Exploration of control strategies for a three-link 2D biped

Teguh Santoso Lembono

December 10, 2020

1 Introduction

The purpose of this report is to present a modelling, simulation, and control of a simplified humanoid robot walking model. The report is divided into two parts. The first presents the kinematics and dynamics analysis of the walking model, the impact map, the numerical integration for simulation, and the standard controller. The second part then presents the proposed controller using an optimal control approach.

2 Theory
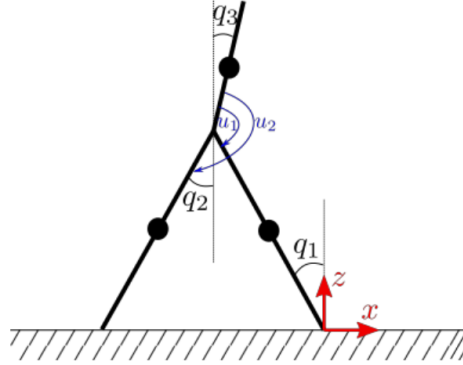
The simplified model of the humanoid is shown in Fig. 2. It is represented by three-links model, two refer to the legs and one for the torso. Each link has a point mass at its center. At each instant, one leg acts as the stance leg that stays stationary, and the other as the swing leg.

2.1 Kinematics

The kinematics can be easily calculated by considering the stance foot location as the origin of the coordinate frame, as shown in Fig. 2. The points of interests are then the location of each mass $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$, the hip $(\boldsymbol{p}_h)$, and the swing foot $(\boldsymbol{p}_{swf})$. Each point $\boldsymbol{p}$ consists of the x and z coordinates

for this 2D model. By simple geometry, we obtain

$$x_1 = 0.5 * l_1 * sin(q_1) \tag{1}$$

$$z_1 = 0.5 * l_1 * cos(q_1) \tag{2}$$

$$x_2 = l_1 * sin(q_1) - 0.5 * l_2 * sin(q_2) \tag{3}$$

$$z_2 = l_1 * cos(q_1) - 0.5 * l_2 * cos(q_2) \tag{4}$$

$$x_3 = l_1 * sin(q_1) + 0.5 * l_3 * sin(q_3) \tag{5}$$

$$z_3 = l_1 * cos(q_1) + 0.5 * l_3 * cos(q_3) \tag{6}$$

$$x_h = l_1 * sin(q_1) \tag{7}$$

$$z_h = l_1 * cos(q_1) \tag{8}$$

$$x_{swf} = l_1 * sin(q_1) - l_2 * sin(q_2) \tag{9}$$

$$z_{swf} = l_1 * cos(q_1) - l_2 * cos(q_2) \tag{10}$$

$$x_t = l_1 * sin(q_1) + l_3 * sin(q_3) \tag{11}$$

$$z_t = l_1 * cos(q_1) + l_3 * cos(q_3). \tag{12}$$

The corresponding velocities can then be computed using chain rule, i.e.

$$\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial t}. \tag{13}$$

Using the symbolic programming in matlab or python, it can be done automatically.

2.2 Dynamics

To obtain the dynamic equation of the locomotion model, the dynamics is divided into two phases: the swing phase and the impact phase. In the swing phase, we assume that the stance foot is not moving (there is sufficient contact force to keep it stationary), while the swing foot is moving to the next contact location. At the point of impact, we assume an instantaneous impact that preserves the momentum, as described in the next section.

To describe the dynamics of the swing phase, we can rely on Lagrangian dynamics. First, we found the expression of the total kinetic and potential energy (T and V),

$$T = \sum_{i=1}^{3} \frac{1}{2} m_i \frac{\mathrm{d}\boldsymbol{p}_i}{\mathrm{d}t}^2 \tag{14}$$

$$V = \sum_{i=1}^{3} m_i g z_i. \tag{15}$$

The Lagrangian is then calculated as

$$L = T - V. \tag{16}$$

With this, the Lagrangian equation of motion can be calculated as

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = \tau_i, \tag{17}$$

where τ_i is the generalized torque. The equations can be computed using the symbolic programming.

