Co-design of biped walkers

Ibai Musatadi, Daniel Gil, Kerman Dañobeitia, Unai Amenabar and Shirin Oshtoudan
Universiteit Gent, Ibai.MusatadiIrazabal, Daniel.GilPorta, Kerman.Danobeitia, Unai.AmenabarArambarri, Shirin.Oshtoudan@UGent.be

***Abstract* – This paper discusses the design of a biped walker, aiming to enhance its walking efficiency and stability. The study was carried out in different parts: in the kinematic analysis phase, the motion of the biped walker is examined to understand its joint angles, velocities, and positions during walking. Afterwards, the energy consumption of the model is analysed to identify areas of inefficiency and potential improvements. Finally, the step optimization stage aims to minimize the input energy with the objective of achieving a more efficient and stable walking pattern. To do so, different optimization methods were tried, and the most optimal one has been chosen.**

INTRODUCTION

On this report, the procedure of a co-design of biped walkers' model creation is explained. A simplified version of a biped walker, more precisely a 5-link model with two links to model each leg and a fifth one for the hip, as depicted on Fig. 1.

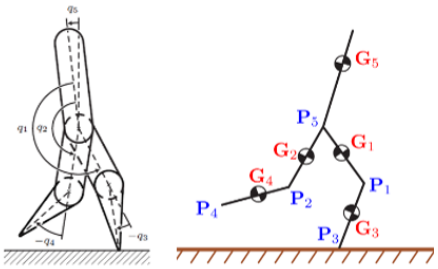

FIGURE 1
BODY ANGLES AND GRAVITY CENTRES DEFINITION

The main goal of the project is to generate a model of a 5-link body, being able to predict the motion of the biped walker at each time instance. Moreover, a step has to be optimized for minimal input toque at each joint. Similarly, it is also aimed to optimize the tibia-femur ratio to further minimize input torques.

On the modelling apart, the system kinematics in single-stance configuration have been calculated. Followed by the continuous dynamics using the Euler-Lagrange method to obtain the equations of motion. Lastly, the heel-strike impact has been added to the model, representing the instantaneous change of the stance leg and the forces generated at that instance.

A visual model of the biped walker has been generated in two different ways; a 3D model using Simulink and a simpler 2D representation of the mathematical model. This visualisation model helps with the verifications, showing the results of the simulations in a more visual way.

Once checked the model, an optimization routine has been applied to calculate the best trajectory for maximum energetic efficiency. Moreover, the tibia-femur length ratio has also been optimized for it.

MODELLING

In this part of the project, the dynamics of the biped walker are modelled and simulated afterwards to proof the generated model is correct.

I. Kinematics

The first step for modelling the biped walker is determining the kinematics. To ease the calculations, the absolute angles of each part of the body have been calculated based on relative ones, resulting in angles θ_1 to θ_5.

Once having the absolute angles, the position of each joint and gravity centre are calculated using forwards kinematics. This is done applying the principle of translational plus rotational matrix transformation, i being the index of the body part.

$$r_i = \sum_{i=0}^{n} \begin{pmatrix} 1 & 0 & l_{i,x} \\ 0 & 1 & l_{i,y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_i & \sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

The only static point of the body is the P3, depicted in Fig. 1, which represents the foot that is in contact with the ground. Therefore, P3 has been set up as origin point (0,0), starting the kinematic calculations from it. Next figure, Fig. 2, shows a figure with an example of the kinematic chain, points (o) show the position of the gravity centres of each link.

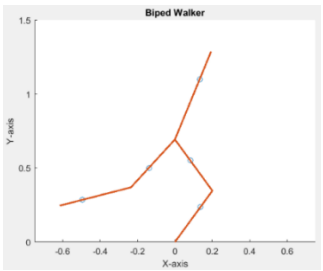

FIGURE 2
PLOT OF THE BODY FOR STEP POSITION

The 2nd figure plot helps verifying before calculation of linear and angular velocity equations. Programs themselves have differential function implemented, `sp.diff()` in *Python*, or `diff` in *MATLAB* calculate automatically the velocities.

