Reactive Walking for the Humanoid Robot Pyrène

Louise Scherrer

Louise SCHERRER

Biomedical Engineering (BIOMED)
2019-2020

LAAS-CNRS
7 avenue de Colonel ROCHE
BP 54200
31031 Toulouse Cedex 4
FRANCE

Reactive Walking for the Humanoid Robot Pyrène

- Marche robot réactive pour le robot humanoïde Pyrène -

from 10/02/2020 to 07/08/2020

Confidentiality : no

Under the supervision of:

- **Company supervisor: Olivier, STASSE, ostasse@laas.fr**
  Present at the defense : yes

- **Phelma Tutor: Franz, BRUCKERT, franz.bruckert@grenoble-inp.fr**

Ecole nationale
supérieure de physique,
électronique, matériaux

Phelma
Bât. Grenoble INP - Minatec
3 Parvis Louis Néel - CS 50257
F-38016 Grenoble Cedex 01

Tél +33 (0)4 56 52 91 00
Fax +33 (0)4 56 52 91 03

http://phelma.grenoble-inp.fr

Contents

List of Figures

Glossary

actuator (in the case of the present system) a mechanism generating motion from a source of energy and a command. For example, a servomotor is an actuator. 10

admittance control an admittance controller takes a force as input (measured, desired or the difference between the two) and imposes a position in output, based on the dynamical relationship that expresses motion resulting from force. 19

CoM Center of Mass. 10

CoP Center of Pressure. 35

DCM Divergent Component of Motion. 20

DoF Degree of Freedom. 8

DSP Double Support Phase. 12

end-effector The controlled target: it can be a hand equipped with a tool for example, or in this case the feet of the robot. 22

FF Free-Flyer joint, representation of the position and orientation of the base of a robot with respect to its environment. 9

Inverse Dynamics represents the methods allowing to compute the forces and momentum that have to be applied to a system so as to obtain the desired motion. The computation requires the modelling of the dynamics of the system. 13

Inverse Kinematics represents the set of methods computing the configuration of the joints of an articular model so as to obtain a desired position and orientation of the component of interest, for example a desired position of the center of mass of the system. In Robotics, it can be used to compute the commanded articular angles that should be transmitted to the motors of the robot so as to respect objectives (for example the position of the hand or feet of a humanoid robot). Motion is obtained from geometrical considerations only. 13

LIPM Linear Inverted Pendulum Model. 11

MPC Model Predictive Control. 17

NMPC Non-linear Model Predictive Control. 13, 23

PG Pattern Generator. 13

position control type of automatic control for which the desired behaviour of the system (the input commands) are given in terms of positions. 19

QP Quadratic Program. 34

RNEA Recursive Newton Euler Algorithm. 17

SoT Stack of Tasks. 13

SSP Single Support Phase. 12

WPG Walking Pattern Generator. 13

ZMP Zero Moment Point. 11

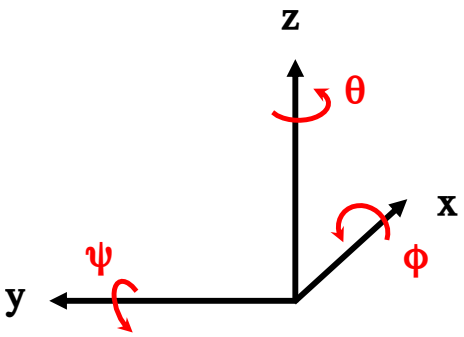


Figure 1: *Axis and rotation convention used throughout the report. ϕ is the roll angle, ψ is the pitch angle and θ is the yaw angle*

1 Introduction

Human-like automatons capable of mimicking the human motions have been imagined for centuries. Leonardo da Vinci has worked on a mechanical knight in armor in the late 15^{th} century. Some very advanced automatons have been built as early as in the 18^{th} century. For example, *la musicienne* was built by Pierre Jaquet-Droz and was capable of playing on a custom-built instrument with very accurate human-like motions. Apart from the mechanical feat that such systems represent, as well as their entertainment potential, the idea of a mechanical companion, capable of freeing humans from certain tasks, has been one motive for consequent research in what has become the field of humanoid robotics.

An impressive and pioneering humanoid robot is the ASIMO walking robot by Honda, that was unveiled in 2000 and capable of dynamic walking, as well as navigating in autonomy in a room, recognizing people and places... Shuuji Kajita explains in his book [Kajita et al., 2009] that humanoid robots present the advantages of being intuitive to interact with (compared to non anthropomorphic robots) and are well adapted to human environments. This should facilitate their integration into facilities designed for humans: biped walking for different grounds and staircases, hands for tool handling and door opening.

Trying to replicate human motions on robots is also a very powerful tool in order to better understand human locomotion as well, which implies that this field of Robotics has important links with some medical fields dealing with walking impairment and rehabilitation technologies for example.

However, despite two decades of progress in humanoid robotics and biped locomotion research since ASIMO was presented, the design and control of such robots for stable walking remains a complex challenge.

1.1 Presentation of the laboratory and context of the internship

1.1.1 Laboratory LAAS-CNRS and its Robotics Department

The LAAS-CNRS, *Laboratory for analysis and architecture of systems* depends on the CNRS (Centre national de la recherche scientifique) and is located in Toulouse (France). Founded in 1967 and currently led by Liviu Nicu, its research activities are organized around four main departments: Information Technology, Automatics, Robotics and Micro and nano technologies. Research is conducted in 26 teams.

The Robotics department is composed of three teams: Gepetto, RAP and RIS with complementary fields of expertise. RIS (for Robotics and its Interactions) is specialized in the architecture and control of robots and drones and their interactions with their environment and the people surrounding them. RAP (for Robotics, Action, Perception) focuses on percep-

Figure 2: *Humanoid robot HRP-2 from the Intelligent Systems Research Institute in Japan*

tion (from the detection and tracking to the identification and interpretation) of data from the environment and the integration of both sensors and algorithms in robotics systems. Finally, Gepetto (the team in which the internship presented in this report has been realized) focuses on the planning, motion generation and control of anthropomorphic robots and biomechanical systems. The Robotics department works on diverse robots, including manipulator arms, 4-wheeled robots and humanoid robots. Many projects are designed with industrial partners, including, for example, Airbus.

1.1.2 Gepetto team

The Gepetto team, founded in 2006 and currently led by Philippe Souères, is specialized in motion generation for anthropomorphic systems, and is now recognized as one of the leading teams in humanoid robotics. Research is conducted with an interdisciplinary approach, considering the robot numerical model, its real counterpart and the human body. The team handles the three levels of fundamental research (theoretical developments in mathematics, dynamics, automatic control...), integration (software packaging of the fundamental results developed, following an open-source objective) and application (distribution of the said software) in various systems, including modelling systems and humanoid robots. The team has developed a powerful control scheme, including a walking motion generator for the humanoid robot HRP-2 (visible on Figure 2). Since 2017, a second humanoid robot called Pyrène is available in the lab (see Figure 3). Gradually, the results on legged locomotion obtained on HRP-2 are being transferred on this robot, and the current research is conducted on it. Part of this research is developed in the context of a project with Airbus called Rob4Fam.

1.1.3 Rob4Fam Project

The Robots For the Future of Aircraft Manufacturing is a joint laboratory created in 2019 between the LAAS-CNRS and the Airbus Advanced Manufacturing Department of Toulouse. It is aiming at transferring the knowledge developed in Gepetto to the Airbus Robotics department, on two platforms: Pyrène and Tiago (a wheeled robot with one arm). My internship has been part of this project.

1.1.4 Presentation of the Talos humanoid robot

The robot that is the subject of this report is the first version of the Talos robots – named Pyrène – developed in collaboration between the company PAL-Robotics (based in Spain) and the Gepetto team at LAAS-CNRS. It can be seen on Figure 3. It can be noted that, being the first of his series, Pyrène is a scientific flagship. It is thus not intended for a direct application. Designed to resemble closely the human figure and abilities, it measures 1.75 m high for 95 kg.

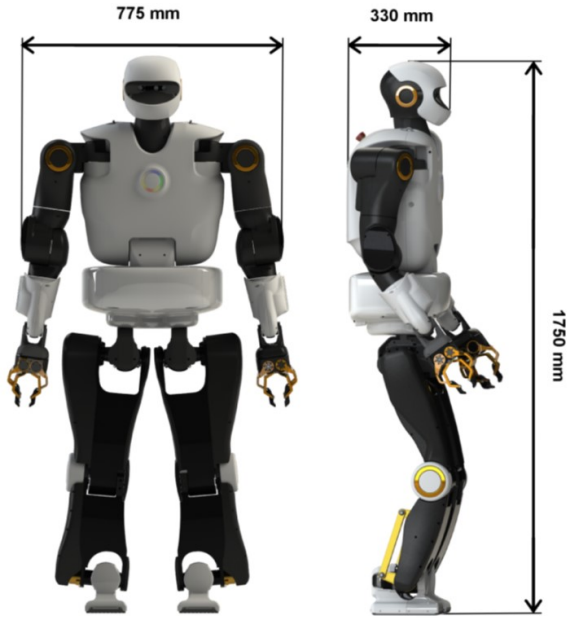


Figure 3: *Humanoid robot Pyrène - Talos model from PAL Robotics*

32 joints actuated with servomotors allow its range of movements (see Figure 4). The joints positioning allow the robot to reproduce human abilities closely. Each joint allows rotation around one axis, providing a total of 32 actuated Degrees of Freedom (abbreviated DoF) to the robot (Stasse et al., 2017). In terms of sensors, an encoder on the joint allows to measure its articular position θ_{joint}, and a torque sensor allows to measure the torque exerted on the joint. The ground reaction forces on the feet of the robot can be measured via torque sensors placed at the ankle level. An Inertial Measurement Unit (IMU) is also available to estimate its position and speed.

The robot features also batteries and an embedded computer and can thus operate in autonomy.

1.2 Internship presentation

A walking control scheme had already been developed and successfully tested by the team on the previous humanoid robot HRP-2, validating a powerful walking pattern generator fast enough for real-time computation, and capable of automatically positioning its foot steps (application examples in Stasse et al., 2009 and Ramirez-Alpizar et al., 2016). On the new Pyrène robot though, being heavier than HRP-2 and the first prototype of its series (Talos model), diverse technical issues such as a flexibility at the hip level, make its walking unstable with the current walking pattern and control scheme. A stabilizer had thus been implemented and partly tested. The internship aimed at improving the current state of the control scheme by working on both the stabilizer and the walking pattern generator, including the validation of the control in real-time allowing the robot teleoperation.

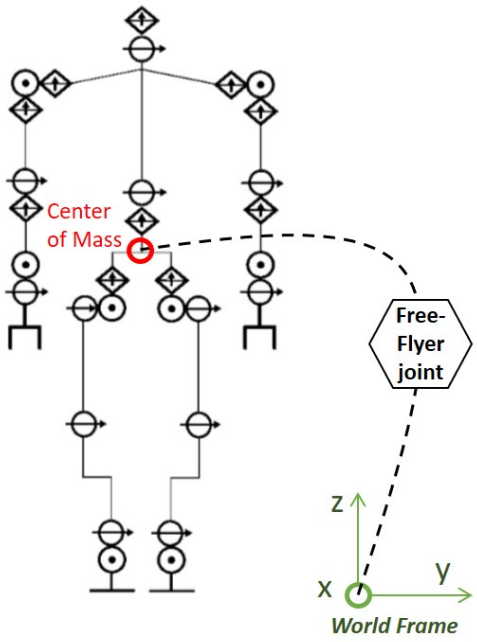


Figure 4: *Joint distribution on Pyrène – each of the robot's joints allows one Degree of Freedom, to which a special free-flyer joint is added, that links the robot to the world frame*

2 Context – State of the art at the beginning of the internship

A humanoid robot can be described as an articulated body, several rigid bodies linked together by actuated joints which provide the *actuated* degrees of freedoms. The total number of Degrees of Freedom of the robot corresponds to those of the actuated joints (see Figure 4), plus the position and orientation of the 'base' of the robot in the world frame (6 coordinates in total), taken at the level of the Center of Mass (CoM). These last 6 DoFs can be seen as an additional joint, named 'free-flyer' (FF) which links the robot to its environment. All those degrees of freedom can vary independently from one another. They are gathered into a vector $q = \begin{bmatrix} FF^{x,y,z,\phi,\psi,\theta} & \theta_{joint_1} & \theta_{joint_2} & ... \end{bmatrix}^\top$ called the configuration vector. The position of each body of the robot in space is uniquely identified given one value of q.

In the case of a position-controlled robot, the control scheme plans stable motions, that are transmitted to the robot via its configuration vector or *control vector*. Others means of control are possible (like torque control for instance), but in the case of this internship, Pyrène is controlled in position only. The equations described in the present report are thus written with this means of control in mind.

2.1 Walking for humanoid robots: the equations behind the movement

Humans achieve walking motions by applying forces on their environment. Intuitively, it is by pressing on the ground with the legs that the body can be moved forward. The balance is maintained via a careful distribution of our body weight over our contacts with the ground – balancing the arms or bending slightly for example. On the robot, this is achieved by planning a stable set of configurations for the actuators. The corresponding degrees of freedom represent the actuated part of the robot. Note that it is not possible to act directly on the free-flyer joint by acting on the actuators. Hence, the DoFs of the free-flyer joint represent the underactuated part of the robot.

The future values of the control vector (38 variables in the case of Pyrène) are planned over a time horizon (1.6 s here), discretized into 1ms time steps (that is a control at 1 kHz). At

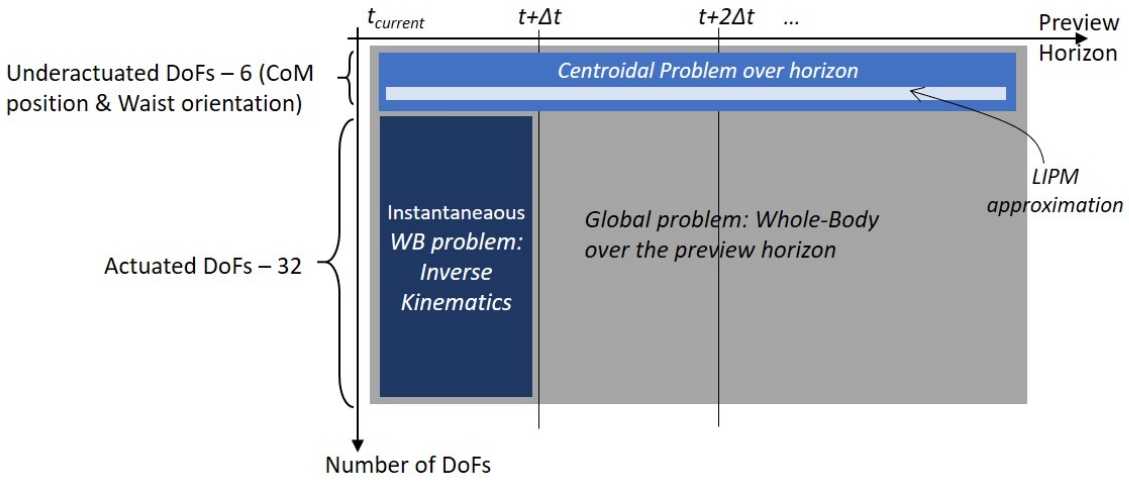


Figure 5: *Scheme of the complexity of the locomotion problem for humanoid robots, based on [Naveau, 2016]. Being able to solve the full problem in the grey rectangle (whole-body trajectory for the full preview future) is a long-run objective. For now though, this problem is impossible to solve in milliseconds. At this speed, it is however possible to solve the dark blue problem: whole-body problem but only for the nearest future, as well as the linearized problem at the level of the Center of Mass only, over the full 1.6 s preview.*

each time step, the future control vectors are updated over the time horizon, which involves solving a problem with close to 10 000 variables in less than 1ms. Moreover, the equations of motion are highly non-linear, resulting in the impossibility (as for now at least) to solve this global or 'Whole-Body' (WB) problem at this rate. The problem is thus separated into sub-parts, easier to solve (please refer to Figure 5) Naveau, 2016. The sub-problem consisting of the centroidal dynamics is described in the following paragraph. The whole-body problem is only solved instantaneously – that is for the very next timestep and is described in Section 2.1.3. The model for the contact forces on the feet of the robot is also part of the complexity of the locomotion problem and is presented in Section 2.1.2. Please note that in the scope of this internship, walking is only considered on flat ground.

In addition to those computational and modelling challenges, the construction of the real robot itself is a challenge on its own, requiring high quality servomotors, very accurate encoders and sensors. Ideally, the control schemes' robustness would allow to overcome the inaccuracies arousing from 'imperfect' robots. In the particular case of this internship, different control challenges have been observed for HRP-2 (on which the control scheme had been successfully implemented) and Pyrène. Those challenges include a different control frequency (HRP-2 used to be controlled at 200 Hz, instead of 1000 Hz for Pyrène), a flexibility at the hip level, and a different stabilizer. For each robot there is thus a different challenge in overcoming the simulation to reality gap, which prevents the direct application of a functional control scheme on a different robot.

2.1.1 Centroidal dynamics

The centroidal approach of the problem considers the dynamics at the level of the Center of Mass (CoM) of the robot, on which the entire dynamics is projected. (As such, it is not a simplification of the problem, but the CoM movement cannot be transmitted directly to the robot, which requires the use of a control scheme, in order to obtain a corresponding articular position for each actuator.)

Considering the robot as a rigid body, the Newton-Euler equations describe the relationship between the motion of its CoM and the forces acting on it:

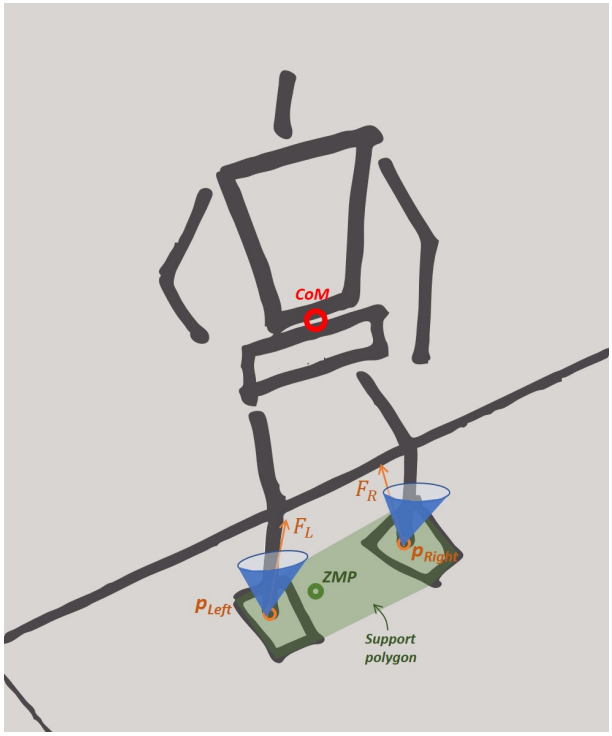


Figure 6: *Scheme of the contact forces expressed at points p_i of each foot and the associated contact wrench cones ; the convex polygon of contact and the ZMP lying in it*

$$\begin{cases} m(\ddot{c} - g) = \sum_i F_i \\ mc \times (\ddot{c} - g) + \dot{L} = p_i \times F_i \end{cases} \tag{1}$$

with m the total mass of the robot, c and $\ddot{c}$ the position and linear acceleration of the CoM respectively, $g = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^\top$ the gravity force vector and F_i the external forces acting on the robot, p_i the application point of the forces (p_{Left} and p_{Right} in Figure 6), and $\dot{L} = \omega \times I\omega$ the part of the angular momentum produced by the repartition of the robot's weight around its CoM. It can be computed from the angular velocity ω and inertia I of a body equivalent to the dynamics of the robot's kinematic tree (as detailed in Orin et al., 2013). (Equations expressed in the global Euclidian frame or World frame.)

