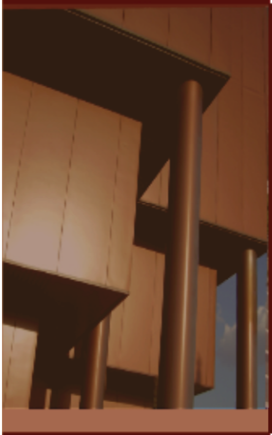
Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

Biomechanical analysis of the reaction forces and moments between feet and ground during gait without force plates data.

Autor: Juan Morales Galera

Tutora: Juana María Mayo Núñez



Departamento de Ingeniería Mecánica y Fabricación

Escuela Técnica Superior de Ingeniería

Sevilla, 2017

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

Biomechanical analysis of the reaction forces and moments between feet and ground during gait without force plates data.

Autor:

Juan Morales Galera

Tutora:

Juana María Mayo Núñez

Catedrática de Universidad

Departamento de Ingeniería Mecánica y Fabricación

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2017

Trabajo de Fin de Grado: Biomechanical analysis of the reaction forces and moments between feet and ground during gait without force plates data.

Autor:	Juan Morales Galera
Tutora:	Juana María Mayo Núñez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, Junio de 2017

El Secretario del Tribunal

Agradecimientos

A mi familia.

A mi tutora Juana por su dedicación, ayuda y paciencia.

Resumen

Este trabajo tiene por objetivo demostrar la viabilidad de la utilización de la dinámica inversa en el cálculo de fuerzas de reacción durante la marcha de un sujeto, sin el uso de la información que nos pueden facilitar las placas de fuerza.

Este documento incluye una descripción detallada de los pasos que se han seguido para alcanzar dichos objetivos. Se realiza un trabajo experimental, seguido de un postprocesado de la información obtenida.

Finalmente los resultados son analizados y se discute el hecho de que pueda o no ser válido el método descrito.

Abstract

The aim of this work is to prove the viability of inverse dynamics in terms of analysing the reaction forces, which are implicated during the human gait, without the data which can be provided by force plates.

This document includes the description of the steps followed to achieve the different goals. An experimental work and the post process of this information are the main phases of the project.

Finally, the results will be discussed and analysed in order to overcome different issues in the near future.

Contents

List of figures

List of tables

1 Introduction

Engineering is the application of science and mathematics by which the properties and the matter sources of energy in nature are made useful to people.

Engineering can also be defined as the application of mathematics, science, economics, social and practical knowledge in order to design, maintain or improve structures, machines or materials.

It is considered that engineering has existed since the ancient times where humans devised fundamental inventions as the lever, the wheel and so on. The modern engineering made its appearance with the development of specialized machinery during the industrial revolution. Since that moment, the progress has kept such an unstoppable rhythm.

That fact situates this society in the middle of an age in which all the fields of science have been mixed in order to achieve goals which were unimaginable fifty years ago.

As an example, medicine and engineering work together many times in terms of solving complex issues. This is how bioengineering was born in the middle of the XX century. This engineering field seeks to integrate quantitative and design approaches to biological systems, encompassing a wide range of specific disciplines.

Biomechanics is known as a subarea of bioengineering, which was mentioned in the previous paragraph. It is defined as the study of the structure and function of biological systems of the living beings' bodies and the science which allows to overcome the issues presented in this document.

1.1 State of the art

In the 19th century some scientists began to study the locomotion opening the field of modern analysis of ground reaction forces with movement. This fact was followed by a depth investigation of human gait. Christian Wilhelm Braune [1] was the one who significantly advanced the science using recent advances in engineering mechanics. Little by little, this field has strengthened its bases until it has become one of the most recognized engineering fields of these ages.

This engineering discipline has a wide range of applications. Its study provides information about relevant forces and moments which are the cause of motion. This fact allows to investigate solutions for determinate pathological problems, for example. The domination of this field could let the bases of high accurate prosthesis implementation and so on.

1.2 Inverse dynamics

Inverse dynamics commonly refers to a method used for computing forces and moments based on the kinematics of a body and its inertial properties, mass and moment of inertia.

In order to calculate forces and torques, inverse dynamics is typically used in the fields of robotics and biomechanics where the information which is provided allows to predict the behaviour of the body that it is being studied.

1.3 Segment body models

In terms of easing, the scientists and engineers tend to divide the human body by segments. This method makes people be capable to work in an easy way. Along the history of biomechanics, there were designed an amount of different human-segment models. Although they are closely similar, each of them has been produced for a particular kind of investigation or field of science.

In this document there will be described a few of these models:

SC-14, [2] which divides the human body in 14 segments. SC-15, [3] which divides the human body in 15 segments. SC-16, [4] which divides the body in 16 segments. SC-18, [5] which divides the human body in 18 segments.

The following pictures show the SC-14 and SC-15 segment models of the human body by Dempter and Hanavan respectively.

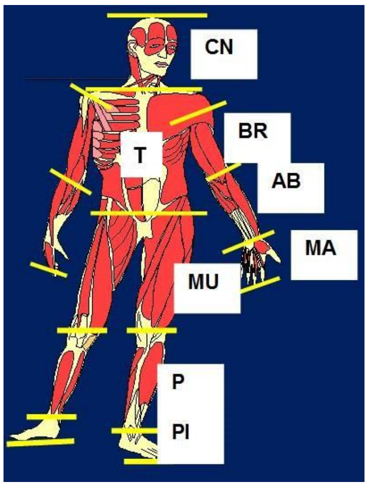


Figure 1: SC-14 segment model.

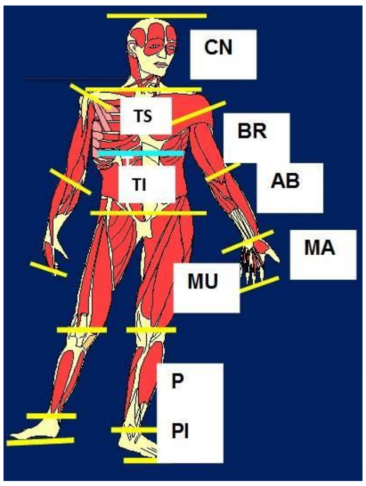


Figure 2: SC-15 segment model.

In this project, a fifteen segments model will be used. The following table describes each of the segments and joints:

	SEGMENT	JOINT		DESCRIPTION
1	RFOO		RAJC	Right foot
2	RSHA	RAJC	RKJC	Right shank
3	RTHG	RKJC	RHJC	Right thigh
4	PEL	SAJC RHJC LHJC		Pelvis

5	LTHG	LKJC	LHJC	Left thigh
6	LSHA	LAJC	LKJC	Left shank
7	LFOO		LAJC	Left foot
8	RHAN		RWJC	Right hand
9	RFAR	RWJC	REJC	Right forearm
10	RUPA	REJC	RSJC	Right upper arm
11	LUPA	LEJC	LSJC	Left upper arm
12	LFAR	LWJC	LEJC	Left forearm
13	LHAN		LWJC	Left hand
14	HEA		NJC	Head
15	TRU	RSJC LSJC NJC SAJC		Trunk

Table 1: Segments and joints.

It must be observed, that segments could be parent or child ones depending on where the work is focusing on. Each child segments are defined as the ones which are relatively moving to the parent ones.

The following picture illustrates how a child and a parent body are related [6]:

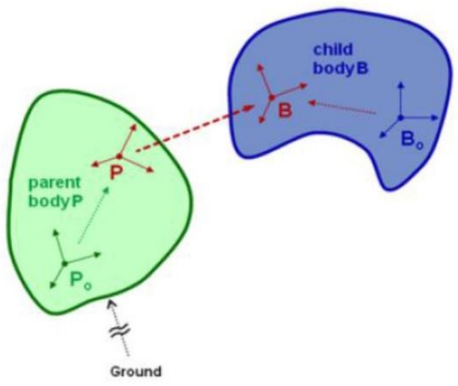


Figure 3: Parent and child bodies.

To overcome the inverse dynamic problem in this study, segments and joints have such an important role. In fact, as it will be seen in the following chapters, they represent one of the most significant difficulties of the process.

1.4 Human gait

Human gait is defined as the locomotion achieved though the movement of human limbs. Human gaits are the various ways in which humans can move.

Human gait can also be defined as the transitory progression of the human body as a whole thanks to the coordinate movements of the body segments.

There are two phases during the gait cycle: Stance phase and Swing phase. During the first one, the foot is on the ground, on the other hand, in the swing phase, this foot is not in contact with the ground and the leg is swinging through in preparation for the next foot strike [7].

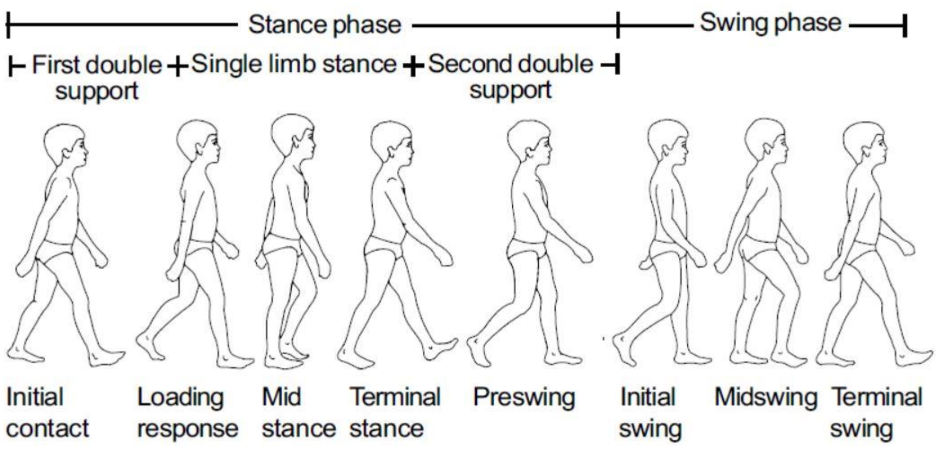


Figure 4: Human gait phases.

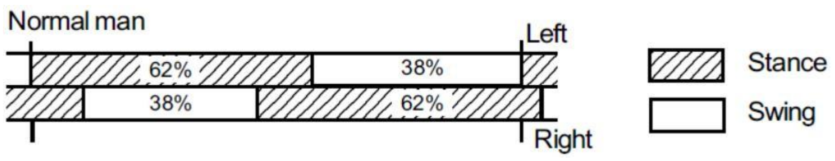


Figure 5: Time spend on each limb during a cycle of a normal patient.

With these two images it can be easily understood the phases of the human gait. It must be known the fact that the gait of a person with some pathological issue is different in terms of time spent by each limb.

1.4.1 Double support

In figure number five it can be seen that there are moments in which both, right and left foot, are in contact with the ground. This is called the double support phase. If it is wanted to calculate the reactions between the feet and the ground during the double support, there is an inconsistent problem due to the fact that there are three forces and three moments per foot and only six equations to solve it. Therefore, apparently, the problem has no solution.

Some studies have already address this issue. In the following, three methods will be described [8].

1.4.1.1 Using reactions measured by force plates

This is the most common way to calculate the ground reactions. Using force plates is the only way the equilibrium problem can be solved without considering the upper body.

This method yields information which can be used as inputs for the solution of the double support problem. It provides three components of both the external reaction forces and moments, all of them taken at the centre of each plate.

In this project force plates have been used in order to compare the inverse dynamic results and the ones provided by them. As it will be described in the following sections, they are also used as a tool to know when the double support appears.

1.4.1.2 Optimized contact model

Transition curves to determinate the reaction during the double support can be used in normal gait applications. However, in pathological gait, where the length of the double support and the swing can be different, it is recommended to work with optimized foot-ground contact.

To achieve this goal, the foot surface is approximated by several spheres. The objective is to adjust the position of these spheres plus their size in order to get the best accuracy.

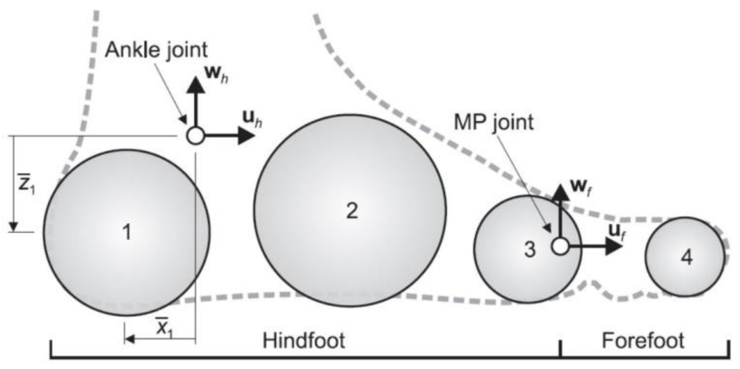


Figure 6: Contact force model by spheres.

1.4.1.3 The Smooth Transition Assumption (STA)

The Smooth Transition Assumption algorithm is based on the assumption that the reaction forces and moments at the trailing foot decay according to a certain law along the double support phase. For that reason, different functions have been used in order to approximate the shape of the reactions at the trailing foot. These exponential and linear functions have been obtained by trial and error.

There are two different functions: one for the antero-posterior reaction force (f_x), while the other one is used for the other five components (f). The following figure illustrates the shape of both functions:

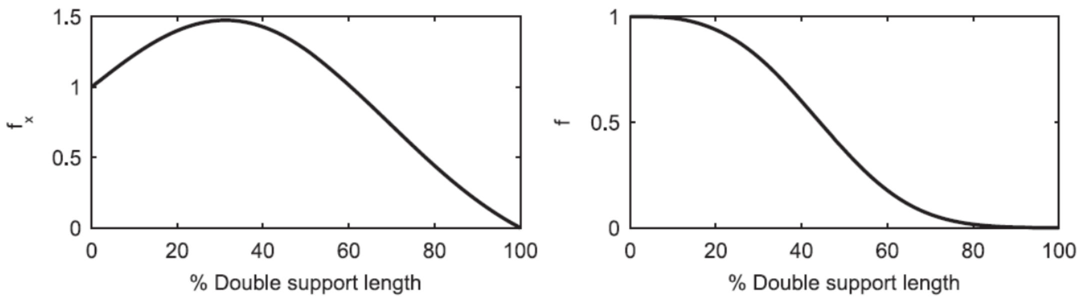


Figure 7: Shape of the functions f_x and f by the STA algorithm.

This method depends on the value of the reactions of the trailing foot just before the double support starts. This forces are $G_{1x}(t_h)$ and the reactions during the double support are calculated as:

$$G_{1x}(t) \ = f_x(t)\, G_{1x}(t_h)\ ; \qquad t_h \le t \le \ t_t \qquad (1)$$

For the antero-posterior component.

$$G_{1x}(t) \ = f(t) G_{1x}(t_h); \qquad t_h \le t \le \ t_t \qquad (2)$$

For the rest of the components. Where t_t is the moment when the double support finishes.

