UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Project Report

Bachelor's degree in Mechanical Engineering

Simulation of movement

Location: EEBE

Autor: Yago Trias Vila
Tutor/a EEBE: Gil Serrancolí

Contents

List of Figures

1 Introduction

The Final Project for the Simulation of Movement course consists of choosing one model of our interest and using the knowledge achieved during the course and laboratories and simulate the movement from a starting position and final position using Dynamic Optimization Methods using Direct Collocation (DC) using CasADI[1][3] as a library inside MatLab.

Dynamic Optimization is an optimal control problem[4] using an Integral cost function (e.g: function summing up all the joint moments or work developed by motors during the movement). It also takes into account three types of constraints:

- Dynamic constraints (e.g: Velocities have to be coordinate derivatives and Equations of motion in implicit formulation).
- Path constraints (e.g: Coordinates and their velocities can't exceed certain bounds).
- Boundary conditions (e.g: Fixed initial and final states).

The Model selected for the project consisted of a 2D model of a skeletal body with 10 degrees of freedom. This model has defined 3 absolute coordinates and the remaining ones are relative with one another.

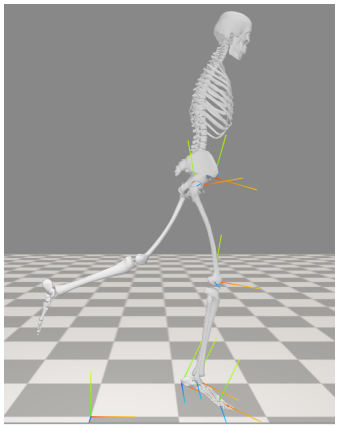

Figure 1: Gait10DoF

2 Software environment

The software used to solve the Dynamic optimization problem is MATLAB[2] using the library CasADI[1] as a numerical optimization framework that uses automatic differentiation.

Figure 2: CasADI

3 Work developed

The Model selected for the project consisted of a 2D model of a skeletal body with 10 degrees of freedom. This model has defined 3 absolute coordinates and the remaining ones are relative with one another. The 1^{st} coordinate is used to defined the pelvis tilt with is the angle of the pelvis coordinate system in relation to the ground and the 2^{nd} and 3^{rd} are the coordinate displacements based on the pelvis base. 4^{th} is the relative angle in relation to the tibia base, 5^{th} is the relative angle in relation to the femur basis and 6^{th} is the relative angle in relation to the hip basis. The remaining ones 7^{th}, 8^{th}, 9^{th} are the same coordinate but for the left leg and the final one 10^{th} is the lumbar extension relative to the hip base.

This model has in the end 10 degrees of freedom as described however it only has 7 joint moments regarding from the 4^{th} to the 10^{th} coordinates. So it is a under torqued model in which the Optimization Solver is going to take a little a little more to compute the optimal solution.

The Optimization solver to be able to compute the joint moments of the coordinates it needs to take into account the contact forces. In this model is defined 4 independent contact forces:

- $F1$ is positioned at the right, just before the toes.
- $F2$ is positioned at the right, at the calcaneus.

- $F3$ is positioned at the left, at the calcaneus.
- $F4$ is positioned at the left, just before the toes.

The script used to individual contact forces need to be provided the penetration (pen) of each point, the velocity of the penetration ($dpen$) and tangential velocity ($dtan$).

$$PoH = \begin{pmatrix} -0.0707 \\ -0.0661 \end{pmatrix},\ AK = \begin{pmatrix} -0.04877 \\ -0.04195 \end{pmatrix},\ l_{femur} = 0.41,\ l_{tibia} = 0.43 \quad (1)$$

The position of the calcaneus and toes position if computed using rotation matrices as defined:

$$P_o = \begin{pmatrix} \cos q_1 & \sin q_1 \\ -\sin q_1 & \cos q_1 \end{pmatrix} \cdot \begin{pmatrix} -q_2 \\ -q_3 \end{pmatrix} \quad (2)$$

$$H = P_o + \begin{pmatrix} \cos q_1 & \sin q_1 \\ -\sin q_1 & \cos q_1 \end{pmatrix} \cdot PoH \quad (3)$$