II. Single-stance continuous dynamics

Once having the kinematic model defined, next step consists of implementing the biped walker's motion energy analysis. Lagrangian equations have been used to calculate the motion of every link. It is determined by the kinetic and potential energy equations.

Kinetic energy equation:

$$T = T_l + T_r = \frac{1}{2}\sum_{n=1}^{5} m_n \cdot \left\|\dot{G_n}\right\|^2 + \frac{1}{2}\sum_{n=1}^{5} I_n \cdot \dot{\theta}^2 \qquad (2)$$

Where:

- m_n : mass of each element.
- $\left\|\dot{G_n}\right\|$: velocity of each gravity centre.
- I_n : inertia.

And the potential energy equation:

$$V = \sum_{n=1}^{5} m_n \cdot g \cdot e_y^T \cdot G_n \qquad (3)$$

Where:

- g: gravity
- $e_y^T \cdot G_n$: height of each gravity centre with respect to the floor.

The Lagrangian equation is calculated subtracting the potential energy to the kinetic energy.

$$L = T - V \qquad (4)$$

Deriving the Lagrangian respect to each relative angle (q_i) and its velocity ($\dot{q}_i$), the equations of motion are determined:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} \qquad (5)$$

From these equations, the mass matrix (M) and the vector matrix (C) are obtained. The vector matrix is calculated by substituting the acceleration terms of the equations of motion with zeros; and $M \cdot e_i$ is obtained substituting the angular speeds and gravity terms with zeros.

$$\ddot{q}_i = e_i = 0 \rightarrow C \qquad (6)$$

$$\dot{q}_i = 0 \text{ and } g = 0 \rightarrow M \cdot e_i \qquad (7)$$

Each mode shape's effect is determined individually to get the mass matrix (M). Each $e_i$is set to zero except the one whose effect is determined. For instance, to obtain the first element of the mass matrix, $e_1$is set to 1 and $e_{2:5}$ to null for the first element in the vector $M \cdot e_i$.

Afterwards, the input torques are introduced (u_1 to u_5). However, the problem states that the hip is not controlled ($u_5 = 0$), so a control matrix B is generated such that:

$$\mathrm{B}u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ 0 \end{bmatrix} \qquad (8)$$

The equation of motion for the biped walker is then:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) - Bu = 0 \qquad (9)$$

This way, the state space model is:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \qquad (10)$$

$$\dot{x} = f_{ss}(x, u) = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M(q)^{-1}\left(\mathrm{B}u - C(q, \dot{q})\right) \end{bmatrix} \qquad (11)$$

III. Solver method

To calculate dynamics, a solver is implemented to determine the motion of the body for a specified time span.

Until this point, *MATLAB* was used as main software but when trying to implement an `ode45` solver (Runge-Kutta 4(5) method), the type of variables that the team was using to do the calculations seemed incompatible with the solver. As a solution was not found, it was decided to switch the entire code to a Python environment.

In Python, the main ODE solver is `odeint`. Unlike *MATLAB*, the default method of the solver for the ordinary differential equations in python is "Adams/BDF switching method". This method estimates the values in between the determined time step for calculation (calculation of the integral), resulting in a much smoother final function. Therefore, this method is very computationally demanding, meaning that it takes quite some time to run any simulation.

For the developed code, the `odeint` solver takes 15 minutes for a 1 second simulation. This happens because the inner step size of the solver is something around $10^{-5}[s]$.

That being the case, it was decided to change the solver method to another one with a lower computation requirement, enough for the application: the Runge-Kutta method with 4th order. This method only evaluates state at the instances that are specified (step-size determined by user). The state is evaluated as such:

$$k_i = f(x_n + h\sum_{j=1}^{i-1} a_{i,j}\, k_j, t_n + c_i h) \qquad (12)$$

Where:

- j: order,
- h: step size
- x_n: state at time n
- $a_{i,j}$: weight of each evaluation.

As the state is evaluated at different time instances for the same given previous state x_n, the step size of the solver should be carefully chosen. If too high, the solver may run into very high values that do not represent the system accurately. A step-size of 0.02 seconds is enough in this system.

