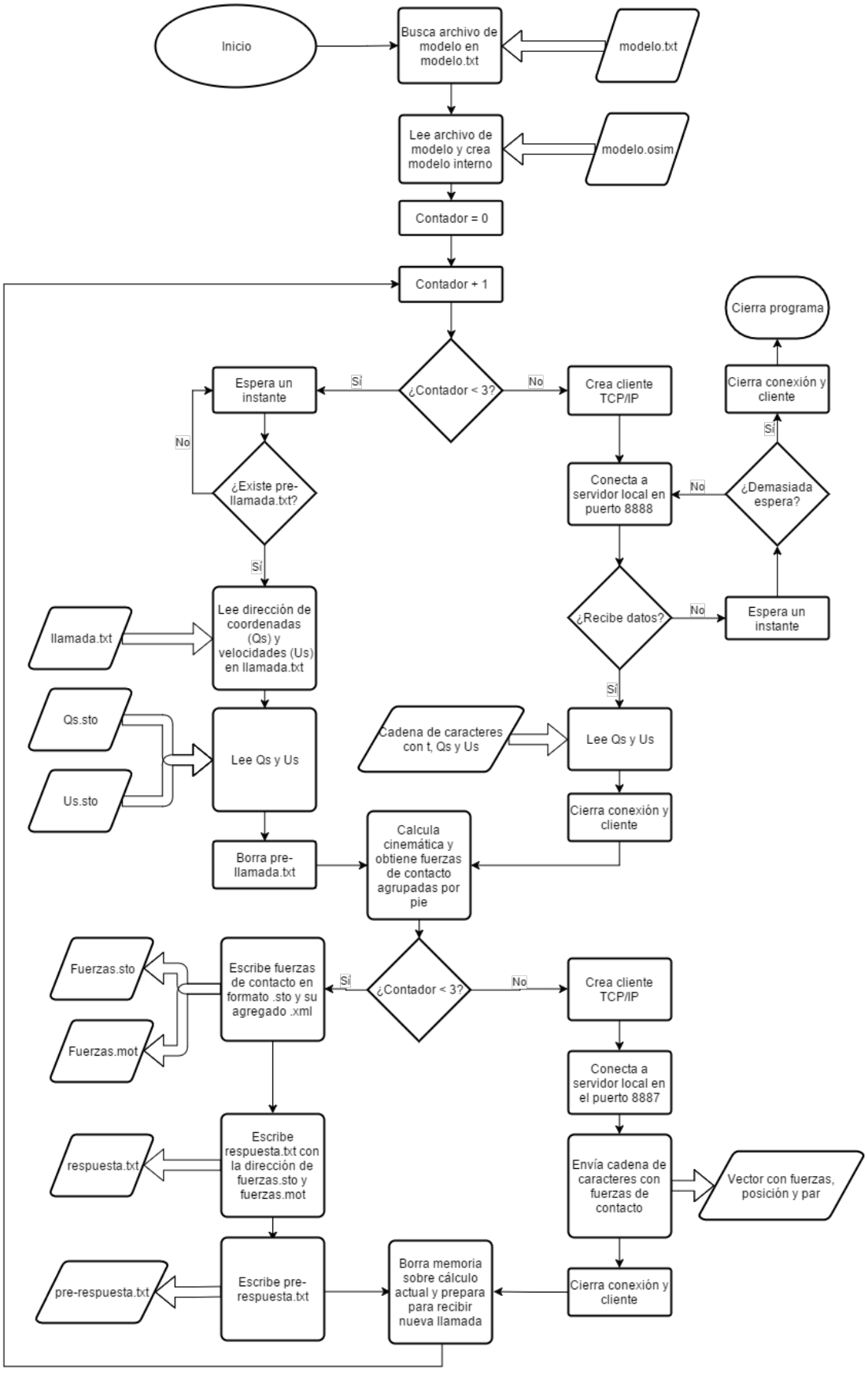

FIGURA 4.3: Funcionamiento de Programa.exe, resaltando las entrada y salida de datos.

Capítulo 5

Resultados

En este capítulo se informará de los resultados obtenidos mediante los métodos explicados en los capítulos anteriores.

5.1. Lugar de las esferas

Se analizaron dos problemas diferentes a la hora de calcular la posición de las esferas. Primeramente se realizó el cálculo de la posición de las esferas respecto a un ciclo de marcha completo referido a ambos pies, y luego se redujo el estudio al pie derecho durante la parte que corresponde aproximadamente a las tres primeras etapas de la fase de apoyo del ciclo de marcha.

