Sapienza Università di Roma

Underactuated Robots

Final Project

Simulation of a 2D Hopping Robot.

Authors:
Fabian Lopez *(1874590)*
Michele Ciciolla *(1869990)*
Jose Bustamante *(1161750)*

Lecturers:
Prof. Leonardo Lanari
Prof. Giuseppe Oriolo

October 27, 2020

Sapienza
Università di Roma

Introduction

In this project, the authors are aimed to simulate the behavior of a 2D underactuated hopping robot, with just one leg, under the action of two decoupled controllers: the first one for the manipulation of the hopping height on the vertical plane and the second one for the control of motion on the horizontal plane. The dynamic model of the hopping robot and the control algorithms are implemented based on the work [1] by Raibert. In that work, Raibert presents a detailed description of the *one-legged hopping* model, characterizes its behavior and introduces four control algorithms, namely, one algorithm to regulate hopping height and three algorithms for achieving balance while hopping in place and running.

This report shows the results of the implementation of the energy based vertical controller and the foot placement and leg sweeping algorithms for horizontal control. Since the cyclic behavior of the hopping system can be represented by using a state machine, the controller has been implemented as an event-driven system with a well defined behavior for each state of the hopping cycle. The implementation has been carried out in Simulink with the aid of the Stateflow library. All the simulation results shown in this report are accompanied with their corresponding SIMULINK and MATLAB live script files so that the reader can verify the results.

1 The Model

In legged systems, legs typically perform two actions during locomotion: varying their own length and changing their orientation with respect to the body. The variation of leg length is performed in order to achieve upward and backward body propulsion, to cushion landings, and to reduce its own moment of inertia and increase its clearance when swung forward. The lengthening and shortening of a leg during these activities is a dynamic action governed by the resonant interaction of three components: leg compliance, body mass and gravity [2]. In humans legs, springy muscles and tendons store and retrieve energy when leg is shortened and lengthened, respectively. Moreover, the propulsion produced by legs swinging back and forth permits the precise location of feet with respect to the center of gravity and the angular momentum variation.

For the development of this project, the model used for the simulations is shown in Fig. 1, which consist of a single springy leg linked to a rigid mass (the *body*) through a hinge-type hip. The leg mass and moment of inertia are M_1 and I_1, respectively, while the same parameters for the body are denoted as M_2 and I_2. The leg center of mass is located a distance r_1 from the lower tip of the leg (the *foot*) and the body center of mass is located a distance r_2 above the hip.

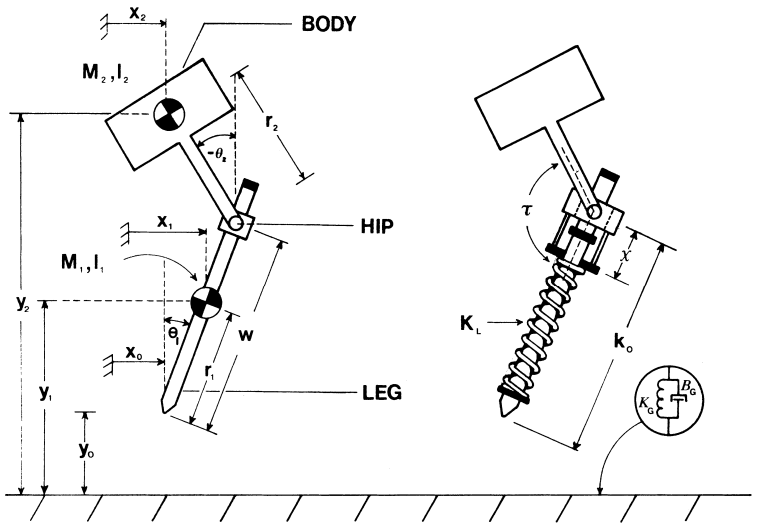


Figure 1: Planar one-legged model used for simulations.

In order to position the leg or body, a control torque τ applied between the body and the leg at the hip is defined as

$$\tau(t) = K_P(\theta_1 - \theta_{1,d}) + K_V(\dot{\theta}_1) \tag{1}$$

where $\theta_{1,d}$ is the desired leg angle and K_P and K_V are feedback gains. The same feedback rule (1) is applied during both stance and flight periods, but with different gains K_P and K_V for each case, as specified in (2).

$$K_P = \begin{cases} 1800 \text{ Nt-m/rad} & \text{for } y_0 \leq 0 \\ 1200 \text{ Nt-m/rad} & \text{otherwise} \end{cases}, \qquad K_V = \begin{cases} 200 \text{ Nt-m-s/rad} & \text{for } y_0 \leq 0 \\ 60 \text{ Nt-m-s/rad} & \text{otherwise} \end{cases} \tag{2}$$

The overall length of the leg is influenced by a spring with one end rigidly connected to the foot, a position actuator fastened to the other end of the spring, and a mechanical stop representing a very stiff spring with damping acting between the foot and one side of the position actuator. The spring and mechanical stop are arranged such that only the spring generates forces when $(w - \chi) < k_0$ and only the mechanical stop generates forces when $(w - \chi) > k_0$, with w and k_0 representing the leg spring length in motion and at rest, respectively, and χ being the length of the position actuator. Therefore, these two scenarios can be modeled by a force F_K acting at the hip, in direction tangent to the leg, described by

$$F_K = \begin{cases} K_L(k_0 - w + \chi) & \text{for } (k_0 - w + \chi) > 0 \\ K_{L2}(k_0 - w + \chi) - B_{L2}\dot{w} & \text{otherwise} \end{cases} \tag{3}$$

where K_L is the spring stiffness, and K_{L2} and B_{L2} are the stiffness and damping coefficients of the mechanical stop.