The ground reaction forces exerted by each foot can be represented by one single force acting on the CoM from its application point on the floor. This point is here the barycenter of the contact points or center of pressure. This force and its moment about the CoM are referred to later as the *net contact wrench*. As representative of the application point of the contact forces, there is no moment induced by the forces on this point called the Zero Moment Point (ZMP) and noted p in the following equations.

The second equation is at the origin of the non-linearity of the system. In order to solve it and obtain a trajectory for the CoM, some simplifications are made, which linearize them. The model is then very similar to a linear inverted pendulum model or LIPM:

- $\dot{L}$ is neglected, that is, the moment induced by the limbs of the robot on its CoM is considered much smaller that that induced by the contact forces Stasse et al., 2018
- the robot's CoM movement is constrained to a horizontal plane, that is the height of the CoM is fixed Kajita et al., 2009

This allows to combine and rewrite equations (1) as follows (after simplification by m):

$$\begin{cases} -c_z\ddot{c_y} + c_y(\ddot{c_z} + g) = p_y(\ddot{c_z} + g) - p_z\ddot{c_y} \\ c_z\ddot{c_x} - c_x(\ddot{c_z} + g) = p_x(\ddot{c_z} + g) - p_z\ddot{c_x} \\ c_x\ddot{c_y} - c_y\ddot{c_x} = p_x\ddot{c_y} - p_y\ddot{c_x} \end{cases} \tag{2}$$

Those equations are not independent: the third one can be written as a linear combination of the first two ones. Knowing that the ZMP is on the floor, $p_z = 0$, and since the CoM is constrained to move on a horizontal plane, c_z is a constant and $\ddot{c_z} = 0$, the following linear relationship can be written between the CoM and the ZMP:

$$\begin{cases} p_x = c_x - \frac{c_z}{g}\ddot{c_x} \\ p_y = c_y - \frac{c_z}{g}\ddot{c_y} \\ p_z = 0 \end{cases} \tag{3}$$

Since the ZMP position is imposed by the contact forces, the CoM trajectory can be deduced from the ZMP trajectory, so as to respect the constraints on the contact forces in order to maintain the contact necessary for a balanced motion. The next paragraph details the constraints on the contact forces model.

2.1.2 Contact forces: the means of movement

Walking involves an alternance of phases, namely a Double Support Phase (DSP) during which both feet are in contact with the ground, and a Single Support Phase (SSP) during which only one foot is in contact with the ground. The succession of foot step positions and the alternance of phases allows to put the robot into motion, so as to follow a given reference CoM trajectory. The balance of the motion is guaranteed if the contact forces fulfill some constraints, expressed here in the unilateral contact model, described as follows, with the Coulomb friction laws:

- the contact forces are not able to pull solids together, only to push them. This results in the normal component of the contact forces being positive: $F^n_{contact} \geq 0$
- in order to have a non-sliding contact, the tangential component $F^t_{contact}$ must fulfill the inequality $\| F^t_{contact} \| \leq \mu F^n_{contact}$ with μ being the friction coefficient (set to 0.7 here). This limit of friction defines a cone (included in Figure 6) inside which $F_{contact}$ must lie.

The contact cone can be linearized by approximating it with a polyhedron, for easier computation. However, taking the contact cone formulation of the contact forces makes a costly computation. Another method to take the contact constraints into account, described by Wieber, 2002 and Caron et al., 2015, is to express the restrictions on the contact forces by limiting the position of the ZMP. Indeed, supposing a single foot in contact with the ground, the motion remains balanced if the weight of the body is maintained over the foot surface. This means that the ZMP lies somewhere on the surface of the foot. When standing on both feet, the body weight is distributed over both feet, meaning that the ZMP lies somewhere in between the feet. A *support polygon* can thus be defined such that if the ZMP lies in it, the contact is balanced. This gives the balance criteria for the robot's motion, as used in the current control scheme Naveau et al., 2017.

Note: this contact model generally leads to an impact force when the contact is set -and thus a discontinuity in the forces, that can be reduced by imposing that the contact be established with no speed at the foot level.

2.1.3 From centroidal trajectory to Whole-Body Motion

The whole-body approach considers the configuration of the whole robot. The whole-body motion is thus described in the space of configuration vectors. The robot's dynamics in the configuration's space can be written as follows:

$$M(\dot{q})\ddot{q} + N(q,\dot{q})\dot{q} + G(q) = \tau + J_c(q)^\top F_{contact}(t) \tag{4}$$

with $q = \begin{bmatrix} q_{FF} \\ q_{joint1} \\ q_{joint2} \\ \dots \\ q_{joint32} \end{bmatrix}$ the configuration vector composed of the free-flyer components

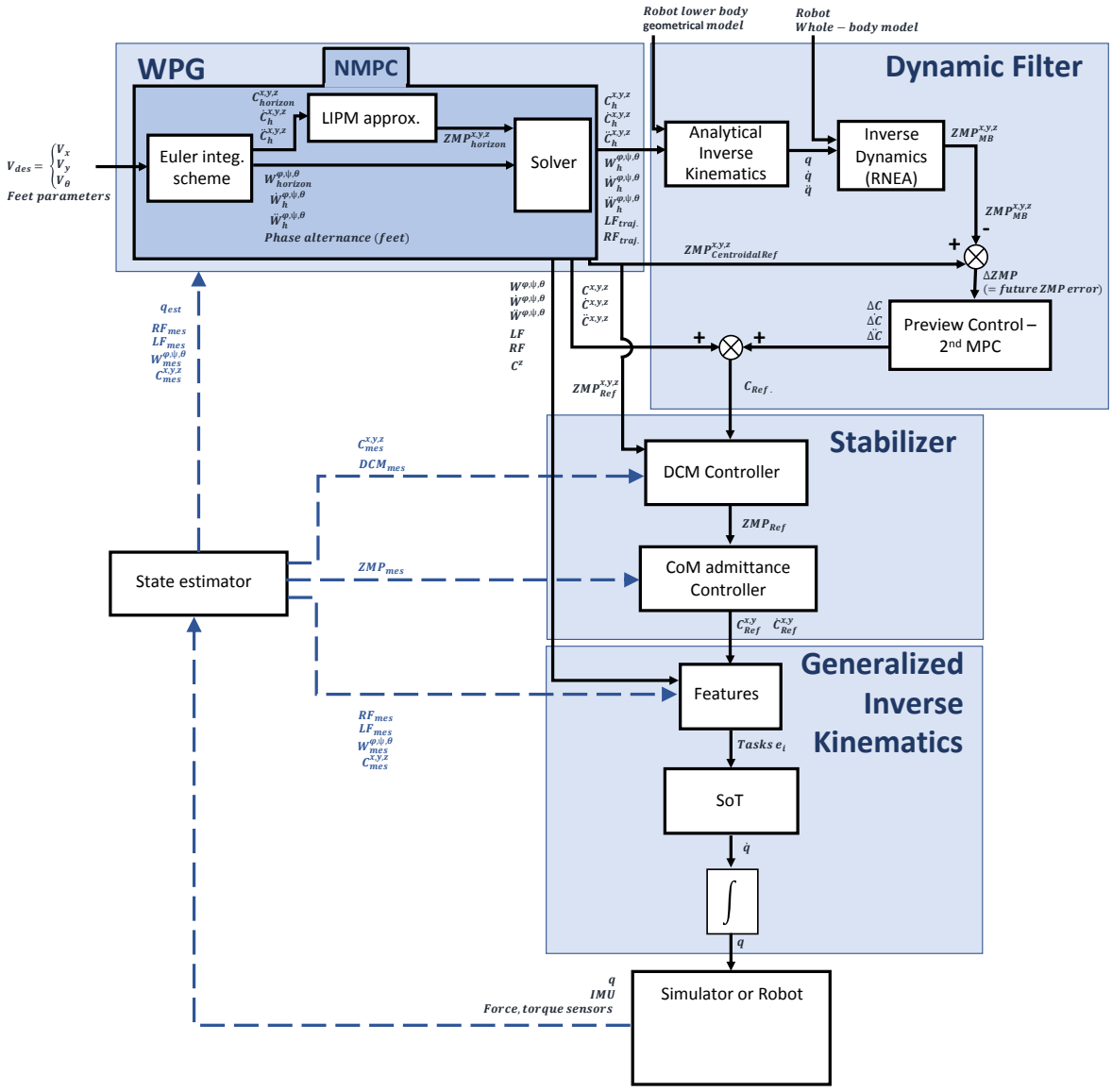
$q_{FF} = [x_{CoM}, y_{CoM}, z_{CoM}, \phi_{Waist}, \psi_{Waist}, \theta_{Waist}]^\top$ (corresponding to the CoM position and Waist orientation) and the articular position of the actuated joints. It is expressed here with Euler angles (the detailed axis and orientation scheme can be seen on Figure 1) but is generally implemented as a quaternion. $M(q)$ is the mass matrix corresponding to a generalized inertia matrix for articulated body systems [Featherstone, 2008], $G(q)$ the gravity force vector, $\tau = \begin{bmatrix} 0_{FF} & \tau_{Actuators} \end{bmatrix}^\top$ the actuators' torques, $F_{contact}$ the vector related to the contact forces, mapped to the configuration space via $J_c(q)^\top$, the contact Jacobian matrix (equal to $\frac{\partial F}{\partial q}$), and $N(q,\dot{q})$ the matrix gathering non-linear effects such as Coriolis and centrifugal forces.

In practise, this equation is not solved in order to retrieve the control vector q directly from it. This would represent an Inverse Dynamics computation, which is outside the scope of this training practice. (It is however the subject of the PhD thesis of Noëlie Ramuzat, PhD student in Gepetto.) Instead, this equation, describing the dynamics of the system, is taken into account when the centroidal problem is solved, so that its solutions (mainly trajectories for the CoM and the feet) are dynamically consistent. This is detailed in Section 2.2.2. Those trajectories are objectives (called Tasks) for the robot. Then, an Inverse Kinematics scheme computes the control law (configuration vector velocity $\dot{q}$), based on the said objectives. This Inverse Kinematics scheme is not computing an analytical solution, but is rather defined as an optimization problem: it is looking for the control law $\dot{q}$ that respects the Tasks at best. Inverse Kinematics has been used with success on the previous robot HRP-2. The software implementing this (called the Stack of Tasks or SoT) is described in the coming section.

2.2 Solving the equations: implementation of the control scheme

The general control scheme of the robot as implemented at the beginning of the internship can be seen on Figure 7. The problem described in the previous section can be efficiently solved using optimal control methods because such methods allow to directly handle constraints in the solving scheme. First, a Walking Pattern Generator (WPG or simply PG) uses a Non-linear Model Predictive Control (NMPC) to compute a reference trajectory for the Center of Mass, the waist orientation and the feet of the robot (staying consistent with the centroidal dynamics and desired direction and speed of motion for the robot). The NMPC is subsampled internally at 200 Hz, which represents a new trajectory planned over the entire horizon every 5 ms. This is described in Section 2.2.1. Even though the LIPM has been used successfully for stable walking motion, the torque induced by the limbs over the movement of the CoM is neglected. In the case of Pyrène, which weighs close to 100 kg, this leads to errors on the realised trajectory that are too strong to maintain a balanced walking. A filter has thus been developed and tested [Naveau et al., 2017] [Naveau, 2016] in order to take into account the influence of the whole body on the CoM trajectory and to correct it (2.2.2). This filter is the part of the control scheme taking into account the Whole-Body dynamics equation (4). Then the corrected CoM trajectory, along with the waist orientation, the feet trajectory and a general upper-body posture are used in the Inverse Kinematics scheme Stack of Tasks to compute the commanded configuration vector sent to the robot (2.2.3).

The stabilizer present on Figure 7 is presented in a separate section 2.3.



Parameters used

C: Center of Mass position
W: Waist orientation
ZMP: Zero Moment Point position
LF, **RF**: Left and Right Foot position
q: control vector
DCM: Divergent Component of Motion

mes stands for measured or estimated
h stands for horizon (values over the entire preview horizon)
Ref: reference or desired

IMU: Inertial Measurement Unit

Equations and implementation details

- Euler Integ. Scheme: eq. 5, Section 2.2.1
- LIPM approx.: eq. 3 (2.1.1), 5 to 7 (2.2.1)
- Solver: eq. 8-12, Section 2.2.1
- Analytical IK: eq. 13, Section 2.2.2
- ID: eq. 14, Section 2.2.2
- Generalized IK: eq. 15 to 21, Section 2.2.3

Figure 7: *Scheme of the Control scheme as implemented at the beginning of the internship. The user specifies V_{des} and feet parameters. Along with the measure of the current state of the robot, they are used to solve the locomotion problem and compute the control vector q. This control vector is sent either to the real robot or to a simulation. The state estimator is not studied here, details can be found in [Flayols et al., 2017]. The* sot-pattern-generator *package to which I contributed is wrapping the WPG and Dynamic Filter blocks (detailed in Section 3).*

2.2.1 Computing a CoM and feet trajectory: NMPC

The current state of the Walking Pattern Generator has been implemented and tested by Naveau et al., 2017: it takes as an input the desired speed of the robot (that is a vector $V_{des} = \begin{bmatrix} V_x & V_y & V_\theta \end{bmatrix}^\top$ with V_θ the angular speed about the vertical axis z), and the current state of the robot, and computes automatically the feet trajectories along with the CoM trajectory and Waist orientation. This strategy, introduced by Herdt et al., 2003, Herdt et al., 2010 and further developed by Naveau et al., 2017 is referred to as "Walking without thinking about it". Before that, the feet positions had to be given in advance and the CoM trajectory computed from them. The addition of the automatic orientation of the foot steps makes the problem non-linear.

Ideally, the equations of motion would be solved over the infinite future. However, as explained in 2.1, the whole-body problem is very complex and only solved instantaneously. As a result, the NMPC computes the waist and feet trajectories over a finite time horizon. Indeed, the influence of the future states for the current one has been studied by Kajita et al., 2003, with the conclusion that 75% of the influence of the future steps is included in the first $0.8s$ of future, in the case of a control period of 5ms for the Model Predictive Control. This means that, in order to plan precisely the coming $0.8s$, double that amount of preview is needed. The preview window is thus chosen to be $1.6s$. The equations are discretized over the time horizon $n.T$ divided into n equal timesteps T, t_k being the present time and t_{k+i} with $i \in \{1; ...; n\}$ the future timesteps over the horizon.

In order to be feasible, the CoM linear acceleration and the angular acceleration of the waist have to be continuous, which means that the jerk has to be at least piecewise constant. The centroidal model of the robot can be seen as a rigid body around the CoM (as defined in Section 2.1.1). Attaching a frame to it allows to define the position and orientation of the robot with a single variable $\mathbb{C}^{x,y,z,\phi,\psi,\theta}$ which corresponds to the CoM position and waist orientation. Considering the LIPM approximation and a piecewise constance jerk $\dddot{\mathbb{C}}(t) = \dddot{\mathbb{C}}_k$ for $t \in [t_k; t_{k+1}]$, the future free-flyer states can be computed using a Euler integration scheme:

$$
\begin{aligned}
\ddot{\mathbb{C}}_{k+1} &= T\dddot{\mathbb{C}}_k + \ddot{\mathbb{C}}_k \\
\dot{\mathbb{C}}_{k+1} &= \frac{T^2}{2}\dddot{\mathbb{C}}_k + T\ddot{\mathbb{C}}_k + \dot{\mathbb{C}}_k \\
\mathbb{C}_{k+1} &= \frac{T^3}{6}\dddot{\mathbb{C}}_k + \frac{T^2}{2}\ddot{\mathbb{C}}_k + T\dot{\mathbb{C}}_k + \mathbb{C}_k
\end{aligned}
$$

which can be gathered into the form:

$$
\begin{pmatrix} \mathbb{C}_{k+1} \\ \dot{\mathbb{C}}_{k+1} \\ \ddot{\mathbb{C}}_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \mathbb{C}_k + \begin{pmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{pmatrix} \dddot{\mathbb{C}}_k \tag{5}
$$

The future time steps $\mathbb{C}_{k+i}$ are then defined by recursion over the control horizon.

The LIPM approximation is then used to compute the ZMP trajectory over the control horizon, from the CoM position $\mathbb{C}^{x,y,z}$.

$$
ZMP^{x,y}_{k+i} = \begin{pmatrix} 1 & 0 & -\mathbb{C}^z/g \end{pmatrix} \mathbb{C}^{x,y}_{k+i} \tag{6}
$$

$$
ZMP^{z}_{k+i} = 0 \tag{7}
$$

The user defines a set of parameters for the feet: the desired time spent on single and double support phase in a step, how distant the feet should be from one another, the desired maximum height for the feet... This allows to define the contact phase for each time step over the control horizon.

The ZMP trajectory, the waist orientation (along with their respective speed and acceleration) and the desired alternance of phases are then given to the Non-linear Model Predictive Control as an equality constraint. The NMPC will then produce the reference feet trajectory LF_{traj} & RF_{traj}, the CoM trajectory and its derivatives $C^{x,y,z}$, the waist orientation and its derivatives $W^{\phi,\psi,\theta}$, and a reference ZMP trajectory $ZMP^{x,y,z}$.

The free variables (Fv) are the jerk of the current foot position and orientation, depending on the phase over the horizon: $\dddot{LF}^{x,y,\theta}$ or $\dddot{RF}^{x,y,\theta}$ (noted ${}^{L,R}F^{x,y,\theta}$); and the jerk of the CoM position $\dddot{C}^{x,y}$

The cost function is the following one:

$$\min_{Fv} \left[\frac{\alpha}{2} J_1(Fv) + \frac{\alpha}{2} J_2(Fv) + \frac{\beta}{2} J_3(Fv) + \frac{\gamma}{2} J_4(Fv) \right] \tag{8}$$

with J_1 and J_2 the linear and angular velocity tracking respectively, J_3 for maximum stability: keeping the ZMP position at the center of the foot sole, J_4 to minimize the CoM jerk:

$$J_1(Fv) = \| \dot{C}_h^x - V_{des}^x \|_2^2 + \| \dot{C}_h^y - V_{des}^y \|_2^2 \tag{9}$$

$$J_2(Fv) = \|^{L,R} F^{\theta} - \int V_{des}^{\theta} dt \|_2^2 \tag{10}$$

$$J_3(Fv) = \|^{L,R} F_h^x - ZMP_h^x \|_2^2 + \|^{L,R} F_h^y - ZMP_h^y \|_2^2 \tag{11}$$

$$J_4(Fv) = \| \ddot{C}_h^x \|_2^2 + \| \ddot{C}_h^y \|_2^2 \tag{12}$$

with $\| \, . \, \|_2$ the L_2 norm (Euclidian norm). The subscript h stands for horizon: the variables are considered on the entire preview.

Respecting the following constraints

- *Equality constraint*: the dynamic of the system has to be respected: the Euler integration scheme result and the LIPM
- 1^{st} *inequality constraint* – Balance: the ZMP has to remain in the support polygon, defined using the feet size and positions
- 2^{nd} *inequality constraint* – Foot step feasibility: given the geometry of the robot and the limits of the actuators, the area of possible feet landing is described geometrically and linearized by a convex polygon

The waist orientation (not included directly into the NMPC) and its derivatives are defined to follow the orientation of the feet (set at the average between the orientation of both feet).