5.1.1. Estudio del ciclo completo en ambos pies

Para el estudio del ciclo completo en ambos pies, se tomaron como referencia los datos de fuerza vertical de contacto del archivo *subject01_walk1_grf.mot*, los datos de posición del archivo *Kinematics_q.sto* y de velocidad del archivo *Kinematics_u.sto*. Se empleó el segmento de tiempo de ambos archivos comprendido entre 0,4 y 1,6, esto es, 1,2 segundos de duración a 60 Hz, unos 721 marcos de comparación. Primero se realizó una optimización inicial de valores de rigidez y amortiguamiento para las nueve esferas. Con esos valores, se realizó una optimización del lugar de las esferas posterior. Y con los valores de esa última optimización posterior se retocó nuevamente la rigidez y disipación para intentar afinar los resultados más. Con tal modelo, se obtuvieron los siguientes 9 parámetros, 7 para la optimización del lugar y 2 para la rigidez y amortiguamiento.
El archivo final de resultados, que sería empleado en los siguiente métodos puede encontrarse en los anexos electrónicos, aunque los resultados de las nueve variables pueden consultarse la siguiente tabla 5.1 y una comparación gráfica de las fuerzas en la figura 5.3.

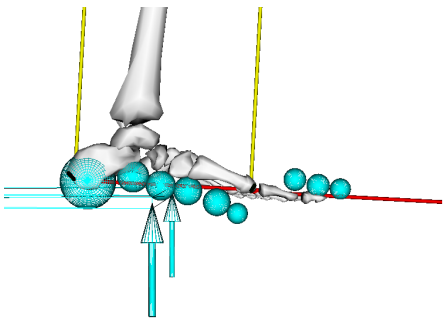

FIGURA 5.1: Posición de las esferas tras la optimización a ciclo completo.

Desviación de las esferas de la parte delantera Se puede observar que para estos resultados obtenidos con el optimizador las esferas de la parte delantera del pie se encuentran notablemente desviadas y desplazadas. La causa de ello se debe a que el optimizador tiene en cuenta la parte final del ciclo de marcha, donde el pie abandona el contacto con el suelo haciendo un último contacto con la punta. Normalmente la parte delantera del pie se hacia dentro, permitiendo un mayor contacto del pie con el suelo en ese momento, pero al no tener el modelo datos sobre el movimiento de la parte delantera del pie respecto al talón, se supone que el pie es un sólido enteramente rígido. Esta disposición de esferas es la forma que el optimizador ha encontrado para aproximar la fuerza de contacto de mejor manera sin tener que penetrar las esferas demasiado en la plataforma.

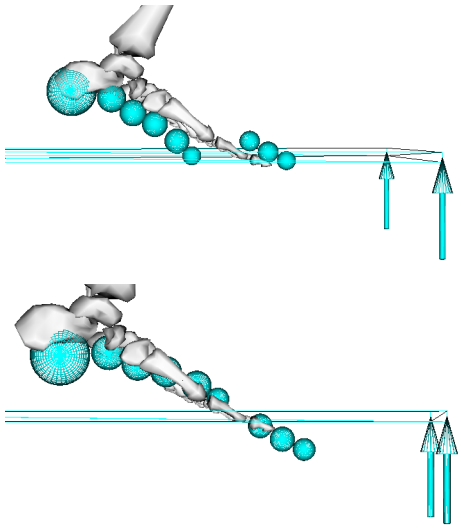

FIGURA 5.2: Comparación de la penetración de las esferas del dedo en el final del ciclo, en un mismo marco (0,69 s). Se puede observar cómo la desviación del primer caso, empleado para ciclo completo, evita una excesiva penetración.

Diferencia entre fuerzas del pie derecho y el pie izquierdo. Al comparar la componente normal de la fuerza de contacto de cada pie 5.3, se observa que la fuerza del pie izquierdo es mucho mayor que la fuerza del pie derecho, mientras que en los datos de referencia ésta es prácticamente igual. La causa de esto es principalmente debido a que en los datos de trayectoria el modelo hunde más el pie izquierdo en la superficie que el pie derecho y dado que una de las premisas a la hora de calcular las esferas ha sido no diferenciar un pie de otro a la hora de escoger los parámetros 3.2, el optimizador tuvo que aceptar el compromiso de tener fuerzas demasiado altas en un pie para que así el otro pudiera tener unas fuerzas de contacto de un orden similar comparable a las de referencia. El momento no tiene ninguna referencia en la gráfica y posiblemente esté invertido en el sentido. Esto es debido a que los datos de referencia iniciales tomados carecían de una información precisa en los momentos de contactos.