For the right leg is used:

$$K_r = H + l_{femur} \cdot \begin{pmatrix} \sin(q_6 - q_1) \\ -\cos(q_6 - q_1) \end{pmatrix} \quad (4)$$

$$A_r = K_r + l_{tibia} \cdot \begin{pmatrix} \sin(q_6 - q_1 + q_5) \\ -\cos(q_6 - q_1 + q_5) \end{pmatrix} \quad (5)$$

$$calcn_r = A_r + \begin{pmatrix} \cos(q_6 - q_1 + q_5 + q_4) - \sin(q_6 - q_1 + q_5 + q_4) \\ \sin(q_6 - q_1 + q_5 + q_4) \cos(q_6 - q_1 + q_5 + q_4) \end{pmatrix} \cdot AK \quad (6)$$

$$toes_r = A_r + \begin{pmatrix} \cos(q_6 - q_1 + q_5 + q_4) - \sin(q_6 - q_1 + q_5 + q_4) \\ \sin(q_6 - q_1 + q_5 + q_4) \cos(q_6 - q_1 + q_5 + q_4) \end{pmatrix} \cdot \left(AK + \begin{pmatrix} 0.2 \\ 0 \end{pmatrix} \right) \quad (7)$$

For the left leg is used:

$$K_l = H + l_{femur} \cdot \begin{pmatrix} \sin(q_7 - q_1) \\ -\cos(q_7 - q_1) \end{pmatrix} \quad (8)$$

$$A_l = K_l + l_{tibia} \cdot \begin{pmatrix} \sin(q_7 - q_1 + q_8) \\ -\cos(q_7 - q_1 + q_8) \end{pmatrix} \quad (9)$$

$$calcn_l = A_l + \begin{pmatrix} \cos(q_7 - q_1 + q_8 + q_9) - \sin(q_7 - q_1 + q_8 + q_9) \\ \sin(q_7 - q_1 + q_8 + q_9) \cos(q_7 - q_1 + q_8 + q_9) \end{pmatrix} \cdot AK \quad (10)$$

$$toes_l = A_l + \begin{pmatrix} \cos(q_7 - q_1 + q_8 + q_9) - \sin(q_7 - q_1 + q_8 + q_9) \\ \sin(q_7 - q_1 + q_8 + q_9) \cos(q_7 - q_1 + q_8 + q_9) \end{pmatrix} \cdot \left(AK + \begin{pmatrix} 0.2 \\ 0 \end{pmatrix} \right) \tag{11}$$

Then *pen*, *dpen* and *dtan* have been calculated using the Symbolic Toolbox in MATLAB

$$pen_1 = -toes_{r,y}, \quad dpen_1 = \frac{dpen_1}{dt}, \quad dtan_1 = \frac{dtoes_{r,x}}{dt} \tag{12}$$

$$pen_2 = -calcn_{r,y}, \quad dpen_2 = \frac{dpen_2}{dt}, \quad dtan_2 = \frac{dcalcn_{r,x}}{dt} \tag{13}$$

$$pen_3 = -calcn_{l,y}, \quad dpen_3 = \frac{dpen_3}{dt}, \quad dtan_3 = \frac{dcalcn_{l,x}}{dt} \tag{14}$$

$$pen_4 = -toes_{l,y}, \quad dpen_4 = \frac{dpen_4}{dt}, \quad dtan_4 = \frac{dtoes_{l,x}}{dt} \tag{15}$$

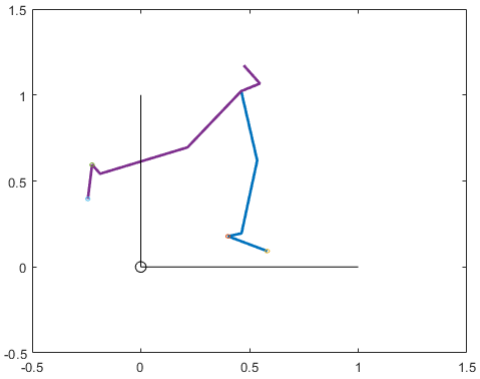

Figure 3: Calculated positions

The Dynamic Optimization using CasADI[1] needs certain constraint as explained before. The path constraints set up for this model are the coordinates and their velocities can't exceed certain bounds. The maximum angular velocity ω_{limit} is defined as 90 degress in 0.5 seconds and the maximum linear velocity v_{limit} is defined as 1 meter in 0.5 seconds.

$$\omega_{limit} = \frac{\pi/2}{0.5}\left(\frac{rad}{s}\right) \qquad v_{limit} = \frac{1}{0.5}\left(\frac{m}{s}\right) \tag{16}$$