Changes in length χ of the position actuator, which is acting between the spring and the hip, can do work on the leg spring to produce an increase or reduction of its stored energy. Furthermore, the leg spring can also absorb energy when it shortens under load of the body, and deliver energy when it lengthens during the upward acceleration of the body.

The position actuator is considered to have a finite response time so that the increase and decrease of the length χ is assumed to follow a quadratic trajectory

$$\chi(t) = \chi_0 + kt^2, \tag{4}$$

with χ_0 being the initial length of position actuator and k a time constant. Moreover, the distance traveled by the actuator in motion (known as the *stroke*) is limited to $\chi_{min} < \chi < \chi_{max}$, with $\chi_{min} > 0$.

The support surface is modeled as a two-dimensional spring K_G and damper B_G, compounded of decoupled vertical and horizontal components. The influence of the surface spring and damper on the hopper appears only when the foot is in contact with the ground, *i.e.*, when the condition $y_0 < 0$ is satisfied, so that the coefficients of spring and damper are zero during flight. Thus, the force exerted by the support surface on the hopper can be represented by two decoupled forces F_x and F_y as

$$F_x = \begin{cases} -K_G(x_0 - x_{TD}) - B_G\dot{x}_0 & \text{for } y_0 \leq 0 \\ 0 & \text{otherwise} \end{cases}, \qquad F_y = \begin{cases} -K_G y_0 - B_G\dot{y}_0 & \text{for } y_0 \leq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

At this point, it is important to remark that the original work [1] present typos in the description of some equations, which leads to the malfunctioning of the simulations. The first one is related to the definition of F_y, which in the original paper uses $B_G y_0$ instead of $B_G\dot{y}_0$, as correctly written in (5).

In order to inject energy into the system during the the hopping cycles, it is necessary to lengthen the position actuator when leg is providing support and shorten it during flight. A removal of energy can also be achieved by changing the phase of these actions, *i.e.*, by shortening the position actuator during support and lengthening it during flight.

The equations of motion for the hopper are described by a system of nonlinear coupled differential equations, as derived in Appedix I of [1]. We would like to remark the presence of typos in these equations, particularly, on the right hand side of equation (43), where the sign of the second summand should be positive and on the term $I_2 w\ddot{\theta}_2$ in equation (44) which is actually $I_2 W\ddot{\theta}_2$.

1.1 The Hooping Cycle and the State Machine

In order for a legged system to balance and to make forward progress the controller must be able to induce periods of support, in which the leg is loaded so that the foot remains immobile, and periods during which the leg is unloaded and then a free movement of the foot can be executed. The alternation of these two periods generate a *hopping cycle*, which for the one-legged robot consists of four well-defined events:

- **Lift-Off:** The moment at which the foot loses contact with the ground.
- **Top:** The moment in flight when the body has peak attitude and vertical motion changes from upward to downward.
- **Touchdown:** the moment the foot makes contact with the ground.
- **Bottom:** The moment in stance when the body has minimum altitude and vertical motion of the body changes from downward to upward.

Since these events follow a regular cyclic progression, a *finite state machine* with four distinct states can be used to represent the hopping cycle such that the state variables of the system can be used to determine the current hopper state and the transition from one state to another.

In this way, aiming at performing simulations for the hopping robot, the whole controller can be seen as an *even-driven system* which can be implemented by using the *Stateflow* library of MATLAB. Particularly, in order to implement the control logic we make use of a *Chart* element which receives as inputs the necessary state variables in order to both decide the transitions between states and to compute the control inputs required to achieve balance and translation of the robot. A schematic of the complete feedback system can be seen in Fig.

2, where the two main components, namely the controller and the system model, are depicted. The complete structure and internal components of the state machine and the model can be found in any of the accompanying *Simulink* files.

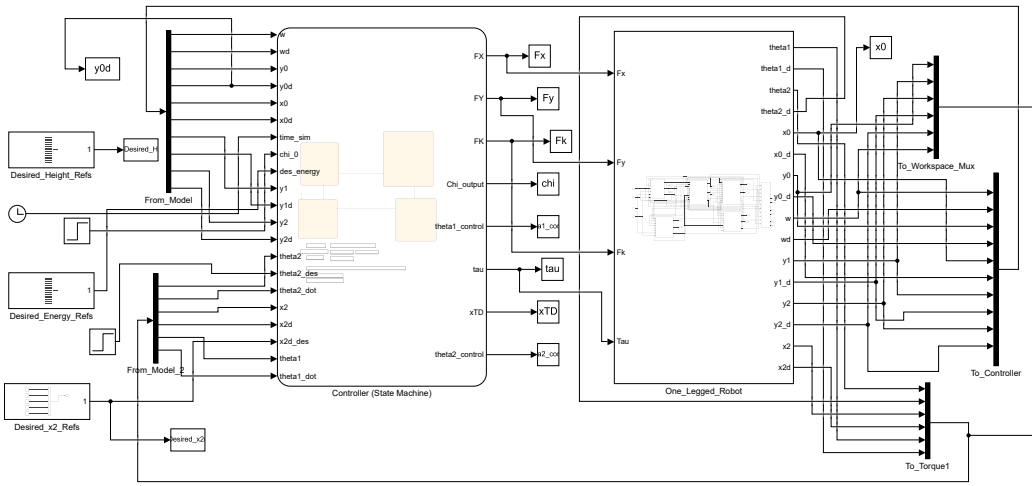


Figure 2: Feedback control for the hopping robot.

2 Vertical Control

The aim of the *Vertical Controller* is to generate stable resonant oscillations that cause the hopper to hop off the ground (thus initiating hopping) and then to ensure the correct tracking of different set-point heights. These tasks can be accomplished by measuring and regulating the system's energy. The vertical controller is a crucial component of the whole controller since it establishes a regular cycle of activity within which horizontal and attitude control can take place.