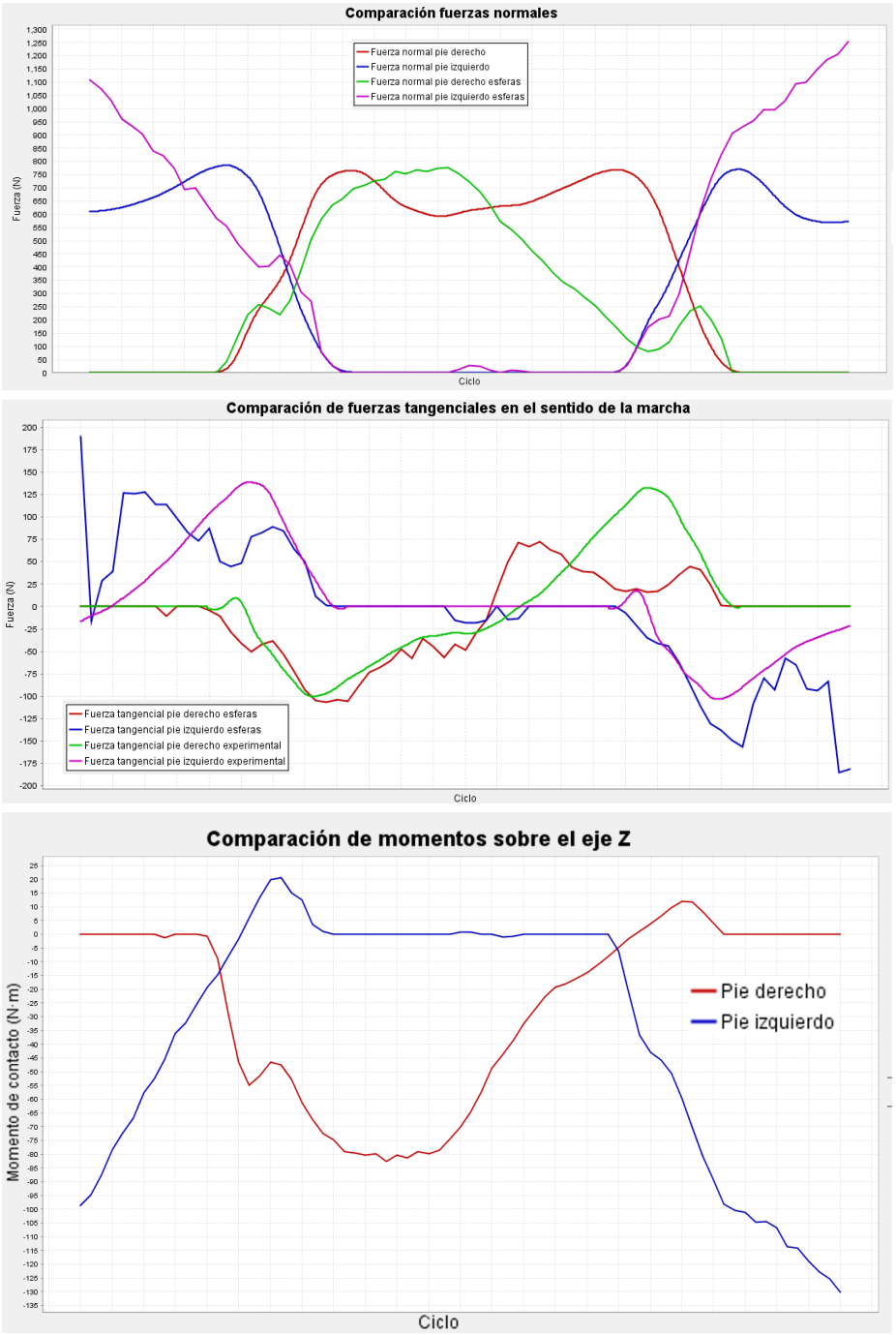

FIGURA 5.3: Comparación de fuerzas en estudio a ciclo completo en ambos pies.

