Development of Hopping Legged Robots

Report submitted in fulfillment of the requirements of
International Linkage Degree Program (ILDP)

by

Kanishk Chaudhary

Bachelor of Engineering (B.E.) in Mechanical Engineering
Birla Institute of Technology and Science, Pilani, India

Supervisor:

Professor Takeshi Takaki, Hiroshima University, Japan

HIROSHIMA UNIVERSITY, JAPAN

January 2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Firstly, I would like to thank Prof Dalip Kumar, Associate Dean (BITS Pilani), the members of International Programmes and Collaboration Division at BITS Pilani (IPCD), Ms. Haruka Takehara and Ms. Minaho Watanabe from the ILDP Office at Hiroshima University for providing me the opportunity to undertake a project at Hiroshima University, Japan, as an audit student.

I am thankful to Dr Suresh Gupta, Associate Dean, Academic Undergraduate Studies Division (AUGSD) and Prof M.S. Dasgupta, Head of Department of Mechanical Engineering at BITS Pilani for allowing me to pursue this project in collaboration with Hiroshima University.

I am grateful to my supervisor Prof Takeshi Takaki from Hiroshima University, for the support and constant guidance throughout the course of this internship. I am also grateful to Dr Amit R. Singh from BITS Pilani, for evaluating my project as a part of my graduation requirements at BITS.

Last, but not the least, I thank my friends and family for their support and encouragement.

ABSTRACT

Development of Hopping Legged Robots

by

Kanishk Chaudhary
Bachelor of Engineering (B.E.) in Mechanical Engineering

Birla Institute of Technology and Science, Pilani

This internship was undertaken at Hiroshima University, Japan, in an online mode, as a part of the International Linkage Degree Program (ILDP). The official timeline of this thesis was from August 2, 2021, to January 4, 2022, in the first semester of the academic session 2021-2022. The task was to simulate a quadruped hopping robot by the end of my project. Initially, I was introduced to an open-source physics engine called "Open Dynamics Engine (ODE)" (Smith, 2001b). Chapter 1 gives an introduction to the ODE environment. Next, a one-legged hopping robot was simulated on ODE, that could move in both 2-dimensional plane and 3-dimensional space (Chapter 2). A parametric study was undertaken (Chapter 3) to further understand, through observation), how changes in various parameters affect the motion and stability of the robot. In Chapter 4 and Chapter 5, the implementation of the one-legged robot algorithm into the biped and quadruped hopping robots have been discussed. Conclusions and future work have been briefly mentioned in Chapter 6

Chapter 1

Introduction to ODE

Open Dynamics Engine is an open-source physics engine written in C/C++ and developed by Russell Smith in 2001. It is mainly used in the applications of rigid-body simulations and collision detection (Smith, 2001b).

ODE also includes a basic graphics package called drawstuff, which uses the OpenGL interface.

The user manual/documentation of ODE is comprehensive and provides details of all important functions which can be used (Smith, 2001a). Apart from this, a beginner tutorial of ODE is available (in Japanese) to familiarise one with the development environment (Demura, 2008). A more detailed evaluation of ODE is also given (Kooijman, 2010). This chapter provides a rough introduction to some important concepts in a generic ODE code.

Although not in abundance, there are some great examples of how ODE has been used to simulate robots in past research. Walking and posture control algorithms to balance a six-legged robot have been developed in ODE (Yıldırım and Arslan, 2018; Arslan and Yıldırım, 2018). ODE has also been used for dynamics calculations for a wheeled mobile robot (Sato et al., 2009).

1.1 World

ODE is initialised by creating a world object, wherein all bodies and joints interact.

1.2 Rigid Bodies

ODE supports 4 types of rigid bodies: sphere, cylinder, capsule, box.

Any rigid body has properties which either remain constant or change over time. The constant properties include:

- Mass of the body
- Position of the center of mass of the body
- 3x3 inertia matrix

The properties that change over time are:

- Position vector
- Linear velocity vector
- Angular velocity vector
- Orientation of the body, denoted either by a quaternion (4x1) or a rotation matrix (3x3).