The basic working principle of the vertical controller is as follows. First of all, based on the desired hopping height H, the controller calculates the required energy to reach the desired height and, in addition, it computes an approximation of the current system's energy. By using this calculations, the controller determines how much energy must be injected into (or removed from) the hopping system and subsequently actuates the position actuator such that the difference between the desired energy and the current energy is compensated. This whole procedure is carried out once every hopping cycle while the robot is on the ground.

2.1 Controller Overview

The first task of the vertical controller is to compute the difference between the desired hopping height E_H and the energy of the next flight E_{FLIGHT} such that the desired energy change $\Delta E_H = E_H - E_{FLIGHT}$ can be calculated. The expressions for calculating these two quantities are given by (6) and (7), from which it can be noticed that E_H is in the form of potential energy while E_{FLIGHT} consists of gravitational potential, kinetic and elastic potential energy components.

$$E_H = M_1 g(H + r_1) + M_2 g(H + k_0 + r_2). \tag{6}$$

$$E_{FLIGHT} = \frac{M_1}{M_1 + M_2}\left(M_1 g y_1 + M_2 g y_2 + \frac{1}{2}M_1 \dot{y}_1^2 + \frac{1}{2}M_2 \dot{y}_2^2 + \frac{1}{2}K_L(k_0 - w + \chi)^2 + \frac{1}{2}K_G y_0^2\right). \tag{7}$$

Once the desired energy change ΔE_H is obtained, the controller calculates the change in position actuator $\Delta\chi$ required to inject (or remove) the desired amount of energy into (or from) the system. Thus, the position of the linear actuator must be extended by an amount determined by

$$\Delta\chi = -(\chi - w - k_0) + \sqrt{(\chi - w - k_0)^2 + \frac{2\Delta E_H}{K_L}}. \tag{8}$$

2.2 Simulation Results

In order to verify the performance of the vertical controller proposed in [1], we have implemented it in Simulink by using the Stateflow library. Fig. 3 depicts the behavior of the hopping system under the action of the vertical controller for a desired foot height of $H = 0.5$ m. It can be noticed how the hopping robot, starting at rest, gradually increases its height to the point of reaching the desired value. After this, a stable hopping behavior is established such that the hopping height is maintained at the desired level.

During the initial instants of time, we can see that to initiate hopping the actuator length $\chi(t)$ considerably increases its value so that the necessary energy to reach the desired vertical energy is injected to the system. Once the stable hopping behavior is attained, the length of the actuator is reduced as the actuator's task becomes only to periodically inject the energy required to maintain hopping.

Now, the last 4 seconds of data from Fig. 3 are replotted in the phase plane shown in Fig. 4. During flight, the constant gravitational acceleration produces a curve resembling a parabolic trajectory, while for the stance period, the mass-spring system causes an harmonic motion. An interesting detail to be remarked from Fig. 4 is the rough part of the curves between LIFT-OFF and TOP which is caused by the damped vibration occurred when the mechanical stop is hit.

The vertical energy of the system is computed for two cycles of fixed height hopping (from $t = 3.686$ s to $t = 5.751$), resulting in the plot depicted in Fig. 5a. For a lossless system the total energy would be represented by a flat energy line but since the hopping system indeed present losses, the total energy curve is not uniform.

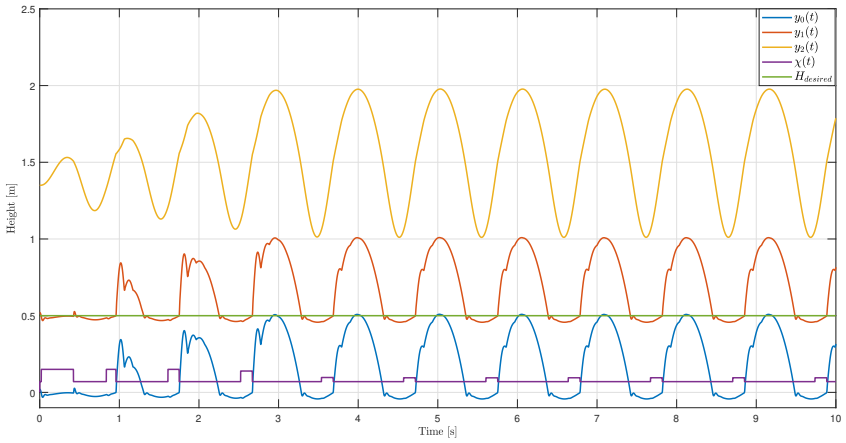


Figure 3: Vertical hopping.