5.1.2. Estudio de la fase de apoyo en el pie derecho

Para este estudio sólo se trató la parte correspondiente al pie derecho de la marcha, desde que acaba de iniciar el contacto hasta que comienza a abandonar la marcha, esto es, las tres primeras etapas de la fase de apoyo. Se modificó la función objetivo durante la optimización para intentar conseguir que el centro de presión de las fuerzas de contacto se acercara lo más posible al experimental.

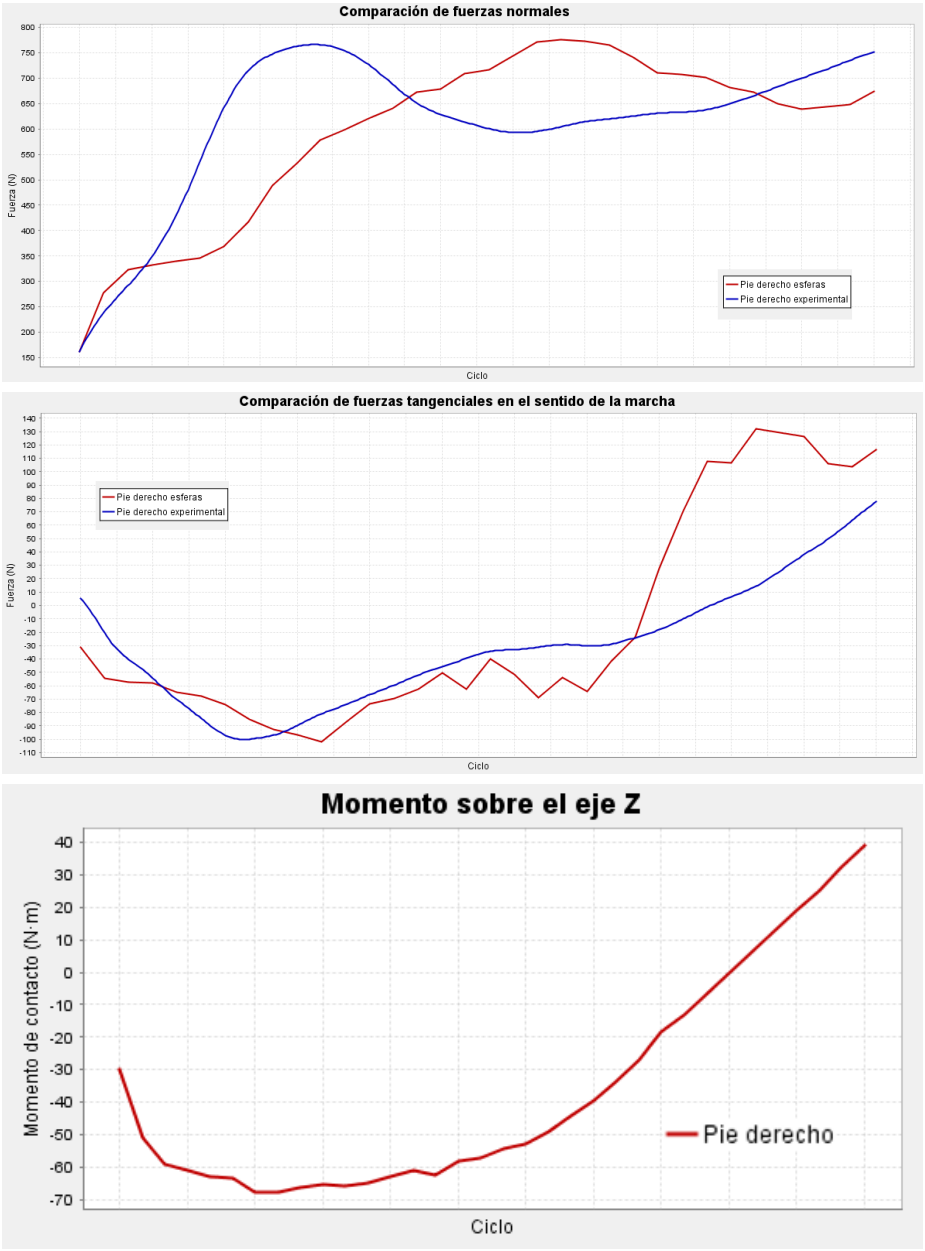

FIGURA 5.4: Comparación de fuerzas en estudio de fase de apoyo en el pie derecho.

Esferas de los dedos Dado que la etapa final de apoyo no está presente en esta parte del estudio, la colocación de las esferas de los dedos responde a una distribución más usual al no ser necesaria la flexión de las articulaciones metatarsofalángicas (la coordenada *mtp_angle* en el modelo).