A free fall scenario has been defined, to verify the dynamics. All inputs are set null, and an arbitrary initial position is given to the body. Consequently, the body starts to fall correctly, effect of the gravity.

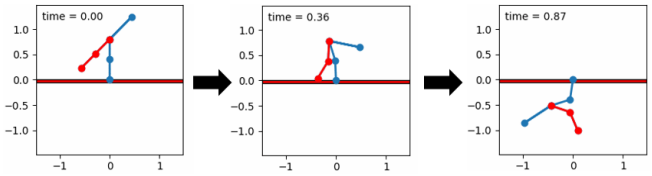

FIGURE 3
GRAVITY COLLAPSE

IV. Heel-strike impact model

The last thing to model in the Biped Walker is the instance where the support foot is changed. The heel-strike impact model governs the relation between the generalised coordinates and velocities before and after impact. In that moment, the impact of the foot with the ground must be calculated and also the change of foot is modelled. For this last, it is enough to generate a matrix so that the angles in the equations of motion are changed:

$$\theta_1 \leftrightarrow \theta_2 \text{ and } \theta_3 \leftrightarrow \theta_4 \tag{13}$$

Assuming that the change of support foot happens instantaneously at time 0 and it does not change the configurations of the motion of the biped walker, the state space can be determined as:

$$q^+ = \lim_{t \uparrow 0} x(t) \text{ and } q^- = \lim_{t \downarrow 0} x(t) \tag{14}$$

That leaves for the relation of generalized momentum before and after the impact,

$$M(q^+)\dot{q}^+ = M(q^-)\dot{q}^- \tag{15}$$

Which implies the impact model is inelastic and results in a sudden decrease of kinetic energy. Note that the vector matrix needs to be changed as well, but in this case, it is enough with the change of angles and does not need to meet any equality.

$$C(q^-) \rightarrow C(q^+) \tag{16}$$

With that, all required changes are applied. And the verification simulation is performed.

SIMULATION

I. *Simulink Simscape*

At start, a 3D model of the Biped Walker was generated in *MATLAB*. It had a 3D visualisation but in fact all the internal parameters and calculations were held in 2D. More precisely, Simulink-Simscape toolbox, which can be used to import 3D CAD models and simulate its motion with a block chain model, automatically generated by the *MATLAB* software when importing. The goal of the model was to link the equations of motion with a real simulation of the 3D model.

Apart from the visualisation of the model, this allows to make the calculations itself. So, the simulation created has a double purpose, on one hand visualisation of the mathematical model and on the other hand, verify the values calculated with the ones measured in Simulink.

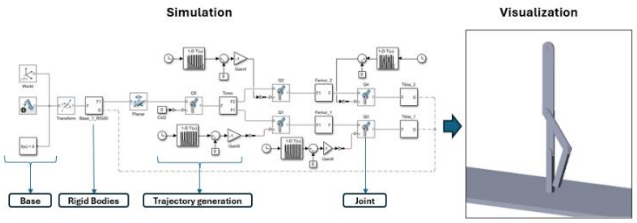

FIGURE 4
SIMULINK MODEL

First, the 5-link biped walker was created in SolidWorks, containing all the information of the system: sizes, masses, inertias… and exported to Simulink. On Simulink all the elements and its connections are divided into blocks and gives the option to control it directly. So, the trajectory of motion can be implemented on the simulation and visualize it.

So, this way, creating a trajectory and implementing on the simulation, the mathematical model has been verified. Each of the joints, apart from controlling it, measures its position, velocity, acceleration and torques.

Moreover, on the simulation also the impact model has been added to it. In each of the corners of the upper side of the tibia, the part that contacts with the floor, a contact block has been added. This block simulates the contact between two solid bodies. But at this point, we realized that simulation used much more complex internal calculations, ending on different results. Due to this reason, this simulation was not developed more, only making use of the simple animation to verify the results.