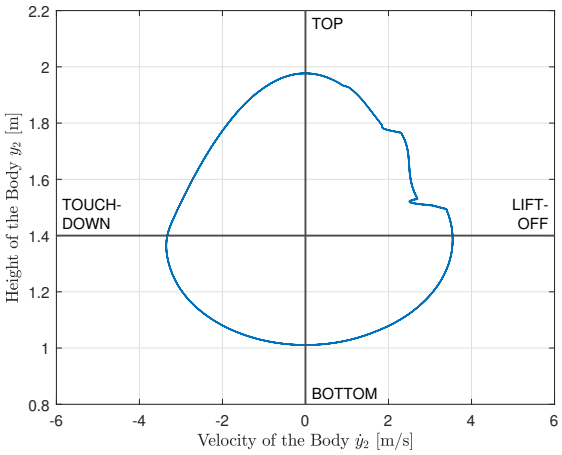
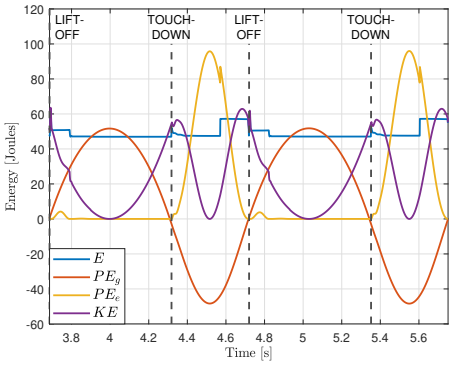
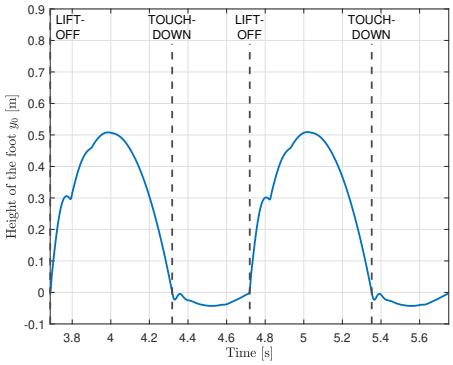


Figure 4: Phase portrait for vertical hopping.

The increase of energy resulting from the actuator lengthening can be seen during the latter part of stance, prior to LIFT-OFF.



(a) Vertical energy.

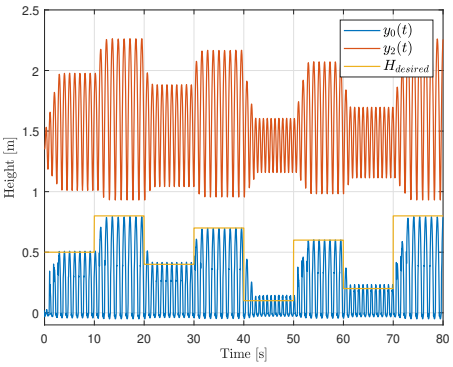


(b) Elevation of foot.

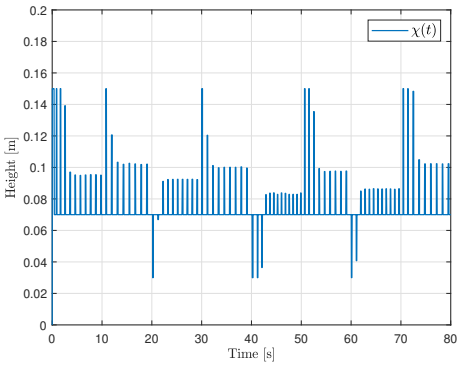
Figure 5: Vertical energy for two hopping cycles at constant hopping height.

Finally, an 80 seconds simulation sequence of vertical hopping for different height set-points is shown in Fig. 6. At times $t = 0$, 10, 20, 30, 40, 50, 60, 70 the corresponding desired foot hopping heights are $H = 0.5$, 0.8, 0.4, 0.7, 0.1, 0.6, 0.2, 0.8. From Fig. 6a, we can see that the foot height properly reaches the desired heights and maintains the elevation over time, thus showing the correct performance of the implemented vertical controller for diverse set-points. Notice that in order to reduce the hopping height, as commanded at times $t = 20$, 40 and

60, the controller initially makes the position actuator to reduce its length (see Fig. 6b) so that energy from the system is removed through active damping. Furthermore, since the maximum position actuator is limited by χ_{max}, the energy that could be injected on a single hopping cycle is limited so that the number of hopping cycles required to achieve the desired hopping height increases as the difference between the current and desired heights increases, as evidenced for the height changes at $t = 50$ and 70.



(a) Foot and hip elevation.



(b) Length of the actuator.

Figure 6: Vertical hopping sequence.

3 Horizontal Control

The *Horizontal Control* addresses the problems of balancing and traveling control for the hopping system such that the following three main tasks must be accomplished by the controller:

- Ensuring that unwanted horizontal motions are not present.
- Generating horizontal motions of adequate velocity when necessary.
- Preventing the locomotion system to tip over.

Foot placement and hip motion of the hopping system can be manipulated through the horizontal controller with the aim of controlling balance and horizontal travel. In particular, tipping and horizontal motion of the hopping system can be influenced by gravity force when the foot is placed with respect the center of gravity. Such foot placement can be adjusted during the flight period to influence attitude and translation during the subsequent stance interval. On the other hand, the pattern of hip motion between the body and leg during stance influences the systems's total angular momentum.

In this project, we implement two of the three algorithms for balance and travel control presented in [1]. Firstly, we implement the *Foot Placement* method in which the horizontal control relies solely on foot placement so that no hip motion is executed during stance. Then, based on the first method, the implementation of the so-called *Leg Sweeping* method, which is an improved foot placement algorithm combined with hip motion during stance, is presented.

3.1 Method 1: Foot Placement

In order to influence the translation and tipping of the hopping system, the hip angle can be manipulated during flight in such a way that the robot's foot touches the ground in a predefined horizontal position. In Method 1, the placement of the foot is determined by two factors: the projection of center of gravity x_{CG} and an error function of state variables x_{ERR}.