5.2. Cálculos en OpenSim

Se realizaron varios intentos de optimización estática y CMC. Se empleó la pareja de programas descritos en el capítulo 4, junto con el archivo de esferas obtenido en el apartado anterior.

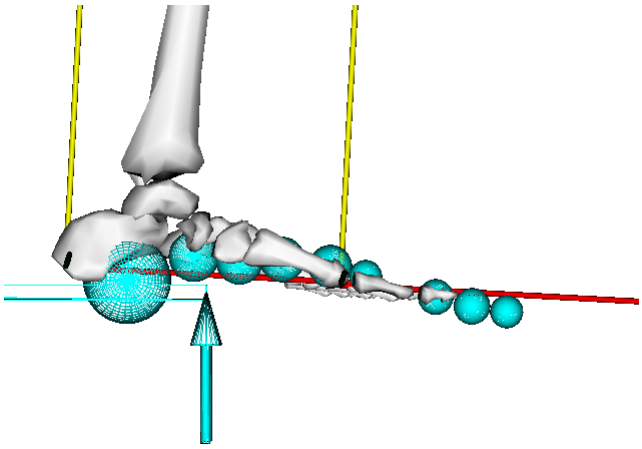

FIGURA 5.5: Posición de las esferas tras la optimización a sólo fase de apoyo.

5.2.1. Optimización estática

En primera instancia se probó a realizar los cálculos de optimización estática con los valores de los actuadores residuales en valor de 1. No funcionó, pues el ciertos momentos la optimización no convergía. Aquí se puede ver un ejemplo del error descrito:

```
SimTK Exception thrown at InteriorPointOptimizer.cpp:261:
  Optimizer failed: Ipopt: Restoration failed (status -2)
OPTIMIZATION FAILED...

StaticOptimization.record:  WARN- The optimizer could not find a solution at time = 1.36667

The model appears too weak for static optimization.
Try increasing the strength and/or range of the following force(s):
   sar_r approaching upper bound of 1
   tfl_r approaching upper bound of 1
   pect_r approaching upper bound of 1
   grac_r approaching upper bound of 1
   iliacus_r approaching upper bound of 1
   psoas_r approaching upper bound of 1
   quad_fem_r approaching upper bound of 1
   rect_fem_r approaching upper bound of 1
   tib_post_r approaching upper bound of 1
   glut_med1_l approaching upper bound of 1
   glut_med2_l approaching upper bound of 1
   glut_med3_l approaching upper bound of 1
   bifemlh_l approaching upper bound of 1
   bifemsh_l approaching upper bound of 1
   add_mag2_l approaching upper bound of 1
   grac_l approaching upper bound of 1
   glut_max1_l approaching upper bound of 1
   glut_max2_l approaching upper bound of 1
   glut_max3_l approaching upper bound of 1
   quad_fem_l approaching upper bound of 1
   gem_l approaching upper bound of 1
   peri_l approaching upper bound of 1
   med_gas_l approaching upper bound of 1
   soleus_l approaching upper bound of 1
   tib_post_l approaching upper bound of 1
```

Pese a que la solución descrita era aumentar la fuerza de los músculos, se consideró más conveniente no modificar el modelo y en vez de ello disminuir la penalización de los actuadores residuales. Así, se pasó a hacer otra simulación modificando los valores óptimos de los actuadores residuales, aplicando un factor de 5 a fuerzas residuales y de 3 a los actuadores residuales.

Parámetro	Unidad	Ciclo completo	Sólo fase de apoyo
Altura de la plataforma h	m	0,0344661	0,0339062
Posición horizontal del grupo talón x_1	m	0,10973	0,134766
Posición vertical del grupo talón y_1	m	0	0,015625
Posición horizontal del grupo dedos x_2	m	0,0722661	0,0908203
Posición vertical del grupo dedos y_2	m	0,0302734	0,0107422
Ángulo en el plano X-Y θ_1 del grupo talón	grados	-8,95249	4,02861
Ángulo en el plano X-Y θ_2 del grupo dedos	grados	21,3669	24,1717
Rigidez de las esferas	$N/m^{1,5}$	46473,4	46473,4
Disipación en el contacto de las esferas	s/m	19,7824	19,7824

CUADRO 5.1: Valores obtenidos por optimización a ciclo completo y a medio ciclo

Para el caso del estudio a ciclo completo, se estudió con unos tiempos de marcha de 0.6 a 1,4 segundos (un ciclo completo, comenzando la fase de apoyo medio del pie izquierdo) y se obtuvieron las fuerzas de contacto del apartado anterior. Aquí se puede observar una comparación de algunas fuerzas musculares y residuales. Obsérvese la gran diferencia en cualquier caso.