2.2.2 Compensating the angular momentum about the CoM: Dynamic Filter

The Center of Mass trajectory is filtered before being sent to the Stack of Tasks which computes $\dot{q}$. This filter, called Dynamic Filter and developed by Naveau et al., 2017 (based on Kajita et al., 2003) allows to take into account the influence of the whole body over the CoM trajectory, which had been neglected so far by the LIPM. It is composed of the following units:

Analytical Inverse Kinematics. Using a geometrical model of the lower part of the robot (describing its bodies and joint relative positions) and the signals computed by the NMPC, an Analytical Inverse Kinematics algorithm is used to compute the corresponding configuration q of the robot.

$$q = IK(Model_{centro}, C^{x,y,z}, \dot{C}^{x,y,z}, \ddot{C}^{x,y,z}, W^{\phi,\psi,\theta}, \dot{W}^{\phi,\psi,\theta}, \ddot{W}^{\phi,\psi,\theta}, LF_{traj}, RF_{traj}) \tag{13}$$

It is then derivated twice by finite difference and q,$\dot{q}$,$\ddot{q}$ are sent to an Inverse Dynamics algorithm.

Inverse Dynamics. Using a Whole-body model of the robot and q and its derivatives, an Inverse Dynamics algorithm named Recursive Newton Euler Algorithm (RNEA) (as described in Featherstone, 2008) is used to compute the external torque that is required in order to get the robot in the given configuration

$$\tau_{ext} = ID(Model_{MB}, q, \dot{q}, \ddot{q}) \tag{14}$$

From τ_{ext}, the corresponding Multi-body ZMP_{MB} of the robot can be directly computed. The future error ΔZMP on the ZMP trajectory (that would be measured if the trajectories from the NMPC were sent directly to the robot) is then

$$\Delta ZMP = ZMP_{Ref} - ZMP_{MB}$$

with ZMP_{Ref} the ZMP reference trajectory computed by the NMPC.

Preview control: MPC. A second Model Predictive Control (MPC), developed by Kajita et al., 2003, uses the future error on the ZMP computed over the horizon in order to compute a correction ΔC on the CoM trajectory computed by the NMPC. That correction, added to the CoM trajectory, allows to obtain the desired ZMP_{Ref} – that is the one computed by the NMPC of the pattern generator.

This MPC is iterated only once, which does not guarantee convergence. However, it does converge in practise and thus produces a satisfactory correction of the CoM trajectory. (On Pyrène, which is heavier and for which the $\dot{L}$ term produces strong deviations – up to 5cm corrections, a second iteration might be required, but has not been used yet.)

Now that it is corrected, the new CoM reference trajectory, along with the feet trajectories, and waist orientation can be sent to the Generalized Inverse Kinematics scheme.

2.2.3 Whole-Body trajectory

The reference trajectories computed by the Walking Pattern Generator and filtered by the Dynamic Filter, are sent to the Inverse Kinematics scheme Stack of Tasks, developed in the team and described in Mansard and Chaumette, 2007 et Mansard et al., 2009. It computes the control law (configuration vector velocity $\dot{q}$) necessary to realize the desired CoM, feet ... positions, that are called the desired *features* and noted s^*. It is then integrated and the resulting configuration vector is sent to the robot.

The following features s_i are considered:

- CoM position $C^{x,y}$
- CoM height C^z (constant)
- Waist orientation $W^{\phi,\psi,\theta}$
- LF position and orientation $LF^{x,y,z,\phi,\psi,\theta}$
- RF position and orientation $RF^{x,y,z,\phi,\psi,\theta}$
- Upper-body configuration (torso, arms, head) – not studied in the context of this internship. It is set to a neutral configuration.

The difference between the current feature and the desired one is called a Task and noted e_i.

$$e_i = s_i - s_i^* \tag{15}$$

The task is accomplished when $\dot{e}_i$ reaches 0 (or as close as possible). A rate of progress in the task is imposed: exponential decay

$$\dot{e} = -\alpha e \tag{16}$$

with α being the control gain of the task, which can be constant, or adaptive in order to control the speed of convergence to the desired feature.

Then

$$\dot{e} = -\alpha(s - s^*) \tag{17}$$

It is assumed that, from the current configuration q of the robot, there is a function f_s at least differenciable, giving the corresponding features of interest s_i (for example the current CoM height):

$$\begin{array}{rccc} f_s & : & \mathcal{C} & \rightarrow & \mathcal{S} \\ & & q & \mapsto & s \end{array}$$

with $\mathcal{C}$ the configuration space and $\mathcal{S}$ the space of the features (for example the Special Euclidian group $SE(3)$). Thus s is linked to q with

$$\dot{s} = \frac{df_s}{dq}\frac{dq}{dt} = J_s\dot{q} \tag{18}$$

with J_s called the Jacobian of the feature. Similarly, the Jacobian of the task J_e can be defined and $\dot{e} = J_e\dot{q}$. $\dot{q}$ can be computed as

$$\dot{q} = J_e^{\dagger}\dot{e} \tag{19}$$

with $J_e^{\dagger}$ the pseudo-inverse of J_e (as introduced by Moore-Penrose), since J_e is usually not square and thus not invertible. This can be equivalently written at the solution to the following optimization problem:

$$\min_{\dot{q}} \parallel J_e\dot{q} - \dot{e} \parallel_2 \tag{20}$$

With multiple tasks, a first solving method consists in weighing the tasks (with λ_i as task weight) and solving the following optimization problem:

$$\min_{\dot{q}} \sum_{e_i} \lambda_i \parallel J_{e_i}\dot{q} - \dot{e}_i \parallel_2 \tag{21}$$

It corresponds to solving all tasks in parallel in the whole configuration space. A second method, which is used in the present case, solves a different representation of the problem with a system of hierarchy of tasks.

If all the tasks are compatible, then the optimization problem described by equation (21) computes the exact solution. Often though, the tasks can not be always fulfilled (for instance if the desired feet positions are further apart than what the robot's legs allow to reach). More importantly, realizing some tasks is paramount compared to others (for example the tasks directly linked to the stability of the robot should not be disturbed by other tasks). The tasks could be weighed, but in the case of the current problem, they are rather *hierarchized*. That is, the solving method ensures that "a task of lower priority cannot influence the realization of the task of higher importance". This is achieved by solving each task in the null space of the task of higher importance before it (please refer to Mansard and Chaumette, 2007 for details). This is possible because the robot is a redundant system: one configuration of the feet or the CoM for example (called the *end-effectors*) can be achieved with several different configurations. Thus a task does not need to act on all the degrees of freedom. Moreover, solving the tasks in a reduced space is what allows to bring the sampling time down to 1 ms in the Stack of Tasks.

The following hierarchy of tasks is used, from highest to lowest priority:

- Upper body posture (keeping it upright)
- Right foot position (later referred to as Contact Task on the right foot)
- Left foot position (Contact Task on the left foot)

- CoM height
- $C^{x,y}$ position
- Waist orientation

The result of the SoT computations, $\dot{q}$, is integrated and the desired configuration q is then ready to be sent to the robot. The lower-level controllers on the robot are in charge of reaching the desired q (this is not discussed in the scope of this internship). The control used here is a high gain position control mode which makes the robot very rigid.

The measure of the current features is what introduces a *feedback* term into the control scheme.

Note: the current feature is estimated by the robot, which introduces a measurement error ϵ in the SoT computations. s_i is in reality $\hat{s}_i = s_i + \epsilon$.

2.3 Stabilizer

The control scheme introduced so far assumes perfect knowledge of the robot's position, perfect models and a perfect realization of the desired motions by the robot. However,

- the robot models are not a perfect representation of the real robot,
- nor is the real robot performing exactly the intended motions,
- the equations of motion are neglecting some terms
- the floor is considered perfectly flat, though the real environment might present a slight slope or small obstacles
- the robot might have to cope with a push or some form of exterior force other than its foot contact forces, considering that it evolves amongst people
- the encoders measure what is considered to be the robot's position, yet the robot's bodies have some flexibility (which is not included in the models), and which the encoders do not measure

The objective of the stabilizer is that the robot be able to cope with all those sources of deviations from its reference behaviour by adding a feedback term, measuring how well the robot is performing the reference movements, and correcting reference trajectories in real time so as to account for the measured deviations.

There are three main strategies for stabilization:

- designing an action at the level of the torso (one example in [Takenaka et al., 2009])
- applying an ankle torque to reject the disturbance directly at the level of the end-effector, that is here, the foot
- re-positioning a foot step: when the disturbance is too large, the current feet positions can not be maintained and balance can only be recovered by adjusting the foot.

The stabilizer described here performs admittance control on the robot. It is based on the one described in [Caron et al., 2019], which has been successfully implemented on another humanoid robot (model HRP-4). It is added in-between the dynamic filter and the SoT (please refer to Figure 7) and it is based on some of the stabilization methods proposed by [Kajita et al., 2009] and used in [Kajita et al., 2001] and [Kajita et al., 2010]. It is based on the following principle: since the robot moves relatively to its environment by applying contact forces on it, the robot

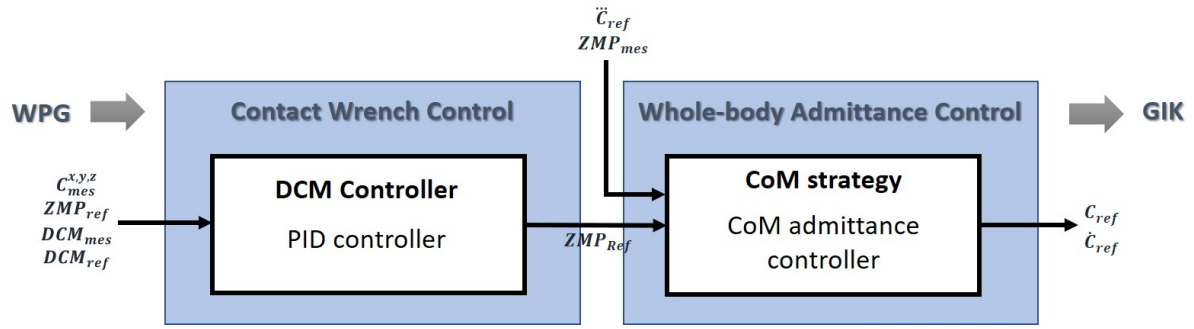


Figure 8: *Scheme of the stabilizer as implemented at the beginning of the internship. The contact wrench control uses a controller of the Divergent Component of Motion (DCM) to compute a new reference of the net contact wrench and of the ZMP trajectory. A CoM admittance controller then adapts the CoM trajectory accordingly.*

can recover its desired position by applying a contact wrench on the environment designed to compensate the measured deviations.

Its state at the beginning of the internship can be seen on Figure 8. The first unit implements a contact wrench control and is described in the following section. Then a whole-body admittance controller, described in Section 2.3.2, adapts the CoM reference trajectory with the result of the contact wrench control.

2.3.1 Contact wrench control by control of the Divergent Component of Motion

The net contact wrench (that is the contact wrench expressed at the level of the CoM) could be computed directly, or it can be computed indirectly by defining the ZMP position p that will produce the said contact wrench.

Stabilizing equation (3) rewritten below

$$\ddot{c}_{x,y} = \omega^2(c_{x,y} - p_{x,y}) \tag{22}$$

(with $\omega = \sqrt{g/c_z}$ the natural frequency of the inverted pendulum described by this equation) can be achieved by controlling both the position and velocity of the CoM. However, the introduction of a variable ξ called the Divergent Component of Motion (DCM) (by Pratt et al., 2006 and Takenaka et al., 2009) can simplify the said control. With

$$\xi = c + \frac{\dot{c}}{\omega} \tag{23}$$

combined to equation 22, a system of two coupled first-order equations can be written:

$$\dot{\xi} = \dot{c} + \omega(c - p) = \omega(\xi - p) \tag{24}$$

$$\dot{c} = \omega(\xi - c) \tag{25}$$

While ξ naturally diverges (equation 24), equation (25) shows that the CoM naturally converges to ξ. A controller on the DCM is thus sufficient to control the CoM deviations.

From the CoM reference trajectory provided by the WPG, the reference DCM can be defined as well. This reference, along with a measure of the DCM (based on the estimation of the base position of the robot) are used in a Proportional Integral controller:

$$\dot{\xi} = \dot{\xi}_{ref} + k_P(\xi_{ref} - \xi_{mes}) + k_I \int (\xi_{ref} - \xi_{mes}) \tag{26}$$

Injecting it in equation 24, the control on the DCM can be used to define a new ZMP reference trajectory from the one defined by the WPG:

$$p_{ref} = p_{PG} - [1 + \frac{k_P}{\omega}](\xi_{ref} - \xi_{mes}) - \frac{k_i}{\omega}\int(\xi_{ref} - \xi_{mes}) \tag{27}$$

Using the new ZMP reference trajectory, the equivalent net contact wrench $W = \begin{pmatrix} F \\ \tau \end{pmatrix}$ can be deduced from equation 22:

$$F = \begin{pmatrix} m\omega^2(c^x - ZMP^x_{ref}) \\ m\omega^2(c^y - ZMP^y_{ref}) \\ mg \end{pmatrix} \tag{28}$$

$$\tau = C \times F \tag{29}$$

The new reference ZMP can be used for CoM admittance control, and the net contact wrench computed from it can be used for end-effector admittance control (which was not implemented in the initial control scheme introduced in Figure 7).

Note: using in the controller the ZMP value that is intended to be sent to the robot (the WPG reference ZMP) represents adding a feedforward term to the controller.

2.3.2 Whole-Body admittance control: CoM strategy

Center of Mass admittance control is performed by adding an acceleration to the CoM with the following law on the CoM speed, using the reference CoM speed from the WPG:

$$\dot{c}_{ref} = \dot{c}_{PG} + \begin{bmatrix} A_{CoM} & 0 & 0 \\ 0 & A_{CoM} & 0 \end{bmatrix}(p_{mes} - p_{ref}) \tag{30}$$

$\dot{c}_{ref}$ defines the CoM velocity corrected based on the one provided by the WPG. The CoM is accelerated in order to bring the measured ZMP closer to the computed one. It is integrated and derived once to provide respectively the corrected CoM position and acceleration reference features in the Inverse Kinematics scheme.

The stabilizer thus defines a new CoM position and velocity reference (replacing the reference previously computed by the WPG) based on a feedback on the real performance of the robot, which are then used for the CoM task of the inverse kinematics entity. Already implemented and tested on the robot at the beginning of the internship, the stabilizer induces, on flat ground, modifications of about 2 to 5 mm in the x and y translation coefficients of the CoM reference trajectories.

2.4 Internship Objectives

Having detailed the current state of the robot control scheme, the internship objectives, briefly introduced in 1.2, are now presented in details:

On the WPG. Some issues remain on the WPG, resulting in lower performances than expected. The first objective of the internship was to develop a simulation allowing the control of the robot in real-time – that is with real-time (or online) generation of the trajectory – matching the desired direction and speed given by the user (the V_{des} vector at the entrance of the control scheme on Figure 7). The V_{des} vector would be computed via a teleoperation script (using keyboard control of the robot motions). An *off-line* version of the simulation was working at this point: the WPG computed trajectories stored in files, that were later read during the simulation using the stabilizer & SoT units. This objective amounted to build the *online* simulation, based on the off-line one. The teleoperation script could later be used on the real robot, adapted for the use of a joystick.

On the stabilizer. The stabilizer designed by Caron et al., 2019 involves an end-effector strategy working in parallel with the CoM strategy in the Whole-body admittance control. The end-effector strategy allows a correction of the feet position in order to produce the reference contact wrenches computed by the control scheme. This involves distributing the net contact wrench on each foot and adding an admittance control at the level of the ankles. At the beginning of the internship, this part of the stabilizer had been coded, but not tested. I was to incorporate it into the simulation, test it, and adapt it if needed to the control scheme of Pyrène (different than the one used in Caron et al., 2019). Ideally, the resulting stabilizer would be tested on the real robot.

2.5 General method

Since I was novice in the field of legged robotics, the first part of the internship had been designed so as to allow me to grow knowledge on both the state of the art and the tools and methods used in the team. It lasted about three months. Then, the work on the stabilizer followed. The detailed organization of the internship can be seen on the Gantt diagram (Figure 26).

Documentation has been an on-going process all along the internship, at first to discover the field, then to understand the implementation choices in the team and later in the internship for trouble-shooting purposes as well.

Meetings were frequently organized with the team members implicated in the work of the internship, in order to define partial objectives and provide help when needed. I could also benefit from help of some PhD students and technical support from some specialized team members.

During the lockdown period (March 15^{th} to July 20^{th}), work was conducted from home. Communication was handled via Riot (a web application using the real-time communication protocol Matrix) and short daily meetings were organized to discuss the daily agenda. That written communication tool was also supplemented by tools allowing screen-sharing and virtual white-boards.

The code is open-source and the team uses the version-control system Git, in order to keep track of the different versions of the tools and to be able to share it conveniently thanks to the online repository Github where all the code is saved. The softwares being in development, a test period follows the frequent updates, in order to ensure that the new pieces of code have the expected behaviour on all configurations. This regularly leads to important debugging sessions.

The implementation of code for the robot goes through three layers of test:

- the equations are checked first in order to ensure that they are dynamically consistent
- the simulation layer comes next: the code is then validated in simulation (where the robot and its environment are reproduced as close as possible to the real ones, including the way the robot 'feels' its environment – more dynamical modeling is taken into account). The internship was thus strongly simulation based. In simulation, a simulator called Gazebo (Koenig and Howard,) allows to watch it and check if its motions are the intended ones. Then, plotting the relevant variables allows to check more thoroughly the results.
- once the simulation has been validated, the code can be tested on the real robot, on which experiments are repeated several times for validation

As a general method, the new features are gradually integrated and tested independently from one another (as long as possible). When ready to be integrated into the main source code, the code is validated on other computers as well.

3 Achievements on the Walking Pattern Generator

Working on the WPG was a way to allow me to grow knowledge on the legged robotics problems, as well as on the team softwares. The work described in this section happened mainly during the first 3 months of internship.

Following the Section 3.1 dedicated to software presentation, the subsequent sections present my contribution towards the real-time teleoperation of Pyrène in simulation, starting with the shift from off-line to online simulation of the walk in Section 3.2 and the writing of the teleoperation script (Section 3.3). Those simulations allowed to discover an issue for the waist orientation of Pyrène (rotation around the vertical axis), for which a solution has been implemented (Section 3.4). Finally, Section 3.5 deals with the difficulties met and how they were overcome.

Please note that, later on, the development made on the stabilizer required more work on the WPG, which is not explained here, but rather in the Stabilizer section (4), given that the WPG was only modified then for the stabilizer's benefit and it did not provide improvements on the WPG directly.

3.1 Software presentation based on the simulation of interest

The implementation of the computation scheme described in the previous section involves:

- computation units (called *entities*) – for instance, the *Pattern Generator* (gathering together the walking pattern generator NMPC and the Dynamic Filter blocks) and the *CoM Admittance Controller* are defined in an entity each
- variables transmitted to and from entities (called *signals*) – for instance *Vref* and $C^{x,y,z}$

The team has been developing its own open-source framework, called *Stack-of-Tasks* to compute the signals. It is composed of several packages, coded in C++ for efficiency purposes, which implement the different entities.

The control scheme is interacting with either a simulated environment and robot or the real robot. In order to coordinate the computations of the Stack-of-Tasks and the real robot or the simulator, the middleware ROS is used. The open-source simulator Gazebo has been used for the simulations.

In this subsection, the software used in the context of this internship is presented, using as example of use, a simulation of the robot walking in simulation. In its state prior to the internship, the WPG was used off-line to compute trajectory files. Thus, the simulation does not include the Pattern Generator entity, it reads the trajectory files, stabilizes the CoM trajectory and sends all of them to the Inverse Kinematics scheme. The robot is simulated in Gazebo. ROS and the main packages involved in the control scheme are presented in the next paragraph. Then, the general simulation launching process is introduced. Finally, the structure of an entity is described and the global *graph of entities* of the simulation is given, which summarizes the software presented in this section.