1.3 Joints

ODE allows two rigid bodies to be connected through the following joints:

- Ball joint
- Hinge joint
- Slider joint
- Contact joint
- Universal joint
- Hinge-2 joint
- Fixed joint
- Angular motor joint
- Linear motor joint
- Plane 2D joint
- Prismatic-Rotoide joint
- Prismatic-Universal joint
- Piston joint
- Double ball-and-socket joint
- Double hinge joint
- Transmission join

Parameters like anchor point, axes of motion, position or angle limits, etc. can be applied to these joints.

1.4 Collision

Geoms are special objects which can be attributed to a rigid body. This specifies the geometry of the body, which is essential in determining the contact points when one geom collides with another. Geoms indicate the geometrical characteristics of an object, like size, shape, position and orientation.

A space is simply a group of geoms. It is a subset of the world wherein the collision is taking place. Spaces help to make the program run faster.

The nearcallback() function is called by ODE whenever a collision between 2 geoms is detected. The user can check which bodies are colliding, and then provide necessary conditions to simLoop() accordingly.

Chapter 2

One-Legged Hopping Robot

This chapter introduces the basic structure of a one-legged hopping robot and the algorithm that governs its motion. This has been derived from (Raibert, 1986).

2.1 Components

A simple model of a one-legged hopping robot consists of 3 bodies: the top body (this will be henceforth referred to as just "the body"), the leg, and the slider.

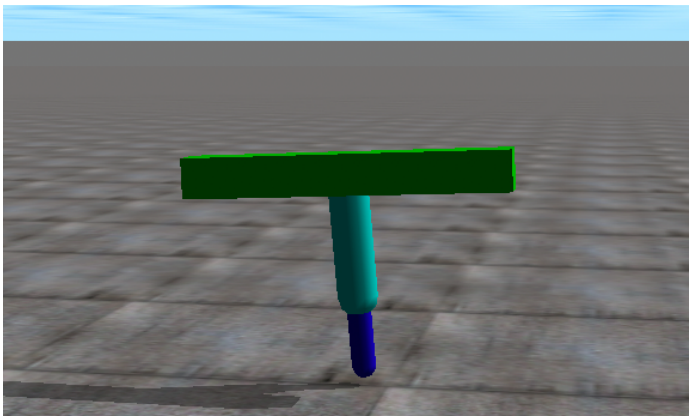

Figure 2.1: Illustration of the one-legged robot made using ODE

Body	
Shape	Box
Dimensions (x,y,z)	(0.8, 0.1, 0.1)
Mass	10
Coordinates of CoM (world)	(0, 0, 1)
Leg	
Shape	Capsule
Dimensions (l, r)	(0.3, 0.05)
Mass	3
Coordinates of CoM (world)	(0, 0, 0.85)
Slider	
Shape	Capsule
Dimensions (l, r)	(0.4, 0.03)
Mass	0.7
Coordinates of CoM (world)	(0, 0, 0.5)

Table 2.1: Physical parameters of the one-legged robot

There are 2 joints connecting these three bodies. For a 2-dimensional motion, a hinge joint connects the leg with the body. For a 3-dimensional motion, there would be a universal joint instead. The leg and the slider are connected through a slider (or prismatic joint). This joint is analogous to a spring having a spring constant of k_s.

2.2 State Variables

The variables which define the position and motion of each of these components are defined below.

Roll, pitch and yaw angles	θ_{Bx}	θ_{By}	θ_{Bz}
Position of the centre of mass	p_{Bx}	p_{By}	p_{Bz}
Linear velocity	v_{Bx}	v_{By}	v_{Bz}
Angular velocity	ω_{Bx}	ω_{By}	ω_{Bz}
Linear acceleration	a_{Bx}	a_{By}	a_{Bz}

Table 2.2: State variables for the body of the one-legged robot

Angular Position	θ_{Lx}	θ_{Ly}
Desired Leg Angle	$\theta_{Lx,des}$	$\theta_{Ly,des}$
Angular velocity	ω_{Lx}	ω_{Ly}
Torque at joint	T_{Lx}	T_{Ly}

Table 2.3: State variables for the leg of the one-legged robot

Slider Length/Position	l_s
Velocity	v_s
Force at joint	F_s

Table 2.4: State variables for the slider of the one-legged robot

Desired Velocity	$v_{x,des}$	$v_{y,des}$
Desired Position	$P_{x,des}$	$P_{y,des}$
Desired Contact Position	$p_{x,des}$	$p_{y,des}$

Table 2.5: Other state variables for the one-legged robot

2.3 Control Algorithm for 2D Motion

The 2D one-legged robot translates along the x-axis and hops along the z-axis, thus making the motion two-dimensional. In order for x-axis translation to occur,

the leg of the robot rotates along the y-axis.