2 Objectives and motivation.

Although with the study of the human gait a several amount of issues has been addressed, there are some limitations which make the science not be able to achieve many purposes or not achieving them as it is desired.

Many of this problems appear with the capture of the kinematics. For this kind of capture, it is normally used an equipment composed by infrared cameras and reflective markers. The human body is divided in segments, as it has already been said in the previous section, and these segments are considered as rigid bodies but, actually they are not. The skin, the contraction of the muscles, among other factors, introduce noise and errors. In order to minimize them, some techniques are applied, for example, constraints between markers or constraints at joints which introduce redundant data called residuals.

With force plates it is possible to know the reaction between the feet and the ground. The inconsistent problem of the double support is easily overcome with them. Moreover, with the information provided by the plates plus the kinematics of the body, the forces which the muscles make in order to yield the movement can be calculated. However, this powerful tool has some limitations. One problem is the work area and another problem is that it is not possible to study a multisegment foot with them.

There are some alternatives to substitute the force plates as the contact methods, which use direct dynamics, or the methods which works using the inverse dynamics.

The main purpose of this study is to check the accuracy of the inverse dynamics calculations compared with the reaction forces and moments provided by two force-plates, between the feet and the ground during gait.

To overcome the main purpose of the study it is necessary to achieve another goals: It is necessary to do an experimental work in terms of getting the kinematics; The order of the segments followed to solve the inverse dynamic problem stars such a big deal, which must be solved by the computing of the computing code; The other goal is to solve the double support problem without the reactions provided by the plates.

3 Methodology

This project has two phases: the measurement of the kinematics and then the computing to process the data. The first phase has been done at the ETSI biomechanical laboratory in the University of Seville. For the second phase, the computing programme MATLAB has been used.

3.1 Data measurement

Vicon Nexus has been the computing programme used to catch the movement of the body. The equipment is constituted by twelve infrared cameras, thirty-nine reflective spheres used as markers, the Vicon Nexus software, a walk path, two force-plates and, obviously, a volunteer to be measured was required.

Once all the material is under control the measurement work can start:

-Firstly, the work environment has to be prepared. The cameras must be in a particular position [9].

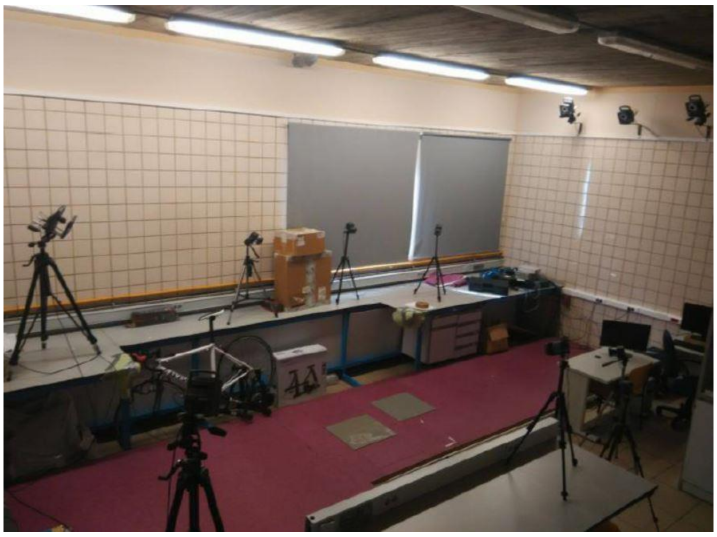


Figure 8: Work area for the capture of the movement, ETSI, University of Seville.

Figure 9: Vicon Nexus camera.

When the work area is prepared, the system must be configured.

-System configuration:

-Cameras have to be calibrated before the measurements. Avoiding the sun light, the quality of the measurements will be better. Each camera works individually and then the software works in terms of combining all the information to recreate the movements. Looking at pictures 14, 15 and 16 it is possible to see the process.

-The volume origin has to be defined. In this case it is set at one of the two force plates. Figures 10 and 19 illustrates where the origin is set.

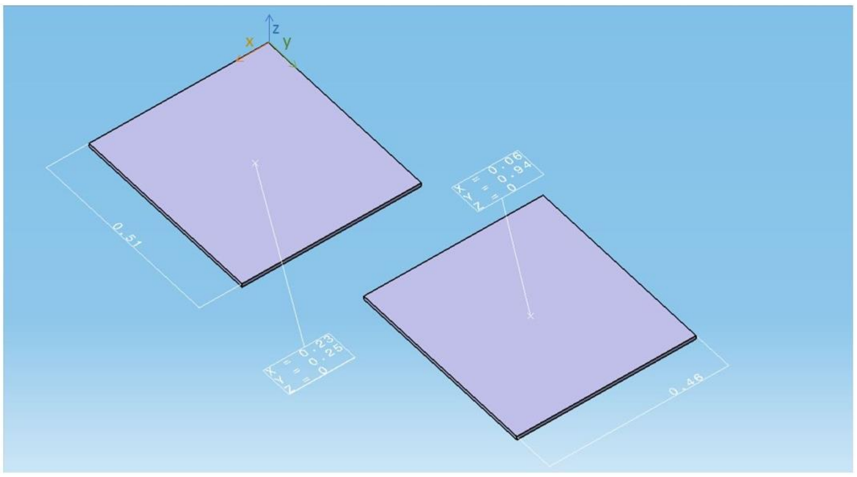


Figure 10: Force plates disposition.

Where the plate in the left is the number one and the other plate the number two.

-The subject configuration is an important step on the grounds that the more accurate the volunteer´s measurements are, the more accuracy will be gotten during the tests.

-It is necessary to know some of the patient´s measurements. The next table describes a few of them:

	Description
Body mass	Patient mass
Height	Patient height
Ankle Width	The medio-lateral distance across the malleoli
Knee Width	The medio-lateral width across the line of the knee axis
Leg length	Distance between the ASIS marker and the medial malleolus
Elbow Width	Width of elbow along the flexion axis
Hand Thickness	Thickness between dorsum and palmar surfaces of the hand
Wrist Width	Thickness of wrist

Table 2: Patient´s measurements.

-Set model: Depending on the sort of work, there will be chosen a concrete model. Vicon Nexus provides a list of models but, it is also possible to create a particular model for a different type of projects. In this case, it will be used a full body model.

-Markers colocation: Markers are the elements which give the position of the different segments or joints. The good colocation of these markers is vital to achieve the best accuracy. The following images illustrate where each marker must be placed.

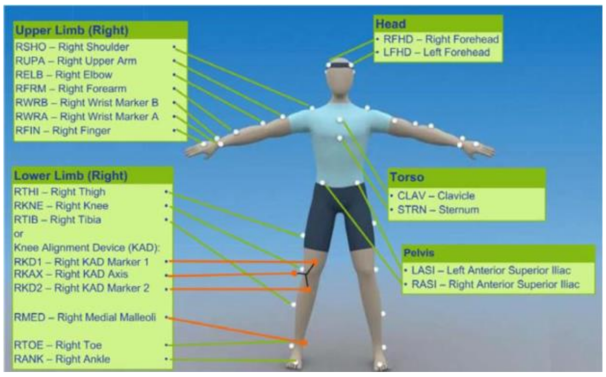


Figure 11: Position of markers.

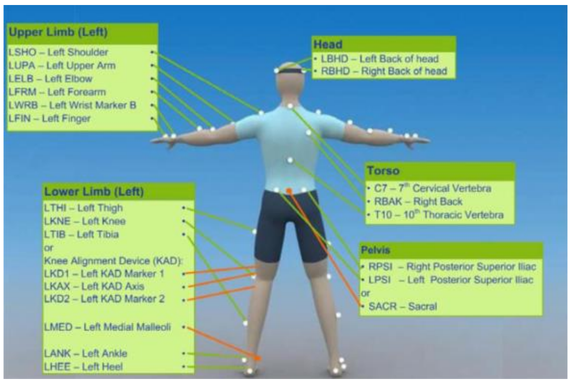


Figure 12: Position of markers.

Figure 13: Marker.

-Once all these steps have been done, the capture can start. The following pictures illustrates this process. It can also be observed the position of the cameras, force plates and marker on the subject.

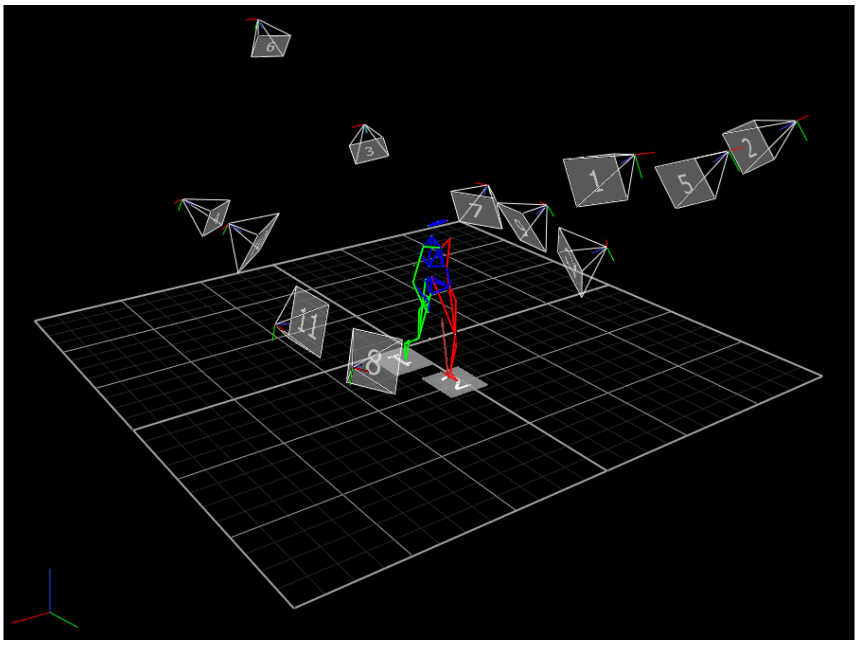


Figure 14: Capture of the human gait with Vicon Nexus.

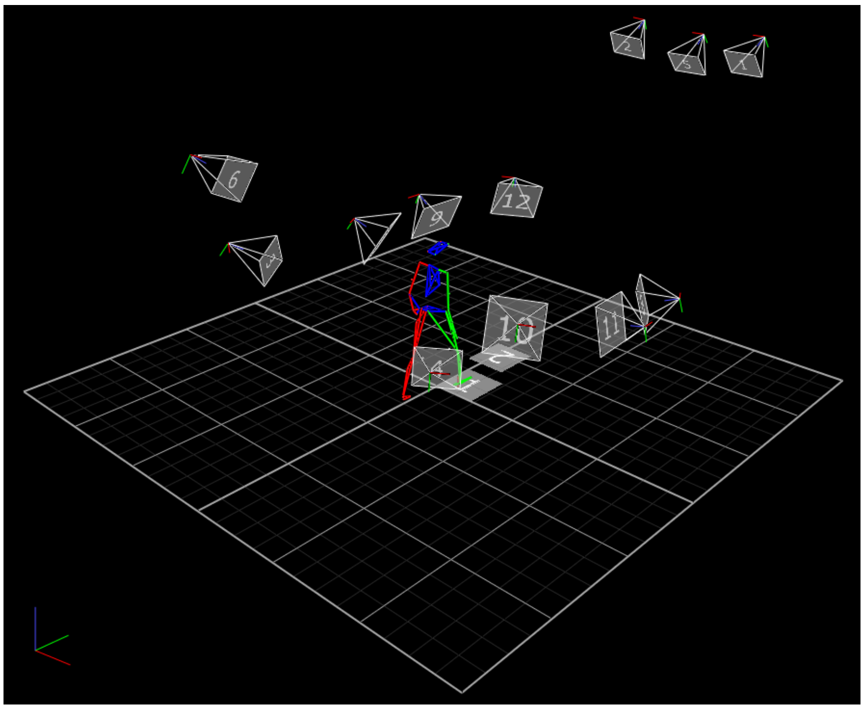


Figure 15: Capture of the human gait with Vicon Nexus.

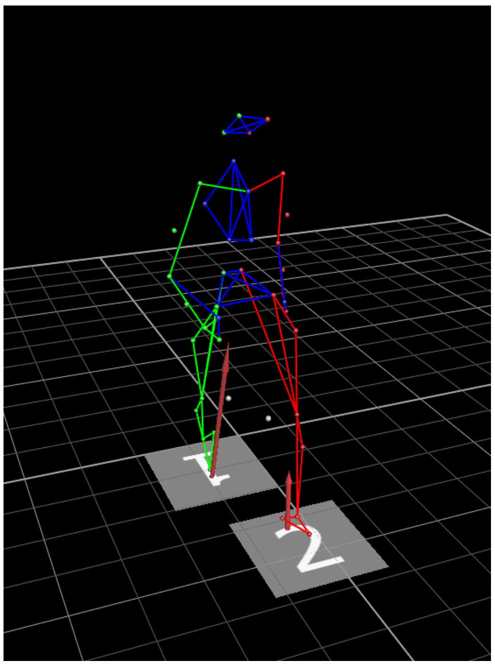


Figure 16: Markers, subject and force plates.

-Data preparation: After the capture, the information has to be processed in order to get the files which will be taken to make the calculations with MATLAB.

3.2 Computing procedure.

MATLAB is the software used in order to process the information provided by Vicon Nexus and the measurements taken at the laboratory.

All the data needed for the calculations of the reaction forces between the body and the ground is now known. Thus, with MATLAB a code will be developed to implement the dynamic equations.

In the human gate, and according to the aim of this project, two phases are distinguished: simple foot support, where only one foot is in contact with the ground, and the double support, where both feet are in contact with the ground.