States bounds during the movement		
Coordinate	States lower bound	States upper bound
Pelvis tilt	-5	90
Pelvis t_x	-2	0.2
Pelvis t_y	-2	0.2
Ankle angle right	-90	90
Knee angle right	-120	10
Hip flexion right	-120	120
Hip flexion left	-120	120
Knee angle left	-120	10
Ankle angle left	-90	90
Lumbar extension	-5	90

Finally the solver needs to have an objective to minimize called the cost function. Initially the cost function is set-up to be the sum of joint moments however the movement obtained does not resemble a real human walking kinematics. As a result other cost function can be added so that the movement obtained is more realistic, such as the sum of coordinate accelerations or the sum of power consumed by the joints.

$$L_1 = \sum_{n=1}^{7} \tau_n^2, \qquad L_2 = \sum_{n=1}^{10} \alpha_n^2, \qquad L_3 = \sum_{n=1}^{7} \tau_n \cdot \omega_n \tag{17}$$

Note that the sum of powers is nos squared over two, as a result the optimization method will try to make it as low as possible meaning in biomechanics that will try to optimize the trajectory so that the input torque produced by the muscles on the joints is as low as possible

The boundary conditions set for the simulations are as initial state all coordinates and velocities to zero, except the vertical translation which the

lower bound is set to $t_y = -0.980$, the upper bound to $t_y = -0.900$ ans the initial guess is $t_y = -0.940$. As final states the horizontal translation $t_x = -1$ and all velocities to zero. The time horizon is $T = 2\ \ (s)$

4 Results

The results obtained on the simulations are quite unique. For the 1^{st}, the cost function was just minimizing the joint moments and solution obtained is:

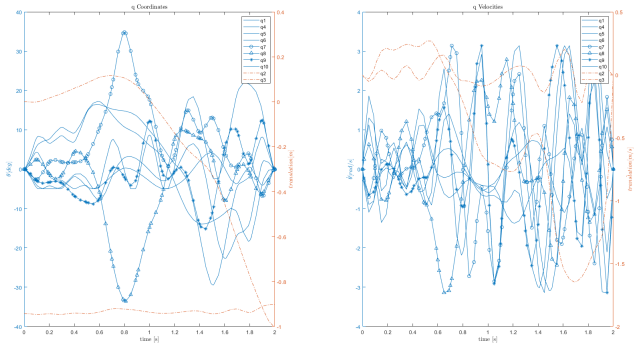

Figure 4: Coordinates and velocities - L_1

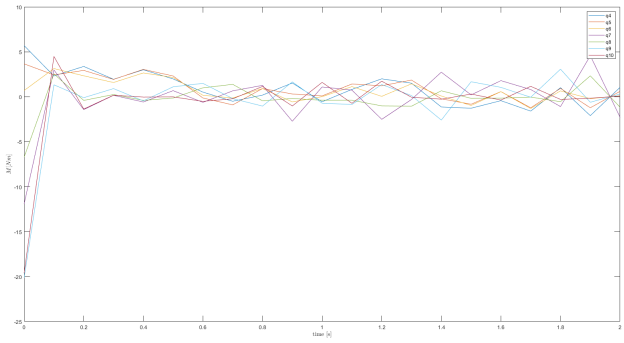

Figure 5: Torques - L_1

The movement obtained is very erratic and does not have any human appealing movement. As a consequence it is added the 2^{nd} cost function L_2 which added minimizing all accelerations, and the results are:

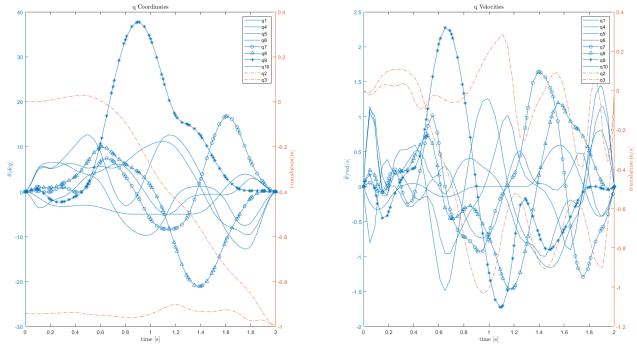

Figure 6: Coordinates and velocities - $L_1 + L_2$

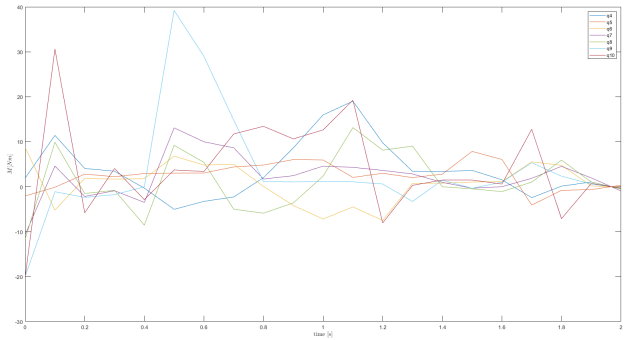

Figure 7: Torques - $L_1 + L_2$

As it can be seen the obtained coordinates and their velocities are a lot smoother and the movement obtained is much more appealing. Even though the movement is very similar to a walking movement, it is decided to add a $3rd$ cost function L_3 minimizing the power generated by the muscles on the joints.

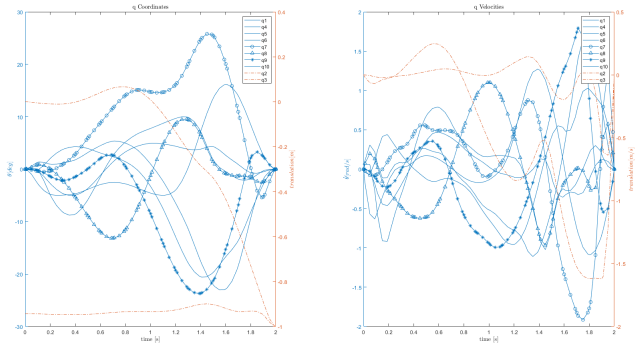

Figure 8: Coordinates and velocities - $L_1 + L_2 + L_3$

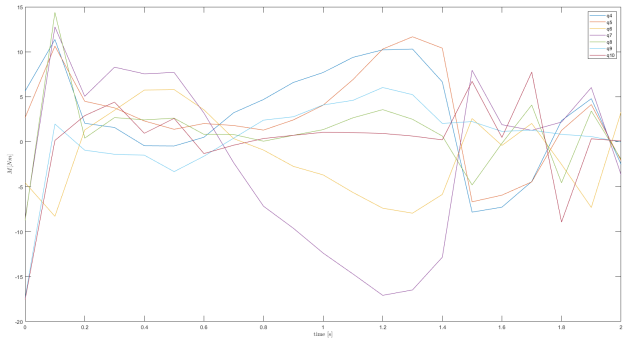

Figure 9: Torques - $L_1 + L_2 + L_3$

Finally the trajectory optimized is a lot more explosive than the last obtained. Also it is interesting to measure the forces between the feet and the floor since is known that the muscles produce the movement by generating forces [5]. The action forces generated by muscles result in reaction forces applied by the ground to the foot, this ground forces can be measured during the walk or run with a force plate however they are quite expensive and the

space where the forces can be measures is very limited. That's why being able to approximate these forces by using a digital model and approximating them is very useful since reduces costs.