There are two primary categories of equations that govern the motion of the one-legged hopping robot (Raibert, 1986).

The first category: Equations (2.1) to (2.3) are for the robot in the sky, before the leg touches the ground. It defines the desired contact position, leg angle and torque that needs to be applied to the legs to achieve the desired leg angle. The variable r here denotes the total length of the leg and maximum slider position.

$$p_{x,des} = \frac{T_g v_{Bx}}{2} + k_6(v_{Bx} - v_{x,des}) \tag{2.1}$$

$$\theta_{Ly,des} = \theta_{By} - \sin^{-1}(\frac{p_{x,des}}{r}) \tag{2.2}$$

$$T_{Ly} = -k_1(\theta_{Ly} - \theta_{Ly,des}) - k_2\omega_{Ly} \tag{2.3}$$

The second category: Equations (2.4) and (2.5) define the torque and thrust applied by the leg in the contact stage, to hop and keep a stable pitch angle.

$$T_{Ly} = k_3\theta_{By} + k_4\omega_{By} + k_5 v_{Bx} \tag{2.4}$$

$$F_s = k_s l_s \tag{2.5}$$

The parameters that govern these equations are k_i, where $i \in \{1, 2, ..., 6\}$ are feedback gains, and k_s, which is the spring constant of the slider joint. The desired velocity is $v_{x,des} = p_{x,des} - p_{Bx}$, keeping within a maximum velocity limit.

2.4 Control Algorithm for 3D Motion

Previously, 2D motion algorithm was discussed. If the robot needs to have the capability for y-axis translation as well, then the algorithm can be repeated again, after changing the axes (Raibert, 1986). Moreover, the joint that connects the leg to the body will change from hinge to universal joint, in order to allow the leg to

k_s	2000
k_1	100
k_2	10
k_3	1000
k_4	300
k_5	280
k_6	0.07
Maximum Velocity	1.2
Maximum Leg Angle	30°
Min and Max Slider Positions	0.10 to 0.17
Ground Time (T_g)	0.16

Table 2.6: Values given to parameters and constants of the one-legged robot

rotate about the x-axis as well. For convenience, the algorithm has been mentioned here again, after adding motion about another axis.

$$p_{x,des} = \frac{T_g v_{Bx}}{2} + k_6(v_{Bx} - v_{x,des}) \tag{2.6}$$

$$p_{y,des} = \frac{T_g v_{By}}{2} + k_6(v_{By} - v_{y,des}) \tag{2.7}$$

$$\theta_{Ly,des} = \theta_{By} - \sin^{-1}(\frac{p_{x,des}}{r}) \tag{2.8}$$

$$\theta_{Lx,des} = \theta_{Bx} - \sin^{-1}(\frac{p_{y,des}}{r}) \tag{2.9}$$

$$T_{Ly} = -k_{1y}(\theta_{Ly} - \theta_{Ly,des}) - k_{2y}\omega_{Ly} \tag{2.10}$$

$$T_{Lx} = -k_{1x}(\theta_{Lx} - \theta_{Lx,des}) - k_{2x}\omega_{Lx} \tag{2.11}$$

$$T_{Ly} = k_{3y}\theta_{By} + k_{4y}\omega_{By} + k_{5y}v_{Bx} \tag{2.12}$$

$$T_{Lx} = k_{3x}\theta_{Bx} + k_{4x}\omega_{Bx} + k_{5x}v_{By} \tag{2.13}$$

$$F_s = k_s l_s \tag{2.14}$$

It can be seen here that the parameters change from k_i to k_{ix} and k_{iy}. Naturally, the value of these parameters will be different as well. For the sake of simplicity and maintaining a symmetry in x and y axes, the shape of the body of the robot was changed. The dimensions taken are (0.8, 0.8, 0.1) instead of (0.8, 0.1, 0.1). This is illustrated in Figure 2.2 below. However, one is free to choose their own dimensions as long as the value of the parameters along both the axes is found for stable motion.