After obtaining the equations for $i = 1, 2, 3$, we can rearrange them in the following form,

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau}. \tag{18}$$

The terms $\boldsymbol{M}(\boldsymbol{q}), \boldsymbol{C}(\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, and $\boldsymbol{G}(\boldsymbol{q})$ cna be computed from (18) as follows. $\boldsymbol{G}(\boldsymbol{q})$ is computed by setting $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$ in (18) to zero. $\boldsymbol{M}(\boldsymbol{q})$ is computed by setting $(\dot{\boldsymbol{q}}, \boldsymbol{\tau})$ in (18) to zero, and substract the term $\boldsymbol{G}(\boldsymbol{q})$ from the equation. Finally, $\boldsymbol{C}(\dot{\boldsymbol{q}})$ is computed by setting $(\ddot{\boldsymbol{q}}, \boldsymbol{\tau})$ to zero, substract the term $\boldsymbol{G}(\boldsymbol{q})$ from the equation, and divide by the corresponding $\dot{\boldsymbol{q}}$.

2.3   Impact Map
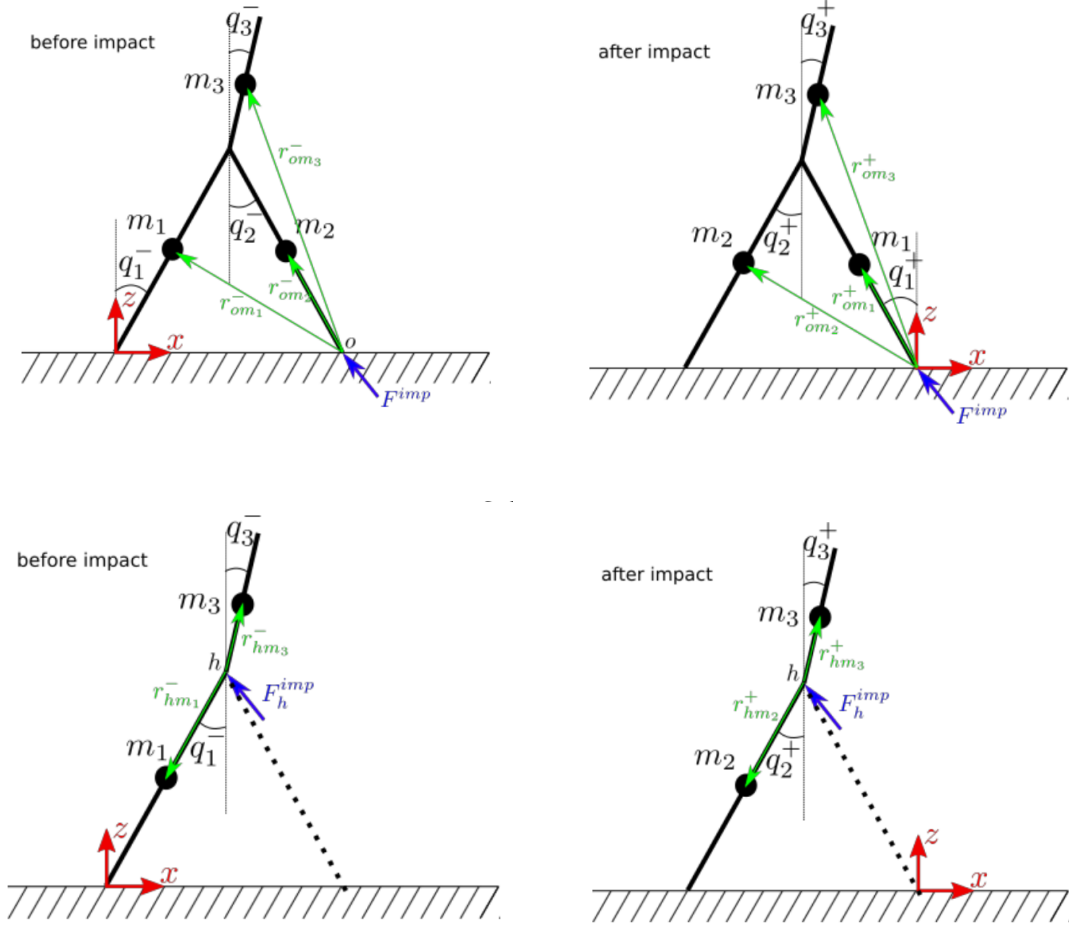
In our model, after the impact we switch the swing foot to become the stance foot, as shown in Fig. 2.3 (left: before impact, right: after impact). The angles are therefore changed as follows,