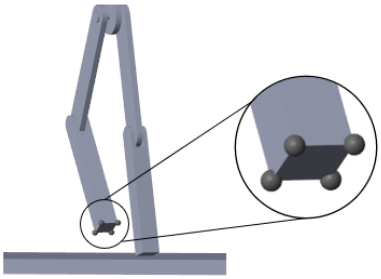

FIGURE 5
DETAIL FOR FOOT SUPPORT POINTS

II. Python

However, as the team was not able to make the *MATLAB* script's `ode45` solver work, as explained before, the entire code was translated to Python environment.

To perform the necessary simulation, a loop is generated, which is defined to simulate a step at each time. When the solver detects that the free foot has a zero or negative height, it ends the simulation process, applies the necessary changes for the heel strike impact model, and starts solving again for the next step. Again, when it detects that the new free foot touches the ground, the solver ends.

To verify the simulation, the body is started in an arbitrary position shown in the left side of Fig. 5 and inputs are set to zero. By doing this, the body is again collapsing due to gravity, but when it senses that it touches the ground, a sudden decrease in kinetic energy happens, and the base foot is changed, resulting in a bounce like motion of the body. Fig 5 shows the initial state (left), instance where the free foot touches the ground (middle), and the position the body takes after 0.2 seconds after heel strike impact (right).

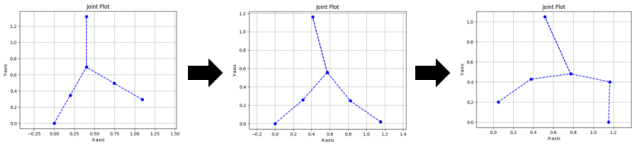

FIGURE 6
SIMULATION OF TWO STEPS

Additionally, an animation of the body has been made in GIF format that shows the entire motion of the body while doing two steps. Furthermore, the free foot is colored red at each time, showcasing that the change of base foot is done correctly.

ENERGY OPTIMIZATION

I. *Configuration*

Once verified that the model without any input works, the next step is to define an initial (x_0) and end state (x_T).

$$x_0 = \begin{pmatrix} q_1(0) \\ \vdots \\ q_5(0) \\ \dot{q_1}(0) \\ \vdots \\ \dot{q_5}(0) \end{pmatrix} \text{ and } x_T = \begin{pmatrix} q_1(T) \\ \vdots \\ q_5(T) \\ \dot{q_1}(T) \\ \vdots \\ \dot{q_5}(T) \end{pmatrix} \quad (17)$$

The goal is to generate a set of input torques (u_1 to u_4) that can move the body from the initial state to the end state within a user defined duration. In addition, the generated input torques must be minimized to cost function:

$$\min_{x(t),u(t)} \int_0^T \frac{1}{2} \|u\|^2 \quad (18)$$

- T: duration of the step.

For the boundary conditions, the problem states that the step must be symmetric (see Fig 7).

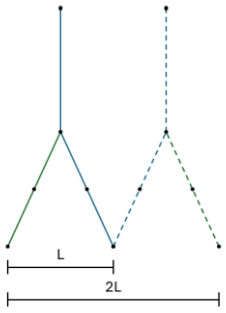

FIGURE 7
OPTIMAL STEP INITIAL AND END STATES

The step length is on this case $2L$, determined by the user. According to Danion et al. (2003), an average person's walking step (stride) speed is approximately 1.12m/s, with a step frequency of 0.92Hz. These values have been set as walking conditions, defining the default stride length and duration from them.

$$L = V_{step} * f_{step} \quad (19)$$

$$T = 1/f_{step} \quad (20)$$

But only the step length and duration are not enough to fully determine the initial and end state, the angular velocities also need to be defined ($\dot{q_1}$ to $\dot{q_5}$). In this case, as for the stride characteristics, there is no exact values, conditions are set: at initial state. The hip joint, P5, absolute vertical velocity must be positive, and at the end state negative.