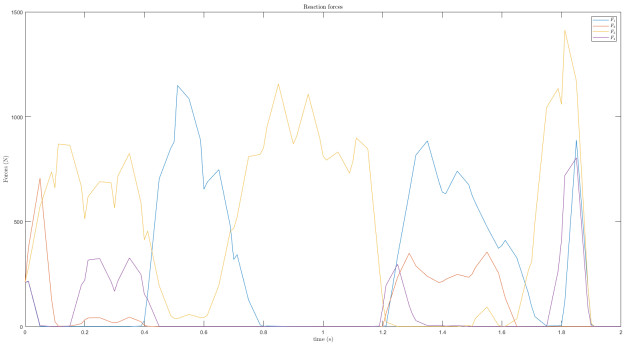

Figure 10: Reaction Forces - $L_1 + L_2$

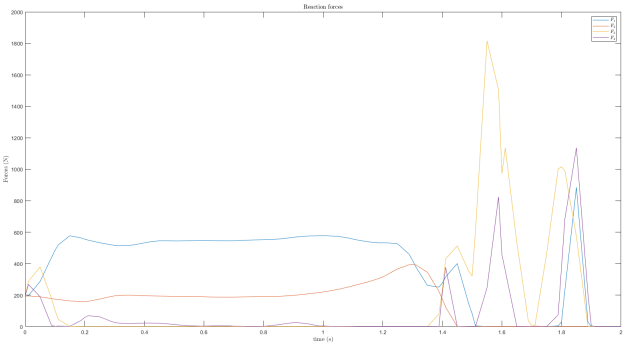

Figure 11: Reaction Forces - $L_1 + L_2 + L_3$

5 Conclusion

In conclusion the simulation of movement project has been a success. The main objective of the project was to get comfortable with MATLAB and the CasADI framework and also with the model selected to perform the dynamic optimization problem. For instance I think that this project helped us a lot to internalise the methods learnt during the course especially the formulation of the kinematics and the dynamic optimization method using direct collocation (DC).

Also, The three-step cost functions (L_1, L_2, L_3) successfully transformed the initial erratic motion into a dynamic, human-resembling walk. Furthermore, the project demonstrated the efficiency of digital models in approximating forces, particularly the reaction forces between feet and floor. This not only validated the muscle-generated movement but also showcased the potential for cost-effective simulations to replace traditional, expensive force plate measurements. In essence, this project not only provides insights into optimizing movement dynamics but also highlights the true potential of digital simulations, making them valuable tools for cost-efficient bio-mechanical analyses and enhancing our understanding of human-like motion.

References

[1] Casadi. https://web.casadi.org/. [Online; accessed 2023-12-11].

[2] Matlab. https://es.mathworks.com/products/matlab.html. [Online; accessed 2023-12-11].

[3] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, jul 11 2018. [Online; accessed 2023-12-12].

[4] Gil Serrancolí and Rosa Pàmies-Vilà. Analysis of the influence of coordinate and dynamic formulations on solving biomechanical optimal control problems. *Mechanism and Machine Theory*, 142:103578, 12 2019. [Online; accessed 2023-12-11].

[5] Thomas K. Uchida and Scott L Delp. *Biomechanics of Movement: The Science of Sports, Robotics, and Rehabilitation*. MIT Press, jan 12 2021. [Online; accessed 2023-12-14].