3.1.1 Controller Overview

First, the horizontal position of the center of gravity x_{CG}, expressed in a coordinate system that translate with the hip, is calculated as

$$x_{CG} = \frac{(r_1 - w)M_1 \sin(\theta_1) + r_2 M_2 \sin(\theta_2)}{M_1 + M_2}. \tag{9}$$

Then, a corrective feedback x_{ERR}, based on a linear combination of state errors, is computed as follows

$$x_{ERR} = K_1(\dot{x}_2 - \dot{x}_{2,d}) + K_2(\theta_2 - \theta_{2,d}) + K_3(\dot{\theta}_2) \tag{10}$$

where $\dot{x}_{2,d}$ and $\theta_{2,d}$ are the desired values for $\dot{x}_2$ and θ_2, respectively, and K_1, K_2 and K_3 are feedback gains. At TOUCHDOWN, the foot of the hopping system is required to be placed in a horizontal position defined by $x_{TD} = x_{CG} + x_{ERR}$, which by using (9) and (10) becomes

$$x_{TD} = w\frac{r_2 M_2 \sin(\theta_2) + (M_1 + M_2)x_{ERR}}{r_1 M_1 + w M_2}. \tag{11}$$

Taking into account the kinematics of the model, we can arrive to the following expression that determines the desired value of θ_1

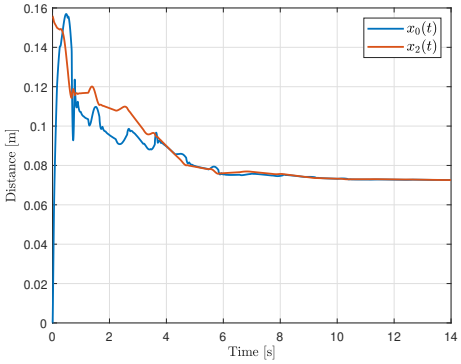
$$\theta_{1,d} = -\arcsin\left[\frac{r_2 M_2 \sin(\theta_2) + (M_1 + M_2)x_{ERR}}{r_1 M_1 + w M_2}\right]. \tag{12}$$

Finally, the control torque given in (1) is used to move the leg to the desired angle $\theta_{1,d}$ given by (12). If instead of controlling $\dot{x}_2$ we want to control x_2, a position controller of the form (13) can be implemented to accomplish this task.

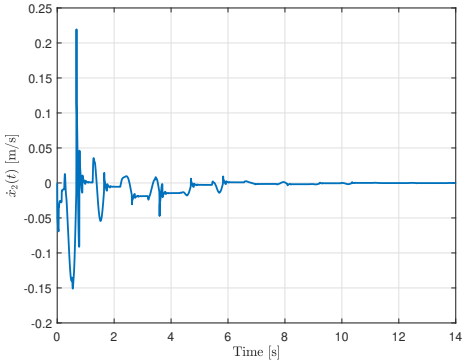
$$\dot{x}_{2,d} = \min\{K(x_2 - x_{2,d}), \dot{x}_{2d,\max}\}. \tag{13}$$

3.1.2 Attitude Correction: Simulation Results

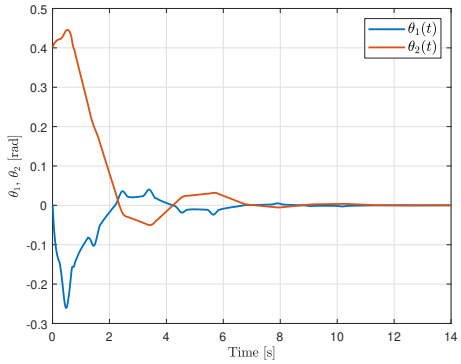
As a first simulation scenario, the implemented foot placement algorithm is required to correct an error in body attitude introduced through the initial conditions. At $t = 0$ we have selected an initial error in body attitude of 0.4 rad ($\theta_2 = 0.4$) and the initial desired height is $H = 0.4$. The correct performance of the horizontal controller based on foot placement can be seen in Fig. 7. In Fig. 7c we can see how the body attitude θ_2 is gradually reduced to the point of reaching a zero value after about 7 hops (approximately 8 seconds). As shown in Fig. 7d, while the horizontal control ensures the balance of the robot, the vertical control works in parallel such that the desired height is attained, even if H is changed as required at $t = 5$ s. Fig. 7b shows that, in order to correct the attitude error, the velocity of the body is required to change continuously and once the robot is balanced its velocity returns to the desired value $\dot{x}_2 = 0$ so that it remains still. Since in this scenario horizontal position is not controlled, the transitory variations in horizontal velocity cause a displacement of the leg with respect to the origin, as shwon in Fig. 7a.



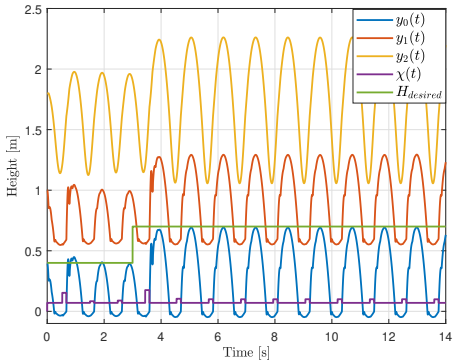
(a) Body and foot displacement.



(b) Body velocity.



(c) Body and leg attitude.

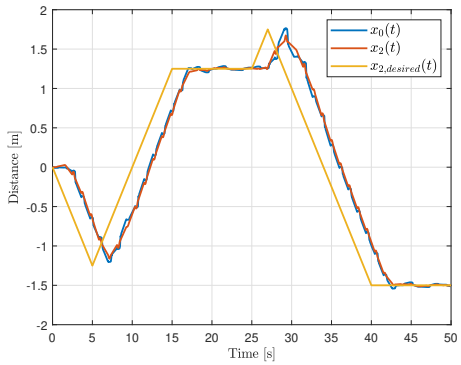


(d) Elevation of the hopping system.

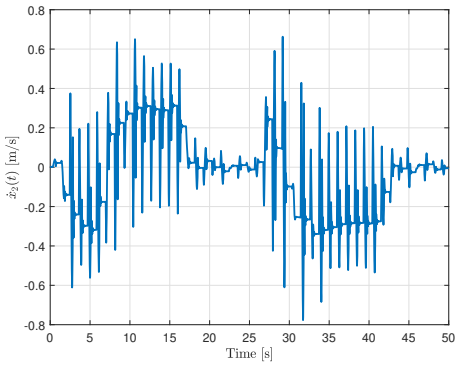
Figure 7: Body attitude correction.

