Report: SLIP models for controlling articulated robotic legs

Stefan Schneyer and Fabian Kolb

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 up to now mainly solved by humans. For moving in challenging terrain with higher velocities hopping is an interesting approach. For simplicity hopping can be simulated with one leg. SLIP models describe the spring-like leg behaviour of animals and humans in a very simple and abstract way. The model reduces the leg to a massless spring with a point mass attached to the top. This abstract SLIP model behaviour is then projected on the dynamics of an articulated leg. In a first stage a closed loop dynamics is generated based on the SLIP model. In the second stage the high order robotic leg tracks this closed loop dynamics and uses a feedback controller to adapt to deviations. The robot state is modelled as automaton. A continuous state captures the position of the robot in the 3D space, the hip angle and knee angle. A discrete state models the transition of flight to stance and stance to flight phase. A control structure is implemented for both the flight and the stance phase. For the flight phase a cascade control consisting of PI and PID controller is used. When touching the ground, the SLIP dynamics can be projected onto the CoG motion of the robotic leg. From there the joint actuator torques which are necessary to generate the operational space forces is calculated. Real robotic legs differ from the simplified SLIP model because the mass and inertia of the leg's segments influence the the robot's motion. Real robots suffer from the impact when the foot strikes the ground. This leads to a loss of kinetic energy which is incorperated in the SLIP model by an energy compensation. Different types of integration are implemented in order to study their effect on the controller stability. The project is implemented in Python. The RBDL library is used, which offers efficient rigid body dynamics algorithms. For visualization Meshup, a tool that was developed with RBDL is used.

II. SLIP model

The SLIP model is an abstraction capturing the locomotion behavior of robots with light, compliant legs. In this model 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:

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

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

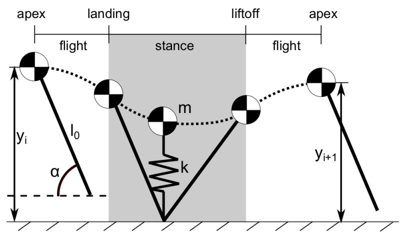

Fig. 1. flight and stance phases of SLIP model [1]

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 SLIP model is a conservative system without any friction or other mechanism to dissipate the dynamics, and thus, their phase space does not shrink over time. Therefore, the apex height and forwards speed are coupled through the system energy and the trajectory is fully characterized by 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 are 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 Fs, 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 $q = [x_{FloatingBase}, y_{FloatingBase}, \phi_{Hip}, \phi_{Hip}]^T$ 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 inertia Matrix $\Lambda_s = (J_sM^{-1}J_s^T)^{-1}$ and the support nullspace $Ns = [I - M^{-1}J_s^T\Lambda_sJ_s]$. The real articulated leg has inertia and mass in each leg segment. Therfore it is affected by the impulse of the foot hitting 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 on 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 are presented in the follwing:
i) task interia:

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

ii) projected coriolis and centrifugal terms

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

iii) projected gravitational term

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

iv) support reduced Jacobian

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

After projecting the stance dynamics of the robotic leg onto the CoG, the SLIP dynamics can now also be imposed onto the CoG. 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 an 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 accuratly. The paragraph follows Hutters work in [2]

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 goal 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 goal foot position can be calculated. This foot position is compared with the current foot position of the acrticulated leg and a desired acceleration of the foot is set. From there the torques required to generate these accelerations of the robot foot are estimated. The relation between joint accelerations and end-effector accelerations is given by the Jacobians. To transform the the accelerations in the joints to actuations the mass matrix and Newton's first law is used.

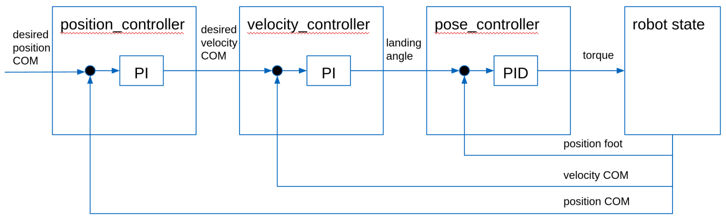

Fig. 2. Flight control

IV. SLIPIC

For the SLIP model the leg is modeled with a masseless spring. In reality the articulated leg does have a mass and inertia for each segment. The impact when the foot strikes the ground therefor influences the velocity of the articulated leg. Therfore the generalized velocity changes through the impact. Like mentioned earlier this velocity change is described throught the support nullspace. This velocity change leads to a loss of kinetic energy:

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

The SLIP model does not account for that energy loss. If the normal SLIP model is used for the SLIP control law in the stance phase this energy loss is not compensated. The potential energy is reduced by the loss of kinetic energy in every jump. This leades to a reduced apex height in every foot strike. To prevent that an extended SLIP model is used which considers the loss of kinetic energy through the impact. This model 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:

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

The spring forces in the SLIP model and thefore also the ground reaction forces (GRF) in the articulated leg are increased such that the energy loss is compensated and the apex height is held constant. Fig 3 visualizes how the virtual spring energy copnesates the loss of kinetic energy.