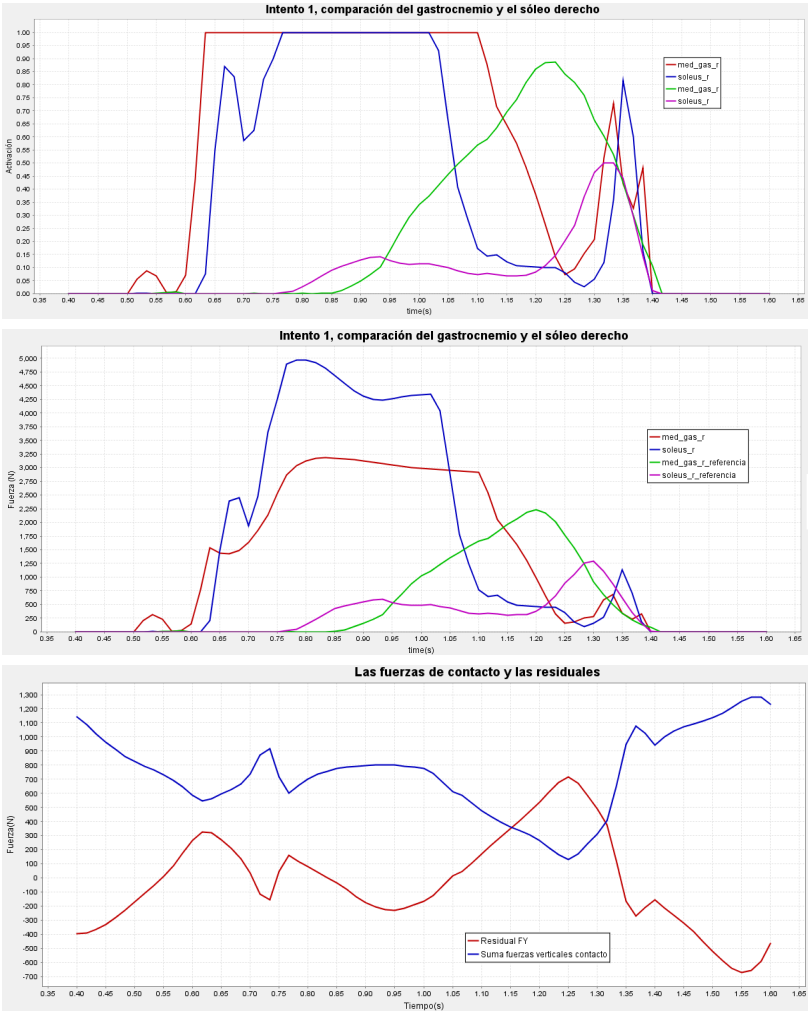

FIGURA 5.6: Comparación de fuerzas y activaciones

El tiempo de cálculo, en un ordenador personal equipado con un procesador AMD A4-3400 a 2,7 GHz fue de 6.111 segundos (1,7 horas) y se hicieron unas 21550 llamadas de cálculo de fuerzas.

5.2.2. CMC

Dada la complejidad del método del CMC, no pudieron hacerse simulaciones con un tiempo de integración que permitiera unos resultados que convergieran con la cinemática requerida; en el ordenador empleado para hacer los cálculos los tiempos de cálculo eran del orden de días, la simulación más larga llevada no pudo calcular más de 0,1 segundos de marcha en un periodo de 7,4 horas. Otra que se intentó para llegar a los 0,2 segundos de simulación tuvo que ser abortada tras más de 36 horas de cálculo. Esto es debido principalmente al problema de que el integrador empleado en la dinámica directa del CMC, al no encontrar un resultado convincente, reduce el paso. Intentos de establecer un paso mínimo han llevado al programa a arrojar error al no encontrar un valor satisfactorio. Se especula que esto puede ser debido a la naturaleza no continua del modelo de esferas, aunque los mismos desarrolladores de la aplicación avisan de que no es posible con el actual programa encontrar una forma adecuada de calcular las fuerzas para el actual algoritmo.

Capítulo 6

Conclusiones

Consideraciones finales

El método descrito en este proyecto de cálculo de fuerzas de contacto mediante esferas tiene bastante potencial de mejora, pero es necesario hacer numerosos cambios sobre él.