$$q_1^+ = q_2^- \tag{19}$$

$$q_2^+ = q_1^- \tag{20}$$

$$q_3^+ = q_3^- . \tag{21}$$

The impact map for the angular velocity can be computed by considering the conservation of angular momentum at the impact at two locations: the impact point (the location of the swing foot at the impact), and at the hip.

Looking at Fig. 2.3 (left), the momentum before the impact can be written as:

$$H_a^- = \sum_{i=1}^{3} m_i \boldsymbol{r}_{om_i}^- \times \dot{\boldsymbol{r}}_i^{\,-} \tag{22}$$

$$H_b^- = m_1 \boldsymbol{r}_{hm_1}^- \times \dot{\boldsymbol{r}}_1^{\,-} \tag{23}$$

$$H_c^- = m_3 \boldsymbol{r}_{hm_3}^- \times \dot{\boldsymbol{r}}_3^{\,-} , \tag{24}$$

and after impact as

$$H_a^+ = \sum_{i=1}^{3} m_i \boldsymbol{r}_{om_i}^+ \times \dot{\boldsymbol{r}}_i^+ \tag{25}$$

$$H_b^+ = m_1 \boldsymbol{r}_{hm_2}^+ \times \dot{\boldsymbol{r}}_2^+ \tag{26}$$

$$H_c^+ = m_3 \boldsymbol{r}_{hm_3}^+ \times \dot{\boldsymbol{r}}_3^+. \tag{27}$$

We can rearrange the equations to obtain $\boldsymbol{H}^- = \boldsymbol{A}_- \dot{\boldsymbol{q}}^-$ and $\boldsymbol{H}^- = \boldsymbol{A}_+ \dot{\boldsymbol{q}}^+$. Then we can obtain the velocities after the impact by solving the linear equation $\boldsymbol{H}^- = \boldsymbol{H}^+$,

$$\dot{\boldsymbol{q}}^+ = \boldsymbol{A}_+^{-1} \boldsymbol{A}_- \dot{\boldsymbol{q}}^-. \tag{28}$$

2.3.1 Exercise Answers

- Q: *What can you say about the potential energy before and after impact?* The potential energy before and after impact does not differ, because the position of each mass remains the same. What changes after the impact is only the kinetic energy.

- Q: *Try $q_m = [pi/6, -pi/6, pi/10], dq_m = [1, 0.2, 0]$. What percentage of the kinetic energy of the biped is lost due to the impact?* There is 27.92% of the kinetic energy lost due to the impact.