ROS Middleware and dynamic-graph package. ROS (for Robot Operating System) is a framework designed to help the writing of robot software, allowing to gather libraries (such as solving, environment mapping libraries...) and tools (plotting, debugging tools...) thanks to a plumbing system, dealing with the communication between any of those libraries [Stasse, 2016]. Information is exchanged via normalized messages written on *topics*, that can be seen as post-its. They are written by computational *nodes* which *publish* them on their topics (just like one would write an item on a post-it). Another node, requiring the information on the said topic, would *subscribe* to it and get any new message as it is published.

The *dynamic-graph* package is part of the Stack-of-Tasks framework and is designed to handle the dependency of the variables in time and related to other signals. Its aim is to avoid the

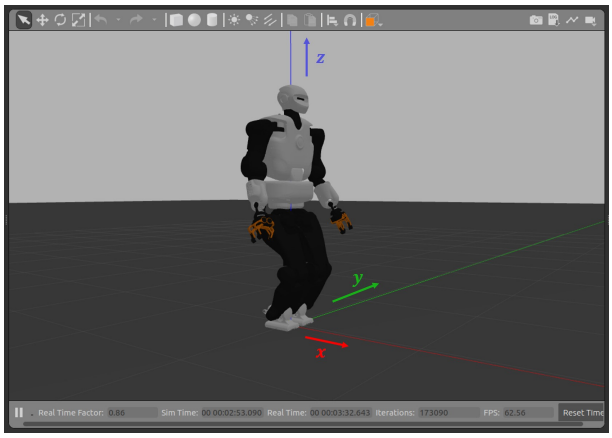


Figure 9: *View of Talos in Gazebo in a world with only a floor and gravity force – Talos is in half-sitting position, corresponding to its initial state and at the origin of the simulation world. The three directions of motions are respectively x – to the front, y – to its left, and z – to the top.*

unnecessary recomputation of data, if, for example, it has not changed since the last call, or if it is not used by any other entity. While ROS is handling the communication between the Python simulation script, the Stack-of-Tasks framework, Gazebo and other tools (plotting for example), the dynamic-graph package is handling the data exchange between the entities *inside* the Stack-of-Tasks framework.

General simulation launching process. For simplicity, the launching scripts and simulation scripts are written in Python.

- *Launching script.* Script gathering the following operations: ROS is started first, then the simulator Gazebo is launched, in which a simple world is simulated (simple ground and gravity force). A model of Talos and its low-level controllers are then added: the Stack-of-Tasks framework is started at this step. The only two entities existing at this time are the one representing the interface with the real or simulated robot (called PYRENE here), and an entity (called Robot Dynamics) reconstructing the position of the robot in the world, based on the configuration vector (corresponding to the initial position of the robot called half-sitting) and its derivative given by the entity representing the robot. The Gazebo interface can be seen on Figure 9.

- *Simulation script execution*: all the necessary entities are created and linked – *plugged* in the Stack-of-Tasks vocabulary – together with the appropriate signals. *dynamic-graph* links the entities together, the trajectory files are read, the control vector is computed and sent to the simulator: Pyrène walks in simulation.

- *User interaction*: the user can interact with the simulation during its execution through direct access to the signals, as well as by the use of tools working with ROS that allow to read the signals values and plot them in real time (for example the tool PlotJuggler was the main plotting tool used during the internship).

Main other packages of the Stack-of-Tasks used during the internship. Only the main packages used during the internship are presented.

- *sot-core*: main package of the Stack-of-Tasks, implementing mainly the solver and tools used for the Inverse Kinematics (tasks definitions, features, projectors...)

- *jrl-walkgen*: package providing the walking pattern generator algorithms, such as the NMPC and the Dynamic Filter [Stasse et al., 2008]
- *Pinocchio*: package implementing the rigid body algorithms described by [Featherstone, 2008], [Carpentier et al., 2019]
- *eiquadprog*: one package implementing quadratic optimization algorithms
- *sot-talos* and *talos_data*: specific packages for the modeling and control of the Pyrène robot

The packages to which I specifically contributed during the internship:

- *sot-pattern-generator*: package wrapping the *jrl-walkgen* package, synchronizing the computations of the WPG signals and gathering them into one single entity
- *sot-talos-balance*: package implementing the stabilizer

The Stack-of-Tasks packages also have dependencies on other packages. The way the framework was installed is the following one: though releases of the Stack-of-Tasks are made regularly, working on the source code of the packages implies to work on a version compiled from the sources directly, so as to be able to access them. The first weeks of the internship have thus been dedicated to the installation of the framework (Linux environment). In order to limit the risk of breaking significant portions of the source code in case of errors, environment chaining has been used. The Stack-of-Tasks packages and their dependencies have been installed in two separate locations on the computer. In a third location, ROS and Gazebo have been installed. Via a precise configuration of the environment variables, the three environments have been linked together. The installation process was provided by a new tutorial. This installation process has been repeated during the lockdown for which a special session was built on the computer.

Structure of an entity and Graph of Entities An entity is defined in a C++ source code file. This file usually defines:

- A class for the entity, such as *PatternGenerator*
- The list of input signals
- The list of output signals (along with the signals they might depend upon for their update)
- Any required computation on the input signals – be it an analytical formula (such as deducing the net contact wrench from the ZMP and CoM position) or an optimization problem (for example, computing the optimum wrench distribution among the feet)
- The definition of each output signals

During the execution of the simulation, the entities are linked together via their input and output signals, defining a graph of entities. The graph of entity for the off-line simulation is available on Figure 23. (For the sake of clarity, most entities have been renamed and thus do not correspond to the implemented names in the Stack-of-Tasks.)

Note: the robot simulated is 'perfect', which means for example that it has no flexibility.

3.2 From offline to online simulation

The simulation described in the previous section was the base for the one that was designed during the internship: an online simulation producing the trajectory variables in real-time, following a desired velocity for the robot (given by the user in real-time), instead of reading pre-computed files. The online simulation requires to insert the Pattern Generator entity (produced

Input Signals	Output Signals	Computations
Current positions for: ▪ Joint positions ▪ CoM ▪ Waist orientation ▪ Feet ▪ Desired Velocity V_{ref}	Reference trajectories for: ▪ CoM ▪ Waist orientation ▪ Feet ▪ CoM speed ▪ CoM acceleration ▪ Phase (now) Computed from the previous ones: ▪ DCM ▪ ZMP	Initialization of the state: Reading the current input signals, initializing the entity variables according to them Building of the robot model: Loading the robot model into a Pinocchio Robot - an object used to define a humanoid robot model later used by Pinocchio for computations Control Step: Considering the current desired velocity, running one step of the global WPG control loop using the data and model stored into the Pinocchio structure.

Figure 10: *Input, output signals and computations of the PatternGenerator entity*

by the package *sot-pattern-generator*) inside the graph of entity of the simulation, in replacement of the file reading entities. The Pattern Generator entity is described first on Figure 10.

The difficulty of that simulation lied in the understanding of the software architecture. The simulation itself, once the PatternGenerator entity had been inserted into the graph of entity, was quickly functional. The graph of entities is reproduced on Figure 13. The validation of the simulation was to be able to make Pyrène walk in simulation by modifying directly the desired speed variable, without inducing falls. The robot behaviour in simulation has then been tested in all directions (forward, backwards, to the left or right laterally and in a turn towards the left or right), with different speeds. The directions transitions have been tested as well, ensuring the ability of the PG to adapt as expected to the changes in trajectory (the automatic feet placement as well as the recomputation of the entire horizon every 5ms being key parts of this WPG). Figure 12 shows the reference trajectories computed in two different simulations. Time lapse images of the robot walking forward can be seen on Figure 11.

From the moment the new order in desired velocity is transmitted, the PG generates a trajectory to adapt to it and the feet start making their way towards the new direction $0.05s$ later, which corresponds to the flying foot being able to change its trajectory when in the air (this can be seen on the left panel of Figure 12: the new velocity order is sent at $7.94s$ and the flying foot starts shifting towards the required direction at time $8.00s$). When asked to stop, the PG is also able to generate the stop in two steps, as can be seen in the Figure 12.

It has also been checked that the following parameters, given to the WPG, are respected no matter the velocity order provided:

- the double support phase lasts $0.2s$ and the single support phase $1.0s$, which means that the robot lengthens its stride when an increased speed is commanded
- the feet are lifted from 5cm at their maximum height
- the linear velocity is bound by the pattern generator to 0.3 m/s
- auto-collision is correctly avoided

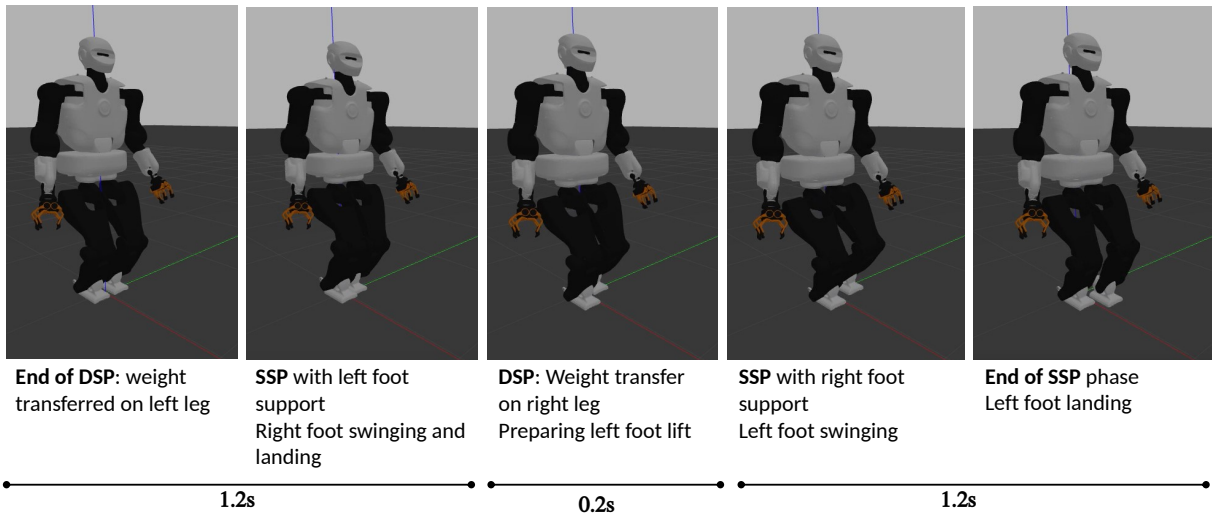


Figure 11: *Time lapse images of the robot executing two steps forward at* 0.1 *m/s speed* ($V_{des} = (0.1, 0.0, 0.0)$)*. DSP stands for Double Support Phase and SSP stands for Single Support Phase.*

- the maximum control value sent to the robot is not reached (which corresponds to the maximum intensity command to the servomotors)

An issue in rotation has however been unravelled: when given orders in rotation (parameter V_θ of V_{ref}), the robot turns its feet but not its waist, resulting, in a few steps, in the impossibility to turn further. A Control Manager unit stops the simulation (and freezes the control) as a result of the impossibility to reach the given reference values. This simulation thus allowed to check the linear walking abilities of the robot in real-time in simulation, and to discover an issue on the transmission or the computation of the waist orientation parameter.

3.3 Script for keyboard teleoperation

In parallel with the writing of the online simulation and the research on the causes of the waist rotation issue, a script has been designed in order to allow the teleoperation of Pyrène via the keyboard. This script has been designed to be adapted later for a joystick control of the real robot. This task was meant as a practise work on ROS, since it is handling the interaction between the user via its keyboard and the robot in simulation.

ROS Publisher/Subscriber principle. In the teleoperation case, the user keystrokes must be detected and communicated to the PatternGenerator entity of the online simulation. To that end, two Nodes (code units connected to ROS) named *pyrene_teleop* and *teleop_listener*, and one topic called *cmd_vel* have been created. When the *pyrene_teleop* detects the user keystroke, it transforms it into a velocity order for the robot and publishes it on the topic *cmd_vel*. The node *teleop_listener* receives the messages published on the topic and transmits it to the pattern generator entity.

Implementation. The Publisher node and topic have been written in a Python script that is meant to be launched in parallel with the simulation. It includes the detection of the keyboard strokes. This is the part that has been designed to be independent from the function taking action on it (updating the velocity orders), so that it can later be easily adapted to a teleoperation using a joystick.

The Subscriber node has been added to the online simulation script.

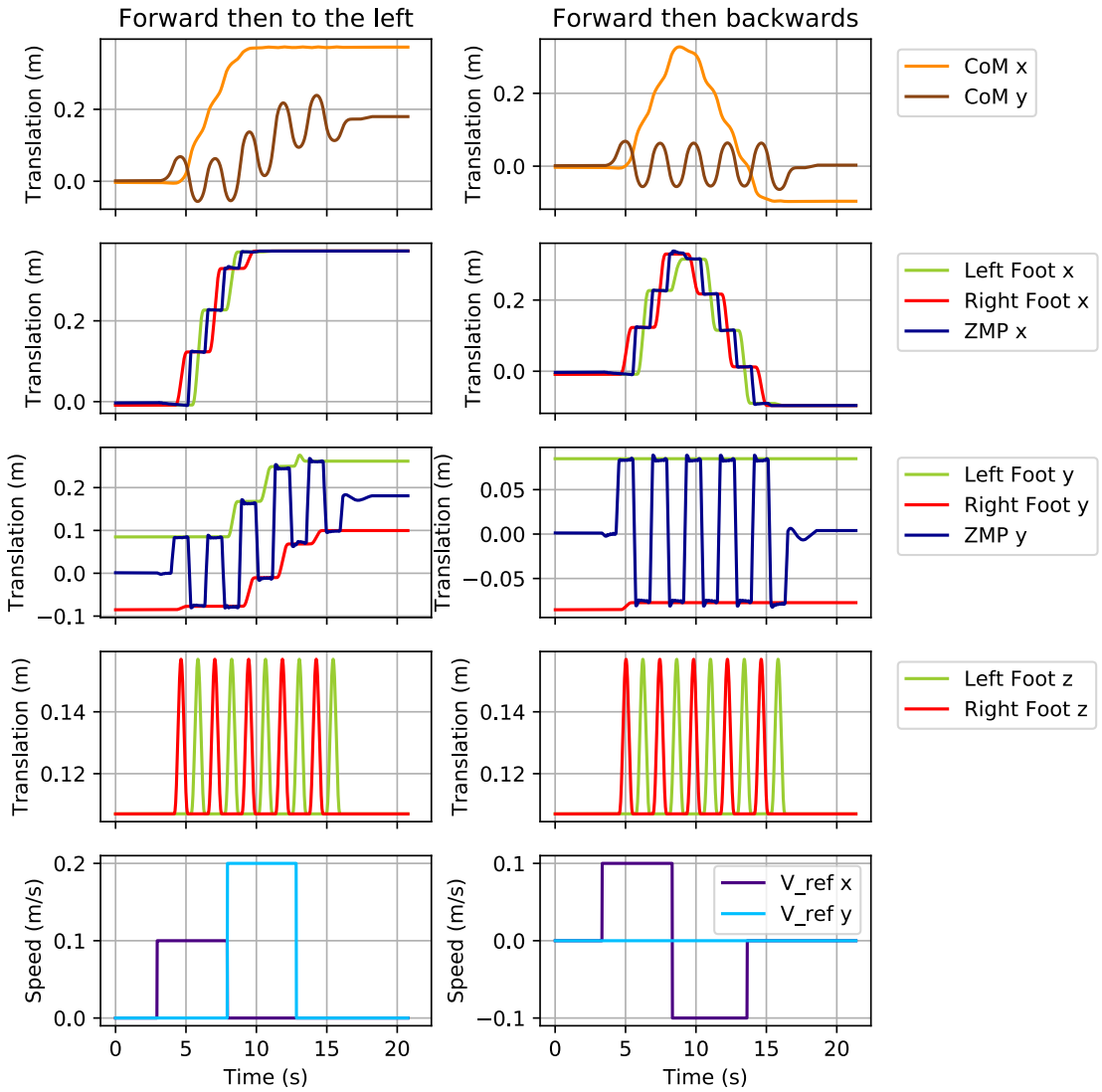


Figure 12: *Reference trajectory computed by the PG in the online simulation, cases of direction transitions, stop at the end of the motion. Left: Forward motion – $V_{ref} = (0.1, 0.0, 0.0)$ – then to the left – $V_{ref} = (0.0, 0.2, 0.0)$. Right: Forward motion – $V_{ref} = (0.1, 0.0, 0.0)$ – then backwards – $V_{ref} = (-0.1, 0.0, 0.0)$*

Validation. It has been checked that the robot changes direction in the simulation, accordingly to the keyboard orders. The topics and the signal for the velocity command (*/sot/pg/velocitydes*) have also been plotted. This allowed to check precisely that the orders are properly taken into account.

It has been included into the *sot-talos-balance* package to be installed automatically with it, and identified by the ROS command *rosrun [script_name]* for convenient launch.

3.4 Solving the waist orientation problem

The following hypothesis have been evaluated:

- the waist signal from the PG is not properly connected to the SoT entity (problem in the Python simulation)
- the waist signal might be connected, but its values not updated by ROS (problem with ROS use)
- the waist signal might not be computed properly by the PG entity (problem in the *sot-pattern-generator* package)

The first two hypotheses have been invalidated by the following tests:

- the graph of entities has been computed from the simulation script. Figure 13 displays the graph of entities for the online simulation. Moreover, a python command allows to print all the signals of the entity and their status (including whether they are plugged or unplugged in the graph). The waist signal appeared correctly inserted into the graph.
- the waist signal values have been plotted (via the ROS topic */sot/pg/waistattitudematrix*): it appeared the signal was not updated (its iteration counter always at 0). A function has been added in the simulation to force its update (and thus re-computation) at each iteration. The iteration counter was then properly incremented, but the waist signal value remained constant, set at its initial value.

From those tests, it seemed that the issue came from the *sot-pattern-generator* package that brings the Pattern Generator entity together. From the design of the control scheme introduced in Section 2.2, the waist orientation should be updated each time the feet orientation is updated. Indeed, the waist orientation is defined directly from the feet orientation. It was previously updated only in a part of the code that was not used anymore with the current robot and the current reference frame used. From this observation, the missing code lines have been added.

The testing of the new signal has been delayed during the debugging of a new version of the software (about 3 weeks), after which the tests were performed, and proved successful: the waist signal is now updated and consistent with the feet orientation. The result is plotted on Figure 14 for a slow rotation speed of 0.01 rad/s (equivalent to 0.6°/s). It has been observed that the rotation of the hip introduces instability in the motion and the robot loses balance when rotating faster than that.

Work remains to be done in order to be able to turn at higher rotation speeds, but this online simulation validates the ability of the current control scheme to make Pyrène walk online in all directions. Those changes on the waist orientation problem have been integrated into the main development branch of the package. In parallel with the resolution of the waist problem, other issues in the package jrl-walkgen have been unravelled by my supervisor Olivier Stasse. The team progress on the WPG during the internship allowed to make the real robot Pyrène walk one step at a time, though it remains unstable for more than one step.

