DODO ALIVE: Simulation and Control

Report Wintersemester 2022/23

Divij Malik
Technische Universität München
divij.malik@tum.de

Harrison Yeh
Technische Universität München
tinghao.yeh@tum.de

Michael Lemanov
Technische Universität München
michael.lemanov@tum.de

***Abstract*—In this paper we present our work and results during the "Dodo Alive!" practical course in the winter semester of 2022-23. We picked up from the Monopod Hopper from last semester which was only constrained to move in the vertical direction and were tasked with adding movement in the horizontal direction. Additionally, this was to be done using the dynamics of the SLIP running model to ensure energy efficiency, conservation and periodicity of movement. The dynamics were taken from the paper from 2010 by Marco Hutter and an available Python code. The second task involved the assembly of the testbed which was taken from the Open Dynamics website.**

I. Introduction

Bio inspired robotics and especially humanoid robots play an increasing role in society and science. Robotic solutions with a structural similarity to humans are especially suitable for tasks which are mainly solved by humans. For moving in challenging terrain with higher velocity, hopping is an interesting approach. For simplicity, hopping is simulated with one leg. SLIP models describe the spring-like leg behavior of animals and humans in a very simple and abstract way. The SLIP model creates a behavior which is then projected onto the articulated leg.

The goal of the team in this semester was to implement the aforementioned SLIP model onto an already existing vertical monopod hopper in Matlab. Additionally, a small extension of the SLIP model in the form of a SLIPic model was to be implemented as well. The target was to attain stable, periodic and energetically conserved forward motion. Apart from this, a second task assigned to the group was the building of a testbed for the hopper in order to test just the vertical motion aspect of it. This testbed was taken from the Open Dynamic Robot Initiative. In this paper, we first go over the dynamics of the SLIP Model and its different phases followed by the additional dynamics of the SLIPic model. The second section elaborates upon the MATLAB implementation of said model. We then explain the structure and assembly of our testbed. We conclude with a summary of the work done this semester as well as helpful tips and guidelines for the next semester to ease the transition.

II. SLIP model

The SLIP model captures the locomotion behavior of robots with light, compliant legs in an abstract way. The body is reduced to a point mass m. The leg is represented by a massless spring with stiffness k_0 and a rest length l_0. For this system the equations of motion are the following:

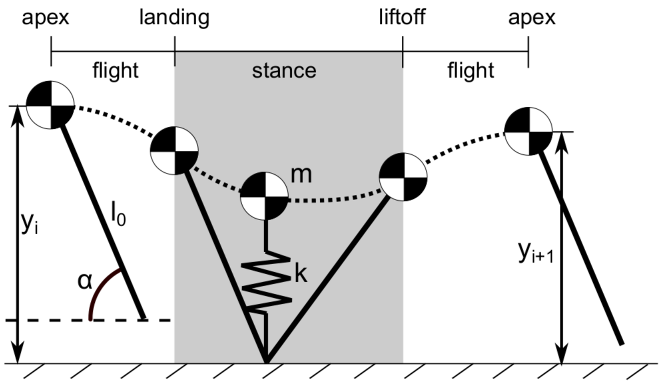

Fig. 1. Flight and stance phases of SLIP model [1]

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ mg \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}, \tag{1}$$

where x and y are the coordinates of the center of mass (COM). F_x and F_y are the ground reaction forces (GRF). The GRFs are 0 during flight and depend on the length of the spring l during compression and the landing angle α of the foot.

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = k_0(l_0 - l) \begin{bmatrix} -cos(\alpha) \\ sin(\alpha) \end{bmatrix}, \tag{2}$$

The SLIP model describes running as a series of apex heights y_i

In Fig 1 it becomes clear how the robot trajectory can be separated in flight and stance phases. The gait behavior is characterized by the forward speed and the apex height. The transition between two consecutive apex heights is defined by the landing angle α:

$$f_{yi+1} = f_{yi}|(\alpha), \tag{3}$$

The landing angle is used to achieve deadbeat stability of the system [1] [2].

III. Control

Different control designs were used for the flight and stance phase.