Although the previous mentioned boundary conditions are the ones that are stated by the problem, some additional constraints have been applied to limit the optimization problem:

- Rotation of each joint: to really mimic the motion of a human body, the joint angles must be limited. According to Hyodo et al. (2017):
  - The rotation of the femur with respect the trunk is limited between 101º and 0º.
  - The rotation of the tibia with respect the femur is has a range of 149º.
- Head movement: for a comfortable walking experience, the head cannot be swinging from one side to the other and must be as stable as possible, so $-10^o < q_5 < 10^o$.

II. *Solver method*

To solve the optimization problem, the *Sympy* library is no longer useful because it cannot handle with these kinds of problems. Therefore, another library is used: CasADi. CasADi, shorten for "Computer Algebra System for Differentiation and Integration," is a powerful open-source software framework designed for numerical optimization and algorithmic differentiation. Developed primarily for solving nonlinear optimization problems with differential algebraic equations (DAEs). It provides efficient and scalable solutions to complex optimization problems.

The way CasADi solver works is described by Fig.8. First, a state space model of the desired simulated object must be provided. In this case, the equation of motion for each joint. Then, the decision variables are defined, which are the necessary variables to perform the optimization problem. Finally, to start solving, the constraints and an initial guess need to be defined. Constraints limit the simulation variables to certain ranges or exact values and the initial guess provides the solver with a first set of what the solution may be. Usually, when solving such a problem, the initial guess

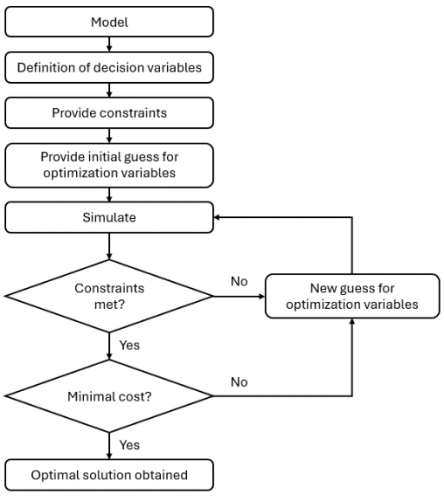

FIGURE 8
CASADI SOLVER METHODOLOGY

When dealing with nonlinear problems such as the motion of a humanoid, CasADi uses the multiple shooting method to calculate the appropriate input series.

The multiple shooting method is a numerical technique commonly used to solve optimal control problems. It breaks down the time interval of the problem into smaller subintervals, called shooting intervals. Each shooting interval is associated with a set of decision variables representing the control inputs and state variables at the interval's boundaries. The method then formulates a set of boundary value problems (BVP) for each shooting interval, where the goal is to enforce continuity between neighboring intervals and satisfy the system dynamics and constraints.

By iteratively solving these BVPs and adjusting the decision variables, the method converges towards a solution that optimizes the given objective function while respecting the system dynamics and constraints. The multiple shooting method is widely used due to its effectiveness in handling complex optimal control problems with nonlinear dynamics and constraints.

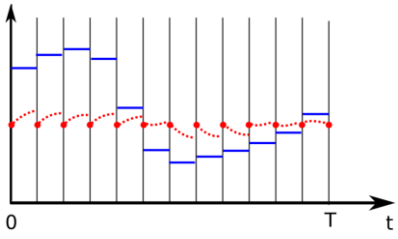

FIGURE 9
MULTIPLE SHOOTING METHOD

In addition, to use the Casadi solver, the entire model must be done with functions and variables provided by the CasADi library. Meaning that the previously obtained motions at the verification stage of the modelling part, with mass matrix and vector matrix substitution method is no longer useful. However, the symbolically obtained mass matrix and vector matrix can be hand copied to a CasADi format, avoiding the entire calculation of the derivatives, and minimizing the learning curve for the CasADi library. This has mainly been decided due to the lack of knowledge of the team about CasADi and the suffered setback of changing the modelling environment from MATLAB to Python.

Hand copying the mass matrix and vector matrix is still a very carefully executed task but can easily be checked by comparing the written code with the output of the symbolically obtained matrices.