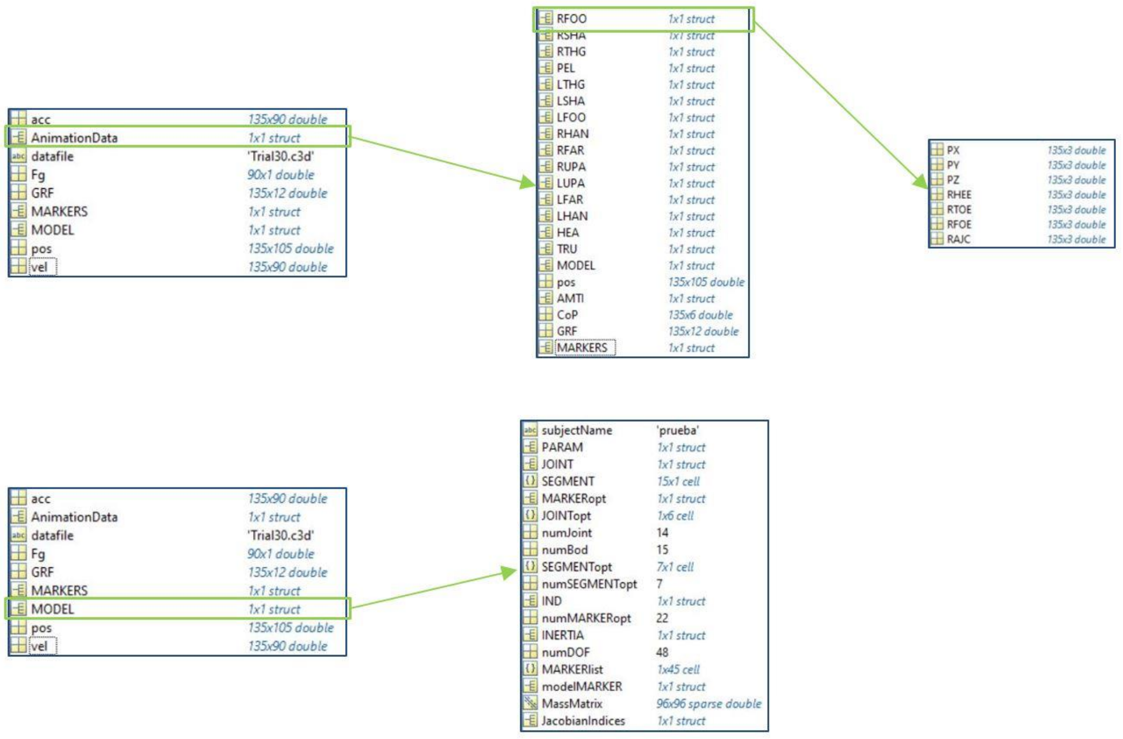
The simple support phase is not as complex as the double support can be. The difference between both are evident: while during the single support only three forces and moments would be unknown, during the double support there would be six forces and six moments. In fact, the double support cannot be faced with a simple equilibrium problem like at the simple support. An amount of scientists and engineers have had this problem. Fortunately, there are few studies which proposes good alternatives to solve the inconsistent system.

3.2.1 General aspects about MATLAB.

Firstly, a programme provided by the ETSI mechanic department, which works with the data given by Vicon Nexus, yields the necessary variables with which the work will be overcome.

	Description
acc	acceleration of each segment
AnimationData	this is a global variable
Fg	weight of each segment
GRF	plate forces and moments
MARKERS	this is a global variable
MODEL	this is a global variable
pos	position of each marker
vel	velocity of each segment

Table 3: Variables which are known.



acc	135x90 double
AnimationData	1x1 struct
datafile	'Trial30.c3d'
Fg	90x1 double
GRF	135x12 double
MARKERS	1x1 struct
MODEL	1x1 struct
pos	135x105 double
vel	135x90 double

RFOO	1x1 struct
RSHA	1x1 struct
RTHG	1x1 struct
PEL	1x1 struct
LTHG	1x1 struct
LSHA	1x1 struct
LFOO	1x1 struct
RHAN	1x1 struct
RFAR	1x1 struct
RUPA	1x1 struct
LUPA	1x1 struct
LFAR	1x1 struct
LHAN	1x1 struct
HEA	1x1 struct
TRU	1x1 struct
MODEL	1x1 struct
pos	135x105 double
AMTI	1x1 struct
CoP	135x6 double
GRF	135x12 double
MARKERS	1x1 struct

PX	135x3 double
PY	135x3 double
PZ	135x3 double
RHEE	135x3 double
RTOE	135x3 double
RFOE	135x3 double
RAJC	135x3 double

acc	135x90 double
AnimationData	1x1 struct
datafile	'Trial30.c3d'
Fg	90x1 double
GRF	135x12 double
MARKERS	1x1 struct
MODEL	1x1 struct
pos	135x105 double
vel	135x90 double

subjectName	'prueba'
PARAM	1x1 struct
JOINT	1x1 struct
SEGMENT	15x1 cell
MARKERopt	1x1 struct
JOINTopt	1x6 cell
numJoint	14
numBod	15
SEGMENTopt	7x1 cell
numSEGMENTopt	7
IND	1x1 struct
numMARKERopt	22
INERTIA	1x1 struct
numDOF	48
MARKERlist	1x45 cell
modelMARKER	1x1 struct
MassMatrix	96x96 sparse double
JacobianIndices	1x1 struct

Figure 17: Global variables.

Figure 5 illustrates how global variables work. Global variables provide an amount of information, for example, the name of the joint in which the work is focusing or joints´ child and parent segments. The well managing of these variables would allow a faster and easier way to work.

3.2.2 Dynamics of the body.

3.2.2.1 Coordinate System.

Some considerations must be taken in order to achieve the aim of this study.

Anatomical planes are hypothetical planes which are used in order to describe the location of structures or the direction of the movement. These planes ease the understanding of the coordinate systems and so on. There are three of them:

1. The sagittal plane is the one parallel to the sagittal suture. It divides the human body into left and right.
2. The frontal plane is a vertical one which divides the body into front and back sections.

3. The transverse plane is a horizontal one which divides the body into upper and bottom sections.

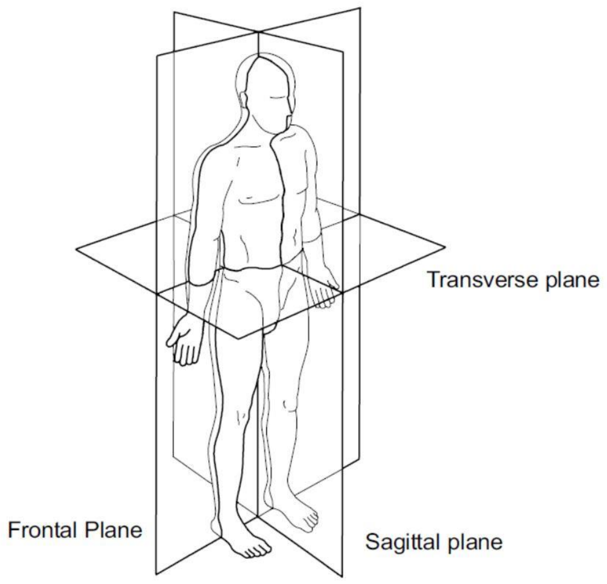


Figure 18: Anatomical planes.

Then, the coordinate systems will be described [10]. The following images illustrate the global and local system coordinate references.

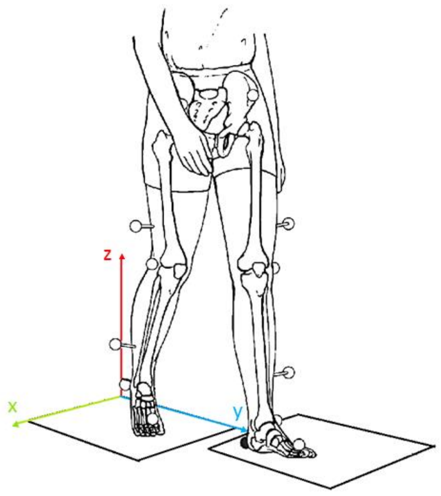


Figure 19: Global coordinate system.

Figure 8 shows the Global Coordinate System where the *x* axis is parallel to the floor and pointing towards the right section of the body; the *y* axis is perpendicular to the *x* axis, parallel to the floor and pointing towards the gait direction. Finally, the *z* axis is perpendicular to the floor plane and pointing towards the upwards.

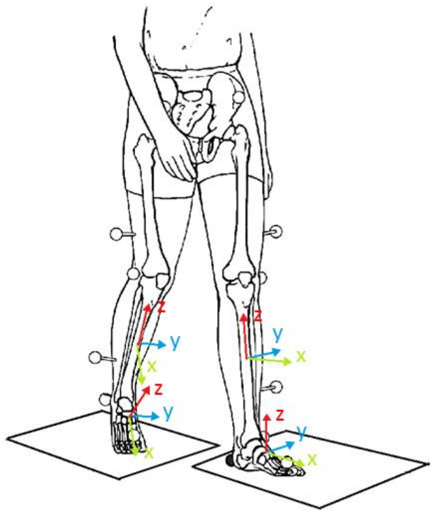


Figure 20: Local coordinate system.

In order to define the local coordinate system, in the static trial, for each segment, the *x* axis is perpendicular to the frontal plane and positive towards the direction of gait. On the other hand, the *y* axis is perpendicular to the sagittal plane, being positive towards the left section of the body. Finally, the *z* axis is perpendicular to the transverse plane being positive towards the upper body.

3.2.2.2 Simple support.

In this project, it is essential to follow a determinate order for the calculations. As it can be imagined, the simple support foot must be the last segment in which force and moment equilibrium will be calculated. It will be also seen the method to face the double support.

As it is known, there are five limbs: head, two arms and two legs. A free limb is the one which is only affected by its weight and inertial forces. At the simple support one leg is not free while all the others limbs are. On the grounds of this fact, arms will be the first segments where the equilibrium will be calculated, then head and the free leg, following this order and ending at the support foot. The following figure illustrates the procedure:

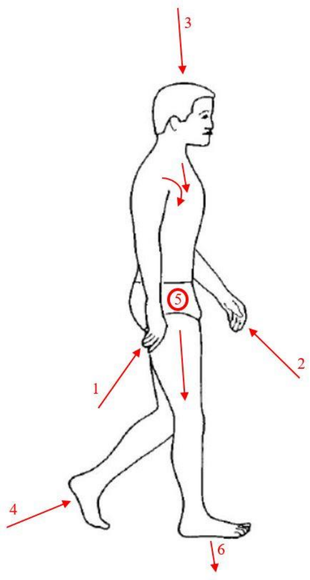


Figure 21: Order followed.

The image above is an example where it is supposed that left foot is the one which is free. When the right foot is the one which is free, the procedure is similar but changing both feet. As it will be seen in this document, for the double support the procedure does not change neither.

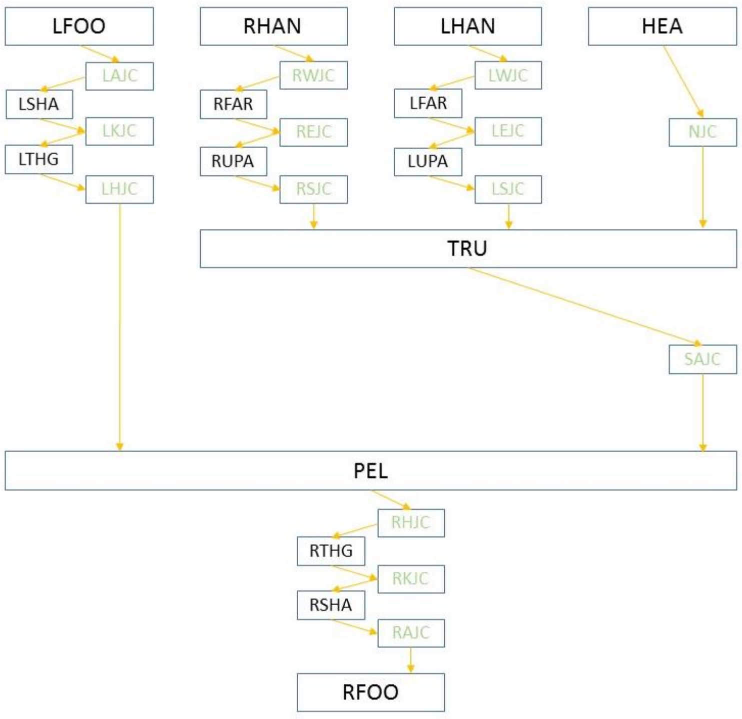


Figure 22: Procedure diagram with left foot free.

The figure 7 illustrates all the segments, joints and the order followed to solve the inverse dynamics when the left foot is free.

Then, the dynamics of ith body segment are determined by the following functions [11]:

NOTATION	
m_i	mass of segment
$\ddot{\vec{r}}_{Ci}$	translational acceleration vector of the ith segment´s mass centre
$\vec{F}_{jk}^{(i)}$	kth joint force vector acting on the ith segment
$\vec{F}_{ek}^{(i)}$	kth external force vector acting on the ith segment
$\vec{g}$	gravitational vector
J_{Ci}	inertia tensor around the mass centre of the ith segment
$\vec{\alpha}_i$	angular acceleration vector of the ith segment
$\vec{\omega}_i$	angular velocity vector of the ith segment
$\vec{M}_{jk}^{(i)}$	kth net muscle moment acting on the ith segment
$\vec{M}_{ek}^{(i)}$	kth external moment acting on the ith segment
$\vec{r}_{jk}^{(i)}$	position vector of the kth joint force from the mass centre of the ith segment
$\vec{r}_{ek}^{(i)}$	position vector of the kth external force from the mass centre of the ith segment
$\vec{F}_{gr}$; $\vec{M}_{gr}$	ground forces and moments of the right foot
$\vec{F}_{gl}$; $\vec{M}_{gl}$	ground forces and moments of the left foot

Table 4: Notation of dynamic equations.

$$m_i \ddot{\vec{r}}_{Ci} = \sum\nolimits_{k=1}^{n_{ji}} \vec{F}_{jk}^{(i)} + \sum\nolimits_{k=1}^{n_{ei}} \vec{F}_{ek}^{(i)} + m_i \vec{g} \text{ ; (3)}$$

$$J_{Ci}\vec{\alpha}_i + \vec{\omega}_i \times (J_{Ci}\vec{\omega}_i) = \sum_{k=1}^{n_{ei}} \vec{M}_{ek}^{(i)} + \sum_{k=1}^{n_{ei}} \left(\vec{r}_{ek}^{(i)} \times \vec{F}_{ek}^{(i)}\right) + \sum_{k=1}^{n_{ji}} \vec{M}_{jk}^{(i)} + \sum_{k=1}^{n_{ji}} \left(\vec{r}_{jk}^{(i)} \times \vec{F}_{jk}^{(i)}\right) \text{ ; (4)}$$

Equation 3 is yielded by combining the translational equations of motion, (eq. (1)), of all **n** body segments. Hence, the sum of external forces is derived:

$$\sum_{i=1}^{n} \sum_{k=1}^{n_{ei}} \vec{F}_{ek}^{(i)} = \sum_{i=1}^{n} \left[m_i \left(\ddot{\vec{r}}_{Ci}^{(i)} - \vec{g} \right) \right] \text{ ; (5)}$$

Equation 4 is yielded by combining the rotational equations of motion, (eq. (2)), of all **n** body segments. Hence, the sum of external moments is derived:

$$\sum_{i=1}^{n}\sum_{k=1}^{n_{ei}}\vec{M}_{ek}^{(i)} = \sum_{i=1}^{n}[J_{Ci}\vec{\alpha}_i + \vec{\omega}_i \times (J_{Ci}\vec{\omega}_i)] - \sum_{i=1}^{n}\sum_{k=1}^{n_{ei}}\left(\vec{r}_{ek}^{(i)} \times \vec{F}_{ek}^{(i)}\right) - \sum_{i=1}^{n}\sum_{k=1}^{n_{ji}}\left(\vec{r}_{jk}^{(i)} \times \vec{F}_{jk}^{(i)}\right) \text{ ; (6)}$$

As the only external forces and moments during the human gait are the ground reactions, equations 3 and 4 can be rewritten as:

$$\vec{F}_{gr} + \vec{F}_{gl} = \sum_{i=1}^{n}[m_i\left(\ddot{\vec{r}}_{Ci}^{(i)} - \vec{g}\right)] \text{ ; (7)}$$

$$\vec{M}_{gr} + \vec{M}_{gl} = \sum_{i=1}^{n}[J_{Ci}\vec{\alpha}_i + \vec{\omega}_i \times (J_{Ci}\vec{\omega}_i)] - \sum_{i=1}^{n}\sum_{k=1}^{n_{ei}}\left(\vec{r}_{ek}^{(i)} \times \vec{F}_{ek}^{(i)}\right) - \sum_{i=1}^{n}\sum_{k=1}^{n_{ji}}\left(\vec{r}_{jk}^{(i)} \times \vec{F}_{jk}^{(i)}\right) \text{ ; (8)}$$

Now, the simple support problem can be solved.