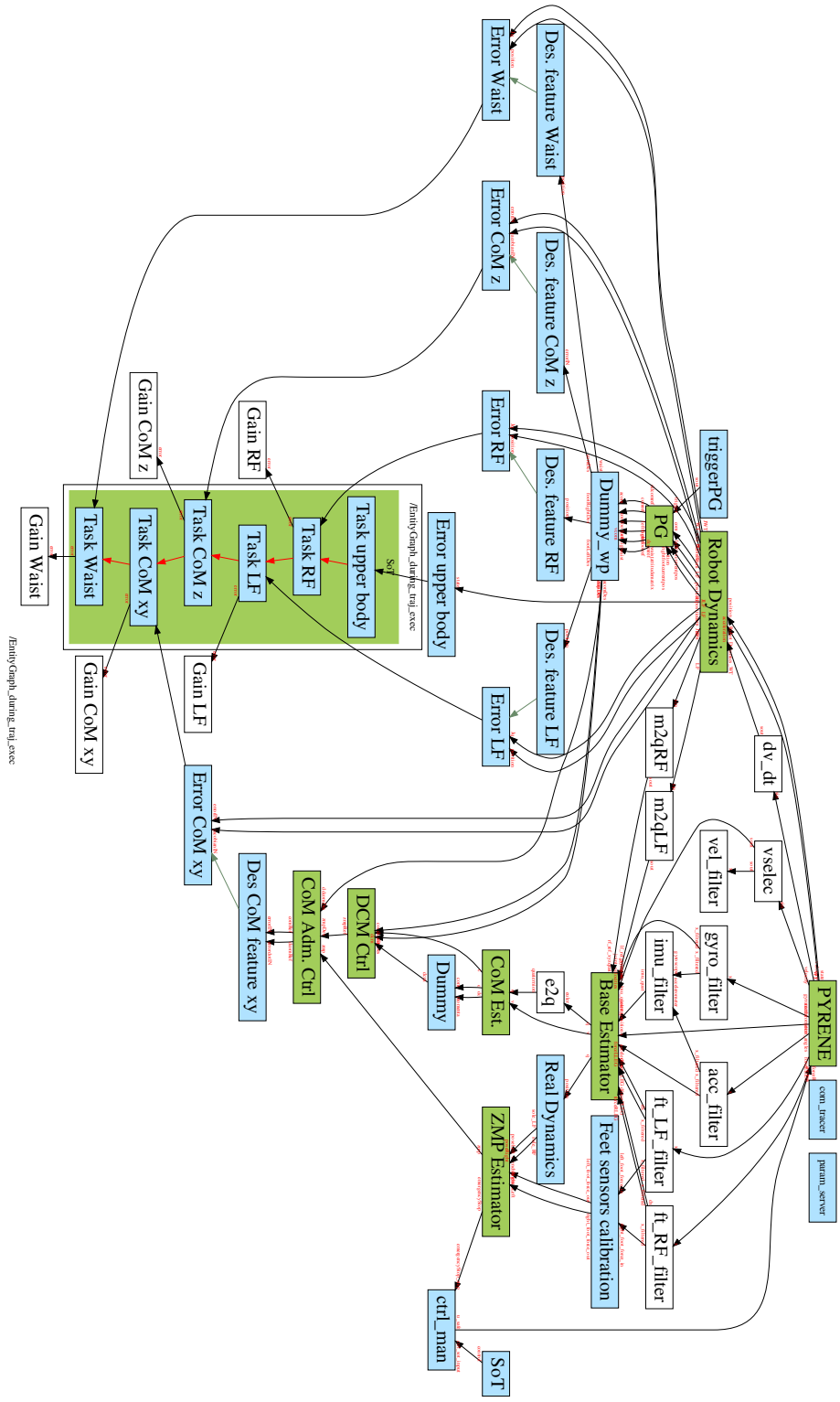


Figure 13: *Entity Graph of the online simulation. The entities are linked via the signals they exchange. The main entities that are also represented on the Control scheme (Figure 7) are colored in green; the white entities are mostly filters for the different sensors and conversion entities (*e2q *means* transformation from Euler to Quaternion representation *of an orientation for example). The 'Dummy' entities are not taking part to the control but only interfaces, mainly adapting signal names for the downstream entities. Please note that some entities have been removed for clarity purposes.*

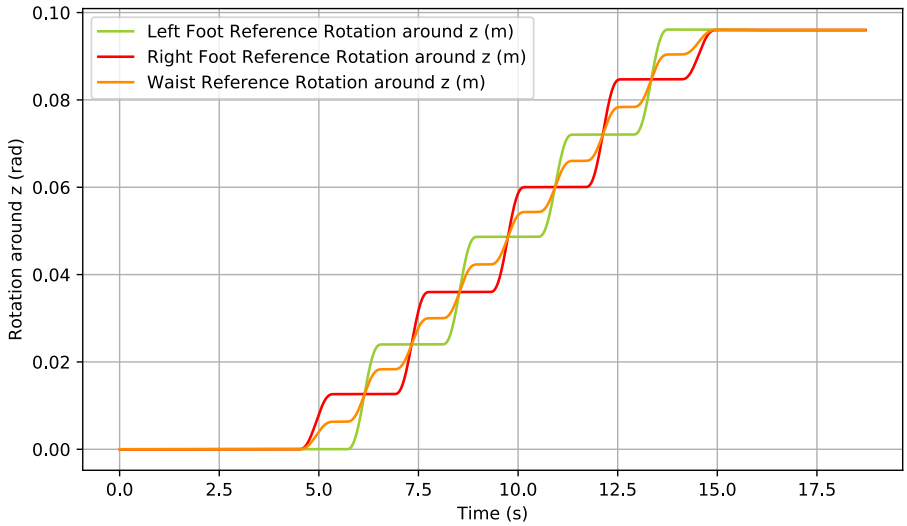


Figure 14: *Waist rotation parameter during simulation is now correctly updated as the average between the feet orientation – Output of the PG entity*

3.5 Difficulties encountered and workaround

The main difficulty I met in getting used to the system comes from the size and complexity of the framework, as well as the fact that little documentation on the code was available. I found Linux search functions to be very helpful in order to look quickly through the entire source code for specific key words.

The software being in development, it happens regularly that an update breaks some features on some user's computers. The update issue mentioned above (which happened at the beginning of lockdown) lasted three weeks, during which I learnt to understand much better how a package is compiled and installed through the catkin tools, as well as the dependencies between the packages and the environment chaining. It allowed me to later integrate and debug my issues with more ease.

Being not experimented in Linux or rigid body dynamics at the beginning of the internship, as well as being a complete beginner in ROS, C++ and the team's framework, an important part of the internship was dedicated to training, either on my own or through reading groups and training sessions, or asking help to those more experienced. The details of my training during the internship is given in Section 5.

The perfect support of the team and the laboratory made the lowkdown period run very smoothly and the communication tools were very well adapted.

Following the results on the PG, two directions of research have been drawn:

- either keeping work on the WPG to improve its performances: increase its maximum speed, improve its rotation ability,
- or working on the stabilizer as originally planned: improve its stability to perturbations.

I have decided to work on the stabilizer for the remaining of the internship.

4 Work on the admittance controller at the ankle

Section 2.3 presented the state of the stabilizer already implemented and tested on the robot prior to the internship. The reader is kindly referred to Figure 8 which summarizes its components: a contact wrench control which adapts the ZMP reference trajectory of the WPG so as to compensate for deviations in the divergent component of motion, followed by a Whole-body admittance control which implements a CoM admittance control, that is the CoM reference position and velocity are adapted to compensate the error in ZMP. That represents a CoM strategy for the Whole-body admittance controller.

My objective on the stabilizer was to implement an *end-effector* strategy on the stabilizer – namely an ankle admittance controller, adapting the feet trajectories to compensate for deviations on the desired contact wrench, directly at the end-effector (feet) level. The controller is acting on the roll (rotation around the x-axis) and pitch (rotation around the y-axis) angles of each foot. That strategy should allow a better tracking of the ZMP reference trajectory.

That stabilization strategy involves the use of the contact force sensors situated on the feet of the robot. Those contact forces are simulated by Gazebo. It was therefore important to first check the consistency between the simulator and the control scheme measurements. That work is presented in Section 4.1. Then, the controller's principle and the (offline) simulation scheme that have been designed to include the new stabilizer parts are described in Section 4.2.1, the design of the new contact tasks is described next (Section 4.2.3). Some work on the WPG was then required (Section 4.3.1) to be able to add the controller in the online simulation. Finally, the unexpectedly poor simulation results required work on the wrench distribution entity (Section 4.4.1).

4.1 Check of the consistency between computation and simulation environment

So far, it has been considered that the simulation environment is perfectly estimating the robot's evolution in space. Similarly, the entities allowing feedback in the control scheme, namely the *Base Estimator*, the *CoM Estimator*, the *ZMP Estimator*, but also the entities called *Robot Dynamics* and *Real Dynamics* on the graph of entities (13) have not yet been discussed. Those entities provide the measured variables used as feedback terms in the SoT and the stabilizer entities. This subsection presents the principle of those state estimators and the bias they might introduce in the coming paragraph. Then, the importance of the accuracy of both the simulator and control scheme state estimators is discussed in the context of the stabilization. This section ends with the measures that have been done to check the consistency between the different state estimators.

4.1.1 The different state estimators of the control scheme

This paragraph details, in the perspective of the stabilization, some of the sources of incertitude that have been listed above in Section 2.3, and in particular, how these translate in the case of simulation.

Model of the robot for simulation. The simulated robot (entity *PYRENE* in the graph of entities) and the simulator Gazebo interact in order to update the state of the robot according to the control vector q and the robot model (mainly the simulation of the low-level servomotor controllers in the robot). The encoders on each servomotor are used for this low-level control loop.

Model of the robot for control (WPG and the Inverse Kinematics scheme). Each task is described by comparing the desired and current measured features. The pattern gen-

erator also requires a measure of the current CoM, feet position and waist orientation. Those measurements are obtained from the entity *Robot Dynamics* which takes as input the values produced by a gyrometer, an accelerometer and feet force sensors on the robot. Those data are transformed into an estimated state velocity, estimated feet and CoM position, and waist orientation, which are then used as inputs for the WPG and SoT entities. Those 'measures' are thus computed based on a robot model, and not actual measurements of the said variables. This introduces discrepancies between the model and the reality.

State estimator for the stabilizer. The stabilizer uses measures of the current CoM, ZMP, feet positions and DCM value that are provided from the state measure of the simulator (based on the measure of the simulated servomotors' encoders values) and the feet force sensors. Those estimated state vector and feet forces are given as input to the *Base Estimator* entity which estimates the CoM position and DCM value from them. That entity is linked to the *Real Dynamics* and *ZMP Estimator* entities that produce estimated feet and ZMP positions, from a robot model as well. Compared to the SoT and WPG entities, the stabilizer thus uses a different means of measurement, which is more directly the result of measurements on the simulated robot.

The study of the detailed implementation of the different state estimators was not in the scope of this internship. However, the implementation choices raise different questions of importance for the stabilizer: does Gazebo correctly position the robot in space? Does the robot models of the simulation and simulator correspond to the models used in the control scheme? Are Gazebo and the simulation consistent on the estimation of the robot state?

4.1.2 Measurement of the state estimator accuracy

The CoM strategy of the stabilizer having already been validated prior to the internship, the present section focuses on the state of the end-effectors.

The following errors on the robot state can be explored:

- The control scheme does not allow to follow the reference trajectories computed by the WPG, in which case the feet end up in a different position to the planned ones.
- The state estimators introduce errors in the positioning of the robot because of rigid mechanics modelling errors for instance, or small axis positioning errors. In this case, both the WPG and the IK schemes are provided with an incorrect 'measure' of the feet positions with which to compute respectively the future reference trajectories and the future control vector.
- The environment of the robot could be different than the one planned by the WPG, for example the floor presents a slope that is not modelled in the WPG, in which case, no matter the precision in the positioning of the feet with respect to the planned trajectories, the feet will hit the ground too soon or too late, and generally not at the right angle.
- (In the case of the real robot, there are also flexibilities existing in the robot bodies, which are not modelled, and which make the feet end up in a different position to the planned one.)

The control scheme should allow the reference trajectories of the WPG to be followed, which had been checked during its implementation. The stabilizer's role is to provide a feedback on the real environment (and the state of the robot in the case of the real robot) and help adapting the reference trajectories to it, that is, to correct the third source of position error. The modelling issue would ideally be small, since compensating for it decreases the ability of the stabilizer to compensate for an environment error as well. That is why the work on the ankle admittance controller started with a measure of the said errors at the feet level.

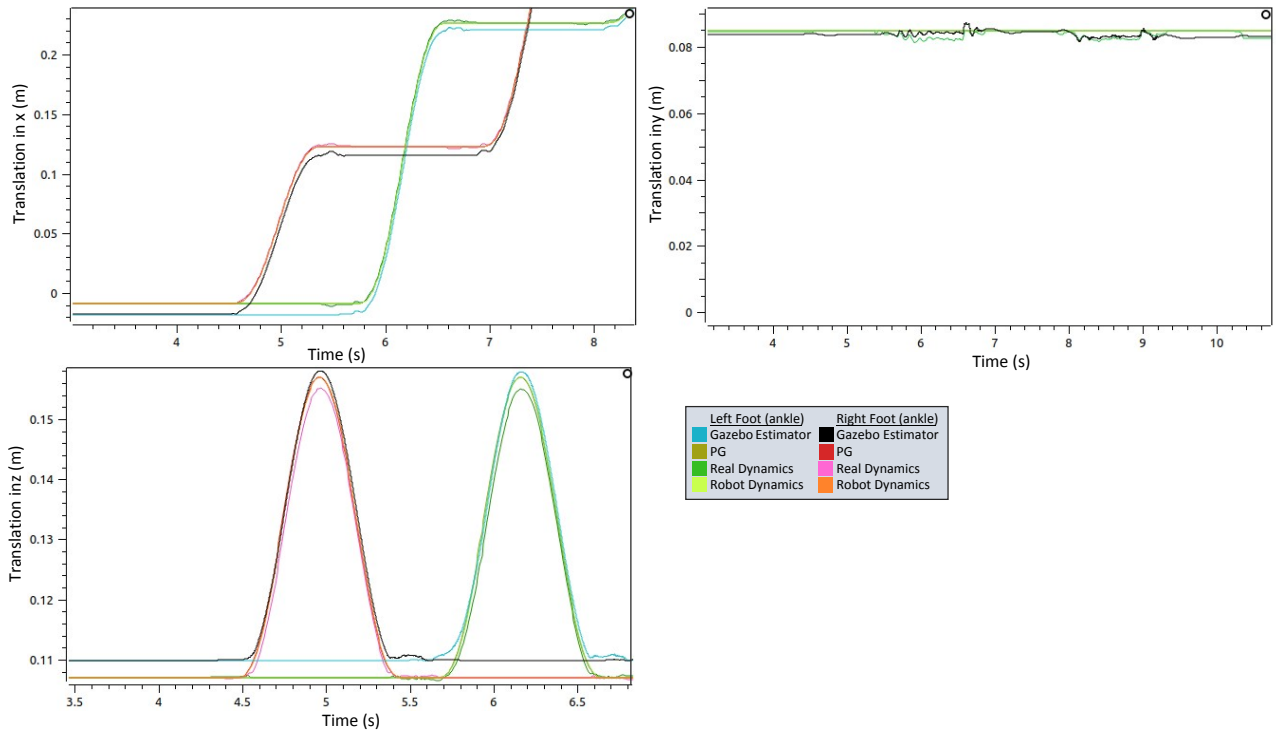


Figure 15: *Feet translation values for the feet in the initial state (until 4.5s) then during the walk between the different estimators and the PG. There is almost no difference between the PG and the estimators of the control scheme, but a larger one with Gazebo's estimator.*

The simulation used for this was making Pyrène walk a few steps in simulation, so as to plot the different feet estimations as well as the reference feet trajectories computed by the WPG and the feet position measured by Gazebo. The errors in the initial state, as well as during the walking movements, have been measured, the resulting plots can be observed on Figures 15.

The feet positions appear well reconstructed by the different state estimators. Gazebo, though, reconstructs a position that is different to the estimators' ones. In particular, for the feet height: the pattern generator planned for a floor position 3mm lower than what Gazebo produces. Thus, the simulation floor has been lowered from the measured difference for the following experiments.

4.2 Ankle admittance controller in the offline simulation

4.2.1 Simulation scheme

Starting from the stabilizer presented on Figure 8, the scheme in Figure 16 has been designed, according to the strategy described in [Caron et al., 2019]: in parallel to a correction at the level of the CoM, another stabilization strategy based on the ankle has been added. The principle is the following one:

1. Wrench distribution among the contacts, solved by optimization in a Quadratic Program (QP) (detailed later in Section 4.4.1)
    - In Single Support Phase: the net contact wrench is applied entirely on the support foot, according to the phase.
    - In Double Support Phase: the wrench is distributed on both feet according to the weight shift from the former support foot to the next one, this is computed by solving an optimization problem described below.

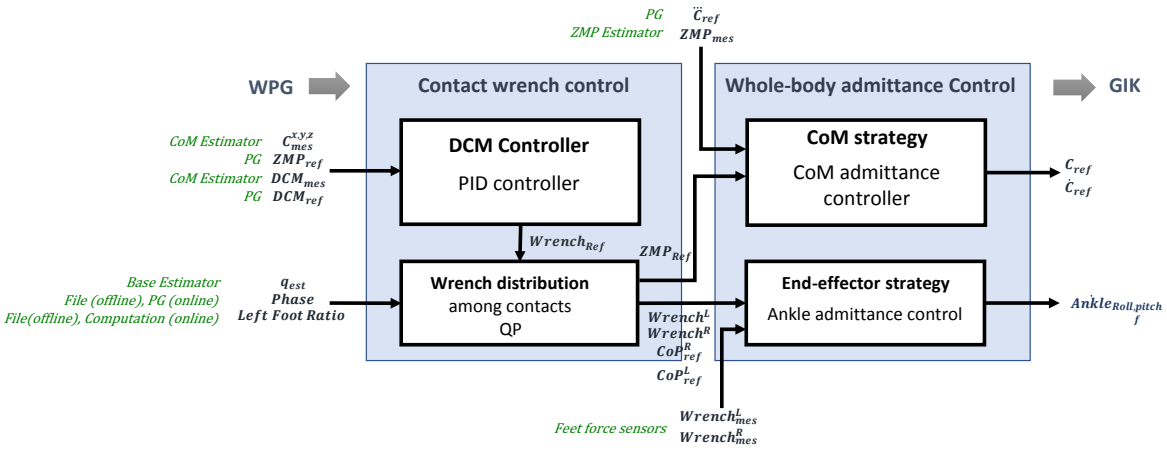


Figure 16: *Full Stabilizer scheme – the green scriptions indicate which entities the facing signals come from. The* Wrench distribution *and* End-effector strategy *(Ankle admittance control) entities are the two entities added to the stabilizer during the internship.*

- From the distributed wrench, the Center of Pressure (CoP), that is the application point of the contact wrench on each foot, is computed. In the case of single support phases, the ZMP of the robot and the CoP of the support foot coincide.

2. Ankle admittance controller: from the distributed wrench, the ideal center of pressure on each foot is drawn, that is the ideal application point of the forces (referred to as $p_{Left,Right}$ in the motion equations of the state of the art Section 2). The torque corresponding to the lever arm of the measured foot force over the ideal CoP $\tau_{L,R} = p_{L,R}^{ideal} \times F_{L,R}^{mes}$ is compared to the measured torque. From the difference, a command in angular velocity is sent on the roll and pitch angles of the ankle of each foot:

$$\dot{\theta}_{R,P} = A_{CoP}(p_{L,R}^{ideal} \times F_{L,R}^{mes} - \tau_{mes}) \tag{31}$$

Starting from a first (untested) version of the wrench distribution and ankle controller entities, they were first integrated into the offline simulation. This required the following changes to the graph of entity:

- The net contact wrench $W = \begin{pmatrix} F \\ \tau \end{pmatrix}$ comes from the DCM controller, which computes it analytically from the adapted ZMP reference it computed, along with the CoM trajectory from the WPG:

$$F = \begin{pmatrix} \frac{mg}{c_z}(c_x^{PG} - ZMP_x^{DCM.Ref}) \\ \frac{mg}{c_z}(c_y^{PG} - ZMP_y^{DCM.Ref}) \\ mg \end{pmatrix}$$
$$\tau = C \times F$$

- The wrench distribution computes a new reference ZMP position based on the result of the wrench distribution, which might be different from the one computed previously by the DCM. Thus, the CoM admittance control is now plugged to the ZMP reference of the wrench distribution entity.