Furthermore, the optimization problem is a very computationally demanding task. Therefore, the running code optimization is also an important aspect to work on. This has been made modifying different aspects. Firstly, lower the Runge-Kutta order from 4 to 3. This fastens the calculation, even if the robustness is decreased. As drawback, the step size needs to be smaller, increasing the computational demand. For that reason, a balanced relation between the order and number of steps have been set.

Similarly, the Casadi internal solver tolerance has been increased significantly, from e^{-6} to e^{-2}, leaving more freedom to the set constraints. Finally, the program's code has been simplified. All this lead on a huge decrease of calculation time, allowing to explore a wider range of initial guesses and constraints. Once the results converged to what seemed the optimal solution, the tolerance is decreased to get the final solution.

III. *Results*

To come up with the optimal solution, several tryouts had to be performed using different configurations of constraints and initial guesses. Each constraint affects the calculation in different ways, while initial guesses help converge faster to the solution. So, it is essential to get both correctly.

Although it is not a direct result from the simulation, it has been observed that when solving an optimization problem, the model must have an adequate number of constraints. For example, if the initial and final state are set as equality constraints for the biped walker, the solver is never able to find an optimal solution, as it is considered as a hard constraint, which cannot be met.

Moreover, the initial guess applied to the solver is also a key aspect to ease computations. For that reason, after the first obtained solution, the next simulations have always the previous input and state set as initial guess. Consequently, constraints and boundary conditions have been set in different sections. Verifying the correct motion of the simulated body for each set of constraints and initial guess.

In the starting case, it is tried to find a path while setting an equality constraint for the initial and final position. In this case, as the constraint where to hard, the model has never able to find optimal scenario. Furthermore, the solver concludes that the optimum movement is doing a pendulum motion going underground, as depicted in the next sequence, Fig. 9, which is not possible.

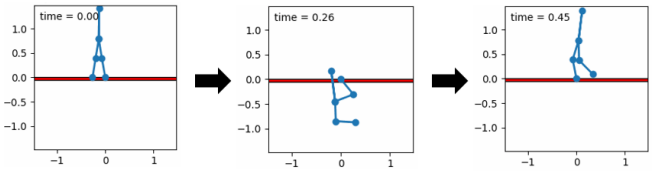

FIGURE 10
FIRST GUESS OF AN OPTIMAL STEP WITHOUT CONSTRAINTS

As the solver is not able to converge to an optimal solution with the first initial set of constraints, the obtained input torques are not being analyzed.

For the next guess, it is decided to leave more freedom for the initial and final state of the body. So, instead of defining the entire state, the position of free foot is the only defined point.

- Initial conditions:
  - $P_5 = [-L, 0]$
  - $\dot{P_5} = [\dot{e_{5,x}} > 0,\ \dot{e_{5,y}} > 0]$
- Final conditions
  - $P_5 = [L, 0]$
  - $\dot{P_5} = [\dot{e_{5,x}} > 0,\ \dot{e_{5,y}} < 0]$

For the next simulation, an absolute positive height has been defined for each point of the body during the entire step duration. In addition, the free foot must go in a positive direction of x during the simulation. These are the obtained results:

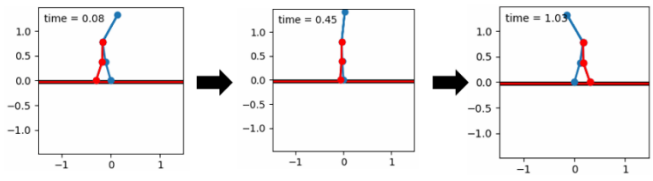

FIGURE 11
SECOND GUESS OF AN OPTIMAL STEP

This motion has a cost of $3.431 * 10^{-5}$, meaning that the solver body is able to go from one point to other with minimal effort. This happens due to the starting velocity of the joints, which help to the desired motion. Fig. 11 shows the evolution of each input torque through the step.