While global reference system will be used for translational equations, for rotational equations local reference system will.

It is also important to know that moments will be taken in the mass centre of feet on the grounds of easing future projects related with this one. Due to that fact, moments provided by force plates will be modified to be compared with the inverse dynamic results.

3.2.2.3 Double support

As it has been already said, the double support supposes a new challenge to deal with. During the double support there is an inconsistent problem due to underspecification and therefore, apparently, the problem has no solution.

Some studies have already address this issue. At this project the method choose was the STA, which was briefly described in a previous section.

The Smooth Transition Assumption (STA)

The STA uses a transition criterion to determinate reaction forces and moments during the double support [12].

The STA work with an algorithm based on the following observations:

1. During the double support phase, ground forces and moments on the trailing foot change smoothly towards zero.
2. The ratios of the ground reactions to their values at contralateral heel strike can be expressed as functions of double support duration.

The method uses a combination of exponential and linear functions in order to approximate the shape of the reactions at the trailing foot. The functions proposed were determined by trial and error with experimental data.

Looking at figure number seven it can be observed the shape of both kind of functions:

$$\frac{F_x}{F_{x0}} = e^{-(t/T_{ds})}; (9)$$

$$\frac{F_x}{F_{x0}} = \left(k_1 e^{-[(t-t_P)/T_{ds}]} - k_2 \frac{t}{T_{ds}}\right); (10)$$

Where the first one is used for all ground reaction components except anterior force. The second function is the one used to determinate the anterior ground reaction which is not monotonic.

F_{x0} is the anterior force at contralateral heel strike. T_{ds} is half the double support time, $T_{ds} = \frac{1}{2}\,t$; t_P is the peak force time, $t_P = \frac{2}{3}\,T_{ds}$; and k_1 and k_2 are both constants whose value are: $k_1 = e^{4/9}$ and $k_2 = \frac{k_1}{2}\,e^{-16/9}$;

Hence, $F_x(0) = F_{x0}$ and $F_x(2T_{ds}) = 0$;

It is required to know the transition times between single and double support. This information can be provided by force plates or using kinematics data. In this case, the data which provide the plates has been used. In this project there are three double support phases hence, it is necessary to seek six values: the time of the beginning and the end of each double support phases.

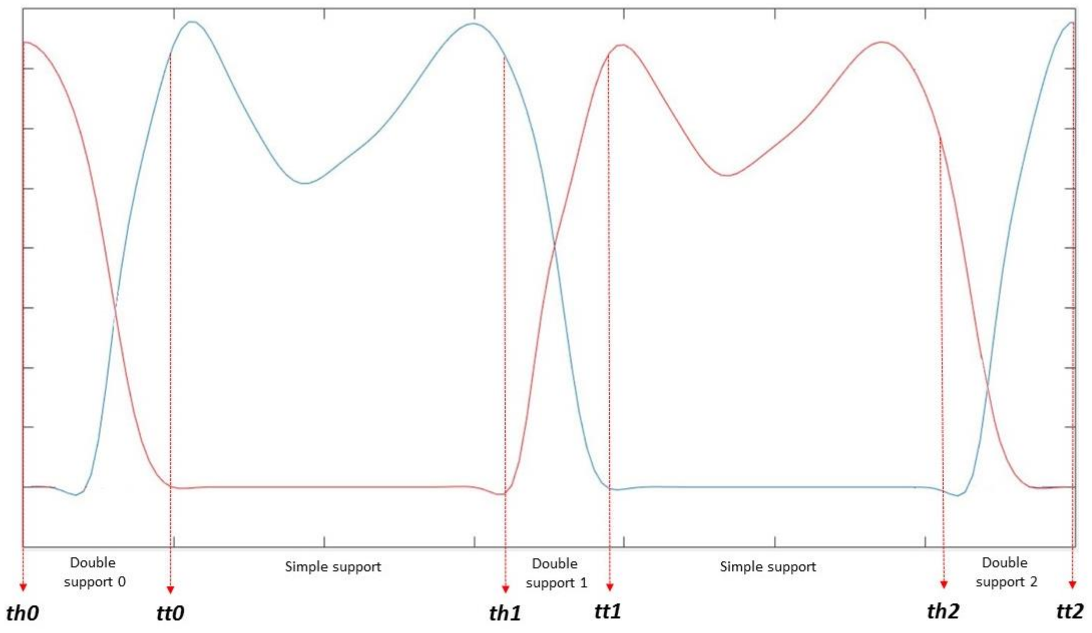


Figure 23: Shape of vertical forces during gait.

This image illustrates an example of the shape of the vertical forces implicated during gait. Where the colours of each curve represent one and the other foot. Force plates only provide the information showed at the picture number 23, however, it is needed to calculate what it is called as “double support 0” and “double support 2” in order to achieve the purpose of this study.

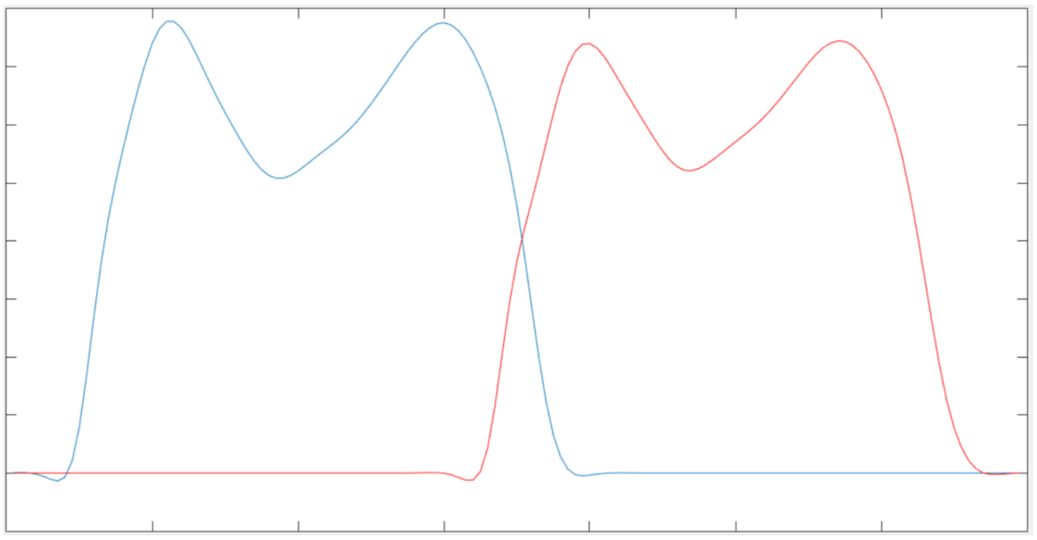


Figure 24: Vertical forces provided by two force plates during gait.

To find the value of th0, tt0, th1, tt1, th2 and tt2 two methods have been used. One for the "double support 1" and the other one for the "double support 0 and 2".

The condition applied to find the times of the double support 1, was that the vertical forces in both plates had to be greater than 5 N.

Th0 and tt2 are the started and the final times of the experiment. To seek the values of tt0 and th2, the first peak and last peak of the vertical forces, provided by the force plates, had to be found. As it is possible to see in the figure number 23, there are four different peaks. Then, the time was selected by subtracting a little percentage of the value of each peak and taking the time of those new values.

Now, the inverse dynamics can be solved during the double support phases. The only difference between the procedure followed in the simple support phase, is that now there is no free foot. The trailing foot has the forces provided by the STA method, therefore, there is no inconsistent system and the problem can be solved.

4 Results and discussion

In this section of the document, the results will be discussed.

As it has been already said, the results will be compared with the forces and moments which the force plates yield.

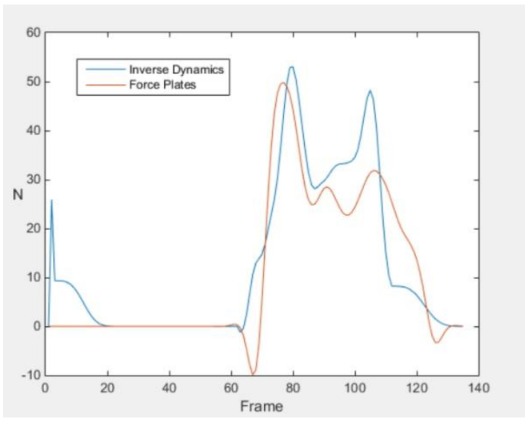


Figure 25: F_x component, left foot.

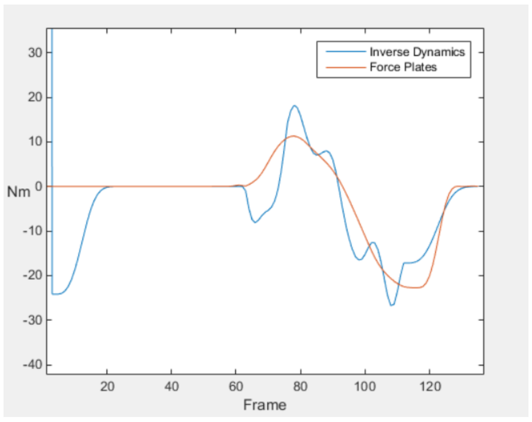


Figure 28: M_x component, left foot.

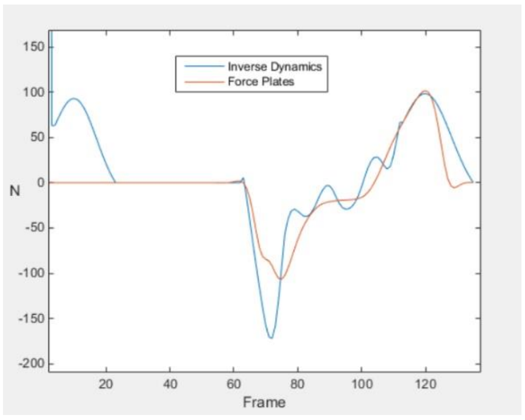


Figure 26: F_y component, left foot.

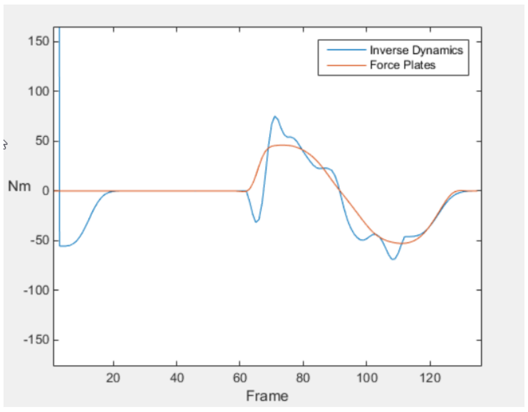


Figure 29: M_y component, left foot.

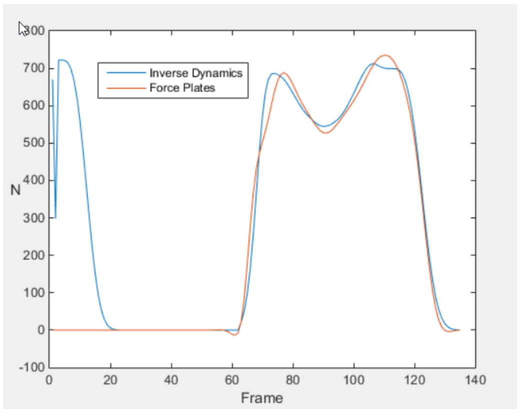


Figure 27: F_z component, left foot.

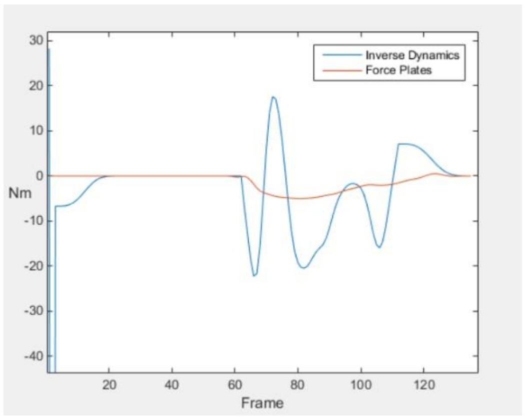


Figure 30: M_z component, left foot.

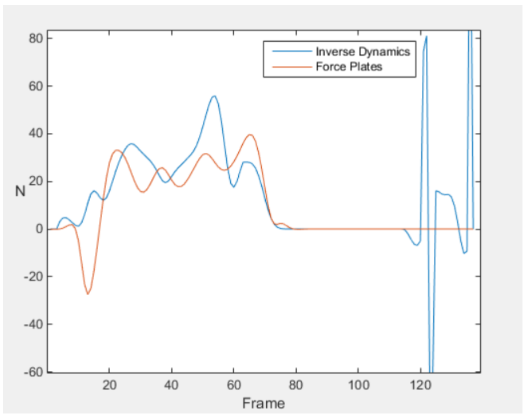


Figure 31: F_x component, right foot.

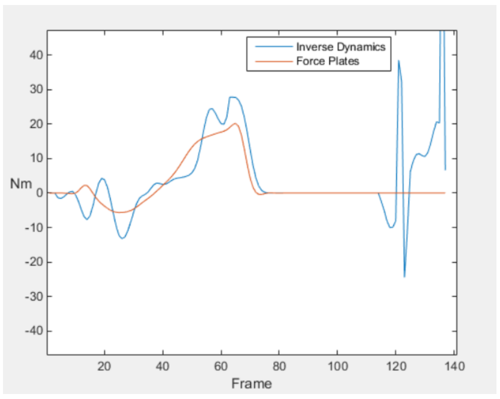


Figure 34: M_x component, right foot.