Uno de los problemas en el cálculo del lugar de las esferas es el hecho de que la articulación metatársica no presentaba ningún movimiento durante la marcha, obligando a obtener unas esferas metatársicas derivadas en la simulación de ciclo completo. Una forma de mejorar este aspecto es realizar la captura de movimientos de la marcha, sobre un sujeto en el que se hayan añadido los marcadores suficientes como para poder determinar así mediante la cinemática inversa la flexión delantera del pie. Con las nuevas coordenadas obtenidas, otro paso de mejora del proceso estaría en optimizar la posición y radio de las esferas, una por una en vez de agruparlas. Con el método actual no parece ser posible lo indicado por la gran cantidad de parámetros a optimizar, pero con un programa más optimizado y realizado en un lenguaje más rápido que Matlab, un método de optimización mejor que la búsqueda de patrón la literatura [24], indica que no es el método más rápido en converger en comparación a otros, puede ser posible.
Sin embargo, el mayor contratiempo de este proyecto se encuentra en la lentitud del cálculo de fuerzas a través de un programa externo hecho en Matlab. El método empleado ralentiza el procesamiento de datos de manera notable, haciendo inviable el cálculo del CMC mediante él. La mejor mejora que puede hacer sería prescindir de las llamadas a Matlab y emplear el mismo código nativo de OpenSim para calcular las fuerzas de Hunt-Crossley, pues ya trae uno. Aunque menos óptimo, pero sí también más rápido que la solución indicada, estaría la opción de emplear la IPA de OpenSim para Matlab, lo que sería una opción a explorar. Empleando estos métodos que agilizaran el proceso, considero que habría sido posible emplear el algoritmo de reducción de residuales, algoritmo que, debido a los problemas presentados de exactitud el cálculo de fuerzas y su lentitud, se antojó de una eficacia dudosa y por ello no se empleó.

Otra dirección en la que efectuar el cálculo es mediante el uso de mallas que simulen la forma del pie, como se ha descrito antes. Posiblemente este método presente también el problema de la flexión de la parte delantera del pie, además de ser, a primera impresión, más caro computacionalmente.

Por último, cabe destacar la relativa novedad de la colocación de las esferas en los metatarsianos en el proceso de ciclo completo, que pese a no resultar todo lo adecuado que debiera ser en cuanto a los resultados finales, sí parece ser una relativa novedad que podría dar pie a la estimación de la flexión del pie en función de la colocación de las esferas en cada momento de la marcha.

Bibliografía

[1] Miguel Aguilar Gutiérrez. *Biomecánica : la física y la fisiología*. Madrid : Instituto de Ciencia de Materiales, 2000, pág. 429. ISBN: 8400079884. URL: http://fama.us.es/record=b1480504{~}S5*spi.

[2] Frank C. Anderson y Marcus G. Pandy. «Dynamic Optimization of Human Walking». En: *Journal of Biomechanical engineering* (2001).

[3] Aristóteles, Elvira Jiménez Sánchez Escariche y Almudena Alonso Miguel. *Partes de los animales ; Marcha de los animales ; Movimiento de los animales*. Madrid : Gredos, 2000, pág. 342. ISBN: 8424922832. URL: http://fama.us.es/record=b1492257{~}S5*spi.

[4] Giovanni Alfonso Borelli (auth.) *On the Movement of Animals*. 1.ª ed. Springer-Verlag Berlin Heidelberg, 1989. ISBN: 978-3-642-73814-2,978-3-642-73812-8.

[5] Rob Barret y Glen Lichtwark. *Matlab OpenSim Pipeline Tools*. URL: https://simtk.org/projects/matlab_tools.

[6] Jason Campbell. *Winsock Networking Tutorials*. URL: http://www.win32developer.com/tutorial/winsock/winsock_tutorial_1.shtm.

[7] H Cenk Güler, Necip Berme y Sheldon R Simon. «A viscoelastic sphere model for the representation of plantar soft tissue during simulations». En: *Journal of Biomechanics* 31.9 (dic. de 2016), págs. 847-853. ISSN: 0021-9290.

[8] Scott L. Delp y col. «An interacive graphics-based model of the lower extremity to study orthopaedic surgical procedures». En: *Transactions on Biomedical engineering* (1990).

[9] *Documentación en línea de OpenSim*. URL: http://simtk-confluence.stanford.edu:8080/display/OpenSim/OpenSim+Support.