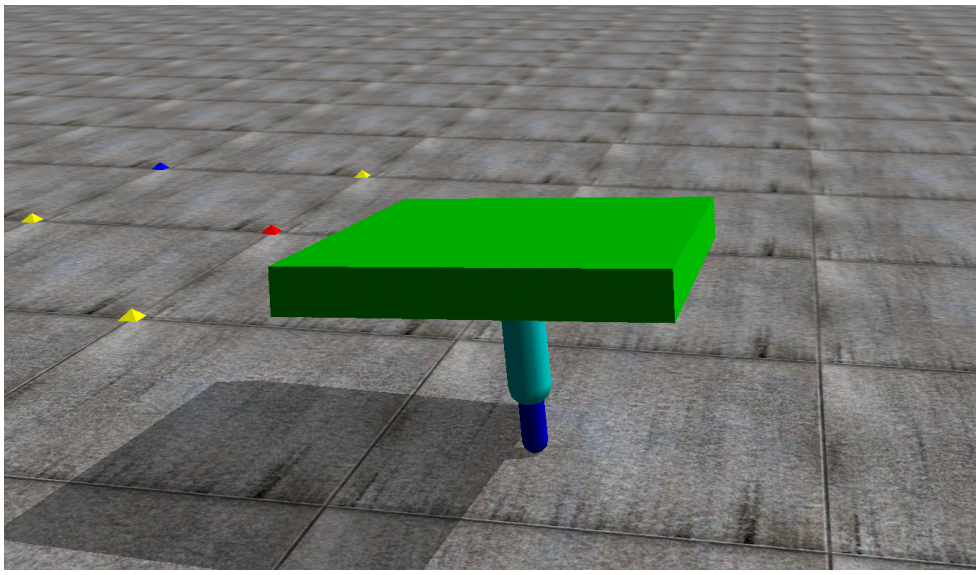

Figure 2.2: Illustration of the 3D one-legged robot, having equal dimensions in x and y axes

2.5 Calculating State Variables

The positions and forces at joints can be taken directly from the given functions in ODE.

A first order low pass and differential filter is used to calculate velocity from displacement, acceleration from velocity, etc. The equations governing the same are

given below.

$$G(s) = \frac{\omega_c}{\omega_c + s} s \tag{2.15}$$

$$y_n = \frac{2\omega_c(x_n - x_{n-1}) + (2 - \omega_c T)y_{n-1}}{2 + \omega_c T} \tag{2.16}$$

The value of ω_c is taken to be 30. T denotes the filter time, which is equal to the time duration between each step of the simulation (0.005).

Chapter 3

Parametric Study

Section 2.3 introduced the main governing equations of motion for the one-legged hopping robot in the two-dimensional space (not considering y-axis motion). These equations include various parameters, which either act like a spring constant (k_s) or as feedback gains (k_i). The parameters k_s, k_1, k_2, k_3, k_4 and k_5 were further studied to understand how their changes affect the overall motion and stability of the robot.

Through trial and error, the 6 parameters were varied when the slider mass m_s, leg mass m_L and body mass m_b were changed. Observations are given in Tables 3.1, 3.2 and 3.3.

3.1 Conditions for stability

The three conditions for stability are given as follows:

1. Pitch angle should be close to 0.
2. Robot should hop at an optimum height to allow leg movement but not let pitch angle get far from 0.
3. Leg should reach close to neutral point at contact.

	Original	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
m_b	10	10	10	10	10	10	10	10	10	10	10	10
m_L	3	3	3	3	3	3	3	3	3	3	3	3
m_s	0.7	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
k_s	2000	2613	2029	4000	2000	2000	2291	2500	2398	3000	4000	2363
k_1	100	100	100	100	100	100	100	100	200	50	100	100
k_2	10	10	10	10	10	10	10	10	10	10	40	5
k_3	1000	1000	1000	1000	1000	2000	500	100	1000	1000	1000	1000
k_4	300	300	300	400	200	300	300	300	300	300	300	300
k_5	280	280	350	280	280	280	280	280	280	280	280	280
	Stable	Stable	Stable	Unstable, tilting for	Stable	Unstable	Stable	Stable, tilting back	Stable, leg moving faster	Unstable, leg too slow	Unstable, leg too slow	Stable, leg moving faster