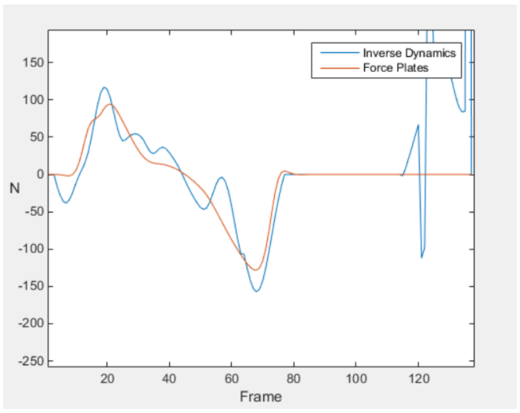


Figure 32: F_y component, right foot.

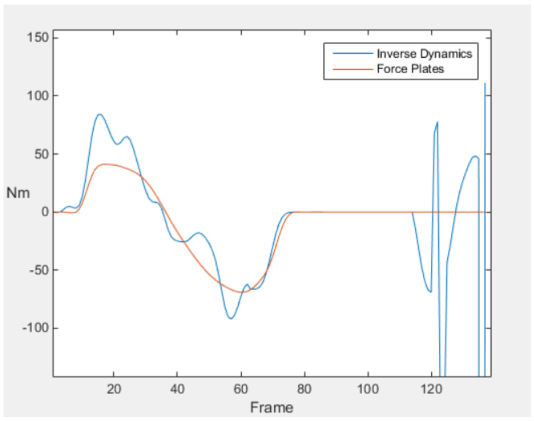


Figure 35: M_y component, right foot.

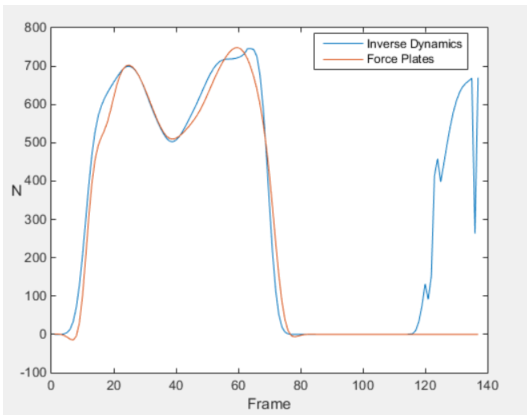


Figure 33: F_z component, right foot.

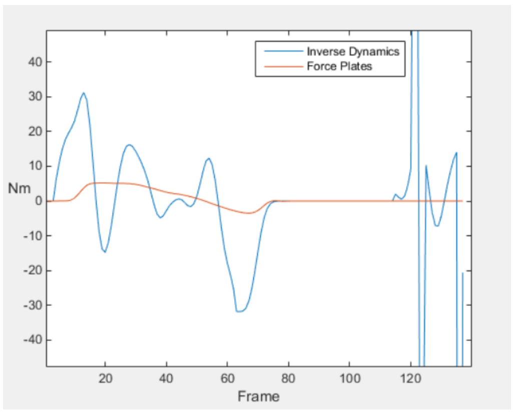


Figure 36: M_z component, right foot.

These are the reaction forces and moments yields by this study. Some comments will be made in order to understand the results.

Firstly, it is possible to observe that the accuracy reached in both feet are great except for graphics number 29 and 35. Both represent the moments in the transverse plane of each foot. As it can be seen, the values are not as high as the ones in the other graphics. Due to that fact, the errors are more appreciable here than in the other reactions. However, the results of these two moments are not very significant for the study of the human gait. Thus, there is not such a big importance in the bad results of these moments.

On the other hand, moments in the sagittal plane plus the vertical forces are, without any doubt, the most important of the project. To achieve this level of accuracy, some parameters have been altered: the double support times.

As it was said in section number 3.3, th0, tt0, th2 and tt2 were calculated finding the peaks of the vertical forces, provided by the force plates, and then applying a percentage. This percentage is the value which must be altered.

In the previous pictures, the values were: $th0 = 4$; $tt0 = 25$; $th2 = 115$; $tt2 = 137$; and for these experimental data, it has been decided that these values were the best ones. Of course, the values must be altered for different future experimental data.

In terms of understanding how to choose these values, it will be seen how the reactions behave when these times change:

If the values change to $th0 = 2$; $tt0 = 23$; $th2 = 115$; $tt2 = 137$;the reactions adopt a worse shape. The following pictures illustrates these changes.

Vertical force at the right foot (figure number 32) compared with the next picture:

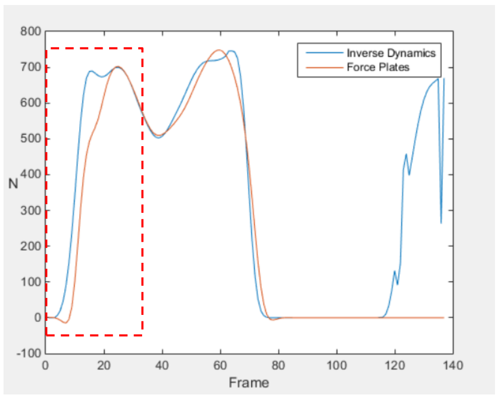


Figure 37: F_z component at the right foot, th0 = 2 and tt0 = 23.

Frontal moment at the right foot (figure number 33) compared with the next picture:

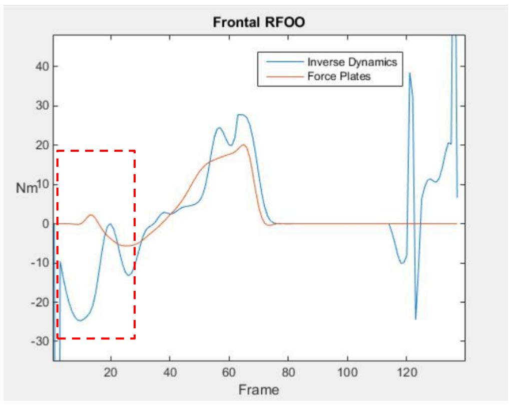


Figure 38: M_x component at the right foot, th0 = 2 and tt0 = 23.

Sagittal moment (figure number 34) compared with the following picture:

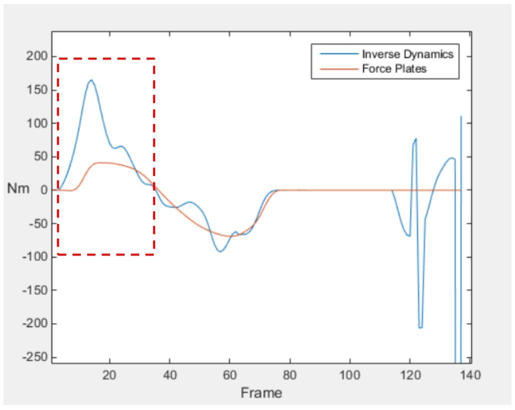


Figure 39: M_y component at the right foot, th0 = 2 and tt0 = 23.

On the other hand, if the values change to $th0 = 4$; $tt0 = 25$; $th2 = 111$; $tt2 = 137$; the graphics of the reactions, related with the left foot, adopt these shapes.

F_x at the left foot (image number 24) compared with:

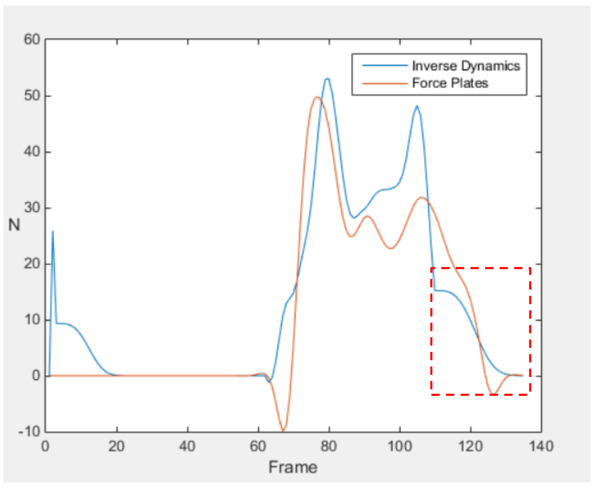


Figure 40: F_x component at the left foot, th2 = 111 and tt2 = 137.

F_y at the left foot (figure number 25) compared with:

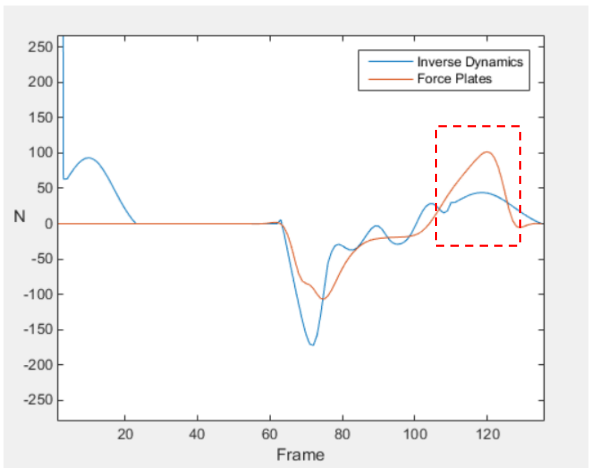


Figure 41: F_y component at the left foot, th2 = 111 and tt2 = 137.

Sagittal moment at the left foot (picture number 28) compared with:

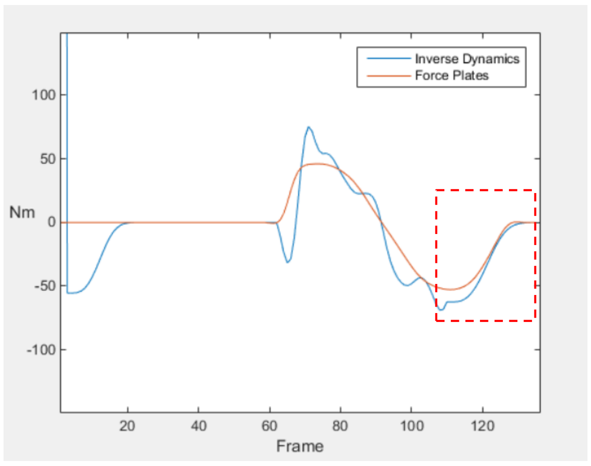


Figure 42: M_y component at left foot, th2 = 111 and tt2 = 137.

There are such a large range of possibilities. Before get the final results, it is recommended to alter the double support times on the grounds of improving them.

Another fact is that although the good shapes of the reactions, there are some areas where irregular peaks appeared. In terms of finding the reason why this happens, there have been done some checks.

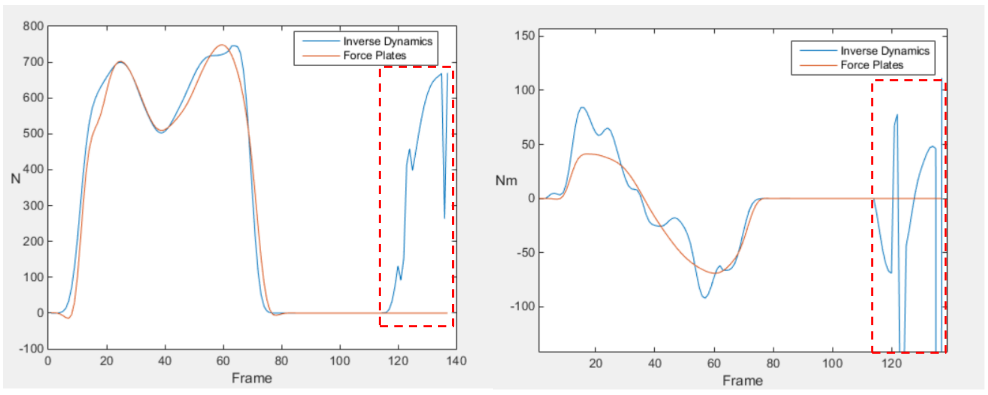


Figure 43: Example of irregular peaks.

The experimental work at the laboratory, can introduce noise. To discard this possibility, the calculations have been done without the inertial forces. While the inertial forces depend on the accelerations data, the external forces do not.

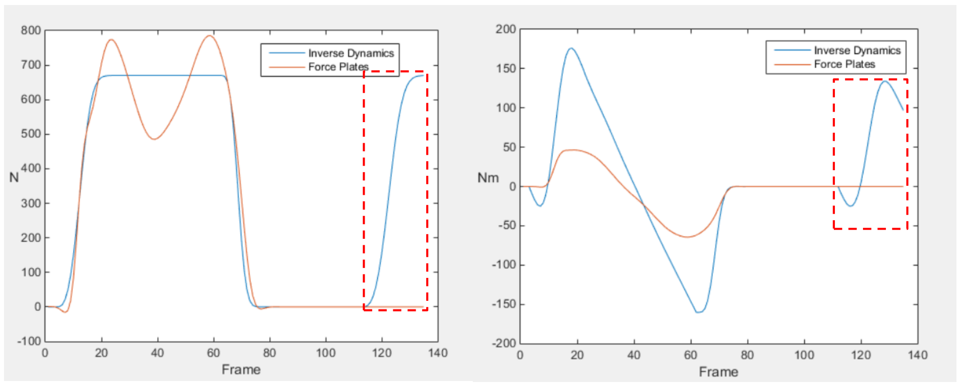


Figure 44: F_z and M_y results without inertia.

It can be observed in the previous figure, that if the inertial forces are not considered in the calculation, the irregular peaks disappear, therefore, the problem is that the capture of the movement, at the experimental work with Vicon Nexus, has not been as much accurate as it should be.

In order to be sure that the program is calculating the reaction forces in a good way when the inertial forces are not considered, there has been done another test: it has been compared the moments yielded by the program during the simple support, with the ones produced by the total weight of the body, at the mass centre of each feet.

The following figures illustrate the method:

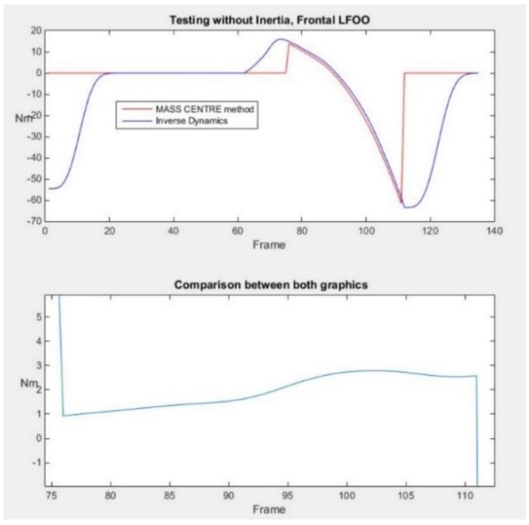


Figure 45: Comparison between M_x without inertia and the mass centre method during the second simple support.