Annex

MATLAB Scripts

Script for computing the penetration, velocity of penetration and tangential velocity based on the model kinematics.

```
function PendPendTan = ComputePendPendTan(Xkm1)
% Model equations
import casadi.*

PendPendTan=MX.zeros(1,12);

q1 = Xkm1(1);
q1dot = Xkm1(2);
q2 = Xkm1(3);
q2dot = Xkm1(4);
q3 = Xkm1(5);
q3dot = Xkm1(6);
q4 = Xkm1(7);
q4dot = Xkm1(8);
q5 = Xkm1(9);
q5dot = Xkm1(10);
q6 = Xkm1(11);
q6dot = Xkm1(12);
q7 = Xkm1(13);
q7dot = Xkm1(14);
q8 = Xkm1(15);
q8dot = Xkm1(16);
q9 = Xkm1(17);
q9dot = Xkm1(18);

% Calculate pos and vel of contact points
PoH = [-0.0707;-0.0661];
a = PoH(1); b = PoH(2);
AK = [-0.04877;-0.04195];
c = AK(1); d = AK(2);
l_femur = 0.41; % m
l_tibia = 0.43; % m

% Computing position
```

```
Po = [cos(q1) sin(q1); -sin(q1) cos(q1)]*[-q2; -q3];
H = Po + [cos(q1) sin(q1); -sin(q1) cos(q1)]*PoH;

%right positions
K_r = H +l_femur*[sin(q6-q1); -cos(q6-q1)];
A_r = K_r + l_tibia*[sin(q6-q1+q5); -cos(q6-q1+q5)];
calcn_r = A_r + [cos(q6-q1+q5+q4) -sin(q6-q1+q5-q4);
    sin(q6-q1+q5-q4) cos(q6-q1+q5-q4)]*AK;
toes_r = A_r + [cos(q6-q1+q5+q4) -sin(q6-q1+q5-q4);
   sin(q6-q1+q5-q4) cos(q6-q1+q5-q4)]*(AK+[0.2;0]);
v_calcn_r = [q1dot*sin(q1)*q2 - q3dot*sin(q1) - c*
   sin(q4 - q1 + q5 + q6)*(q4dot - q1dot + q5dot +
   q6dot) - q2dot*cos(q1) - l_femur*cos(q1 - q6)*(
   q1dot - q6dot) + b*q1dot*cos(q1) - a*q1dot*sin(q1
   ) + l_tibia*cos(q5 - q1 + q6)*(q5dot - q1dot +
   q6dot) - d*cos(q4 - q1 + q5 + q6)*(q4dot - q1dot
   + q5dot + q6dot) - q1dot*cos(q1)*q3; q2dot*sin(q1
   ) - q3dot*cos(q1) - d*sin(q4 - q1 + q5 + q6)*(
   q4dot - q1dot + q5dot + q6dot) + q1dot*sin(q1)*q3
    + l_femur*sin(q1 - q6)*(q1dot - q6dot) - a*q1dot
   *cos(q1) - b*q1dot*sin(q1) + l_tibia*sin(q5 - q1
   + q6)*(q5dot - q1dot + q6dot) + c*cos(q4 - q1 +
   q5 + q6)*(q4dot - q1dot + q5dot + q6dot) + q1dot*
   cos(q1)*q2];
v_toes_r = [q1dot*sin(q1)*q2 - q3dot*sin(q1) - q2dot
   *cos(q1) - l_femur*cos(q1 - q6)*(q1dot - q6dot) -
    sin(q4 - q1 + q5 + q6)*(c + 1/5)*(q4dot - q1dot
   + q5dot + q6dot) + b*q1dot*cos(q1) - a*q1dot*sin(
   q1) + l_tibia*cos(q5 - q1 + q6)*(q5dot - q1dot +
   q6dot) - d*cos(q4 - q1 + q5 + q6)*(q4dot - q1dot
   + q5dot + q6dot) - q1dot*cos(q1)*q3; q2dot*sin(q1
   ) - q3dot*cos(q1) - d*sin(q4 - q1 + q5 + q6)*(
   q4dot - q1dot + q5dot + q6dot) + q1dot*sin(q1)*q3
    + cos(q4 - q1 + q5 + q6)*(c + 1/5)*(q4dot -
   q1dot + q5dot + q6dot) + l_femur*sin(q1 - q6)*(
   q1dot - q6dot) - a*q1dot*cos(q1) - b*q1dot*sin(q1
   ) + l_tibia*sin(q5 - q1 + q6)*(q5dot - q1dot +
   q6dot) + q1dot*cos(q1)*q2];