Table 3.1: Observations when slider mass m_s was varied

	Original	**1**	**2**	**3**	**4**	**5**	**6**	**7**
m_b	10	10	10	10	10	10	10	10
m_L	3	6	6	6	6	6	6	6
m_s	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
k_s	2000	4172	12923	2711	2498	2900	4500	4500
k_1	100	100	100	100	100	100	200	100
k_2	10	10	10	10	10	10	10	5
k_3	1000	1000	1000	1000	1000	500	1000	1000
k_4	300	300	300	300	200	300	300	300
k_5	280	280	280	350	280	280	280	280
	Stable	Stable, body leaning forward	Stable, jumping too high	Stable, slow	Stable, smooth	Stable, slow	Stable, body leaning forward, faster leg in air	Stable, body leaning forward, faster leg in air

Table 3.2: Observations when leg mass m_L was varied

	Original	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
m_b	10	20	20	20	20	20	20	20	20	20	20	20
m_L	3	3	3	3	3	3	3	3	3	3	3	3
m_s	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
k_s	2000	2602	3687	2423	2328	2000-3000	3100	2700	2000-5000	2500	2500	2400
k_1	100	100	100	200	100	100	100	100	100	100	100	100
k_2	10	10	10	10	5	10	10	10	10	10	10	10
k_3	1000	1000	1000	1000	1000	500	500	750	1000	1000	1000	1000
k_4	300	300	300	300	300	300	300	300	200	250	300	300
k_5	280	280	350	280	280	280	280	280	280	280	320	350
	Stable	Stable	Stable	Stable, slow	Stable, slow	Unstable, leaning back	Stable, slow, leaning back	Stable, slow, leaning back	Unstable, leaning back	Stable, slow, leaning back	Stable, slow, leaning back	Unstable

Table 3.3: Observations when body mass m_b was varied

m_b	min k_s	max k_s
10	1670	4256
20	2602	3687
30	3749	4509
40	4659	5200

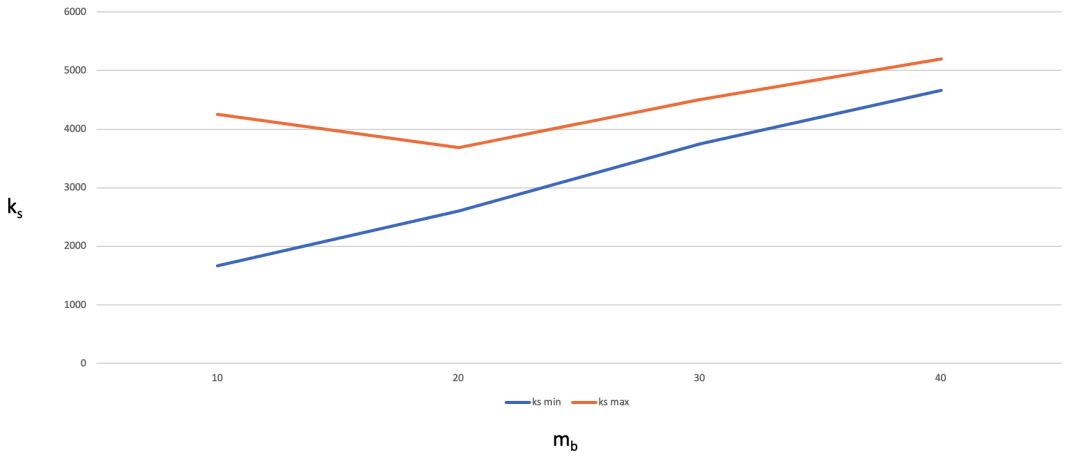

Figure 3.1: Graph depicting the stable region of k_s when m_b is varied (Upto T=10, other parameters are default)

3.2 Conclusions

Through the observations given in the above tables, the following conclusions can be made regarding the effect of these parameters on the motion and stability of the robot.