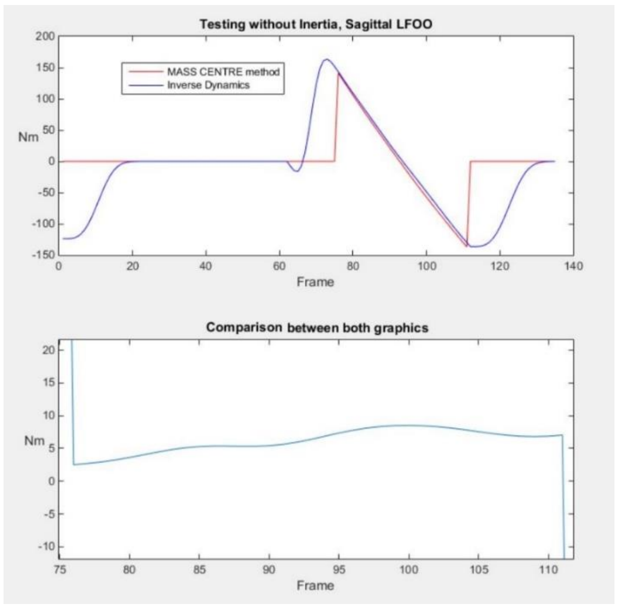


Figure 46: Comparison between M_y without inertia and the mass centre method during the second simple support.

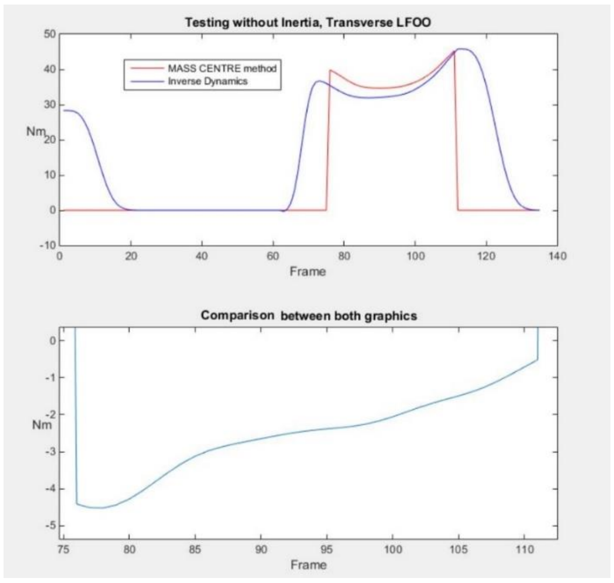


Figure 47: Comparison between M_z without inertia and the mass centre method during the second simple support.

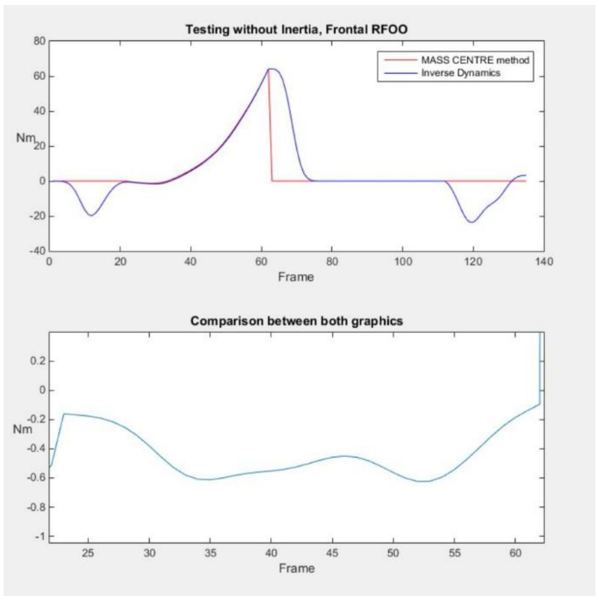


Figure 48: Comparison between M_x without inertia and the mass centre method during the first simple support.

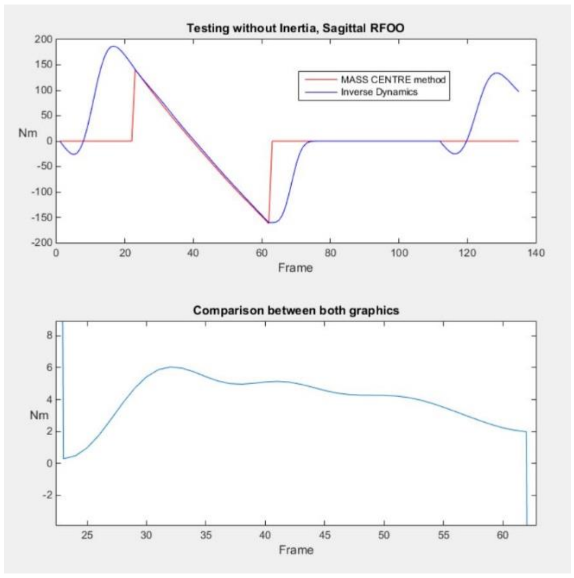


Figure 49: Comparison between M_y without inertia and the mass centre method during the first simple support.

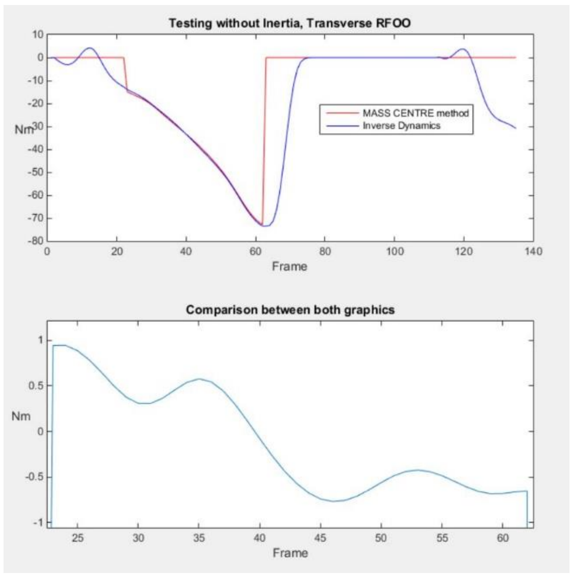


Figure 50: Comparison between M_z without inertia and the mass centre method during the first simple support.

The second graph of each image, illustrates the error commit. Without no doubt, it can be said that the results are almost the same. This method supports the viability of the calculations which have been done.

5 Conclusions and future work

As a conclusion, it can be said that all the purposes have been achieved. The order and method followed by the code to calculate the equilibrium equations have yielded the results expected. It can be observed that the STA method suppose a good alternative to solve the double support problem. Moreover, now it is known that the parameters related with the double support are essential to achieve good results in these kind of projects. Finally, after seeing the results, there is no doubt, that the capture of the movement is also vital to avoid irregular behaviour of the reactions.

Although all the evidence suggest that the results made this study a successful method for the calculation of the reactions between feet and ground, there are some notes that must be made for future studies:

1. It has been followed the STA algorithm to calculated the double support phases, however, the results obtained for the M_z component are not what they were expected.
2. Some of the curves described shapes which have not been understood. The reason why this happens should be studied in future projects to improve the accuracy.

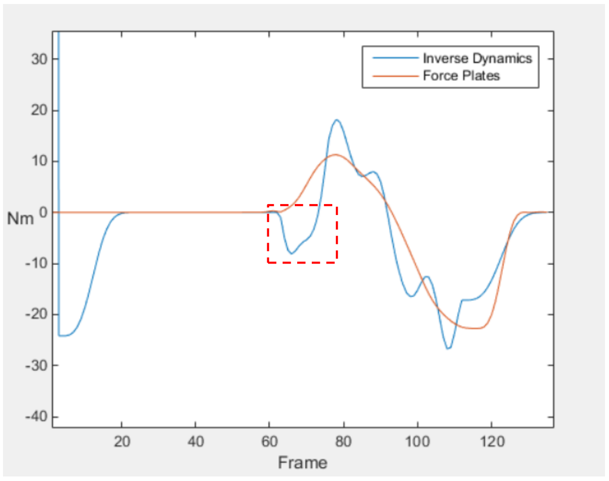


Figure 51: $M_x$component, left foot. Example of shape misunderstood.

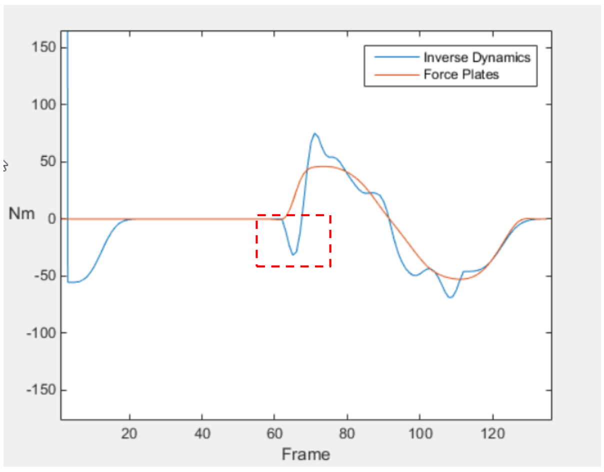


Figure 52: $M_y$component, left foot. Example of shape misunderstood.

Considering these notes and the goals achieved, new challenges can be faced now.

The Oxford foot is a model created in 2001 which divides the foot in three segments to be able to study it in an easy way while, at the same time, a better accuracy can be reached.

In this project the foot was studied as a unique segment, however, the results aimed, allow to widen the horizons and work with an Oxford foot model. For that reason, the next step would be to modify this project in order to study this model.

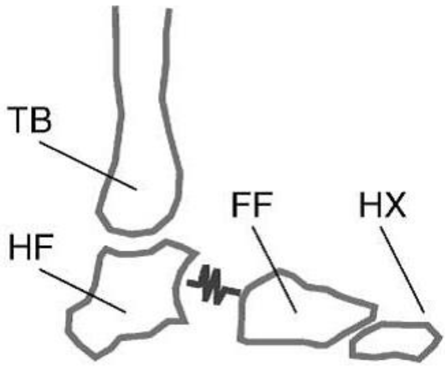


Figure 53: Oxford foot model.

As it can be seen in the image above, the foot is now divides in the Hallux segment (HX), the Forefoot segment (FF) and the Hindfoot one (HF). This means that the inverse dynamic problem will become more complex than the one exposed at this document.

The increasing of the number of equations and limbs will suppose a new challenge to deal with. On the other hand, this future work exposed will yield results closer to the reality.