```

```
%left positions
K_l = H +l_femur*[sin(q7-q1); -cos(q7-q1)];
A_l = K_l + l_tibia*[sin(q7-q1+q8); -cos(q7-q1+q8)];
calcn_l = A_l + [cos(q7-q1+q8+q9) -sin(q7-q1+q8+q9);
    sin(q7-q1+q8+q9) cos(q7-q1+q8+q9)]*AK;
toes_l = A_l + [cos(q7-q1+q8+q9) -sin(q7-q1+q8+q9);
   sin(q7-q1+q8+q9) cos(q7-q1+q8+q9)]*(AK+[0.2;0]);
v_calcn_l = [q1dot*sin(q1)*q2 - q3dot*sin(q1) - c*
   sin(q7 - q1 + q8 + q9)*(q7dot - q1dot + q8dot +
   q9dot) - q2dot*cos(q1) - l_femur*cos(q1 - q7)*(
   q1dot - q7dot) + b*q1dot*cos(q1) - a*q1dot*sin(q1
   ) + l_tibia*cos(q7 - q1 + q8)*(q7dot - q1dot +
   q8dot) - d*cos(q7 - q1 + q8 + q9)*(q7dot - q1dot
   + q8dot + q9dot) - q1dot*cos(q1)*q3; q2dot*sin(q1
   ) - q3dot*cos(q1) - d*sin(q7 - q1 + q8 + q9)*(
   q7dot - q1dot + q8dot + q9dot) + q1dot*sin(q1)*q3
    + l_femur*sin(q1 - q7)*(q1dot - q7dot) - a*q1dot
   *cos(q1) - b*q1dot*sin(q1) + l_tibia*sin(q7 - q1
   + q8)*(q7dot - q1dot + q8dot) + c*cos(q7 - q1 +
   q8 + q9)*(q7dot - q1dot + q8dot + q9dot) + q1dot*
   cos(q1)*q2];
v_toes_l = [q1dot*sin(q1)*q2 - q3dot*sin(q1) - q2dot
   *cos(q1) - l_femur*cos(q1 - q7)*(q1dot - q7dot) -
    sin(q7 - q1 + q8 + q9)*(c + 1/5)*(q7dot - q1dot
   + q8dot + q9dot) + b*q1dot*cos(q1) - a*q1dot*sin(
   q1) + l_tibia*cos(q7 - q1 + q8)*(q7dot - q1dot +
   q8dot) - d*cos(q7 - q1 + q8 + q9)*(q7dot - q1dot
   + q8dot + q9dot) - q1dot*cos(q1)*q3; q2dot*sin(q1
   ) - q3dot*cos(q1) - d*sin(q7 - q1 + q8 + q9)*(
   q7dot - q1dot + q8dot + q9dot) + q1dot*sin(q1)*q3
    + cos(q7 - q1 + q8 + q9)*(c + 1/5)*(q7dot -
   q1dot + q8dot + q9dot) + l_femur*sin(q1 - q7)*(
   q1dot - q7dot) - a*q1dot*cos(q1) - b*q1dot*sin(q1
   ) + l_tibia*sin(q7 - q1 + q8)*(q7dot - q1dot +
   q8dot) + q1dot*cos(q1)*q2];


% F1
pen1 = -toes_r(2);
dpen1 = -v_toes_r(2);
```

```
dtan1 = v_toes_r(1);
% F2
pen2 = -calcn_r(2);
dpen2 = -v_calcn_r(2);
dtan2 = v_calcn_r(1);
%F3
pen3 = -calcn_l(2);
dpen3 = -v_calcn_l(2);
dtan3 = v_calcn_l(1);
%F4
pen4 = -toes_l(2);
dpen4 = -v_toes_l(2);
dtan4 = v_toes_l(1);

PendPendTan(1) = pen1;
PendPendTan(2) = dpen1;
PendPendTan(3) = dtan1;

PendPendTan(4) = pen2;
PendPendTan(5) = dpen2;
PendPendTan(6) = dtan2;

PendPendTan(7) = pen3;
PendPendTan(8) = dpen3;
PendPendTan(9) = dtan3;

PendPendTan(10) = pen4;
PendPendTan(11) = dpen4;
PendPendTan(12) = dtan4;

end
```