- Q: *Plot the percentage of the kinetic energy loss due to impact as a function of angle α where $q^- = [\alpha, -\alpha, 0]$ and α varies from 0 to $\pi/4$. Assume that $\dot{q}^- = [1, 0.2, 0]$. Include your plot.*
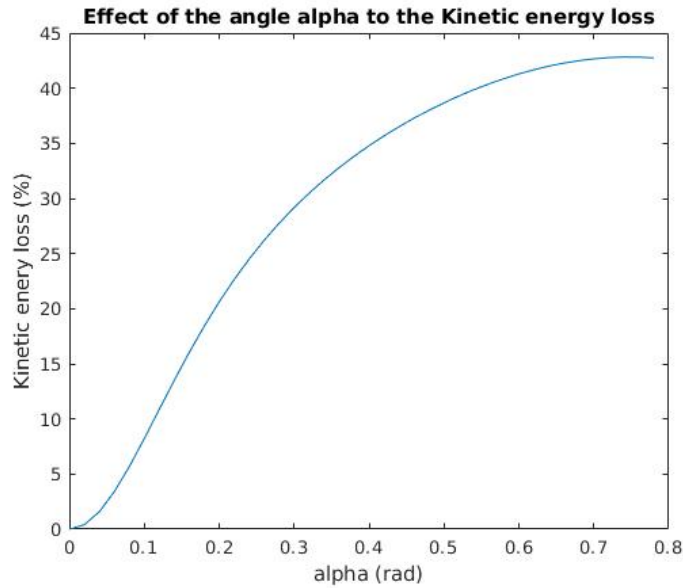
- Q: *The bigger α is the bigger is the step length. Based on your answer to question 3, what is the relation between step length and energy loss at impact given a fixed $\dot{q}^-$?* As the step length grows bigger, the energy loss increases.

2.4 Model Validation and Animation

2.4.1 Exercise Answers

- *In the animations, what does a real-time factor of 1 mean? How about a real-time factor less than 1?*

- *How does "skip" in animate.m effect the real-time factor and the speed of the animation?*

- *What is the role of r0 in animate.m?*

2.5   Numerical Integration

2.6   Control and Optimization

The system during the swing phase is underactuated, i.e. there are three degrees of freedom to control but only two control inputs, as shown in Fig. **??**. In the standard controller, we first choose the first control input u_1 to control the torso angle to be at a desired value α, whereas the second control input u_2 is used to control the swing foot angle q_2. Expressing these as virtual constraints in terms of new variables y_1 and y_2, we set

$$y_1 = q_3 - \alpha \tag{29}$$
$$y_2 = -q_2 - q_1. \tag{30}$$

Differentiating this constraint, we obtain

$$\dot{y}_1 = \dot{q}_3 \tag{31}$$
$$\dot{y}_2 = -\dot{q}_2 - \dot{q}_1. \tag{32}$$

We then set the controller as PD control to drive y_1, y_2 to zeros,

$$u_1 = k_1^P y_1 + k_1^D \dot{y}_1 \tag{33}$$
$$u_2 = k_2^P y_2 + k_2^D \dot{y}_2. \tag{34}$$

Now what is left is to choose the parameters $(k_1^P, k_1^D, k_2^P, k_2^D, \text{and} \alpha)$, as well as the initial state $(\boldsymbol{q}^0, \dot{\boldsymbol{q}}^0)$. We use the function *fmin* in Matlab to optimize these parameters, given an objective function.

3   Control

3.1   Overview of Control Strategy

In this work, I rely on optimal control to design an efficient locomotion controller. The advantage of this formulation is that the resulting motion is optimal w.r.t. the desired criteria, which can be expressed as an objective function and constraint(s). A high level specification (such as minimizing energy, tracking a desired foot trajectory, tracking a desired torso angle) can be translated quite easily to the objective function.

However, there are several challenges in implementing the optimal controller to this problem. Firstly, most optimal control strategy is time-based. That is, given a time horizon (e.g. 1 second), solving the optimal control problem will give us a specific control and state trajectory that minimizes the objective function while satisfying the constraints. Some variants are time independent, e.g. when the time horizon is infinite, but it is not suitable for this locomotion problem (where the desired behavior highly depends on time and event, and not stationary). Since the impact timing is difficult to predict precisely (especially given some disturbance), a purely time-based controller is difficult to be used here. Secondly, the dynamics model of our locomotion is discontinuous due to the impact map. This is a big problem for optimal control, as most methods would require the gradient of the dynamics function.