1. Increasing k_s will make the robot jump higher.
   $k_s \uparrow \implies p_{bx} \uparrow$

2. Increasing k_1 will make the leg reach the neutral point faster.
   $k_1 \uparrow \implies \omega_L \uparrow$

3. Decreasing k_2 will make the leg reach the neutral point faster.
   $k_2 \downarrow \implies \omega_L \uparrow$

4. Decreasing k_3 will make the body tilt backwards.
   $k_3 \downarrow \implies \theta_{by} \downarrow$

5. Decreasing k_4 will make the body tilt backwards.
   $k_4 \downarrow \implies \theta_{by} \downarrow$

6. Increasing k_5 will make the body tilt backwards.
   $k_5 \uparrow \implies \theta_{by} \downarrow$

Chapter 4

Biped Robot

The model for biped robot includes adding an extra leg and slider to the same one-legged robot in the previous chapter. For the sake of simplicity and planar motion, the second leg is attached to the body at the same point as the first leg.

The dimensions and mass of the additional leg and slider is same as the first leg. The joints and their parameters are also the same as the first leg.

Leg 1	
Angular Position	θ_{L1}
Desired Leg Angle	$\theta_{L1,des}$
Angular velocity	ω_{L1}
Torque at joint	T_{L1}
Leg 2	
Angular Position	θ_{L2}
Desired Leg Angle	$\theta_{L2,des}$
Angular velocity	ω_{L2}
Torque at joint	T_{L2}

Table 4.1: State variables for the two legs of the biped robot

Slider 1	
Slider Length/Position	l_{s1}
Velocity	v_{s1}
Force at joint	F_{s1}
Slider 2	
Slider Length/Position	l_{s2}
Velocity	v_{s2}
Force at joint	F_{s2}

Table 4.2: State variables for the sliders of the biped robot

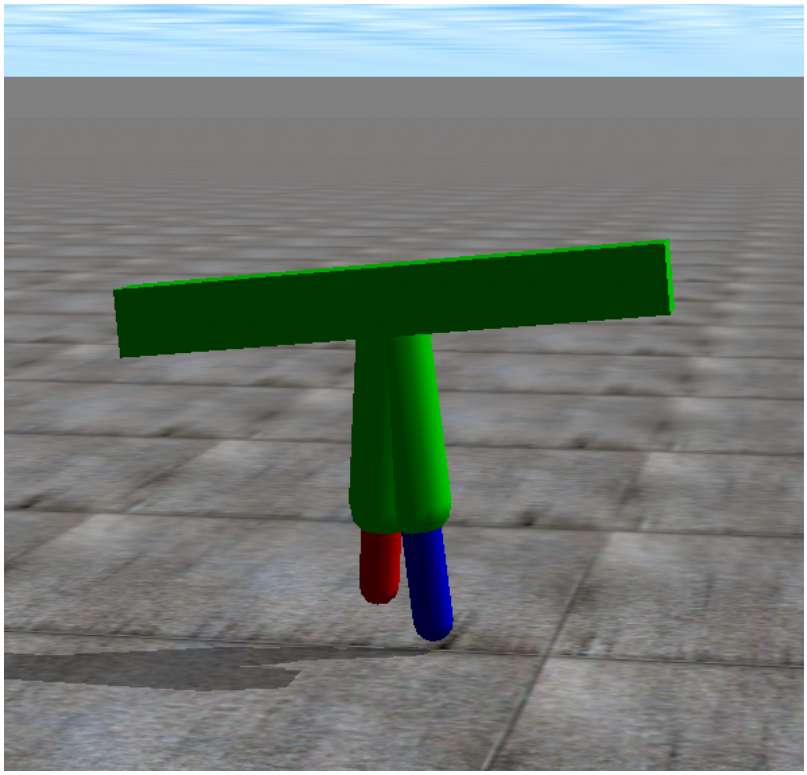

Figure 4.1: Illustration of the biped robot

Body	
Shape	Box
Dimensions (x,y,z)	(0.8, 0.1, 0.1)
Mass	10
Coordinates of CoM (world)	(0, 0, 1)
Legs (both)	
Shape	Capsule
Dimensions (l, r)	(0.3, 0.05)
Mass	0.2
Coordinates of CoM (world)	(0, 0, 0.85)
Slider	
Shape	Capsule
Dimensions (l, r)	(0.4, 0.03)
Mass	0.1
Coordinates of CoM (world)	(0, 0, 0.5)

Table 4.3: Physical parameters of the biped robot

4.1 Control Algorithm

The equations which govern the motion of the one-legged hopping robot can also be used in the biped robot, where the motion of each leg corresponds to the motion of the one-legged robot. The only difference here is that there is an additional switching unit that assigns the necessary equation of motion to each leg.