3.1.3 Position Control: Simulation Results

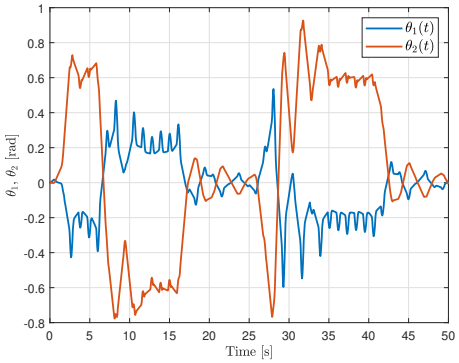
In this second case, the foot placement algorithm is used to follow a desired horizontal position $x_{2,d}$ with the aid of the position controller (13). The response of the foot placement algorithm to changes in the desired horizontal position can be seen in Fig. 8. The desired horizontal position is represented by a curve that continuously changes the set-point $x_{2,d}$, so that the hopping system is required to track a desired trajectory as shown in Fig. 8a. It can be noticed that, although not immediately, the hopping system travels through all the desired trajectory, which shows a correct performance of the implemented foot placement algorithm. From Fig. 8c, we can see that balance of the robot is maintained at any instant of time through fast coordinated variations of body and leg angles. Moreover, as depicted in Fig. 8b, the body velocity $\dot{x}_2$ changes accordingly to the desired horizontal position, becoming zero when the robot is required to remain still.



(a) Body and foot displacement.



(b) Body velocity.



(c) Body and leg attitude.

Figure 8: Horizontal position control.

3.2 Method 2: Leg sweeping

The algorithm implemented in this section is based on a generalization of the foot placement approach. As for the foot placement algorithm, the *Leg Sweeping* algorithm places the foot to control balance and tipping but in addition it controls forward velocity by manipulating the hip angle during stance.

The idea behind the leg sweeping algorithm is to make the leg to reproduce a motion that is symmetrical about the point halfway through stance, as shown in Fig. 9. During this symmetrical motion, the center of gravity spends about an equal time both in front of the support point and behind it which results in a zero average gravitational tipping moment throughout stance, thus avoiding tipping of the robot.

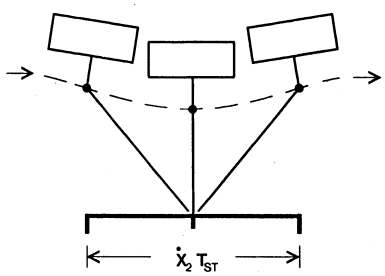


Figure 9: Symmetrical motion for the one-legged robot.

Therefore, one of the objectives of the leg sweeping controller is to determine the locus of points over which the center of gravity will travel during the next stance period. This locus is called a *CG print* and its length is simply the product of the forward velocity $\dot{x}_2$ and the duration of stance T_{ST}. Thus, the idea is that in steady state the controller places the foot exactly in the center of the CG print. However, when attitude deviations are present, the controller places the foot forward or behind the center of the CG print to correct the error based on a linear combination of state errors.

The system's forward velocity depends on the forces arising between the ground and the foot and can be

influenced by the controlled motion of the hip. If the hopping system is required to follow a constant forward velocity, then the horizontal components of all the forces acting on the system must sum zero. This task is accomplished by the sweeping algorithm which calculates a target angle for the hip at each moment during stance such that, under nominal conditions, the resultant force acting on the foot is vertical. However, when deviations of the forward velocity from its desired value are present then the controller induce accelerating or retarding forces in order to correct the velocity error.

3.2.1 Controller Overview

For the implementation of the leg sweeping algorithm, it is required to have prior knowledge of the duration of stance T_{ST}, the horizontal velocity of the body $\dot{x}_2$ and the geometry of the mechanical system so that the an appropriate leg angle for TOUCHDOWN and a sweeping function for the leg can be calculated. In our implementation, the state machine facilitates obtaining the duration of stance at each hop cycle which is calculated by saving a time point t_{TD} when the system enters the TOUCHDOWN-BOTTOM state and subtracting it to a time point recorded when the system exits the state BOTTOM-LIFTOFF. Therefore, the horizontal distance traversed during stance (the length of the CG print) is

$$\Delta x_{STANCE} = \dot{x}_2 T_{ST}. \tag{14}$$

Combining (14) with (11), we obtain the new equation for foot placement as

$$x_{TD} = w\frac{r_2 M_2 \sin(\theta_2) + (M_1 + M_2)x_{ERR}}{r_1 M_1 + w M_2} + \frac{\Delta x_{STANCE}}{2}. \tag{15}$$

Now, by using the information provided by (14) and (15), the sweeping function at time t during stance can be represented by

$$x(t) = x_{TD} - \frac{(t - t_{TD})}{T_{ST}}\Delta x_{STANCE}. \tag{16}$$

Finally, taking the kinematics of the robot into account, the desired leg angle during stance can be found to be

$$\theta_{1,d}(t) = -\arcsin\left(\frac{x(t)}{w}\right) \tag{17}$$

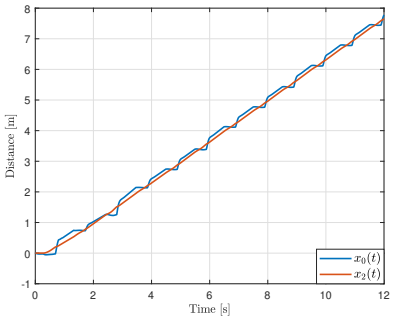
To accommodate forward acceleration, a compromise between the actual and desired velocity can be used to calculate the length of the CG print as follows

$$\Delta x_{STANCE} = \begin{cases} (\dot{x}_2 - \Delta\dot{x}_{\max}) & \text{for } \dot{x}_{2,d} < (\dot{x}_2 - \Delta\dot{x}_{\max}) \\ (\dot{x}_2 + \Delta\dot{x}_{\max}) & \text{for } \dot{x}_{2,d} > (\dot{x}_2 + \Delta\dot{x}_{\max}) \\ \dot{x}_{2,d} T_{ST} & \text{otherwise} \end{cases} \tag{18}$$