To overcome these issues, I propose to use the following approach. First, the optimal control is solved for only one step until impact, so that the discontinuous event at the impact is not considered. Considering that the locomotion is a periodical motion, designing an optimal trajectory for one step

should be sufficient, provided that the state after the impact returns to the initial state. Secondly, we need a controller that is not purely time based or feed forward only. One option is to use a feedback controller (e.g. PD) to track the desired optimal trajectory. However, there is a specific optimal control formulation that provides this automatically, i.e. Iterative Linear Quadratic Regulator (ILQR). Solving ILQR gives us both the optimal trajectories and the optimal feedback controller (note that 'optimal' here refers to local optimum, not the global one). Finally, to choose the optimum parameters for the optimal control, we again rely on an optimization solver (fmin) to ensure that the controller is stable.

3.2 Iterative Linear Quadratic Regulator (ILQR)

A general discrete OCP consists of a cost function

$$C(\boldsymbol{x}, \boldsymbol{u}) = \sum_{t=0}^{T-1} c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) + c_T(\boldsymbol{x}_T, \boldsymbol{u}_T), \tag{35}$$

subject to the dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{36}$$

$(\boldsymbol{x}, \boldsymbol{u})$ are the state and control trajectories for the time horizon T. ILQR solves this optimal control problem by iteratively formulating a sub problem as an LQR problem and solve it to obtain a better solution at each iteration.

Let us consider the current guess $(\boldsymbol{x}^k, \boldsymbol{u}^k)$, where k is the iteration index. Given the general cost function in (35), we approximate it as a quadratic function around $(\boldsymbol{x}^k, \boldsymbol{u}^k)$,

$$c_t(\delta\boldsymbol{x}_t, \delta\boldsymbol{u}_t) = \frac{1}{2} \begin{bmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{c}_{\boldsymbol{xx},t} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{c}_{\boldsymbol{uu},t} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_{\boldsymbol{x},t} & \boldsymbol{c}_{\boldsymbol{u},t} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_t \\ \delta\boldsymbol{u}_t \end{bmatrix}, \quad (37)$$

where $\boldsymbol{c}_{\boldsymbol{x},t}, \boldsymbol{c}_{\boldsymbol{u},t}, \boldsymbol{c}_{\boldsymbol{xx},t}$, and $\boldsymbol{c}_{\boldsymbol{uu},t}$ are the cost function's first and second order derivatives with respect to $\boldsymbol{x}$ and $\boldsymbol{u}$. Similarly, we can approximate the dynamics in (36) using the linear approximation

$$\delta\boldsymbol{x}_t = \boldsymbol{A}_t \delta\boldsymbol{x}_t + \boldsymbol{B}_t \delta\boldsymbol{u}_t, \tag{38}$$

where $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ are the derivatives of the dynamics $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ with respect to $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$, respectively. The derivatives are evaluated at the current guess $(\boldsymbol{x}^k, \boldsymbol{u}^k)$. At this stage, we have quadratic costs and linear dynamics as functions of $\delta\boldsymbol{x}$ and $\delta\boldsymbol{u}$. This is therefore a time-varying LQR problem, of which the variables of interest are $\delta\boldsymbol{x}$ and $\delta\boldsymbol{u}$, and the solution can be computed analytically.

By the end of ILQR iteration, we will have a locally optimal solution as state and control trajectories $(\boldsymbol{x}^*, \boldsymbol{u}^*)$, as well as the optimal feedback term $\boldsymbol{K}_t$. To use this feedback term, at online execution we compute the control command by

$$\boldsymbol{u}_t = \boldsymbol{u}_t^* + \boldsymbol{K}_t(\boldsymbol{x}_t - \boldsymbol{x}_t^*). \tag{39}$$

This optimal feedback term serves as a locally optimal PD controler to track the optimal trajectories $(\boldsymbol{x}^*, \boldsymbol{u}^*)$.