6 Matlab code

```
function [Fres cdg2 moment2 GRF2 Fres4 Fin ]=solveEquiWithoutGRF (GRF,pos,vel1,acc1,AnimationData,MARKERS,datafile)

close all
clc
global MODEL
global AMTI

off=1; %activate or deactivate Fin

%######################
[position,JointPos] = PlugInGait(MARKERS,datafile);
%######################
Fres4 = zeros(size(GRF,1),6*MODEL.numBod);
moment2=zeros(size(GRF,1),3);
cdg2=zeros(size(GRF,1),3);
sumInertialForces = zeros(1,3);
Fin = zeros(size(GRF,1),6*MODEL.numBod);
Fext =zeros(6*MODEL.numBod,1);
for k=1:MODEL.numBod
    Fext (6*k-3,:) = -MODEL.MassMatrix(6*k-5,6*k-5)*9.81;
end
Fres = zeros(size(GRF,1),6*MODEL.numBod);
Fapl= zeros(size(GRF,1),6*MODEL.numBod);Fapl1= zeros(size(GRF,1),6*MODEL.numBod);
Fapl2= zeros(size(GRF,1),6*MODEL.numBod);Fapl3= zeros(size(GRF,1),6*MODEL.numBod);
Fapl4= zeros(size(GRF,1),6*MODEL.numBod);Fapl5= zeros(size(GRF,1),6*MODEL.numBod);
p=zeros(MODEL.numJoint,4);
pJ=zeros(MODEL.numJoint,3);
pC=zeros(MODEL.numJoint,3);
pJ2=zeros(MODEL.numJoint,3);

%finding double support times
[th0, tt0, th, tt, th2, tt2] = DsupportTime(GRF);

u=8; u2=2; u3=3; u4=2; u5=1; u6=5; u7=1;%AUXILIARY VARIABLES FOR LIMBS
q=1; k=1; h=0; %TO ENSURE THE ORDER
goDsupport0=1;  goDsupport2=0;

for i=1:size(GRF,1)

          while k<=length(MODEL.SEGMENT)
                    %RIGHT AND LEFT ARM
                    if (strcmp(MODEL.SEGMENT(k),MODEL.SEGMENT(u))) && q==1
                              while ~(strcmp(MODEL.SEGMENT(k),MODEL.SEGMENT(u+u2)))

                                          p(k,1:4) = pos(i,7*k-3:7*k);
                                          w(k,1:3) = vel1(i,6*k-2:6*k);
                                          pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                          R = Rot_Mat(p(k,1:4));

                                          seg=MODEL.SEGMENT{k,1};
                                          if strcmp(seg,'RHAN') || strcmp(seg,'LHAN')
                                                    child=MODEL.JOINT.(seg){1,1};
                                          else
                                                    parent=MODEL.JOINT.(seg){1,1};
                                                    child=MODEL.JOINT.(seg){2,1};
                                          end
```

```
                        pJ(k,1:3)=AnimationData.(seg).(child)(i,:); %next joint position
                        vec=(pC(k,1:3)-pJ(k,1:3));

                        % Inertial Forces
                        H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                        Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off; %Fin son las -F de inercia
                        Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                        sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);

                        if strcmp(seg,'RHAN') || strcmp(seg,'LHAN')
                                %Final forces in Joint
                                Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)';
                                Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)')+ (R' * cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                        else
                                pJ2(k,1:3)=JointPos.(parent)(i,:); %position of the previous joint
                                vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                %Forces from the previous Joint
                                Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(k-u5)-5:6*(k-u5)-3);
                                R_parent = Rot_Mat(p(k-u5,1:4));
                                Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(k-u5)-2:6*(k-u5)))';
                                %Final forces in Joint
                                Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl(i,6*k-5:6*k-3);
                                Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec2,Fapl(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                        end

                        if u==8
                              k=k+1;
                        else
                              k=k-1;
                        end
                end
                p(k,1:4) = pos(i,7*k-3:7*k);
                w(k,1:3) = vel1(i,6*k-2:6*k);
                pC(k,1:3)=pos(i,7*k-6:7*k-4);
                R = Rot_Mat(p(k,1:4));


                seg=MODEL.SEGMENT{k,1};
                child=MODEL.JOINT.(seg){2,1};
                parent=MODEL.JOINT.(seg){1,1};
                pJ(k,1:3)=AnimationData.(seg).(child)(i,:);

                % Inertial Forces
                H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                pJ2(k,1:3)=JointPos.(parent)(i,:);
                vec=(pC(k,1:3)-pJ(k,1:3));
                vec2=(pJ2(k,1:3)-pJ(k,1:3));
                %Forces from the previous Joint
                Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(k-u5)-5:6*(k-u5)-3);
                R_parent = Rot_Mat(p(k-u5,1:4));
                Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(k-u5)-2:6*(k-u5)))';
                %Final forces in Joint
                Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl(i,6*k-5:6*k-3);
                Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec2,Fapl(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' *
cross(vec,Fext(6*k-5:6*k-3)')')';

                if u==13
                          q=2;
                end
                u=13; u2=-2; k=1; u5=-1;
```

```
end

%HEAD
if (strcmp(MODEL.SEGMENT(k),'HEA'))  && q==2

        p(k,1:4) = pos(i,7*k-3:7*k);
        w(k,1:3) = vel1(i,6*k-2:6*k);
        pC(k,1:3)=pos(i,7*k-6:7*k-4);
        R = Rot_Mat(p(k,1:4));

        seg=MODEL.SEGMENT{k,1};
        child=MODEL.JOINT.(seg){1,1};
        pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
        vec=(pJ(k,1:3)-pC(k,1:3));

        % Inertial Forces
        H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
        Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
        Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
        sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
        %Final forces in Joint
        Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)';
        Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)')- (R' * cross(vec,Fres(i,6*k-5:6*k-3))')';

        q=3; k=1;
end

%TRUNK
if (strcmp(MODEL.SEGMENT(k),'TRU')) && q==3

           seg=MODEL.SEGMENT{k,1};
           %defining next and previous joints
           child=MODEL.JOINT.(seg){4,1};
           parent1=MODEL.JOINT.(seg){1,1};
           parent2=MODEL.JOINT.(seg){2,1};
           parent3=MODEL.JOINT.(seg){3,1};

           p(k,1:4) = pos(i,7*k-3:7*k);
           w(k,1:3) = vel1(i,6*k-2:6*k);
           pC(k,1:3)=pos(i,7*k-6:7*k-4);
           R = Rot_Mat(p(k,1:4));
           pJ(k,1:3)=AnimationData.(seg).(child)(i,:);

           % Inertial Forces
           H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
           Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
           Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
           sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
           vec=(pC(k,1:3)-pJ(k,1:3)); %moments at the next joint
           %Forces from the previous Joints
           %RUPA where k=10
           pJ2_1(k,1:3)=JointPos.(parent1)(i,:);
           vec2_1=(pJ2_1(k,1:3)-pJ(k,1:3));
           Fapl1(i,6*k-5:6*k-3)=-Fres(i,6*(10)-5:6*(10)-3);
           R_parent1 = Rot_Mat(p(10,1:4));
           Fapl1(i,6*k-2:6*k)=-R'*R_parent1*(Fres(i,6*(10)-2:6*(10)))';
           %LUPA where k=11
           pJ2_2(k,1:3)=JointPos.(parent2)(i,:);
           vec2_2=(pJ2_2(k,1:3)-pJ(k,1:3));
           Fapl2(i,6*k-5:6*k-3)=-Fres(i,6*(11)-5:6*(11)-3);
           R_parent2 = Rot_Mat(p(11,1:4));
           Fapl2(i,6*k-2:6*k)=-R'*R_parent2*(Fres(i,6*(11)-2:6*(11)))';
```

```
                                %HEA where k=14
                                pJ2_3(k,1:3)=JointPos.(parent3)(i,:);
                                vec2_3=(pJ2_3(k,1:3)-pJ(k,1:3));
                                Fapl3(i,6*k-5:6*k-3)=-Fres(i,6*(14)-5:6*(14)-3);
                                R_parent3 = Rot_Mat(p(14,1:4));
                                Fapl3(i,6*k-2:6*k)=-R'*R_parent3*(Fres(i,6*(14)-2:6*(14)))';

                                Fapl(i,6*k-5:6*k)=Fapl1(i,6*k-5:6*k)+Fapl2(i,6*k-5:6*k)+Fapl3(i,6*k-5:6*k);
                                %Final forces in Joint
                                Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl1(i,6*k-5:6*k-3)-Fapl2(i,6*k-5:6*k-3)-Fapl3(i,6*k-5:6*k-3);
                                Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec2_1,Fapl1(i,6*k-5:6*k-3))')'-(R' * cross(vec2_2,Fapl2(i,6*k-5:6*k-3))')'-
(R' * cross(vec2_3,Fapl3(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';

                                q=4;
                                k=1;
                    end

                    %FREE LEG
                    if  i==1 || i==2 || i==3 || (i>tt && i<th2) %this condition depends on the value of th0
                              if (strcmp(MODEL.SEGMENT(k),'RFOO')) && q==4
                                        while ~(strcmp(MODEL.SEGMENT(k),'RTHG'))
                                                    if strcmp(seg,'RFOO')
                                                              %Final forces in Joint
                                                              Fres(i,6*k-5:6*k)=zeros(1,6);
                                                    end
                                                    p(k,1:4) = pos(i,7*k-3:7*k);
                                                    w(k,1:3) = vel1(i,6*k-2:6*k);
                                                    pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                                    R = Rot_Mat(p(k,1:4)); u5=1;

                                                    seg=MODEL.SEGMENT{k,1};
                                                    if strcmp(seg,'RFOO')
                                                              child=MODEL.JOINT.(seg){1,1};
                                                    else
                                                              parent=MODEL.JOINT.(seg){1,1};
                                                              child=MODEL.JOINT.(seg){2,1};
                                                    end
                                                    pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                                    vec=(pC(k,1:3)-pJ(k,1:3));

                                                    % Inertial Forces
                                                    H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' *
w(k,:)';
                                                    Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                                    Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                                    sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);

                                                    if strcmp(seg,'RFOO')
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                                    else
                                                              pJ2(k,1:3)=JointPos.(parent)(i,:);
                                                              vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                                              %Forces from the previous Joint
                                                              Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                                              R_parent = Rot_Mat(p(k-u5,1:4));
                                                              Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
```

```
                                                    end

                                                    k=k+1;
                                        end
                                        p(k,1:4) = pos(i,7*k-3:7*k);
                                        w(k,1:3) = vel1(i,6*k-2:6*k);
                                        pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                        R = Rot_Mat(p(k,1:4));

                                        seg=MODEL.SEGMENT{k,1};
                                        child=MODEL.JOINT.(seg){2,1};
                                        parent=MODEL.JOINT.(seg){1,1};
                                        pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                        vec=(pC(k,1:3)-pJ(k,1:3));

                                        % Inertial Forces
                                        H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                                        Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                        Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                        sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                                        pJ2(k,1:3)=JointPos.(parent)(i,:);
                                        vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                        %Forces from the previous Joint
                                        Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                        R_parent = Rot_Mat(p(k-u5,1:4));
                                        Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                        %Final forces in Joint
                                        Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                        Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-
3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';

                                        q=5;u3=5;u4=2;
                                        k=1;
                              end
                    elseif  i>tt0 && i<th
                              if (strcmp(MODEL.SEGMENT(k),'LFOO')) && q==4
                                        while ~(strcmp(MODEL.SEGMENT(k),'LTHG'))
                                                    if strcmp(seg,'LFOO')
                                                              %Final forces in Joint
                                                              Fres(i,6*k-5:6*k)=zeros(1,6);
                                                    end
                                                    p(k,1:4) = pos(i,7*k-3:7*k);
                                                    w(k,1:3) = vel1(i,6*k-2:6*k);
                                                    pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                                    R = Rot_Mat(p(k,1:4)); u5=-1;

                                                    seg=MODEL.SEGMENT{k,1};
                                                    if strcmp(seg,'LFOO')
                                                              child=MODEL.JOINT.(seg){1,1};
                                                    else
                                                              parent=MODEL.JOINT.(seg){1,1};
                                                              child=MODEL.JOINT.(seg){2,1};
                                                    end
                                                    pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                                    vec=(pC(k,1:3)-pJ(k,1:3)); %moments at the next joint

                                                    % Inertial Forces
                                                    H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' *
w(k,:)';
                                                    Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                                    Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                                    sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                                                    if strcmp(seg,'LFOO')
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)';
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)')+ (R' * cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-
3)')')';
                                                    else
                                                              pJ2(k,1:3)=JointPos.(parent)(i,:);
                                                              vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                                              %Forces from the previous Joint
                                                              Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                                              R_parent = Rot_Mat(p(k-u5,1:4));
                                                              Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                                    end

                                                    k=k-1;
                                        end
                                        p(k,1:4) = pos(i,7*k-3:7*k);
                                        w(k,1:3) = vel1(i,6*k-2:6*k);
                                        pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                        R = Rot_Mat(p(k,1:4));

                                        seg=MODEL.SEGMENT{k,1};
                                        child=MODEL.JOINT.(seg){2,1};
                                        parent=MODEL.JOINT.(seg){1,1};
                                        pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                        vec=(pC(k,1:3)-pJ(k,1:3));

                                        % Inertial Forces
                                        H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                                        Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                        Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                        sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                                        pJ2(k,1:3)=JointPos.(parent)(i,:);
                                        vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                        %Forces from the previous Joint
                                        Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                        R_parent = Rot_Mat(p(k-u5,1:4));
                                        Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                        %Final forces in Joint
                                        Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                        Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-
3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';

                                        q=5;u3=3;u4=-2;
                                        k=1;
                              end

                   %########################### Double support 1 ############################
                    elseif i>=th && i<=tt

                                  if (strcmp(MODEL.SEGMENT(k),'RFOO')) && q==4
                                        if h==1
                                              Fx0(1,1:6)=Fres(i-1,1:6);
                                              h=0;
                                        end
                                        Fres(i,6*k-5:6*k)=PRUEBADSUPPORT(Fx0,th,tt,i);
                                        u5=1;
                                        while ~(strcmp(MODEL.SEGMENT(k),'RTHG'))
                                                    p(k,1:4) = pos(i,7*k-3:7*k);
                                                    w(k,1:3) = vel1(i,6*k-2:6*k);
                                                    pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                                    R = Rot_Mat(p(k,1:4)); u5=1;

                                                    seg=MODEL.SEGMENT{k,1};
                                                    if strcmp(seg,'RFOO')
                                                              child=MODEL.JOINT.(seg){1,1};
                                                    else
                                                              parent=MODEL.JOINT.(seg){1,1};
                                                              child=MODEL.JOINT.(seg){2,1};
                                                    end
                                                    pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                                    vec=(pC(k,1:3)-pJ(k,1:3));

                                                    % Inertial Forces
                                                    H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' *
w(k,:)';
                                                    Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                                    Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                                    sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);

                                                    if strcmp(seg,'RFOO')
                                                              Rright=R;
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))- (R' * cross(vec,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                                    else
                                                              pJ2(k,1:3)=JointPos.(parent)(i,:);
                                                              vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                                              %Forces from the previous Joint
                                                              Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                                              R_parent = Rot_Mat(p(k-u5,1:4));
                                                              Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                                    end

                                                    k=k+1;
                                        end
                                        p(k,1:4) = pos(i,7*k-3:7*k);
                                        w(k,1:3) = vel1(i,6*k-2:6*k);
                                        pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                        R = Rot_Mat(p(k,1:4));

                                        seg=MODEL.SEGMENT{k,1};
                                        child=MODEL.JOINT.(seg){2,1};
                                        parent=MODEL.JOINT.(seg){1,1};
                                        pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                        vec=(pC(k,1:3)-pJ(k,1:3));

                                        % Inertial Forces
                                        H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                                        Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                        Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                        sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                                        pJ2(k,1:3)=JointPos.(parent)(i,:);
                                        vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                        %Forces from the previous Joint
                                        Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                        R_parent = Rot_Mat(p(k-u5,1:4));
                                        Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                        %Final forces in Joint
                                        Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                        Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-
3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';

                                        q=5;u3=5;u4=2;
                                        k=1; goDsupport0=0; goDsupport2=1;
                                  end
                    %######################### end of the double support 1 ###########################


                    %################## double support 0 & double support 2 ######################
                    elseif (i>=th0 && i<=tt0)  ||  (i>=th2 && i<=tt2)

                                  if (strcmp(MODEL.SEGMENT(k),'LFOO')) && q==4
                                        if h==0
                                              Fx0(1,1:6)=Fres(i-1,6*k-5:6*k);
                                              h=1;
                                        end
                                        if goDsupport0==1
                                                  Fres(i,6*k-5:6*k)=PRUEBADSUPPORT(Fx0,th0,tt0,i); %esto es mi Fres directamente, CUIDADO PORQUE AHORA MIS INCÓGNITAS SERÁN LAS Fapl
                                                  u5=-1;
                                        elseif goDsupport2==1
                                                  Fres(i,6*k-5:6*k)=PRUEBADSUPPORT(Fx0,th2,tt2,i);
                                                  u5=-1;
                                        end

                                        while ~(strcmp(MODEL.SEGMENT(k),'LTHG'))
                                                    p(k,1:4) = pos(i,7*k-3:7*k);
                                                    w(k,1:3) = vel1(i,6*k-2:6*k);
                                                    pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                                    R = Rot_Mat(p(k,1:4)); u5=-1;

                                                    seg=MODEL.SEGMENT{k,1};
                                                    if strcmp(seg,'LFOO')
                                                              child=MODEL.JOINT.(seg){1,1};
                                                    else
                                                              parent=MODEL.JOINT.(seg){1,1};
                                                              child=MODEL.JOINT.(seg){2,1};
                                                    end
                                                    pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                                    vec=(pC(k,1:3)-pJ(k,1:3));

                                                    % Inertial Forces
                                                    H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' *
w(k,:)';
                                                    Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                                    Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                                    sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);

                                                    if strcmp(seg,'LFOO')
                                                              Rleft=R;
                                                              %Final forces in Joint
                                                              Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                              Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))- (R' * cross(vec,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                                    else
                                                              pJ2(k,1:3)=JointPos.(parent)(i,:);
                                                              vec2=(pJ2(k,1:3)-pJ(k,1:3));
```

```
                                                        %Forces from the previous Joint
                                                        Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                                        R_parent = Rot_Mat(p(k-u5,1:4));
                                                        Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                                        %Final forces in Joint
                                                        Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                                        Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                                end

                                                k=k-1;
                                        end
                                        p(k,1:4) = pos(i,7*k-3:7*k);
                                        w(k,1:3) = vel1(i,6*k-2:6*k);
                                        pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                        R = Rot_Mat(p(k,1:4));

                                        seg=MODEL.SEGMENT{k,1};
                                        child=MODEL.JOINT.(seg){2,1};
                                        parent=MODEL.JOINT.(seg){1,1};

                                        pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                        vec=(pC(k,1:3)-pJ(k,1:3));

                                        % Inertial Forces
                                        H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                                        Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                        Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                        sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                                        pJ2(k,1:3)=JointPos.(parent)(i,:);
                                        vec2=(pJ2(k,1:3)-pJ(k,1:3));
                                        %Forces from the previous Joint
                                        Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(k-u5)-5:6*(k-u5)-3);
                                        R_parent = Rot_Mat(p(k-u5,1:4));
                                        Fres(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(k-u5)-2:6*(k-u5)))';
                                        %Final forces in Joint
                                        Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fres(i,6*k-5:6*k-3);
                                        Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fres(i,6*k-2:6*k))-(R' * cross(vec2,Fres(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-
3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';

                                        q=5;u3=3;u4=-2;
                                        k=1;
                                end

                end
                %############### end of the double support 0 & double support 2 #####################

                %PELVIS
                if (strcmp(MODEL.SEGMENT(k),'PEL')) && q==5
                        seg=MODEL.SEGMENT{k,1};
                        if u3==5
                                parent4=MODEL.JOINT.(seg){1,1}; u6=3; u7=1; %Right foot free
                                child=MODEL.JOINT.(seg){2,1};
                        else
                                parent4=MODEL.JOINT.(seg){2,1}; u6=5; u7=-1; %Left foot free
                                child=MODEL.JOINT.(seg){1,1};
                        end
                        parent5=MODEL.JOINT.(seg){3,1};

                        p(k,1:4) = pos(i,7*k-3:7*k);
                        w(k,1:3) = vel1(i,6*k-2:6*k);
                        pC(k,1:3)=pos(i,7*k-6:7*k-4);
```

```
                              R = Rot_Mat(p(k,1:4));

                              pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                              vec=(pC(k,1:3)-pJ(k,1:3));

                              % Inertial Forces
                              H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                              Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                              Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                              sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                              %Forces from the previous Joints
                              %TRU where k=15
                              pJ2_5(k,1:3)=JointPos.(parent5)(i,:);
                              vec2_5=(pJ2_5(k,1:3)-pJ(k,1:3));
                              Fapl5(i,6*k-5:6*k-3)=-Fres(i,6*(15)-5:6*(15)-3);
                              R_parent5 = Rot_Mat(p(15,1:4));
                              Fapl5(i,6*k-2:6*k)=-R'*R_parent5*(Fres(i,6*(15)-2:6*(15)))';

                              if   (i>=th0 && i<=tt0) || (i>=th && i<=tt) || (i>=th2 && i<=tt2)
                                        pJ2_4(k,1:3)=JointPos.(parent4)(i,:);
                                        vec2_4=(pJ2_4(k,1:3)-pJ(k,1:3));
                                        Fres(i,6*k-5:6*k-3)=-Fapl(i,6*(u6)-5:6*(u6)-3);
                                        R_parent4 = Rot_Mat(p(u6,1:4));
                                        Fres(i,6*k-2:6*k)=-R'*R_parent4*(Fapl(i,6*(u6)-2:6*(u6)))';

                                        %Final forces in Joint
                                        Fapl(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl5(i,6*k-5:6*k-3)-Fres(i,6*k-5:6*k-3);
                                        Fapl(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl5(i,6*k-2:6*k)-Fres(i,6*k-2:6*k))-(R' * cross(vec2_5,Fapl5(i,6*k-5:6*k-3))')'- (R' *
cross(vec2_4,Fres(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                              else
                                        %RTHG o LTHG con k=u6
                                        pJ2_4(k,1:3)=JointPos.(parent4)(i,:);
                                        vec2_4=(pJ2_4(k,1:3)-pJ(k,1:3));
                                        Fapl4(i,6*k-5:6*k-3)=-Fapl(i,6*(u6)-5:6*(u6)-3);
                                        R_parent4 = Rot_Mat(p(u6,1:4));
                                        Fapl4(i,6*k-2:6*k)=-R'*R_parent4*(Fapl(i,6*(u6)-2:6*(u6)))';

                                        %Final forces in Joint
                                        Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl4(i,6*k-5:6*k-3)-Fapl5(i,6*k-5:6*k-3);
                                        Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl4(i,6*k-2:6*k)-Fapl5(i,6*k-2:6*k))- (R' * cross(vec2_4,Fapl4(i,6*k-5:6*k-3))')'-(R' *
cross(vec2_5,Fapl5(i,6*k-5:6*k-3))')'+ (R' * cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                              end

                              q=6;
                              k=1;
                    end

                    %FOOT IN CONTACT
                    if (strcmp(MODEL.SEGMENT(k),MODEL.SEGMENT(u3))) && q==6
                              while ~(strcmp(MODEL.SEGMENT(k),MODEL.SEGMENT(u3+u4)))
                                            p(k,1:4) = pos(i,7*k-3:7*k);
                                            w(k,1:3) = vel1(i,6*k-2:6*k);
                                            pC(k,1:3)=pos(i,7*k-6:7*k-4);
                                            R = Rot_Mat(p(k,1:4));

                                            seg=MODEL.SEGMENT{k,1};
                                            child=MODEL.JOINT.(seg){1,1};
                                            parent=MODEL.JOINT.(seg){2,1};
                                            pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                                            vec=(pC(k,1:3)-pJ(k,1:3));

                                            % Inertial Forces
```

```
                                           H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                                           Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                                           Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                                           sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                                           pJ2(k,1:3)=JointPos.(parent)(i,:);
                                           vec2=(pJ2(k,1:3)-pJ(k,1:3));

                                           if  (i>=th0 && i<=tt0) || (i>=th && i<=tt) || (i>=th2 && i<=tt2)
                                                     %Forces from the previous Joint
                                                     if strcmp(seg,'RTHG') || strcmp(seg,'LTHG')
                                                               %PEL k=4
                                                               Fapl(i,6*k-5:6*k-3)=-Fapl(i,6*(4)-5:6*(4)-3);
                                                               R_parent = Rot_Mat(p(4,1:4));
                                                               Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fapl(i,6*(4)-2:6*(4)))';
                                                     else
                                                               Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(k-u7)-5:6*(k-u7)-3);
                                                               R_parent = Rot_Mat(p(k-u7,1:4));
                                                               Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(k-u7)-2:6*(k-u7)))';
                                                     end
                                                     %Final forces in Joint
                                                     Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl(i,6*k-5:6*k-3);
                                                     Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec2,Fapl(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                          else
                                                     %Forces from the previous Joint
                                                     if strcmp(seg,'RTHG') || strcmp(seg,'LTHG')
                                                               %PEL k=4
                                                               Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(4)-5:6*(4)-3);
                                                               R_parent = Rot_Mat(p(4,1:4));
                                                               Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(4)-2:6*(4)))';
                                                     else
                                                               Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(k-u7)-5:6*(k-u7)-3);
                                                               R_parent = Rot_Mat(p(k-u7,1:4));
                                                               Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(k-u7)-2:6*(k-u7)))';
                                                     end
                                                     %Final forces in Joint
                                                     Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl(i,6*k-5:6*k-3);
                                                     Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec2,Fapl(i,6*k-5:6*k-3))')'+ (R' *
cross(vec,Fin(i,6*k-5:6*k-3))')'- (R' * cross(vec,Fext(6*k-5:6*k-3)')')';
                                           end

                                           if u4==2
                                                  k=k+1;
                                           else
                                                  k=k-1;
                                           end
                             end
                             p(k,1:4) = pos(i,7*k-3:7*k);
                             w(k,1:3) = vel1(i,6*k-2:6*k);
                             pC(k,1:3)=pos(i,7*k-6:7*k-4);
                             R = Rot_Mat(p(k,1:4));

                             seg=MODEL.SEGMENT{k,1};
                             child=MODEL.JOINT.(seg){1,1};
                             pJ(k,1:3)=AnimationData.(seg).(child)(i,:);
                             vec=(pJ(k,1:3)-pC(k,1:3)); %Moments at the mass centre

                             % Inertial Forces
                             H = ([cross([1 0 0],R'*w(k,:)');cross([0 1 0],R'*w(k,:)');cross([0 0 1],R'*w(k,:)')]) * MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) * R' * w(k,:)';
                             Fin(i,6*k-5:6*k-3) =(MODEL.MassMatrix(6*k-5:6*k-3,6*k-5:6*k-3)* acc1(i,6*k-5:6*k-3)')*off;
                             Fin(i,6*k-2:6*k) = (MODEL.MassMatrix(6*k-2:6*k,6*k-2:6*k) *  R' * acc1(i,6*k-2:6*k)'+H)*off;
                             sumInertialForces = sumInertialForces + Fin(i,6*k-5:6*k-3);
                             if  (i>=th0 && i<=tt0) || (i>=th && i<=tt) || (i>=th2 && i<=tt2)
```

```
                                        %Forces from the previous Joint
                                        Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(k-u7)-5:6*(k-u7)-3);
                                        R_parent = Rot_Mat(p(k-u7,1:4));
                                        Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(k-u7)-2:6*(k-u7)))';
                                        %Final forces in Joint
                                        Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl(i,6*k-5:6*k-3);
                                        Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec,Fapl(i,6*k-5:6*k-3))')';
                             else
                                        %Forces from the previous Joint
                                        Fapl(i,6*k-5:6*k-3)=-Fres(i,6*(k-u7)-5:6*(k-u7)-3);
                                        R_parent = Rot_Mat(p(k-u7,1:4));
                                        Fapl(i,6*k-2:6*k)=-R'*R_parent*(Fres(i,6*(k-u7)-2:6*(k-u7)))';
                                        %Final forces in Joint
                                        Fres(i,6*k-5:6*k-3)=Fin(i,6*k-5:6*k-3)-Fext(6*k-5:6*k-3)'-Fapl(i,6*k-5:6*k-3);
                                        Fres(i,6*k-2:6*k)=(Fin(i,6*k-2:6*k)-Fext(6*k-2:6*k)'-Fapl(i,6*k-2:6*k))- (R' * cross(vec,Fapl(i,6*k-5:6*k-3))')';
                              end
                              %plate Forces and moments at the mass centre of each foot
                              if (i>=th && i<=tt)
                                        Rleft=R;
                              elseif  (i>=th0 && i<=tt0) ||  (i>=th2 && i<=tt2)
                                        Rright=R;
                              else
                                        Rleft=R;
                                        Rright=R;
                              end
                                        vec3=(AMTI.pos(1:3)-pos(i,7*1-6:7*1-4)); %From the mass centre of the right foot to the centre of the right plate
                                        GRF2(i,1:3)=GRF(i,1:3);
                                        GRF2(i,4:6)=(Rright'*(GRF(i,4:6)+cross(vec3, GRF(i,1:3)))')';

                                        vec3=(AMTI.pos(4:6)-pos(i,7*7-6:7*7-4)); %From the mass centre of the left foot to the centre of the left plate
                                        GRF2(i,7:9)=GRF(i,7:9);
                                        GRF2(i,10:12)=(Rleft'*(GRF(i,10:12)+cross(vec3, GRF(i,7:9)))')';

                              [cdg moment]=MassCentre(Fext, pos, i, R, pC)
                              moment2(i,1:3)=moment(i,1:3);
                              cdg2(i,1:3)=cdg(i,1:3);

                   k=length(MODEL.SEGMENT); q=1;u=8;u2=2;u3=3;u4=2;u5=1; u6=5; u7=1;
                   end
          k=k+1;
          end
k=1;
end
fres2(:,1:6)=Fres(:,1:6);
fres2(:,7:12)=Fres(:,37:42);
multiplot (fres2, GRF2)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Double support times                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [th0, tt0, th, tt, th2, tt2] = DsupportTime(GRF)
h=-1;
th0=4;
tt2=size(GRF,1);
          [z , y]= findpeaks(GRF(:,3));
          for i=1:length(z)
                    if z(i)>700 && h==-1
                              m = y(i);
                              while (GRF(y(i),3)-GRF(m,3))<1
                                        tt0 = m;
                                        m=m-1;
                              end
                              h=0;
                    end
          end
          for i=1:size(GRF,1)
               if GRF(i,3)>5 && GRF(i,9)>5 && h==0
                         th=i;
                         h=1;
               elseif GRF(i,3)<5 && GRF(i,9)>5 && h==1
                         tt=i;
                         h=2;
               end
          end
          [a , b]= findpeaks(GRF(:,9));
          for i=1:length(a)
                    if a(i)>700
                              m=b(i);
                              while (GRF(b(i),9)-GRF(m,9))<20 %20
                                        th2 = m;
                                        m=m+1;
                              end
                    end
          end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Mass centre method                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [cdg moment]=MassCentre(Fg, pos, i, R, pC)
clc
global MODEL

num=zeros(1,3);
den=0;

moment=zeros(135,3);
for k=1:length(MODEL.SEGMENT)
          Fg2(k,1)=Fg(6*k-3);
end
          oo=pos(i,7*1-6:7*1-4);%Right foot
%           oo=pos(i,7*7-6:7*7-4);%Left foot
          for k=1:15
                    pC(k,1:3)=pos(i,7*k-6:7*k-4);
                    vec = pC(k,1:3)-oo;
                    num = num + Fg2(k)*vec;
                    den = den + Fg2(k);
          end
          cdg(i,1:3)=num/den;
          moment(i,1:3)=(R'*cross(cdg(i,:), [0 0 670])')';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Plot                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function multiplot(Fres2,GRF)
          for i=1:12
                    figure(i)
                    plot (Fres2(:,i))
                    hold on
                    plot (GRF(:,i))
                    legend ('Inverse Dynamics', 'Force Plates')
                    xlabel ('Frame')
                    if i==4 || i==5 || i==6 || i==10 || i==11 || i==12
                              ylabel ('Nm')
                    else
                              ylabel ('N')
                    end
                     set(get(gca,'YLabel'),'Rotation',0)

          end

end
```