- The phase signal is a new signal in the control scheme, needed for the wrench distribution (new file provided to the offline simulation).

- The wrench distribution takes into account the ratio of total robot weight that is put on each foot during the DSP. This can be done by adding a signal. *LeftFootRatio* that varies from 0 (flying foot just landed or just taking off, no weight on the foot) to 1 (support foot: weight totally transferred on the foot) during each DSP.

At this point, the controller is plugged for its input signals, and can thus compute its output ones, but it is not yet integrated into the control loop: the output ankle rotation speeds are not yet transmitted to the robot via the SoT. The computed values of the controller can thus be tested first, to ensure a correct behaviour. This requires new testing simulations.

4.2.2 Design of testing simulations

The tests in simulation have different objectives:

1. During the design of the simulation scheme, and when modifying the controllers: test the controller behaviour.

2. When satisfied with the behaviour: test the stability improvement, that is to what obstacles/changes in environment it is now able to cope with, and how much the ZMP tracking is improved.

Small changes in the simulation environment allow to visualize how the controller signals evolve (and how the robot walk is impaired by it). Different changes can occur:

- The foot hits the floor with the wrong orientation (only with its heel for example). In the present case of flat ground walking, this was not encountered.

- The floor is not at the planned altitude: either it is too low and the foot does not land before starting to transfer weight on the leg, or it comes too soon, for example if there is a small obstacle the robot steps on. That latter case is the one studied for the present simulation.

A simulation environment including a stick of varying height and position has been designed. The initial state of such a simulation can be seen on Figure 17. The stick has been designed with the same characteristics as the floor in terms of friction coefficients and collision parameters. As the controller outputs are not yet transmitted to the SoT, the simulation is used to check the behaviour of the controller. The roll and pitch commanded velocities for the left foot (which is the foot stepping on the stick) can be seen on Figure 18. The roll commanded angle is decreased when the robot steps on the stick, at the end of the SSP ($9.6s$): this corresponds to tilting to the right so as to compensate for the unexpected load on the left foot. Then, at the end of the DSP, the reverse behaviour is observed as the robot should transfer its weight on the left leg. The main variation of the pitch angle happens just before $10s$, in the middle of the DSP: the controller aims at bringing the robot's CoM forward, pitching its ankle towards the top. This is to compensate for the torque driving the robot backwards when the foot is tilted upwards by the stick.

4.2.3 Integration into the contact tasks

Two different implementations of the controller outputs into the SoT have been tested. The first implementation was inspired by the available functions that had been previously designed along the ankle controller. It involved creating four additional tasks for the inverse kinematics scheme, in parallel with the existing contact tasks: one for the desired roll and pitch velocity for each foot computed by the ankle controller. The second implementation was the integration of the controller outputs directly into the existing two contact tasks.

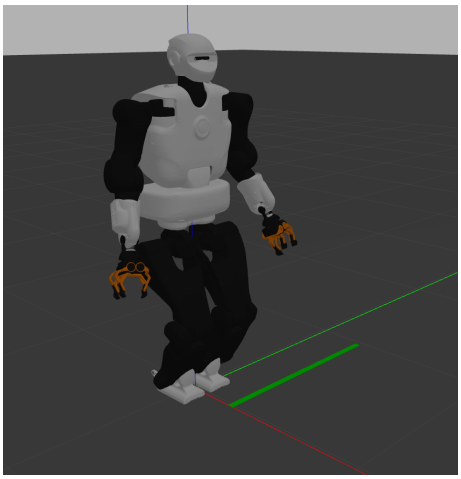


Figure 17: *Initial state of Pyrène in simulation with a stick as obstacle, 25cm ahead of him, 4mm high and 3cm large. In this configuration, Pyrène steps on it with the tip of the left foot at its second foot step.*

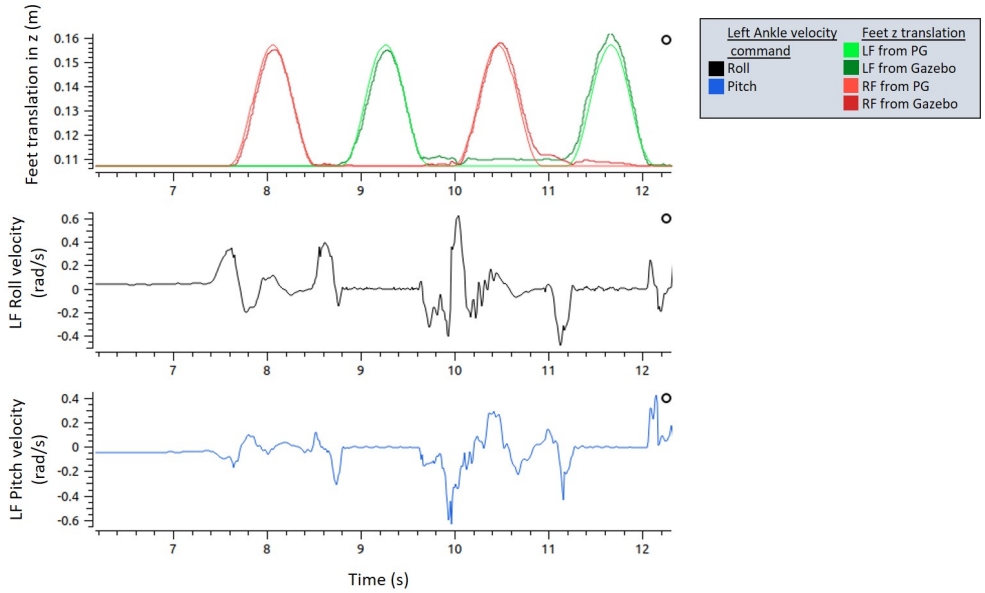


Figure 18: *Ankle controller velocity output. Pyrène walks on the stick at time 9.6s – The ankle controller gain is set at 0.01 – The controller is not yet plugged to the SoT.*

The creation of four additional tasks did not bring any result: when those tasks were set at a low priority (after the contact tasks), they did not bring any change to the control vector of the robot. That is probably due to the fact that the SoT was unable to allocate DoFs for those tasks. When put at high priority though, the realization of those tasks disturbed the robot in its other tasks, resulting in control outbursts and falls: they probably represent too many tasks for the control. The robot was not able to stabilize its static position.

Going back to equation 15 introduced in the state of the art (2.2.3) and defining the tasks e with respect to the features s^* (desired) and s (measured):

$$\dot{e} = \dot{s} - \dot{s}^* = -\alpha J^\dagger (s - s^*)$$

The following equation can be derived:

$$\dot{q} = -\alpha J^\dagger (s - s^*) + J^\dagger \dot{s}^* \tag{32}$$

where s^* is here the foot position given by the PG and $\dot{s}^*$ the reference velocity vector for the roll and pitch angles computed by the ankle controller. At this point, $\dot{s}^*$ is thus not the derivative of s^*. Though tasks like the CoM task on x and y are using a reference feature in both position and velocity, this would not be consistent for the contact tasks without modifying the current features.

The features for the contact tasks are initially taking only the foot position as reference. In order to keep the same kind of feature, the desired velocity computed by the controller can be integrated over a time step and incorporated into a new reference position for the feet.

The feet trajectories are stored in homogeneous matrices, which gather the translation part of the position (x, y and z coordinates) and the orientation part (rotation along each axis, expressed as a quaternion). Those poses (position and orientation) belong to the Special Euclidian group SE(3), and have the following form:

$$P \cong \begin{pmatrix} x \\ y \\ z \\ q_x \\ q_y \\ q_z \\ q_w \end{pmatrix} \tag{33}$$

The feet velocity is represented by a 6D-vector gathering the linear velocity along the x, y and z axis, as well as the angular velocity along the same axis. Integration of a feet velocity into a feet pose (position and orientation) requires a special operator in the SE(3) group. Such operators are implemented in the Pinocchio package, along with many derivative and integration tools used in robot control. The *integrate* function takes as input the current position and velocity and computes the position that would be obtained if the object at the current position were moving at the current velocity for one second.

With that tool, a new output signal of the ankle controller entity has been added, so as to output the adapted reference feet positions. It is meant to replace the reference feet trajectory from the PG in the reference features for the contact tasks.

Upon looking at the phase signal file, it seemed that the DSP was not of the appropriate duration. The contact phase information is however very important since it rhythms the wrench distribution. As a result, the wrench distribution profile looked too harsh, with no real transition from one foot to the other during the DSP. It has thus been decided to move directly to the online simulation and to test the full control scheme direclty online.

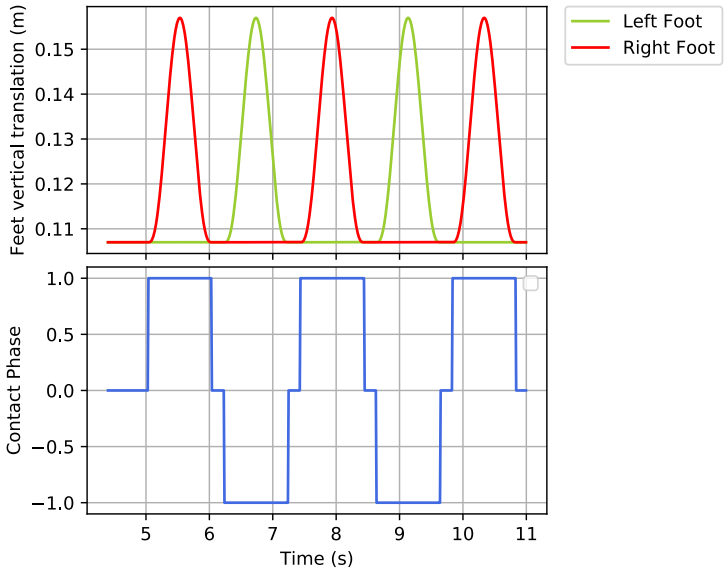


Figure 19: *Output Contact Phase Signal from the PG – 0 stands for DSP, 1 for LF support and -1 for RF support*

4.3 Adaptation to online simulation

The two additional signals required by the wrench distributor (LeftFootRatio and Contact Phase) are provided by additional files in the offline simulation. They are not naturally computed by the PG entity (the files were computed analytically after the original trajectory files had been produced). On the online simulation, the computation of the signals has to be added directly into some entities. This is the subject of the two coming paragraphs.

4.3.1 Enabling the phase signal reading from the WPG

The phase signal has been added to the output signals of the pattern generator.

The phase signal is an integer, either 0 (DSP), 1 (Left support foot) or −1 (Right support foot). It is directly deduced from a variable of the *PatternGenerator* entity keeping track of the state of the feet (whether they are support or flying foot).

Computed directly from the pattern generator, the phase signal ended up being more consistent that the one included in the files. In particular, the first and last node of DSP are now placed exactly when the swinging foot takes off or lands (please refer to Figure 19), which means that the DSP now has a much more accurate duration.

This signal was also needed for a PhD student of the team, Noëlie Ramuzat, with whom the final version of this signal has been designed: we merged our two versions into the final one and integrated it into the current version of the package. This signal is now available by default when installing the package.

4.3.2 Redefining the Left Foot Ratio

This Section focuses on the implementation of this signal for the online simulation and represents one major change from the implementation of Stéphane Caron. The way the Left Foot Ratio is used in the wrench distribution is detailed in the section after this one. The files provided for the left foot ratio had been designed empirically and for early tests prior to the internship, with a linear variation from 0.4 to 0.6 (instead of a theoritical variation from 0 to 1) during the DSP (and had inadvertently been designed as the Right Foot Ratio instead). That imposes

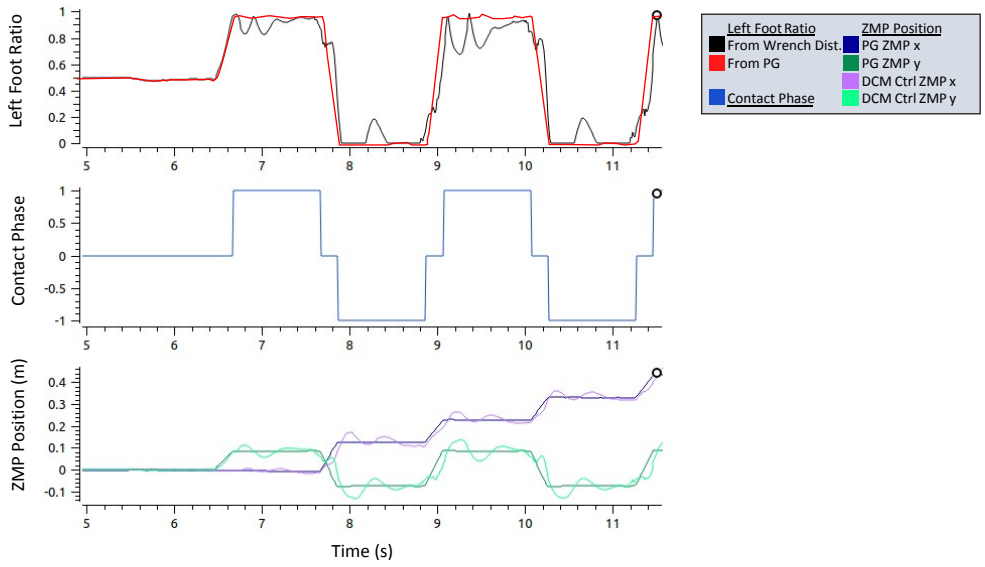


Figure 20: *Top: Left foot ratio signal as now implemented in the Wrench Distribution entity (in black); the red curve shows what an ideal ratio would look like, it takes the ZMP trajectory of the WPG directly instead of the one adapted by the DCM controller. Middle: Contact Phase Signal. Bottom: ZMP trajectories that explain the Left foot ratio shape: the DCM controller actually computes a rather important correction.*

an arbitrary variation of the ratio, without taking into account the fact that the WPG already indirectly defined a distribution of the robot weight, and hence feet forces when it optimized the ZMP trajectory. (In the implementation of Caron et al., 2019, the WPG is implemented differently and the Left Foot Ratio is really meant to decide which repartition of the robot weight should be applied.) Thus, instead of imposing a linear transfer of weight from one foot to the other during DSP which might collide with what the PG has planned, it would be more consistent to define the left foot ratio using the PG.

Though modifying the WPG so as to output the wrench distribution directly would be possible, it requires rather important changes in the package *jrl-walkgen* and it was decided to first implement it in the wrench distribution entity. In order to take into account the PG, the left foot ratio was defined as follows (definition inspired from Kajita et al., 2010):

$$LeftFootRatio = \frac{|ZMP_{x,y} - RF_{x,y}|}{|LF_{x,y} - RF_{x,y}|} \tag{34}$$

$|LF_{x,y} - RF_{x,y}|$ defining the distance on the floor between the feet sole centers and $|ZMP_{x,y} - RF_{x,y}|$ the distance between the ZMP and the right foot.

The LeftFootRation is clamped between 0 and 1 for safety.

The ZMP used in the present control scheme is the one from the DCM controller, as this controller adapted the WPG reference ZMP trajectory to minimize the DCM. This was inserted as an internal signal to the wrench distribution entity, with the addition of the following input signals: Left and Right Foot reference trajectories from the WPG and reference ZMP trajectory defined by the DCM Controller. The signal can be seen on Figure 20.

The Left Foot Signal is now consistent with the WPG signal. Its variation during the DSP is the one of interest since this signal is used in the wrench distribution QP only; thus, the rather poor behaviour of the signal in SSP is not concerning. From that point, the online simulation now possesses all its signals and is fully plugged in the control scheme. A graph of entity of the final simulation can be seen on Figure 25, as well as the final control scheme of the full simulation on Figure 24. In order to test the validity of the Left foot ratio variation, the QP

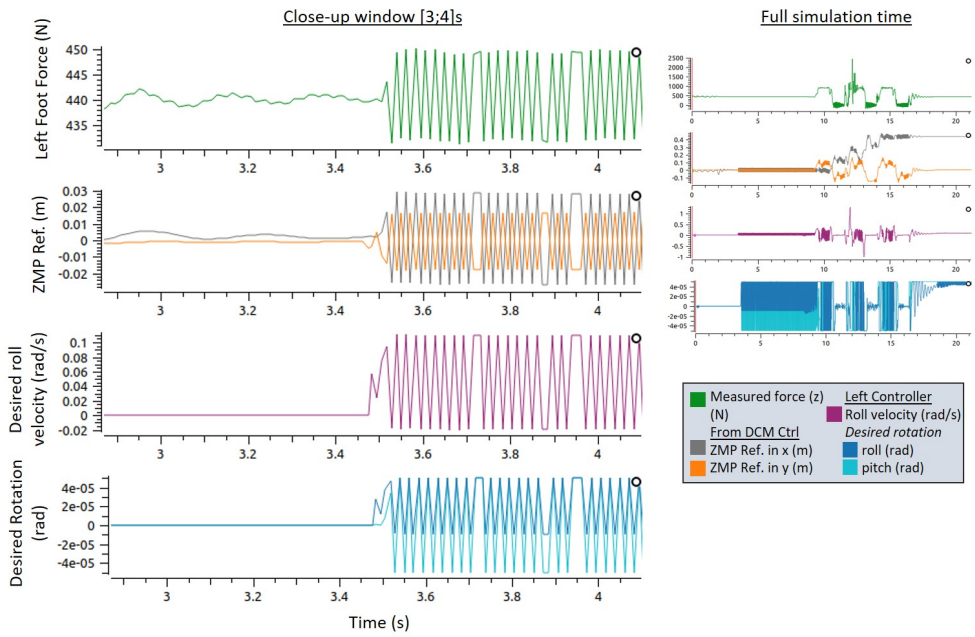


Figure 21: *Oscillatory behaviour – Full simulation and close-up with an ankle admittance control gain of 0.01 rad.m/N. The oscillations start at time 3.45s when the controller gain is switched from 0 to 0.01 after initialization. Most variables of the control scheme oscillate with the controller.*

and the distributed feet wrenches can be compared to the measured feet wrenches.

4.4 Work on the wrench distribution

The tests in simulation with the previously described simulation trigger strong oscillatory behaviour as soon as the ankle controller gain values are set to their non-zero value, if this value exceeds 0.001 rad.m/N (Figure 21 shows such oscillations). Such a gain value is 10 times smaller than the expected value and thus does not induce any noticeable adaptation of the feet to the ground. The oscillations have the following characteristics:

- High frequency (comparable or larger than the sampling frequency of 1kHz)
- Instantaneously triggering an oscillatory behaviour of most signals (including the feet forces measured and computed by the QP, the measured feet position on the ground, the measured ZMP, the ankle controller output signals...)
- High amplitude (same order of magnitude as the signal values)

The oscillations induce the vibration of the entire robot. The main hypothesis for this behaviour was a modelling issue on the wrench distribution (including the definition of the left foot ratio), inducing incompatible wrench targets from the WPG and the ankle controller.

Other hypothesis included a deeper modelling problem, a DCM controller gain that was too high (since Stéphane Caron identifies in his article that high gains on the DCM controller disturb the correction the ankle controller can bring and might lead to oscillations).

Trying to solve that oscillatory behaviour has been the work of the last six weeks of the internship. The behaviour of the QP has been checked first: it is detailed in Section 4.4.1 and its results have been compared to the expected and measured wrenches (Section 4.4.3), which led to the discovery of a model issue in the control scheme. In parallel, the ankle controller has been improved with a clamping feature (Section 4.4.2). In the very last days of internship, some

final tests were designed with Noëlie Ramuzat, that led to some new clues, but the controller is not yet fully functional.