A. Stance control

The mechanics of a robotic leg under support conditions is the following:

$$M(q)\ddot{q} + b + g + J_s^T F_s = S^T \tau, \tag{4}$$

with the mass matrix M , the coriolis and centrifugal force vector b , the gravitational force g , the ground contact force F_s, the corresponding support Jacobian J_s, the actuation torques $\tau = [\tau_{Hip}, \tau_{Knee}]^T$ and the actuator selection matrix S limiting the actuation to hip and knee joints. The generalized coordinates $q = [x_{MainBody}, y_{MainBody}, \phi_{Hip}, \phi_{Knee}]^T$ describe the current robot state. Since we assume no slip between foot and ground during contact. The following conditions apply:

$$\dot{x}_s = J_s \dot{q} = 0, \tag{5}$$

$$\ddot{x}_s = J_s \ddot{q} + \dot{J}_s \dot{q} = 0, \tag{6}$$

This reduces the mechanics of a robotic leg to a support consistent description:

$$M\ddot{q} + N_s^T(b + g) + J_s^T \Lambda_s \dot{J}_s \dot{q} = (SN_s)^T \tau, \tag{7}$$

with the support space inertia Matrix $\Lambda_s = (J_s M^{-1} J_s^T)^{-1}$ and the support nullspace $Ns = [I - M^{-1} J_s^T \Lambda_s J_s]$. The real articulated leg has inertia and mass in each leg segment. Therefore it is affected by the impact when the foot hits the ground. The change of the generalized velocity is described by the nullspace $\dot{q}^+ = N_s \dot{q}^-$ The stance dynamics can now be projected onto the center of gravity (CoG) of the articulated leg. This allows to compute the operational space force F:

$$\Lambda^* \ddot{r}_{CoG} + \mu^* + p^* = F, \tag{8}$$

The operational space force F do relate to the robot actuation like the following:

$$\tau = J^{*T} F, \tag{9}$$

The different terms which are used in 8 and 9 are presented in the following:
i) task inertia:

$$\Lambda^* = (JM^{-1} SN_s J^*)^{-1}, \tag{10}$$

ii) projected coriolis and centrifugal terms

$$\begin{aligned}\mu^* &= \Lambda^* J_{CoG} M^{-1} N_s^T b - \Lambda^* \dot{J}_{CoG} \dot{q} \\ &+ \Lambda^* J_{CoG} M^{-1} J_s^T \Lambda_s \dot{J}_s \dot{q},\end{aligned} \tag{11}$$

iii) projected gravitational term

$$p^* = \Lambda^* J_{CoG} M^{-1} N_s^T g, \tag{12}$$

iv) support reduced Jacobian

$$J^* = J_{CoG} M^{-1} (SN_s)^T (SN_s M^{-1} (SN_s)^T)^{-1} \tag{13}$$

After projecting the stance dynamics of the robotic leg onto its CoG, the SLIP dynamics can be imposed on the CoG motion of the articulated leg. Then the equations of motion of the SLIP model in 1 can be solved for the $\ddot{x}, \ddot{y}$ which are then be inserted for $\ddot{r}_{CoG}$ in 8. This enables the SLIP control law:

$$\tau = J^{*T}(\Lambda^* \frac{1}{m}(F_{leg} + mg) + \mu^* + p^*) \tag{14}$$

If we assume to have a perfect model of the articulated leg and the CoG motion of the SLIP model is feasible, using the SLIP control law the CoG of the articulated leg will follow the SLIP trajectory accurately. This paragraph follows Hutter's work in [2]

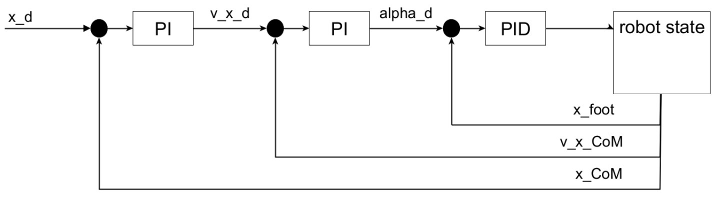

Fig. 2. Flight control

B. Flight control

A cascade control is used in the flight phase to track the landing angle α. The cascade control consists of two PI and one PID controller and is shown in Fig 2. The position controller uses the difference of the desired and current position of the CoG to calculate the desired velocity of the CoG. The velocity controller uses the difference of the goal and current velocity of the CoG to calculate the desired angle of attack in the SLIP model. From this angle, the desired foot position can be calculated. This foot position is compared with the current foot position and a desired acceleration of the foot is set. Consequently, the torque required to generate this acceleration is estimated.