7 References

[1] https://en.wikipedia.org/wiki/Biomechanics

[2] Dempster W. (1955) Space requirements for the seated operator. Wade technical report 55159.

[3] Hanavan E. (1964.) A mathematical model of the human body. Technical Report 64-102.

[4] Zatsiorsky V, Seluyanov V, Chugunova L (1990). In vivo body segment inertial parameters determination using a gamma-scanner method. In N. Berme & A. Cappozzo (Eds.), Biomechanics of Human Movement: Applications in Rehabilitation, Sports and Ergonomics (pp. 186-202).

[5] Hatze, H. (1980). A mathematical model for the computational determination of parameter values of Anthropometric segments. Journal of Biomechanics, 13, 833-843.

[6] Antonio López Iruzubieta (2013). Trabajo de fin de carrera 'Modelo biomecánico OpenSim de miembro específico'.

[7] Christopher L Vaughan, Brian L Davis, Jeremy C O.Connor (1992). 'Dynamics of human gait'.

[8] Urbano Lugrís, Jairo Carlín, Rosa Pàmies-Vilà, Josep M. Font-Llagunes, Javier Cuadrado (2013). 'Solution methods for the double-support indeterminacy in human gait'. Multibody Syst Dyn (2013) 30:247–263.

[9] Jacobo Guajardo-Fajardo Caballos (2016). Trabajo de fin de grado 'Análisis cinemático durante la marcha mediante el Modelo Oxford del Pie'.

[10] José David Jarmell Carrasco (2016). Trabajo de fin de grado 'Oxford foot model kinetic analysis during the stance phase in gait'.

[11] Parviz E. Nikravesh (1988). 'Computer-Aided Analysis of Mechanical Systems'.

[12] Lei Ren, Richard K. Jones, David Howard. (2008). 'Whole body inverse dynamics over a complete gait cycle based only on measured kinematics'. Journal of Biomechanics 41 2750-2759.