4.4.1 Quadratic Program for the Wrench Distribution

The wrench distribution is performed by a quadratic programming taking the net contact wrench (at the level of the CoM) as input, along with the desired weight repartition over the feet (provided by LeftFootRatio). Its objective is to define w_{left} and w_{right} at the level of the center of mass, from the net contact wrench.

The QP is solving the following problem in DSP, with w_{left} and w_{right} as free variables:
The cost function is weighing three objectives:

- Respect the net contact wrench w_{net}: $\| w_{left} + w_{right} - w_{net} \|^2$ (weight: 10 000)
- Minimize ankle torques $\| \tau_{left} + \tau_{right} \|^2$ (weight: 10)
- Follow the left foot ratio as close as possible for the vertical component of the force (written here ρ in short): $\| (1-\rho) f^z_{left} - \rho f^z_{right} \|^2$ (weight: 1)

The following constraints are applied:

- Contact stability: the forces must lie inside the contact cone. This is formulated in a large matrix gathering the inequality constraints that describe the cone
- Avoid sending low-force targets to the controller for stability purposes, by setting a minimum vertical component for the forces of 15 N

The output left and right wrenches are expressed at the level of the CoM and must be expressed at the feet level to be used by the ankle controller. The transformation matrix used is available in Pinocchio, which stores all the relevant robot model and data.

The optimisation algorithm used is *EiquadprogFast*, a powerful solver designed in the team and based on the Eigen package (a common library for linear algebra).

The resulting wrench distribution computed can be seen on Figure 22, along with the measured feet forces along the vertical axis.

The distributed wrenches seem consistent with the measured forces, however on closer look, the different force components show up to a 30N error and the distributed torque can show up to a 50 N/m error (when the measured torque value was below 10N/m), even in static posture. Such an error could lead to the observed oscillations. The study of that difference was the following step of the research on the oscillation issue and is detailed in Section 4.4.3. Before that, an output speed clamp for the ankle controller has been added in parallel to the investigations on the wrench distribution and is presented in the following section.

4.4.2 Ankle Controller speed output clamp

Upon extensive research in the source code of the control scheme described in Caron et al., 2019 (WPG and Stabilizer) in parallel with a PhD student of the team, Fanny Risbourg (who was also implementing a similar stabilizer on an exoskeleton), it appeared that the control speed of the ankle controller was actually coded to be clamped. That agrees with the wish to bring the controller gain higher for more important correction, but without risking a commanded rotation speed the ankle cannot reach. The clamping value was put at 0.1 rad/s (or 6 °/s) which is below the ankle speed limit. Though this improved significantly the controller behaviour in the case of the exoskeleton, it did not prevent the oscillations on Pyrène, and thus it is hard to assess precisely its benefit and tune its bounds as long as the oscillations remain.

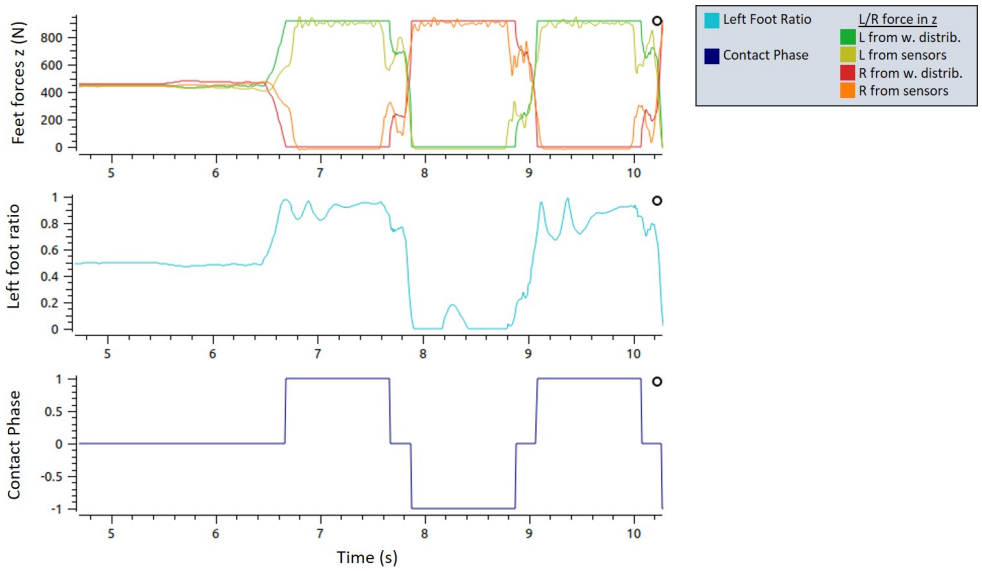


Figure 22: *Top: Surface vertical force computed by the wrench distribution and measured by the force sensors (the measured wrench is filtered by the feet sensors calibration at this point). The force increase measured during the DSP is mainly due to a rather strong impact force. Middle: Left foot ratio. Bottom: Contact phase.*

4.4.3 Study of the static forces and Modelling issues discovery

In static position, the expected feet forces can be easily computed analytically. Since the oscillations appear even in a static position at the initial state, the expected static feet forces have been computed in order to check the consistency of the wrench distribution.

The Newton and Euler equations are reminded below (it is still supposed that there is no exterior force, other than the feet forces and gravity):

$$\cancel{mc}^{\,0} = f_{left} + f_{right} + m(-ge_z) \tag{35}$$

$$\cancel{\dot{L}}^{\,\sim 0} + mc \times [\cancel{\ddot{c}}^{\,0} - g] = p_{left} \times f_{left} + p_{right} \times f_{right} \tag{36}$$

In static position, the barred terms are null or neglected (case of $\dot{L}$), and the following system of 6 equations remains:

$$\begin{cases} f^x_{left} + f^x_{right} = 0 \\ f^y_{left} + f^y_{right} = 0 \\ f^z_{left} + f^z_{right} = mg \\ m \begin{bmatrix} c^x \\ c^y \\ c^z \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} p^x_{left} \\ p^y_{left} \\ p^z_{left} \end{bmatrix} \times \begin{bmatrix} f^x_{left} \\ f^y_{left} \\ f^z_{left} \end{bmatrix} + \begin{bmatrix} p^x_{right} \\ p^y_{right} \\ p^z_{right} \end{bmatrix} \times \begin{bmatrix} f^x_{right} \\ f^y_{right} \\ f^z_{right} \end{bmatrix} \end{cases} \tag{37}$$

Using a least-square solving function from the Eigen package, this system is solved for its 6 force variables.

This computation was performed at the beginning of the simulation, it yielded to different observations:

- even in static position, there is a difference between the left and right wrenches computed by the QP, which reflects the LeftFootRatio value of 0.507 (instead of 0.5). It appeared to be due to the fact that the WPG generates an initial position of the CoM that has a

non-zero value on the y component ($0.001m$), hence the theoretical weight repartition is slightly tilted towards the left foot.

- The feet forces in the x and y directions being negligible compared to the vertical force, the sum of the z components of the forces should be equal to the total gravity weight of the robot (measurements in Gazebo). The wrench distribution computes a total vertical force of 915N, whereas the static feet forces computation computes 885N. That latter weight agrees with the feet force sensors.

- The feet sensors calibration entity has been checked as well. It displayed an offset that was usually used on the real robot. The offset has thus been put to zero for the simulation.

The investigation on this total force difference allowed to discover a problem on the robot model used throughout the simulation scheme. Indeed, the wrench distribution uses one robot model, whereas the WPG uses a reduced one. They both store a different robot mass (93.3 kg and 90.3 kg) which accounts perfectly for the difference in the measured total vertical force.

In parallel to my investigations, a PhD student of the team, Noëlie Ramuzat, working on the WPG for torque control on Talos, met this model inconsistency problem as well, by looking for the reason of an inconsistency in CoM height of the robot in simulation, as well as a wrong feet placement. Both my supervisor and Noëlie have then worked on solving this issue, that is by ensuring that only a single robot model is loaded at the beginning of the simulation and provided to all entities. This issue impacted several packages and took about three weeks to solve.

During the last week of internship, the model issue was solved on Noëlie's installation and the simulation could be tested on it. The following differences were noted:

- The initial state of the robot measured by Gazebo is now much closer to that planned by the WPG (for the feet: zero difference along the x and y components, 3mm difference in the z direction).

- Almost perfect correspondence between the wrench target of the wrench distribution and the theoretical static forces computed (less than 4N difference).

- Almost perfect correspondence of the static vertical forces (about 4N maximum difference) between the desired wrench and the measured one, which was the main expected result. However, no improvement was observed on the other force components: there is still up to 30N (forces) and 50 N/m (torque) difference between the desired and measured wrench. This could be due to the use of a different frame of reference in which the feet sensors are considered by Gazebo and the control scheme. Namely, Gazebo seems to refer to the center of the feet body for the feet sensors, whereas the control scheme defines the forces at the level of the ankle joints.

- Same oscillatory behaviour as before.

- The oscillations' amplitude seemed lower than before, and particularly the LeftFootRatio was now showing consistent values throughout the simulation. That decrease in oscillations allowed Pyrène to walk over and past the 4mm stick with an ankle controller gain of 0.01 Nm/rad, which was impossible before due to stronger oscillations. That still does not represent an improvement on the stabilizer compared to the use of a CoM admittance controller only, since the oscillatory issue is still not solved (thus, no improvement on the ZMP tracking is observed).

Those results tend to indicate that the wrench distribution is valid, at least in the static state. There is for now no explanation for the remaining difference between the measured and desired wrenches (which now concerns all but the vertical force component). It is also surprising

that such small force differences would trigger the oscillations, as they represent an error of a few kilograms which is frequently measured when using the real robot.

4.5 Clues for future tests

First, computing the desired wrenches for each foot direclty from the WPG might bring more accurate results and especially results that are more consistent with the reference trajectories computed by the WPG, including smoother transition between the SSP and DSP.

The team work of the last days of internship with Noëlie Ramuzat allowed to test a few different configurations on her computer, on which the model issue was solved. They are listed below:

- Bringing up the control gain of the CoM task on x and y for the feature in position significantly decreased the amplitude of the oscillations, to the point where the controller, though still oscillating, was not diverging anymore. Increasing the gain should be a good solution for better stabilization as soon as the issue on the wrenches is solved.

- Noëlie implemented a low-pass filter on the desired feet orientation computed by the ankle controller. It filtered the remaining oscillations (the ankle controller gain was maintained at 0.01) and was given as reference feature to the contact tasks. From that point, the ankle controller unfiltered outputs (both the desired ankle roll and pitch velocity and new feet orientation) were not oscillating anymore. That filter introducing a slight delay in the response, it is likely that the oscillations come from the feedback on the controller. This represents a practical solution for the present state of the controller, which allows the controller to be less sensitive to the remaining feet sensors' issues.

- The robot was able to pass a 5mm high stick, which is a very small improvement on the stabilization. (This represents a 1° tilt of the foot when such stabilizers are expected to stabilize the robot on up to 10° slopes.) That improvement was however not observed in terms of ZMP tracking. This could be further studied with various environments, including slopes, where the behaviour might be better.

4.6 Difficulties and workaround

The main difficulty that occurred to me during this part of the internship was trouble-shooting in such a large architecture.

I mainly plotted as much data as I could from my simulation, building additional output signals of entities such as the PG in order to see exactly what was computed.

In parallel, in order to find possible sources of the instability of my controller, I mainly looked for more articles of similar controllers, as well as for the documentation and the comments that had been written by Stéphane Caron in his implementation (since his source code is also available on Github). In this way, I discovered major differences between what was written in his article and what had been implemented, mainly a 10-fold factor in the ankle admittance controller gain value used (the article recommended 0.1 rad.m/N whereas the implemented gain was 0.01 rad.m/m) and the clamping of the angular speed that were not mentioned in the published paper either. Apart from those observations, the code was actually very well commented and documented and thus provided a very valuable and detailed supplement to the article.

5 Training during the internship

Along with the documentation I studied on my own, a lecture group has been organized in order to work in groups on documentation. In the first months, the book by Featherstone Featherstone, 2008 has been read.

In parallel, robotics lectures have been organized by a team researcher, Nicolas Mansard, two hours per week on average, on the following subjects: Geometry (Direct and Inverse), Kinematics and Dynamics (Direct and Inverse as well), Optimal Control. The main theoretical expressions have been studied, along with algorithms to solve them, including optimization algorithms.

Both the lecture group and the robotics classes were very beneficial to me and helped me understand more thoroughly the theoretical principles behind the work of the internship.

A remote Summer School was also offered from July 7^{th} to 9^{th} on the project Memory of Motion (MEMMO), which is one of the projects of the team, focused on three of the tools developed in the team:

- Pinocchio,
- Crocoddyl: very efficient optimal control solver designed to work with Pinocchio for the kinematic computations. It aims at solving the full locomotion problem (grey rectangle on Figure 5),
- TSID: for Task Space Inverse Dynamics.

The tutorials on the tools were accompanied by theoretic classes on kinematic planning, optimal control and contact modelling. Even though the summer school took place near the end of the internship, the opportunity to ask questions to the developers of the tools (mainly Pinocchio for me) was very beneficial.

6 Conclusion

During this internship on humanoid robot locomotion, I was able to learn how to use the team's software for the control and motion planning of a humanoid robot, and significantly increased my knowledge on rigid body dynamics and optimal control. I contributed to two different components of the control scheme of the humanoid robot Pyrène: the Walking Pattern Generator and the Stabilizer.

Comparing the present achievements with the initial internship objectives, the Pattern Generator has been included into an online simulation and keyboard control on the robot in simulation has been implemented. However, the PG remains currently unable to make the real robot walk more than a step at a time.

Regarding the stabilizer, progress has been made on the integration of an end-effector strategy designed to improve the CoM strategy stabilization that had already been validated. The controller has been adapted for the team's control scheme and its different components have been tested independently. Issues remain on the full control scheme that prevent its validation. This stabilizer is implemented from scratch on Pyrène, whereas the one of the previous HRP-2 robot had benefited from 30 years of experience of its makers. That part of the control is in fact tightly linked to most of the control variables which have strong influence on each other. I lacked time to find a final solution for the ankle controller's oscillatory behaviour, even though the work accomplished with Noëlie Ramuzat at the very end of the internship provided new clues on that issue with the implementation of a low-path filter on the ankle controller's output feet trajectories.

I discovered humanoid robotics and the locomotion problem during this internship, which I found fascinating. I would be very interested in working on the understanding of human locomotion through research on legged robots.

At the beginning of my internship, I was pondering over the idea to apply for a PhD thesis, or, more generally, to head towards a research activity in the future. I am now convinced that I would be very happy to keep working in research in the fields of Robotics that are linked to Biomechanics and their possible applications in the medical field, such as prosthetics or exoskeletons. This represents a long-time dream that guided my studies and was only consolidated during this internship.

7 Acknowledgements

First, I would like to express my deep gratitude to my supervisor Olivier Stasse for his outstanding guidance and kind support throughout the internship, as well as for offering me the chance of working in the field I am passionate about. I am extremely grateful for the comprehensive teaching in legged Robotics he provided to me. I don't think I could have had a more enriching experience.

I feel very privileged to have worked in the Gepetto team and would like to express my gratitude to all its members for their kindness and the incredibly positive atmosphere they have created in the team. Though I will only quote here those I directly worked with, I am grateful to all *Gepettists*, as well as to the members of the two other Robotics team of the laboratory I had the chance to meet. I would like to thank Guilhem Saurel for his technical support and his all time availability for software issues. I am also grateful for the very instructive conversations I had with Isabelle Maroger, Kevin Giraud Esclasse and Noëlie Ramuzat who helped me particularly at the beginning of my internship to get familiar with the software. Later, I benefited from very valuable insights from Joseph Mirabel, Noëlie Ramuzat, Fanny Risbourg and Florent Lamiraux. I am especially grateful to Noëlie, with whom the very last tests have been conducted. I can not thank her enough for the time she spent on my simulation in the hope that we could finally solve the stabilizer's issues.

As to my academic training, I would like to thank my supervisor and Biology teacher Franz Bruckert for his profitable classes and particularly for transmitting his great interest in Physiology to me.

My thanks extend to PHELMA for the great training I was offered, as well as to the laboratory LAAS-CNRS for the opportunity to do this internship within its walls. During the lockdown period, I greatly appreciated the support and follow-up from both of them.

Finally, I am very thankful to the proof-readers of the present report, for their very valuable advices and corrections: Olivier Stasse, my classmate and friend Sophie Piot and my parents.

References

[Caron et al., 2019] Caron, S., Kheddar, A., and Tempier, O. (2019). Stair climbing stabilization of the HRP-4 humanoid robot using whole-body admittance control. *IEEE International Conference on Robotics and Automation*.

[Caron et al., 2015] Caron, S., Pham, Q.-C., and Nakamura, Y. (2015). Stability of surface contacts for hu-manoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. *ICRA: International Conference on Robotics and Automation*, pages 5107–5112.

[Carpentier et al., 2019] Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., and Mansard, N. (2019). The pinocchio c++ library — a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. *IEEE*.

[Featherstone, 2008] Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Springer US.

[Flayols et al., 2017] Flayols, T., Del Prete, A., Wensing, P., Mifsud, A., Benallegue, M., and Stasse, O. (2017). Experimental evaluation of simple estimators for humanoid robots. *IEEE-RAS 17th International Conference on Humanoid Robotics*, pages 889–895.

[Herdt et al., 2010] Herdt, A., Diedam, H., Wieber, P.-B., Dimitrov, D., Mombaur, K., and Diehl, M. (2010). Online walking motion generation with automatic foot step placement. *Advanced Robotics*, 24:719–737.

[Herdt et al., 2003] Herdt, A., Perrin, N., and Wieber, P.-B. (2003). Walking without thinking about it. *International Conference on Robot Systems (IROS)*, pages 190–195.

[Kajita et al., 2009] Kajita, S., Hirukawa, H., Harada, K., and Yokoi, K. (2009). *Introduction à la commande des robots humanoïdes*. Springer-Verlag France, Springer Paris Berlin Heidelberg New York. Traduction française de Humanoid Robotics par Sophie Sakka.

[Kajita et al., 2003] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 2:1620–1626 vol.2.

[Kajita et al., 2010] Kajita, S., Morisawa, M., Miura, K., Nakaoka, S., Harada, K., Kaneko, K., Kanehiro, F., and Yokoi, K. (2010). Biped walking stabilization based on linear inverted pendulum tracking. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4489–4496.

[Kajita et al., 2001] Kajita, S., Yokoi, K., Saigo, M., and Tanie, K. (2001). Balancing a humanoid robot using backdrive concerned torque control and direct angular momentum feedback. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, 4:3376–3382 vol.4.

[Koenig and Howard,] Koenig, N. and Howard, A. Gazebo simulator. http://gazebosim.org/.

[Mansard and Chaumette, 2007] Mansard, N. and Chaumette, F. (2007). Task sequencing for sensor-based control. *IEEE Transactions on Robotics*, 23:60–72.

[Mansard et al., 2009] Mansard, N., Stasse, O., Evrard, P., and Kheddar, A. (2009). A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. *In International Conference on Advanced Robotics (ICAR)*.

[Naveau, 2016] Naveau, M. (2016). *Advanced human inspired walking strategies for humanoid robots*. PhD thesis, Universite Toulouse III Paul Sabatier.

[Naveau et al., 2017] Naveau, M., Kudruss, M., Stasse, O., Kirches, C., Mombaur, K., and Souères, P. (2017). A reactive walking pattern generator based on nonlinear model predictive control. *Robotics and Automation Letters*.

[Orin et al., 2013] Orin, D., Goswami, A., and Lee, S.-H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35.

[Pratt et al., 2006] Pratt, J., Carff, J., Drakunov, S., and Goswami, A. (2006). Capture point: A step toward humanoid push recovery. *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 200–207.