Constant parameters values for the optimization simulations

```
1
2  %Set constant parameter values
3
4  % Time horizon
5  T = 2;
6  namefile = 'motion5.mot';
7
8  %Number of coordinates and torque controls
9  nq=10;
10 ncT=7;
11
12 %Set model variables and model equations
13
14 % Declare model variables
15 q1 = MX.sym('q1'); % pelvis tilt
16 q2 = MX.sym('q2'); % pelvis translation x
17 q3 = MX.sym('q3'); % pelvis translation y
18 q4 = MX.sym('q4'); % ankle angle right
19 q5 = MX.sym('q5'); % knee angle right
20 q6 = MX.sym('q6'); % hip flexion right
21 q7 = MX.sym('q7'); % hip flexion left
22 q8 = MX.sym('q8'); % knee angle left
23 q9 = MX.sym('q9'); % ankle angle left
24 q10 = MX.sym('q10'); % lumbar extension
25 q1dot = MX.sym('q1dot');
26 q2dot = MX.sym('q2dot');
27 q3dot = MX.sym('q3dot');
28 q4dot = MX.sym('q4dot');
29 q5dot = MX.sym('q5dot');
30 q6dot = MX.sym('q6dot');
31 q7dot = MX.sym('q7dot');
32 q8dot = MX.sym('q8dot');
33 q9dot = MX.sym('q9dot');
34 q10dot = MX.sym('q10dot');
35 x=[q1 q1dot q2 q2dot q3 q3dot q4 q4dot q5 q5dot q6
      q6dot q7 q7dot q8 q8dot q9 q9dot q10 q10dot];
36
37 % uT1 = MX.sym('uT1');
```

```
% uT2 = MX.sym('uT2');
% uT3 = MX.sym('uT3');
uT4 = MX.sym('uT4');
uT5 = MX.sym('uT5');
uT6 = MX.sym('uT6');
uT7 = MX.sym('uT7');
uT8 = MX.sym('uT8');
uT9 = MX.sym('uT9');
uT10 = MX.sym('uT10');

uT=[uT4 uT5 uT6 uT7 uT8 uT9 uT10];

ua_q1=MX.sym('ua_q1');
ua_q2=MX.sym('ua_q2');
ua_q3=MX.sym('ua_q3');
ua_q4=MX.sym('ua_q4');
ua_q5=MX.sym('ua_q5');
ua_q6=MX.sym('ua_q6');
ua_q7=MX.sym('ua_q7');
ua_q8=MX.sym('ua_q8');
ua_q9=MX.sym('ua_q9');
ua_q10=MX.sym('ua_q10');
ua=[ua_q1 ua_q2 ua_q3 ua_q4 ua_q5 ua_q6 ua_q7 ua_q8
   ua_q9 ua_q10];

% Call Computation Contact Forces
CF = MX.zeros(1,8);

F=external('F','gait10dof.dll');
xdot = [q1dot; ua_q1; q2dot; ua_q2; q3dot; ua_q3;
   q4dot; ua_q4; q5dot; ua_q5; q6dot; ua_q6; q7dot;
   ua_q7; q8dot; ua_q8; q9dot; ua_q9; q10dot; ua_q10
   ];

% Cost function
L = sum(uT.^2)+sum(ua.^2)+sum(uT.*(x(8:2:nq*2)));
%sum(x(2:2:nq*2).^2) %angular velocities
%sum(ua.^2); % angular accelerations
%sum(uT.^2) % joint moments o torques
%sum(uT.*(x(8:2:nq*2))); % power consumed
```

```
74
75 % Boundary conditions
76 %fixed initial states
77 initial_states_lb=[0; 0; 0; 0; -0.980; 0; 0*(2*pi
      /360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0; 0*(2*
      pi/360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0;
      0*(2*pi/360); 0];
78 initial_states_ub=[0; 0; 0; 0; -0.900; 0; 0*(2*pi
      /360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0; 0*(2*
      pi/360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0;
      0*(2*pi/360); 0];
79 initial_states_ig=[0; 0; 0; 0; -0.940; 0; 0*(2*pi
      /360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0; 0*(2*
      pi/360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0;
      0*(2*pi/360); 0];
80 %fixed final states
81 final_states_lb=[0; 0; -1; 0; -0.940; 0; 0*(2*pi
      /360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0; 0*(2*
      pi/360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0;
      0*(2*pi/360); 0];
82 final_states_ub=[0; 0; -1; 0; -0.900; 0; 0*(2*pi
      /360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0; 0*(2*
      pi/360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0;
      0*(2*pi/360); 0];
83 final_states_ig=[0; 0; -1; 0; -0.940; 0; 0*(2*pi
      /360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0; 0*(2*
      pi/360); 0; 0*(2*pi/360); 0; 0*(2*pi/360); 0;
      0*(2*pi/360); 0];
84
85 %state bounds during the movement
86 omega_limit = (pi/2)/0.5; % 90   in 0.5 sec
87 v_limit = 1/0.5; % 1 meter in 0.5 seconds
88
89 states_lb=[-5*(2*pi/360); -omega_limit; -2; -v_limit
      ; -2; -v_limit; -pi/2; -omega_limit; -120*(2*pi
      /360); -omega_limit; -120*(2*pi/360); -
      omega_limit; -120*(2*pi/360); -omega_limit;
      -120*(2*pi/360); -omega_limit; -pi/2; -
      omega_limit; -pi/2; -omega_limit];
90 states_ub=[pi/2;   omega_limit;  0.2;  v_limit;
```

```
   0.2;  v_limit;  pi/2;  omega_limit;  10*(2*pi
   /360);   omega_limit;  120*(2*pi/360);
   omega_limit;  120*(2*pi/360);  omega_limit;
   10*(2*pi/360);   omega_limit;  pi/2; omega_limit;
      pi/2;  omega_limit];
states_ig=zeros(nq*2,1);
%control bounds
controls_lb=-1000*ones(ncT,1);
controls_ub= 1000*ones(ncT,1);
controls_ig=zeros(ncT,1);
%acceleration bounds (implicit form)
accelerations_lb=-500*ones(nq,1);
accelerations_ub= 500*ones(nq,1);
accelerations_ig= zeros(nq,1);

% Control discretization
N = 20; % number of control intervals (20 intervals/
   second reasonable)
```