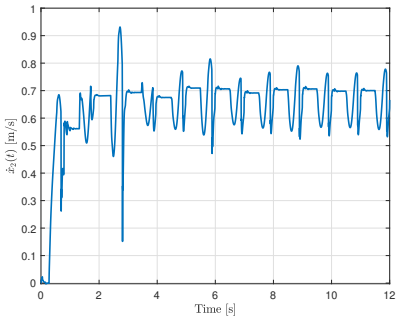
with $\Delta\dot{x}_{\max}$ being a limit to the magnitude of sudden changes in desired velocity.

3.2.2 Running at Constant Horizontal Rate: Simulation Results

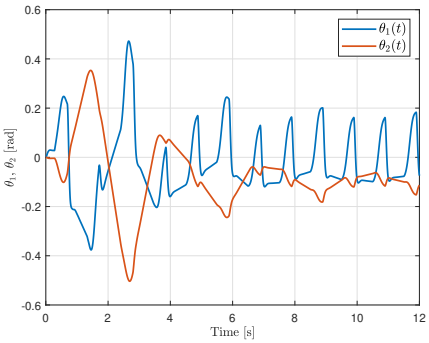
Aiming at testing the implementation of the sweeping algorithm, we have set an scenario in which the robot is required to execute an horizontal translation at 0.7 m/s. The simulation results for this scenario can be seen in Fig. 10. In particular, Fig. 10b shows that the horizontal velocity is properly controlled presenting only transitory variations arising from the inevitable required changes in the body angle θ_2. The distance traveled by the robot is shown in Fig. 10a where the curve for x_2 evidences the regularity of the body movement while the curve for x_0 presents periodical irregularities representing the stance period in which the foot remains still. On the other hand, Fig. 10c depicts the evolution of body and leg angles produced by the sweeping algorithm in order that the center of gravity of the legged systems follow the CG print. Specifically, during the initial instants in which the velocity is gradually increased (approximately up to $t = 4$) , the controller mainly procures the stabilization of the system so that the CG print motion is not strictly satisfied. However, when the system reaches the steady state, the controller focuses on the placement of the foot on the center of the CG print such that the CG print trajectory can be followed as much as possible. This fact can be evidenced looking at Fig. 10c for $t > 5$, in which we can see that the leg angle θ_1 (controlled by the sweeping algorithm during stance) presents periodical transitions from negative to positive values which represents the back and forth swing motion characterizing the desired movement for the leg sweeping method (see Fig. 9).



(a) Body and foot displacements.



(b) Body velocity.



(c) Leg and body attitudes.

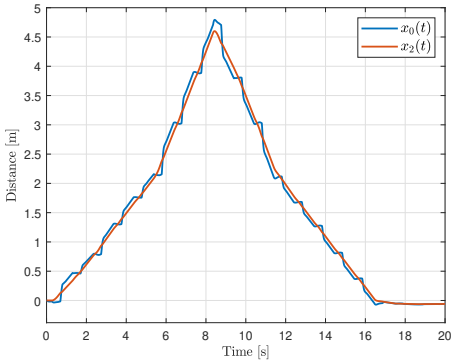
Figure 10: Running at constant horizontal rate.

3.2.3 Running at Varying Horizontal Rate: Simulation Results

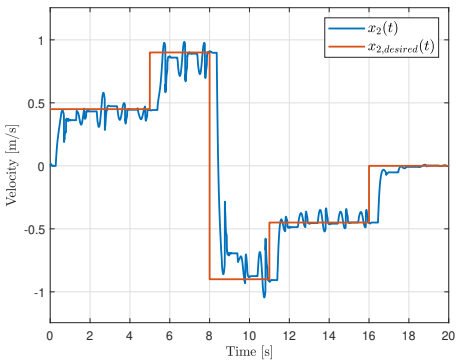
In this last simulation case, we show the correct performance of the sweeping algorithm for regulating forward velocity when the system is required to accelerate. Fig. 11 shows the results for a scenario in which the hopping system is initially required to accelerate to 0.45 m/s and 0.9 m/s, then we decelerate the system to -0.9 m/s and -0.45 m/s and finally the robot is slowed to stop. From Fig. 11b, it can be noticed that the sweeping controller is able to track with good precision the four desired velocities. The resultant displacement of the robot is depicted in Fig. 11a in which we can notice a regular body trajectory for both the accelerating and decelerating motions. Since the resultant velocity is zero, at the final time the robot is correctly placed very close to the initial position, in this case the origin. As in the constant velocity case presented previously, the body and leg attitude, depicted in Fig. 11c, follow an evolution that permits the fulfillment of the CG print trajectory in steady state.