IV. SLIPic

For the SLIP model, the leg is modeled with a massless spring. In reality, the articulated leg does have mass and inertia in each segment of the leg. This leads to loss of velocity at every impact, which in turn leads to a loss of kinetic energy:

$$\begin{aligned}\Delta E &= 0.5 m_{CoG} (|\dot{r}_{CoG}^-|^2 - |\dot{r}_{CoG}^+|^2) \\ &= 0.5 m_{CoG} \dot{q}^{-T} (J_{CoG}^T J_{CoG} - \\ &\quad N_s^T J_{CoG}^T J_{CoG} N_s) \dot{q}^-,\end{aligned} \tag{15}$$

The SLIP model does not account for that energy loss, and therefore, using it would lead to continually decreasing apex height till the leg eventually comes to a standstill. To prevent that, an extended SLIP model is used which is called SLIP with impact compensation (SLIPic). In this model, the spring of the SLIP model is pre-compressed at the landing such that the loss of kinetic energy in the articulated leg equals the energy stored in the spring:

$$\Delta E = 0.5 k \Delta l^2, \tag{16}$$

The spring forces in the SLIP model and therefore also the ground reaction forces (GRF) in the articulated leg are

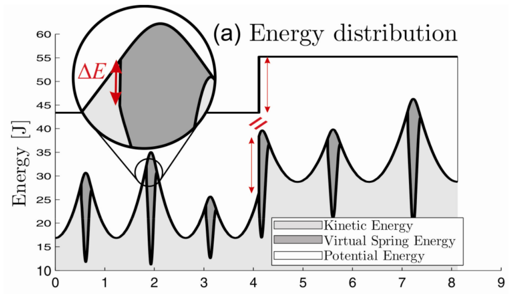

Fig. 3. Impact compensation in SLIPic [2]

increased such that the energy loss is compensated and the apex height is held constant. Fig 3 visualizes how the virtual spring energy compensates the loss of kinetic energy.

V. Implementation

For the implementation we introduced new files to the already existing file-structure in Matlab to compute and simulate the desired hopping since prior to it, the monopod could only hop in the vertical direction. The main files essential for our simulation are *derive_dynamics.m*, *simulate_monopod_hutter.m*, *get_delta_l.m*, *flight_controller.m* and *stance_controller.m*. In the first file we use the symbolic math toolbox to first compute all the necessary dynamics equations, which are further used with simulation values. The second file is the main file for the simulation. This means initialization is done and the different phases of movement are detected in order to apply the correct controller files from above. One is for the stance phase which as the name suggests, is used during ground contact of the robots foot and the other one is for the flight phase. Both files are used to compute the joint torques that result in the desired movement and become an input for a function, that delivers the differential equation to be solved to the ODE45 solver. Contrary to the implications of the file name, only during flight phase a control scheme is applied. That means that both files compute torques that are resulting in the control of the robots motion, but during flight phase an additional controller is running.
During the stance phase, we use the SLIPic model to describe the desired contact forces and then calculate the corresponding joint torques required to produce these forces. The additional controller is the previously mentioned cascaded PID controller scheme that is applied to realize a target position control. The control scheme contains two PI and one PID controller to find the desired velocity, angle of attack, and joint torques by passing the value, that has been calculated as the previous controllers output as the target to relate the desired velocity to joint torques gradually. By only changing the angle of attack of the foot before impact, we can control the bouncing direction and then further control the center of mass velocity to the target position. The so far not addressed file *get_delta_l.m* contains the calculation of the precompression of the modeled spring that is necessary to adjust for energy loss as explained in the SLIPic model to conserve the overall energy to generate periodic motions. Due to the structure of the whole simulation, we had to introduce global variables to the code. The desired motion of the robot is defined by a positional goal in horizontal or x-direction. This means the robot will jump in the direction of this position as fast as limit values we define allow, slow down close to the position and try to hop in place at the desired position. During the whole time the energy of the system is conserved. The simulations structure is visualized in Figure 4.