3.3 ILQR formulation for the locomotion

For the locomotion problem, we consider the initial state $\boldsymbol{x}_0$ to be the state right after an impact, and the end state $\boldsymbol{x}_T$ to be at the next predicted impact. It turns out that formulating the objective function for this problem is complicated, due to the event detection. I previously try using an objective function to only reach the desired end state $\boldsymbol{x}_T$, but this often results in failure because the swing foot touches the ground before reaching $\boldsymbol{x}_T$. In this event based detection, the swing foot trajectory

is very important. Another variable that is also important is the torso angle ($\boldsymbol{q}_3$). This needs to be kept small enough, otherwise the motion is often unstable.

After a lot of trials, the objective function that is found to be working is to track a desired foot swing trajectory, while keeping the torso angle to be around a desired angle. The running cost, i.e. the cost for $t = [0, T-1]$ is set to be

$$c(\boldsymbol{x}_t, \boldsymbol{u}_t) = (\boldsymbol{r}_{swf,t} - \boldsymbol{r}_{swf,t}^*)^\top \boldsymbol{W}(\boldsymbol{r}_{swf,t} - \boldsymbol{r}_{swf,t}^*) + \boldsymbol{u}_t^T \boldsymbol{R} \boldsymbol{u}_t, \tag{40}$$

while the terminal cost at $t = T$ is set to be

$$c(\boldsymbol{x}_T, \boldsymbol{u}_T) = (\boldsymbol{r}_{swf,T} - \boldsymbol{r}_{swf,T}^*)^\top \boldsymbol{W}_T(\boldsymbol{r}_{swf,T} - \boldsymbol{r}_{swf,T}^*). \tag{41}$$

The constraint on the control input is a box constraint,

$$-30 \le \boldsymbol{u}_i \le 30 \quad \text{for} \quad i = 1, 2. \tag{42}$$

To solve this constrained ILQR problem, I use the open source solver Crocoddyl. It uses the box-FDDP algorithm that accepts infeasible initial guess $(\boldsymbol{x}^0, \boldsymbol{u}^0)$.

3.4 Optimization Strategy

To use the optimal control formulation, we have to choose the following parameters $\boldsymbol{\Phi}$:

- The initial state $(\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0$

- The desired torso angle α

- The desired goal position for the swing foot $x_{swf,T}^*$

Although optimal control formulation is powerful, finding the optimal parameters for a desired objective of this particular locomotion problem turns out to be challenging. Instead of starting from random initial guess, I rely on the standard controller discussed in Section 2.6. For a given desired criteria (e.g. a desired speed, step length etc.), first I optimize this standard controller to achieve the criteria, and observe the corresponding parameters $\boldsymbol{\Phi}$. Using this with the ILQR often results in failure, i.e. unstable controller, as the ILQR is still time based and depend a lot on the impact event. So we need to further optimize $\boldsymbol{\Phi}$ to obtain a good controller.

To make the optimization tractable, I do it step by step. First, I optimize for only one foot step. Since a stable controller will have a stable limit cycle, the next state after the impact should be very close to the initial state. So I design the objective function for the optimization to be

$$f(\boldsymbol{\Phi}) = (\boldsymbol{x}_0 - \boldsymbol{x}_T^+)^T \boldsymbol{x}_0 - \boldsymbol{x}_0, \tag{43}$$

where $\boldsymbol{x}_0$ is the initial state, and $\boldsymbol{x}_T^+$ is the final state after the impact (i.e. the initial state for the next foot step). By ensuring that $\boldsymbol{x}_0$ is as close as possible to $\boldsymbol{x}_T^+$, we can have a stable limit cycle.

In practice, optimizing for only one foot step only makes a stable controller for a few foot steps, as the error diverges. When this happens, the optimization is run again, but this time with a longer foot steps, and initialized using the previously obtained parameters. Doing this iteratively often results in a controller that is stable for a large number of foot steps, even to infinity.

3.5 Results and Discussion

3.6 Conclusion