[Ramirez-Alpizar et al., 2016] Ramirez-Alpizar, I. G., Naveau, M., Benazeth, C., Stasse, O., Laumond, J.-P., Harada, K., and Yoshida, E. (2016). Motion Generation for Pulling a Fire Hose by a Humanoid Robot. *16th IEEE-RAS International Conference on Humanoid Robotics (HUMANOIDS 2016)*.

[Stasse, 2016] Stasse, O. (2016). Robot operating system introduction. Course support on ROS, unpublished.

[Stasse et al., 2009] Stasse, O., Evrard, P., Perrin, N., Mansard, N., and Kheddar, A. (2009). Fast foot prints re-planning and motion generation during walking in physical human-humanoid interaction. *9th IEEE-RAS International Conference on Humanoid Robots*, pages 284–289.

[Stasse et al., 2017] Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Mirabel, J., del Prete, A., Souères, P., Mansard, N., and et al., F. L. (2017). Talos: A new humanoid research platform targeted for industrial applications. *International Conference on Humanoid Robotics, ICHR, Birminghamn United Kingdom*.

[Stasse et al., 2018] Stasse, O., Naveau, M., and Kudruss, M. (2018). Walking for humanoid robots. Internal report.

[Stasse et al., 2008] Stasse, O., Verrelst, B., Wieber, P.-B., Vanderborght, B., Evrard, P., Kheddar, A., and Yokoi, K. (2008). Modular architecture for humanoid walking pattern prototyping and experiments. *Advanced Robotics*, 22:6–7.

[Takenaka et al., 2009] Takenaka, T., Matsumoto, T., and Yoshiike, T. (2009). Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1091.

[Takenaka et al., 2009] Takenaka, T., Matsumoto, T., Yoshiike, T., Hasegawa, T., Shirokura, S., Kaneko, H., and Orita, A. (2009). Real time motion generation and control for biped robot-4(th) report: Integrated balance control. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1601 – 1608.

[Wieber, 2002] Wieber, P.-B. (2002). On the stability of walking systems. *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*.

A Entity Graph of the Off-line simulation

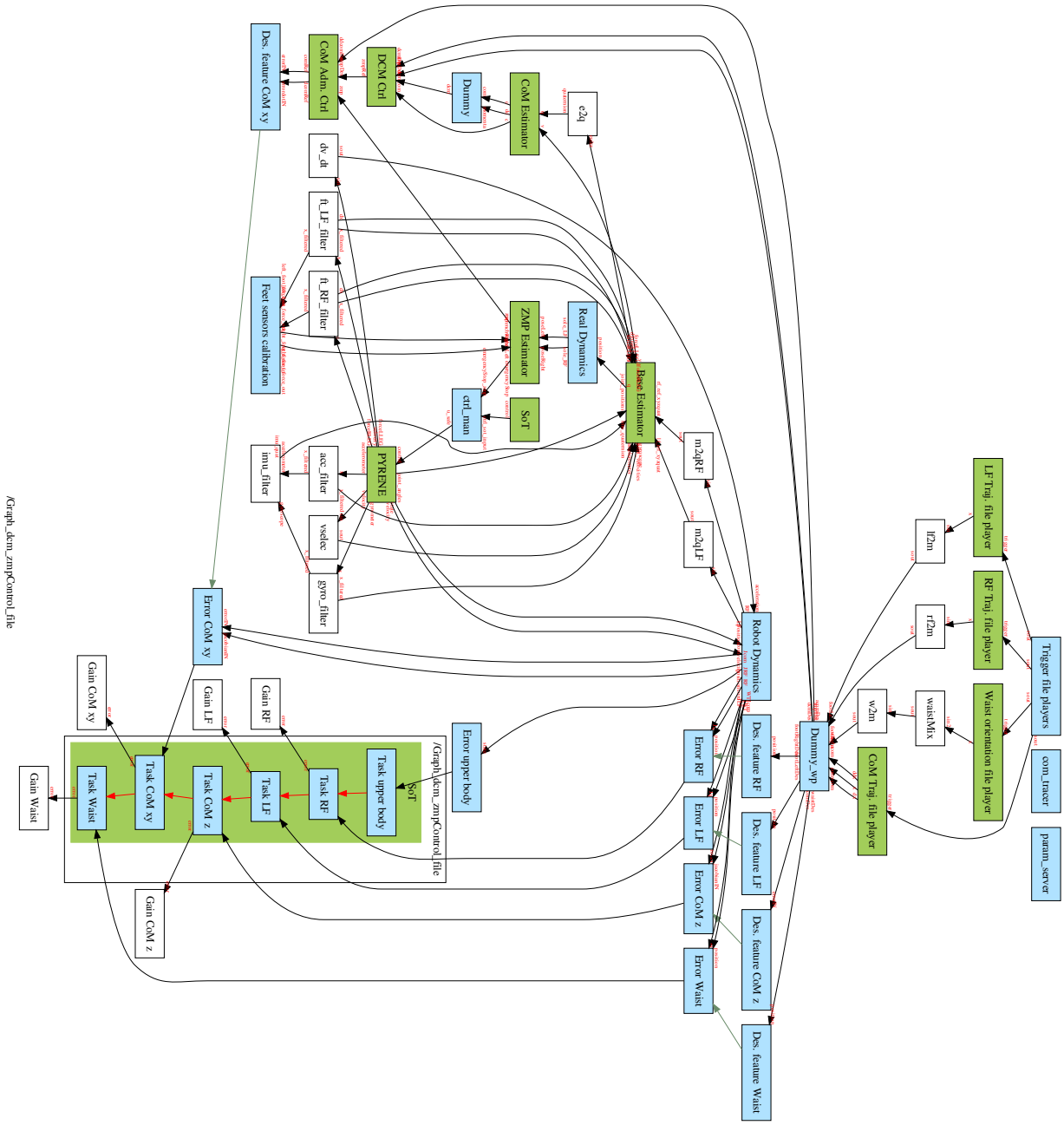
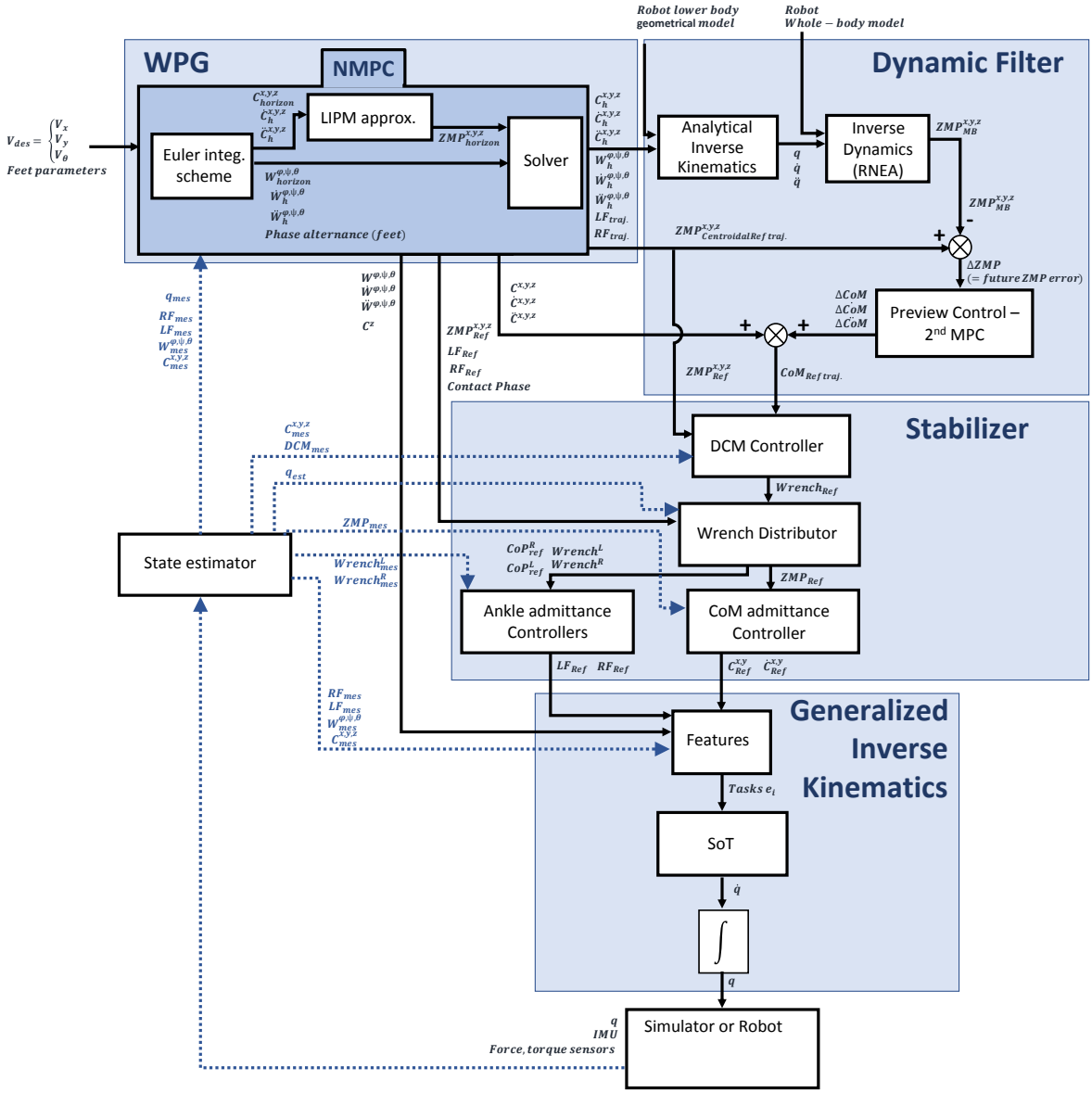


Figure 23: *Entity Graph of the Off-line simulation. This is the initial state of the simulation that was used to build the online simulation. The PatternGenerator entity is here replaced by several entities reading trajectory files computed off-line.*

B About the final online simulation with ankle admittance controllers

B.1 Control scheme



Parameters used

C: Center of Mass position
W: Waist orientation
ZMP: Zero Moment Point position
CoP: Center of Pressure on each foot
LF, **RF**: Left and Right Foot position
q: control vector
DCM: Divergent Component of Motion

mes stands for measured or estimated
h stands for horizon (values over the entire preview horizon)
Ref: reference or desired

IMU: Inertial Measurement Unit

Equations and implementation details

- Euler Integ. Scheme: eq. 5, Section 2.2.1
- LIPM approx.: eq. 3 (2.1.1), 5 to 7 (2.2.1)
- Solver: eq. 8-12, Section 2.2.1
- Analytical IK: eq. 13, Section 2.2.2
- ID: eq. 14, Section 2.2.2
- Generalized IK: eq. 15 to 21, Section 2.2.3
- Wrench distributor: Section 4.4.1
- Ankle Admittance Controllers: eq. 31 , Section 4.2.1

Figure 24: *Final Control scheme of the online simulation with the wrench distribution and the ankle controllers.*

B.2 Entity Graph

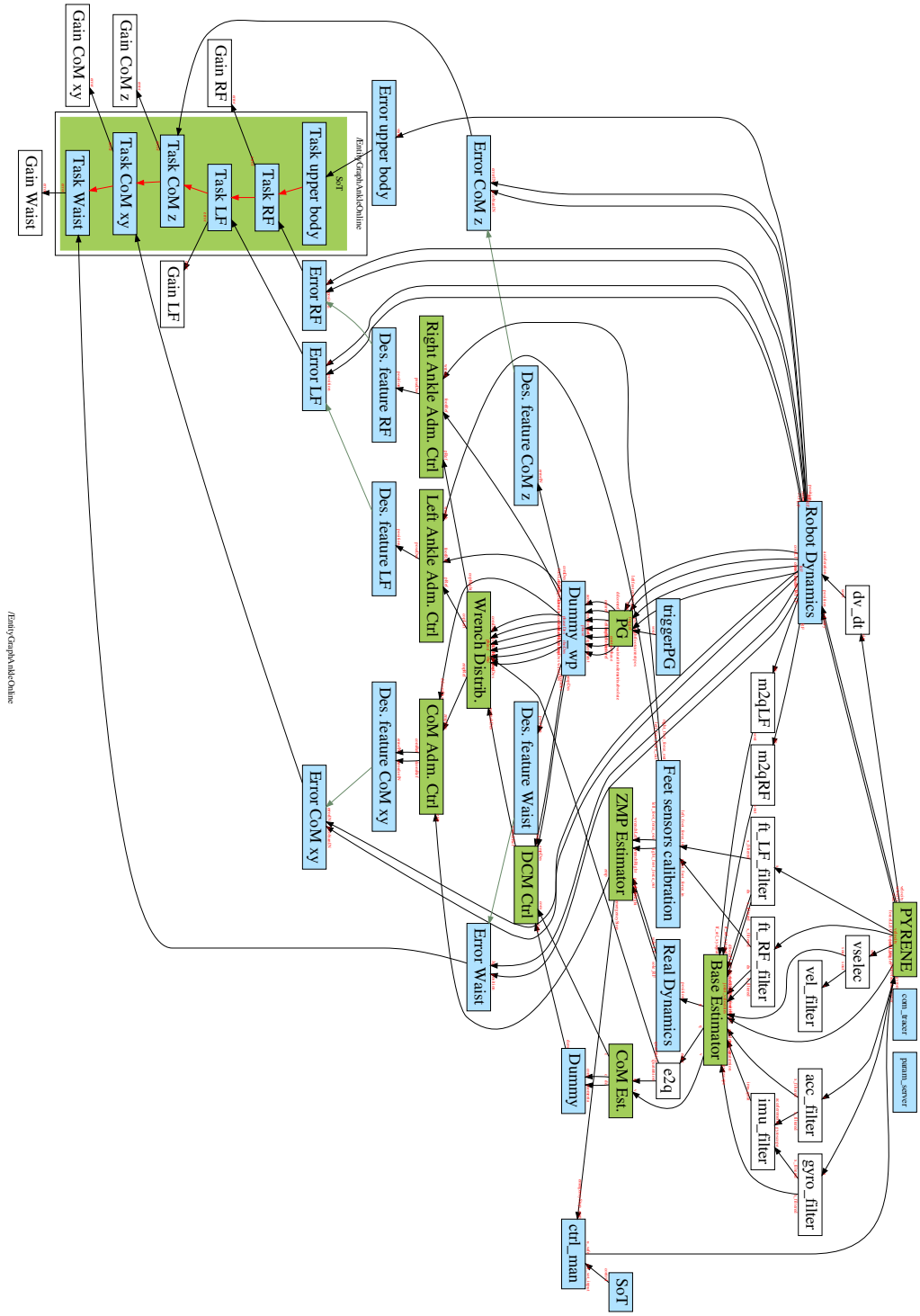


Figure 25: *Entity Graph of the final online simulation with Ankle Admittance Controllers. Each controller computes new reference positions for one foot, which is transmitted to the SoT instead of the reference initially computed by the PG in the online simulation without this part of the stabilizer.*

C Gantt Diagram

The Diagram on Figure 26 shows the schedule of the internship. It is very close to the originally planned one.

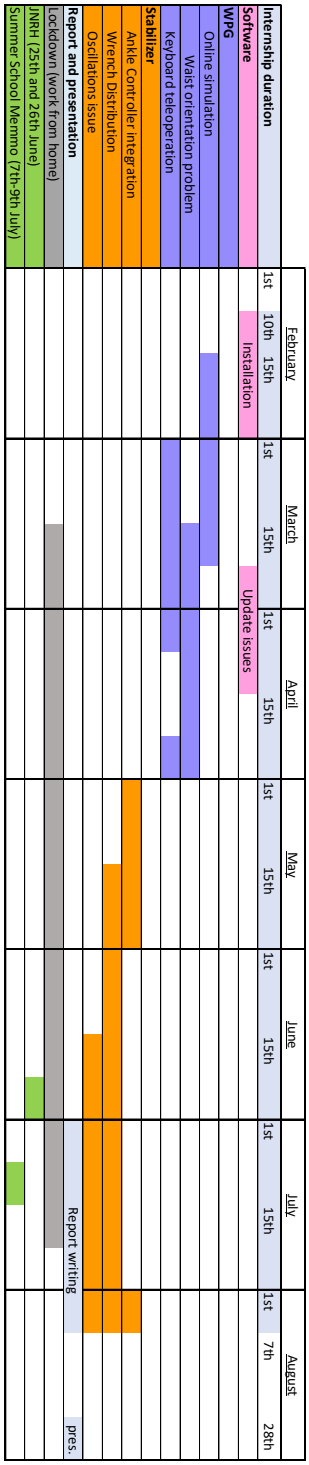


Figure 26: *Gantt Diagram. Schedule of the internship. Some events have been added, such as the JNRH (*Journées Nationales de la Robotique Humanoïde*) and the Summer School MEMMO, as well as the lockdown period during which work was conducted from home. I was back in the laboratory during the last three weeks of internship.*

Résumé

Si les robots humanoïdes existent en pensée depuis plusieurs siècles et peuvent être réellement construits depuis plusieurs décennies, reproduire les mouvements humains sur ces robots présente aujourd'hui encore de nombreux défis. En particulier, le problème de la locomotion bipède requiert des modèles mécaniques et dynamiques complexes, ainsi que des outils de contrôle et de résolution numériques très performants. Le passage de la simulation au robot réel se révèle également crucial. Le stage présenté dans ce rapport se concentre sur la locomotion du robot humanoïde Pyrène (série Talos de la société PAL-Robotics). Il s'inscrit dans un travail d'adaptation des schémas de contrôles développés et validés pour un précédent robot humanoïde sur ce nouveau robot Pyrène. Le schéma de contrôle comporte notamment un générateur de mouvements, un stabilisateur de marche et un solveur de cinématique inverse générant la commande en position fournie au robot. Au cours du stage, un premier travail sur le générateur de mouvements à permis de concevoir une simulation de téléopération du robot en temps réel. Ensuite, divers développements sur le stabilisateur de marche du robot ont été menés, qui visent à intégrer une stratégie de stabilisation supplémentaire. Il s'agit d'ajouter un contrôleur en admittance au niveau des chevilles du robot, permettant d'adapter l'orientation de celles-ci au moment de l'atterrissage du pied, afin que le robot puisse s'adapter à son environnement réel. Plusieurs problèmes de modélisation du robot ont été soulevés dans le schéma de contrôle. Ils ont été partiellement résolus pendant le stage. La nouvelle stratégie de stabilisation est aujourd'hui intégrée dans le schéma de contrôle et partiellement validée.

Summary

If the idea of human-shaped robots is several centuries old and if humanoid robots can be built for real for some decades, reproducing human motions on these robots is still very challenging today. In particular, the bipedal locomotion problem requires complex mechanical and dynamical models, as well as very powerful control and numerical solving tools. The simulation to reality gap is also a crucial step. The internship presented in this report focuses on the locomotion of the humanoid robot Pyrène (Talos series from the company PAL-Robotics). It serves an objective of adaptation of the control scheme that had been developed and validated on a former humanoid robot, on this new robot Pyrène. The control scheme includes a motion generator, a walking stabilizer and an inverse kinematics solver that generates the command in position transmitted to the robot. During the internship, work was first conducted on the motion generator. It allowed to design a simulation for the real-time teleoperation of the robot. Then, diverse developments on the walking stabilizer have been made, that aimed at integrating an additional stabilizing strategy. It involves adding an admittance controller at the level of the ankles of the robot, allowing it to adapt its ankle orientation when landing the foot, so that the robot can adapt to its real environment. Issues on the modelling of the robot in the control scheme have been raised. They have been partly solved during the internship. The new stabilizing strategy is now integrated in the control scheme and partially validated.