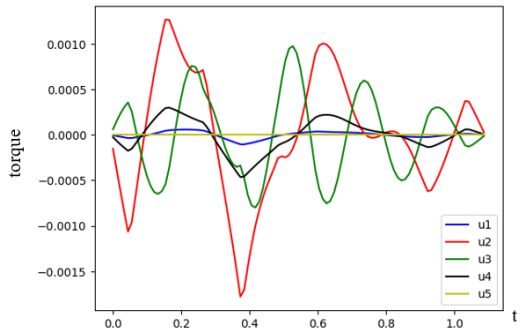

FIGURE 12
INPUT TORQUE EVOLUTION OF SECOND GUESS

Although this is the optimal input torque value according to the simulation, the obtained motion is not feasible for a human. Walking with such a swinging motion of the back is not realistic. Therefore, despite expecting higher final cost, it is decided to limit the rotation of the trunk.

For the same reason, the joint rotation angles are also limited to the values according to the first section of this chapter. And once these constraints are set, the final simulation is performed:

This last simulation shows the result of the motion when all desired constraints are met. As with the first set of constraints, the free foot goes as low to the ground as possible. This happens because maintaining the foot low, the potential energy of the leg is minimal, minimizing the required input torque to keep the body above the ground.

The final motion simulation results therefore in the following torque graph:

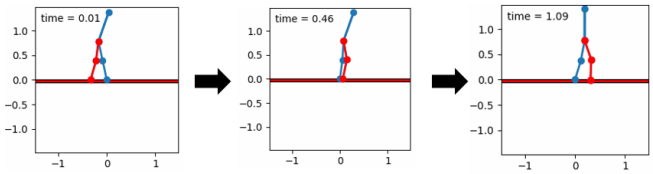

FIGURE 13
FINAL GUESS OF AN OPTIMAL STEP

In contrast with the previous simulation, the resulting torques are significantly higher. As the body is trying to control the stability of the trunk without controlling the hip, the other joints must compensate for equilibrium. However, the main reason to have higher torque is that the initial velocity of all body parts is controlled. This means that the trunk's movement cannot trigger a pendulum motion of the legs that results in a nearly null torque requirement. Consequently, the cost increases by more than 10^{10}%, to a value of 25919,51 and the local values increase by 10^{4}%.

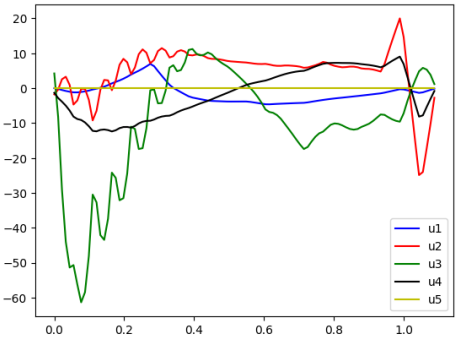

FIGURE 14
INPUT TORQUE EVOLUTION OF FINAL GUESS

CO-OPTIMIZATION

Co-optimization, the simultaneous optimization of multiple interconnected systems or processes, plays a pivotal role in addressing complex real-world problems across various domains. By considering the interactions and dependencies between interconnected components, co-optimization enables more holistic and efficient solutions than traditional isolated optimization approaches.

I. *Configuration*

In the case of this project, to further minimize the input torques required, the tibia/femur ratio may be optimized. Until this point in this ratio has been a fixed parameter, but from this point on, the tibia/femur ratio parameter (θ) is added to the optimization problem.

$$\theta = \frac{l_{\theta,f}}{l_{\theta,t}} \tag{21}$$

The tibia/femur ratio does not only change the length of these components; the mass, gravity center position and inertia, also change with respect to θ:

- Mass:

$$m_{\theta,i} = \frac{m_{0,i}}{l_{0,i}} * l_{\theta,i} \tag{22}$$

- Gravity center

$$r_{\theta,i} = \frac{r_{0,i}}{l_{0,i}} * l_{\theta,i} \tag{23}$$

- Inertia. The tibia and the femur are assumed to be rod that rotate in one of their ends.

$$I_{\theta,i} = \frac{1}{3} * m_{\theta,i} * {l_{\theta,i}}^2 \tag{24}$$

To obtain an optimized θ, the team has opted to just introduce the tibia/femur ratio as an added optimization variable. That way, the model will have three optimization parameters: state (X), input torques (U) and θ. The cost function keeps being the same.