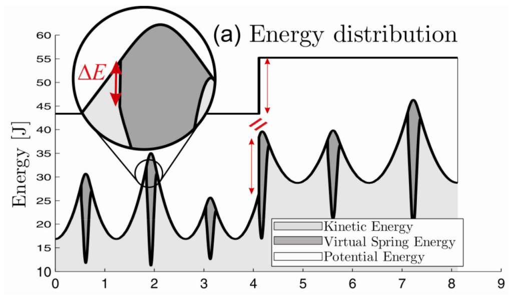

Fig. 3. impact compensation in SLIPic [2]

V. Structure

A. Hybrid Automaton

B. Implementation

VI. Solver

The calculate the robots state we iteratively do a control, forward dynamics and integration step. In the control step we calculate the requred robot actuation from the current robot state. For this we use our stance and flight controller described in Section III. In teh forward dynamics the robot state derivative is calculated from the current robot state and the robot actuation. Integrating this robot state derivative returns the robot state for the next time step. The step size of the solver loop determines the accuracy of the solution for the robot state. The step size is derermined by the steps the integrator takes.

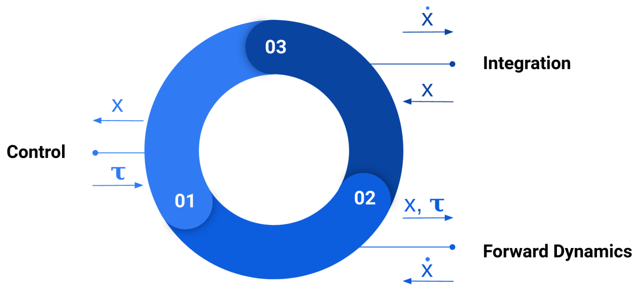

Fig. 4. Solver iteration

In each iteration a state transition from stance to flight or flight to stance phase is checked. The jump function uses the solved robot state from this iteration for the discrete state transition. In a first attempt the ivp solver by SciPy is applied. This solver uses the Runge-Kutta method of order 5(4) to numerically integrate a system of ordinary differential equations given an initial value. The robot state derivative i is integrated until the end of the integration interval is reached or an event function is fullfilled. The guard functions which mark the transition between the discrete states are used as event functions. The ivp solver will find an accurate robot state at time t when the event function is zero. This has the advantage that the robot state is known exactly at the state transition and can be used in the jump function. The ivp solver uses variable step size for each integration step. This variable step size causes unstable behaviour in the system. The PID controller in the flight control numerically differentiates the foot position error. A differentiation with a variable step size might cause problem and makes it hard to tune the PID controller. Fig 5 shows that after 4 jumps the system becomes unstable.

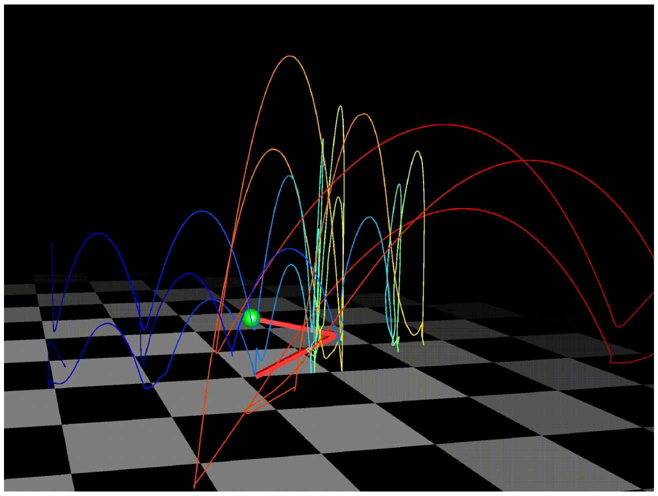

Fig. 5. ivp solver

In a second attempt the Runge-Kutta method of order 5(4) with a fixed step size is evaluated. This has the disandvatage that no root-finding is used. After each integration step the guard functions are checked to make sure whether a discrete state transition did occur. Compared to the ivp solver the robot state at the transition is not as accurate with this integration method. Choosing a small enough step size makes this problem insignificant. The fixed step size helps a lot to make the flight controller more stable. Fig 6 shows the good results with this method.

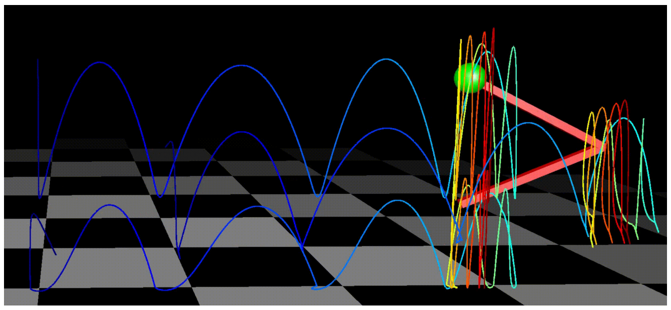

Fig. 6. RK45 solver

VII. Conclusion

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.

[3] J. Grimes and J. Hurst, *THE DESIGN OF ATRIAS 1.0 A UNIQUE MONOPOD, HOPPING ROBOT*, 09 2012, pp. 548–554.