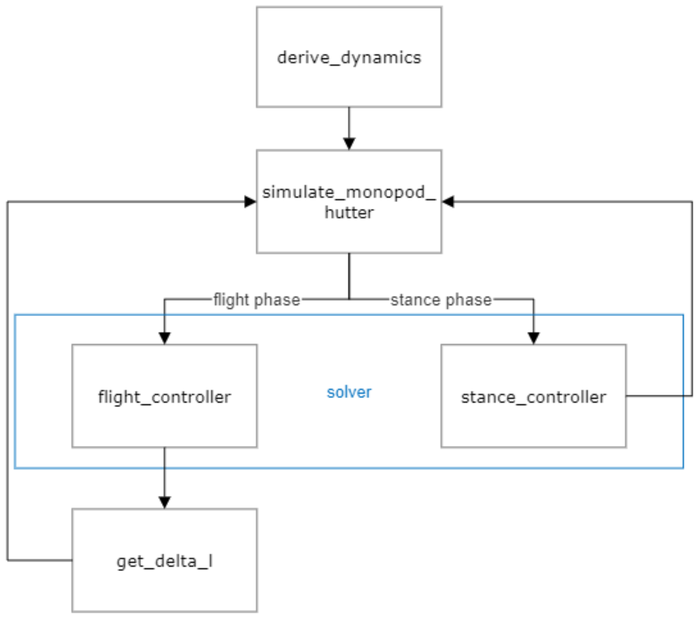

Fig. 4. File Structure

VI. Testbed

A. The Leg Test Stand

For the testbed, we use the latest version (Version 2.0) of the leg test stand from the Open Dynamic Robot Initiative website. This test stand allows us to evaluate the performance of a single leg of 2 degrees of freedom with external sensors. The teststand has a linear guided vertical degree of freedom and a carriage that holds the leg and the motor driver electronics. The test stand can be equipped with a pull-up-motor module for automated resetting to the initial position for machine learning experiments. The test stand available on the website consists of electronic components, off-the-shelf components and 3D Printed parts.

B. Assembly

Our assembly of the test stand differs in a few aspects from that of the Open Dynamic Initiative due to varying implementation strategies and requirements. The first difference is that in the electronic components. While the leg test stand in reference uses a combination of a Masterboard and a Microdriver to control the leg, we will be using an O-drive. We also do not require any of the sensors mentioned on the website. As far as the 3D printed parts are concerned, apart from the Master Board Support and the Motor Driver Electronics Attachment

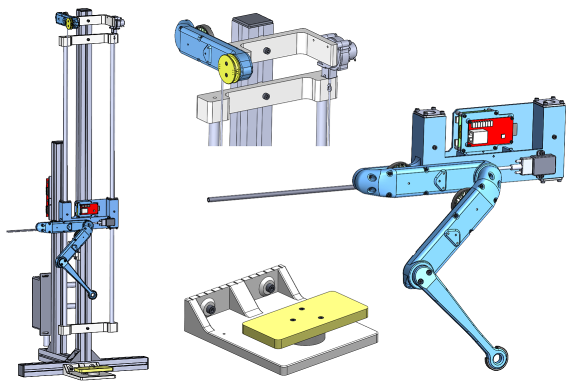

Fig. 5. The Leg Test Stand Overview

which aren't required because of our O-drive and the String Potentiometer, Pullup Motor Mount and Pulley which aren't required because we do not use a motor to pull the leg up, the rest of the parts were printed and assembled using the instructions available.

C. Observations

Although the test stand itself is assembled, it is not ready for testing because of the unavailability of the leg. Some friction was also observed between the parts - mainly between the carriage and linear motion rods - the cause of which is speculated to be minor measurement difference between the 3D printed parts and the off-the-shelf components, and the next course of action would be to smooth these things out to make the test stand ready for action.

VII. Summary

The SLIPic model was implemented successfully to control the forward or backward jumping of the robotic leg. Some significant errors have been discovered and fixed, such as a wrong calculation of the pre-compression of the modeled spring in order to account for energy loss. Now the energy is conserved, which can be seen in the constant apex height in jumps that are not close to the control target. Furthermore some corrections to the dynamics were made, since the robotic leg was not producing the correct behaviour.

Additionally the linear test stand was assembled and is theoretically ready to be used, although problems like high friction between the carriage and linear motion rods might need to be addressed first.

VIII. Conclusion

In conclusion, the simulation of the robotic leg produces the desired motion and the results are satisfying. The robotic leg is able to jump forward and backward without falling over and has also the possibility to hop in place if it reaches a desired location.

For future projects it would be great to extend the simulation from a monopod to a biped with the same hopping behaviour and add or extend the current approach so the biped is also stable while standing and therefor does not need to jump constantly. Aside from that an interesting project idea is introduction of non flat ground surfaces and the challenge to control the robot even on these surfaces.

References

[1] A. Wu and H. Geyer, "Highly robust running of articulated bipeds in unobserved terrain," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2558–2565, Sept 2014.

[2] M. Hutter, C. D. Remy, M. A. Höpflinger, and R. Siegwart, "Slip running with an articulated robotic leg," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4934–4939.