[10] Mariano Garcia. «The Simplest Walking Model: Stability, Complexity, and Scaling». En: *Journal of Biomechanical Engineering* 120 (2 abr. de 1998).

[11] L.A. Gilchrist y D.A. Winter. «A two-part, viscoelastic foot model for use in gait simulations». En: *Journal of Biomechanics* 29.6 (1996), págs. 795-798. ISSN: 00219290.

[12] H. Hatze. *The meaning of the term 'biomechanics'*. 1974.

[13] Robert Hooke y T. A. Jeeves. «" Direct Search" Solution of Numerical and Statistical Problems». En: *J. ACM* 8.2 (abr. de 1961), págs. 212-229. ISSN: 0004-5411.

[14] K. H. Hunt y F. R. E. Crossley. «Coefficient of Restitution Interpreted as Damping in Vibroimpact». En: *Journal of Applied Mechanics* (1975).

[15] K R Kaufman y col. «Physiological prediction of muscle forces-I. Theoretical formulation». En: 40.3 (1991), págs. 781-792. ISSN: 0306-4522.

[16] Emile H. L. Aarts (auth.) Peter J. M. van Laarhoven. *Simulated Annealing: Theory and Applications*. Springer Netherlands, 1987.

[17] E. J. Marey y Ém. Alglave. *Movement*. Londres : W. Heinemann, 1895, pág. 350. URL: https://archive.org/details/movement00mare_0.

[18] D Meglan y N Berme. *A 3D passive mechanical model of the human foot for use in locomotion synthesis*. English. 1993.

[19] M Millard, J McPhee y E Kubica. *Multi-step forward dynamic gait simulation*. Vol. 12. 2009, págs. 25-43. ISBN: 9781402088285.

[20] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. The MIT Press, 1996.

[21] Pedro Filipe Da Silva Moreira. «Developtment of a three-dimensional contact model for the foot-ground interaction in gait simulations». Tesis de mtría. Universidade do Minho, 2009.

[22] Mike Peasgood, Eric Kubica y John McPhee. «Stabilization of a Dynamic Walking Gait Simulation». En: *Journal of Computational and Nonlinear Dynamics* 2.1 (jul. de 2006), págs. 65-72. ISSN: 1555-1415.

[23] A Pedotti, V V Krishnan y L Stark. «Optimization of muscle-force sequencing in human locomotion». English. En: 38.1 OP - In Mathematical Biosciences 1978 38(1):57-76 (1978), pág. 57. ISSN: 00255564.

[24] Donald A. Pierre. *Optimization theory with applications*. New York: Dover Publications, 1986, pág. 612. ISBN: 0-486-65205-X.

[25] Peter Rydesäter. *TCP/UDP/IP Toolbox 2.0.6*. URL: http://es.mathworks.com/matlabcentral/fileexchange/345-tcp-udp-ip-toolbox-2-0-6.

[26] J. Javier Sánchez Lacuesta, Jaime Prat y Enrique Alcántara Alcover. *Biomecánica de la marcha humana normal y patológica*. Valencia : Instituto de Biomecánica de Valencia, 1999, pág. 444. ISBN: 84-923974-6-2. URL: http://fama.us.es/record=b1441782{~}S5*spi.

[27] Ahmed A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, 2005.

[28] Kevin B Shelburne y Marcus G Pandy. «A musculoskeletal model of the knee for evaluating ligament forces during isometric contractions». En: *Journal of Biomechanics* 30.2 (dic. de 2016), págs. 163-176. ISSN: 0021-9290.

[29] Darryl G. Thelen. «Adjustment of Muscle Mechanics Model Parameters to Simulate Dynamic Contractions in Older Adults». En: *ASME Journal of Biomechanical Engineering* (2003).

[30] Darryl G. Thelen y Frank C. Anderson. «Using computed muscle control to generate forward dynamics simulations of human walking from experimental data». En: *Journal of Biomechanics* (2005).

[31] Marek Wojtyra. «Multibody Simulation Model of Human Walking». En: *Mechanics Based Design of Structures and Machines* 31.3 (ene. de 2003), págs. 357-379. ISSN: 1539-7734.

[32] Gary T. Yamaguchi y Felix E. Zajac. «A planar model of the knee joint to characterize the knee extensor mechanism». En: *Journal of Biomechanics* (1989).

[33] F. E. Zajac. «Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control». En: *Critical Reviews in Biomedical Engineering* (1989).