This is called simultaneous approach, so the direct multiple shooting formalism is extended with the model parameters. This way, the sparsity of the corresponding nonlinear program (after transcription of the continuous problem) is hampered resulting in increased computation time.

II. *Results*

For the calculation of the optimal tibia/femur ratio, the simulation for the optimal input torque is used as base. It is basically the same, the only difference between them is that in the new simulation, all physical parameters are defined with respect the tibia/femur length ratio. The ratio is set as an optimization variable whose value will be calculated to obtain the minimum cost.

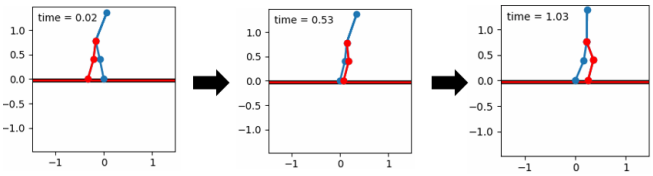

FIGURE 15
SOLUTION FOR MOTION OF CO-OPTIMIZATION PROBLEM

The previous sequence shows that the body has nearly the same motion as in the final guess with fixed bone length (section III in Energy Optimization). However, it does results in a slight change of the joint angles due to the modified tibia/femur ratio.

By doing the simultaneous approach the optimal tibia/femur ratio obtained is 0.47. Resulting in the length of the femur and tibia:

- $l_{f,opt} = 0.376[m]$
- $l_{t,opt} = 0.424[m]$

However, in reality, a human body has a ratio of something around 0.7, according to own measurements. This may mean that there is some kind of error in the model or that the constraints that are being applied do not hold for a real human body.

All in all, the new tibia/femur ratio minimizes the total cost to 21525, which a reduction of 17% compared to the situation where the same constraints are applied with fixed bone lengths. This highlights how a minor change in the design can affect the final efficiency of an application. This being a consequence of the interconnectivity of the parameters and the relation between them.

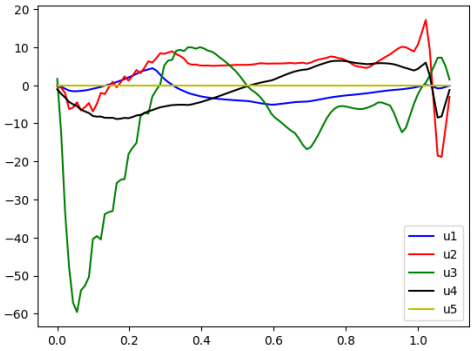

FIGURE 15
EVOLUTION OF INPUT TORQUES IN CO-OPTIMIZATION PROBLEM

CONCLUSION

The most important aspect to perform the project has been the configuration of the Casadi solver when solving the optimization problem. The most important aspects of it, such as the initial guesses for the optimization variables and the applied constraints play a pivotal role in the running duration and success of the simulation.

A good initial guess leads to a significant decrease of the iterations as the initial data set may already be close to the actual solution. Moreover, there may be multiple optimization variables, which can be related to each other. A

good mesh at the initial guess is another aspect than can lead to a decrease in iteration. For example, when doing an initial guess for the state, the input torque should go along with change in state.

Also, equality constraints are usually a very limiting factor to the simulation. The model has less freedom to explore situations that may be a better solution for the final state that is desired. Therefore, it is better to avoid equality constraints and applying inequality ones.

REFERENCES

[1] F. Danion, E. Varraine, M. Bonnard, J. Pailhous. August 2003. “Stride variability in human gait: the effect of stride frequency and stride length”. *Gait & Posture (18)*, pp. 69 - 77.

[2] Kashitaro Hyodo, Tadashi Masuda, Junya Aizawa, Tetsuya Jinno, and Sadao Morita. April 2017. “Hip, knee, and ankle kinematics during activities of daily living: a cross-sectional study”. *Brazilian journal of physical therapy*.

[3] S Bhardwaj, A A Khan and M Muzammil. 2019. “Knee torque estimation in sit to stand transfer”. *Journal of Physics: Conference Series*.