3.3 Control Effort

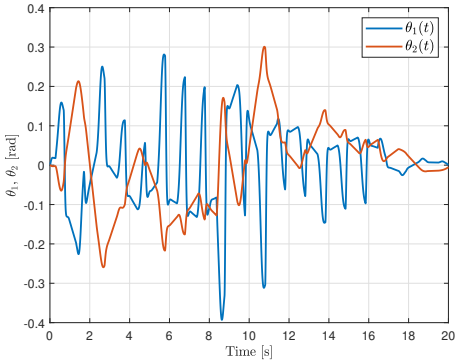
In this final section, we briefly describe the control effort required in each of the four simulation cases in which the foot placement and leg sweeping algorithms were used for horizontal control of the hopping system. First, the torque applied to the hip for the scenario in which the controller was required to correct an initial attitude error is depicted in Fig. 12a where it can be seen that, as it was expected, the required initial torque is considerably high since the controller tries to quickly counteract the initial deviation in order to avoid the immediate falling of the robot. However, once the falling of the robot is prevented, the amount of torque required to eliminate the attitude deviation is considerably reduced. On the other hand, for the control of the horizontal position, we can see from Fig. 12b that torque magnitude remains inside adequate values and it does not present sudden variations of high magnitude during all the traveled trajectory. For the leg sweeping algorithm, Fig. 12c depicts the presence of picks in the applied torque which arise periodically when the robot leave the ground (just after LIFT-OFF). This sudden change in the torque magnitude seems to be used to accommodate the robot posture such that the center of the CG print can be reached during flight. Apart from that sudden variation, the magnitude of the applied torque using the leg sweeping algorithm have a quite regular shape and its magnitude is reduced gradually as steady state is attained. Finally, when the system is required to accelerate, Fig. 12d shows that no relevant variations appear with respect to Fig. 12c and the same pattern is periodically repeated.



(a) Body and foot displacements.

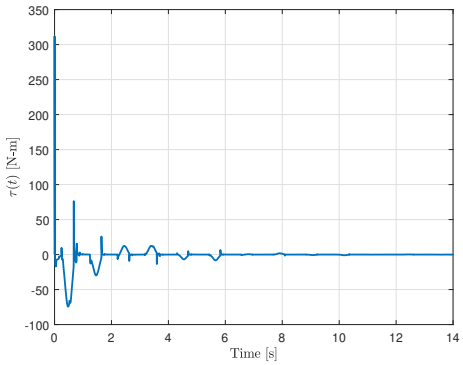


(b) Body velocity.

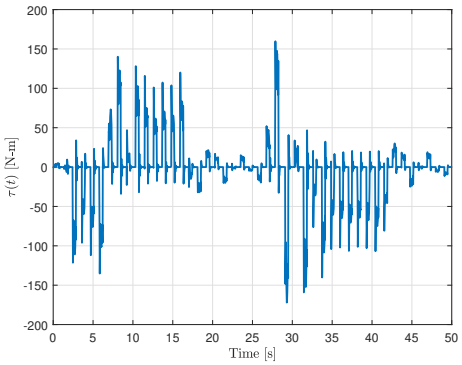


(c) Leg and body attitudes.

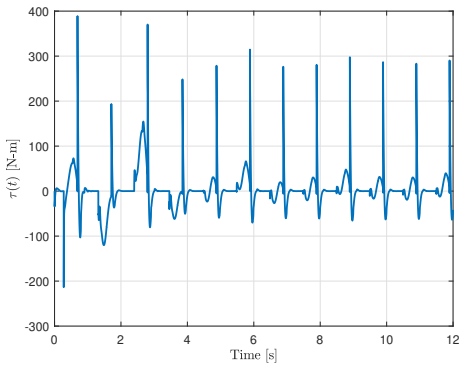
Figure 11: Running at varying horizontal rate.



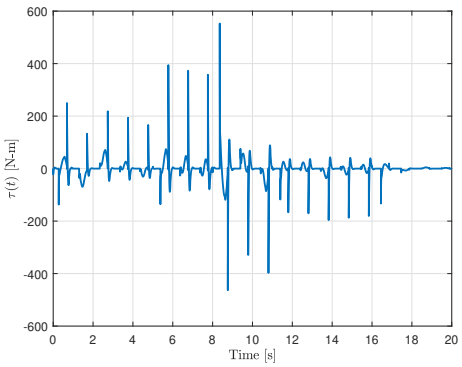
(a) Case: Body attitude correction.



(b) Case: Horizontal position control.



(c) Case: Constant horizontal velocity.



(d) Case: Varying horizontal velocity.

Figure 12: Torque applied at the hip for the four simulation cases.

Conclusions

Vertical and horizontal control of a one-legged hopping robot can be effectively achieved by using two decoupled control mechanisms, thus avoiding a strong coordination between both controllers, which simplifies the control design.

Regulation of the vertical position of the leg can be achieved by using a energy based controller that calculates the energy required to attain the desired height and injects it to the system at each hop through the actuation of a linear position actuator. Once the hopping at the desired height is established, it is maintained by a controlled periodic activation of the actuator.

Balancing the robot and moving it from place to place can be achieved by using a horizontal controller that enforces the correction of errors in body and leg attitudes and ensures that the horizontal velocity of the robot is well controlled. In particular, two methods for horizontal control have been studied: the foot placement algorithm and the leg sweeping algorithm. In the former, the foot is placed with respect to the projection of the center of gravity during flight while the hip is fixed during stance. The latter method uses the so-called CG print to compute both the location of the foot step and a hip trajectory that controls forward travel.

Three control algorithms, one for vertical and two for horizontal control, were implemented during the development of this project. The results presented in this report show the correct implementation of the controllers and, additionally, all the corresponding simulation files are available to the reader for the verification of the results and for obtaining additional results through the modification of the simulation parameters.

References

[1] Marc H Raibert. Hopping in legged systems—modeling and simulation for the two-dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):451–463, 1984.

[2] Giovanni A Cavagna. Elastic bounce of the body. *Journal of applied physiology*, 29(3):279–282, 1970.