There is a resting leg and an active leg in every snapshot of the biped robot's gait. The active leg is the one where the control algorithm is applied, and at this time, the other leg is retracted to avoid contact with the ground.

The switching algorithm to assign this active and resting leg is as follows.

1. Two variables, State1 and State2, check the contact of Leg1 and Leg2 respectively with the ground - 1 if the leg touches the ground, and 0 if it doesn't.

2. Is either leg in the sky (SKY1 and SKY2) or touching the ground (GROUND1 and GROUND2)?.

3. The variable legState is initiated with the value 1. Four if-else conditions are implemented in the simulation loop:

```
if(SKY1    && SKY2    && legState == 4) legState = 1;
if(GROUND1 && SKY2    && legState == 1) legState = 2;
if(SKY1    && SKY2    && legState == 2) legState = 3;
if(SKY1    && GROUND2 && legState == 3) legState = 4;
```

4. A switch-case can be implemented based on these 4 values of legState.

   - legState = 1 denotes both legs are in sky, leg 1 should prepare for landing
   - legState = 2 denotes leg 1 is in contact with the ground, leg 2 should be retracted
   - legState = 3 denotes both legs are in sky, leg 2 should prepare for landing
   - legState = 4 denotes leg 2 is in contact with the ground, leg 1 should be retracted

5. The slider joints have an additional medium limit. To ensure that the resting leg does not come in contact with the ground, the active leg that is in contact has a minimum slider joint length that exceeds the original minimum length by some pre-defined value. This ensures that while the leg is retracting and gaining potential energy, the leg length does not reach the minimum length of the resting leg. This is achieved by varying the dParamLoStop property of dJointSetSliderParam function. When the legs are in the sky, this property is again reverted to the original setting.

Assuming leg 1 is active and leg 2 is resting, the equations used to control the resting leg are:

$$\theta_{L2,des} = -\theta_{L1} \tag{4.1}$$

$$T_{L2} = k_1(\theta_{L2,des} - \theta_{L2}) - k_2\omega_{L2} \tag{4.2}$$

$$F_{s2} = constant \tag{4.3}$$

The constant is taken to be -100 (negative force for retraction).

4.2 Parameters

The parameters for equations (2.1) to (2.5) were changed to achieve a stable and smooth gait. The ratio of the body's mass to the leg's mass should be more than 10:1, so that the leg motion does not significantly impact the pitch angle of the body.

k_s	400
k_1	100
k_2	10
k_3	800
k_4	200
k_5	40
k_6	0.07
Maximum Velocity	1.5
Maximum Leg Angle	30°
Min, Medium and Max Slider Positions	0.0, 0.05, 0.075
Step Time	0.004
Ground Time (T_g)	0.096

Table 4.4: Values given to parameters and constants of the biped robot

Chapter 5

Quadruped Robot

5.1 Components

The quadruped robot includes four legs placed in a symmetrical nature. The dimensions and positions taken are illustrated in Figure 5.1.

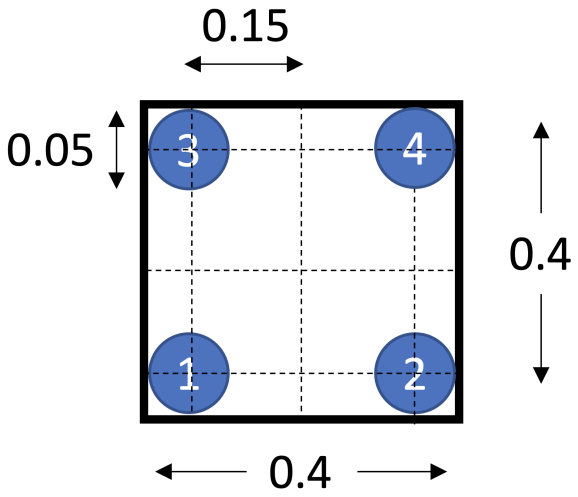

Figure 5.1: Positioning of legs in a four-legged robot (bottom view

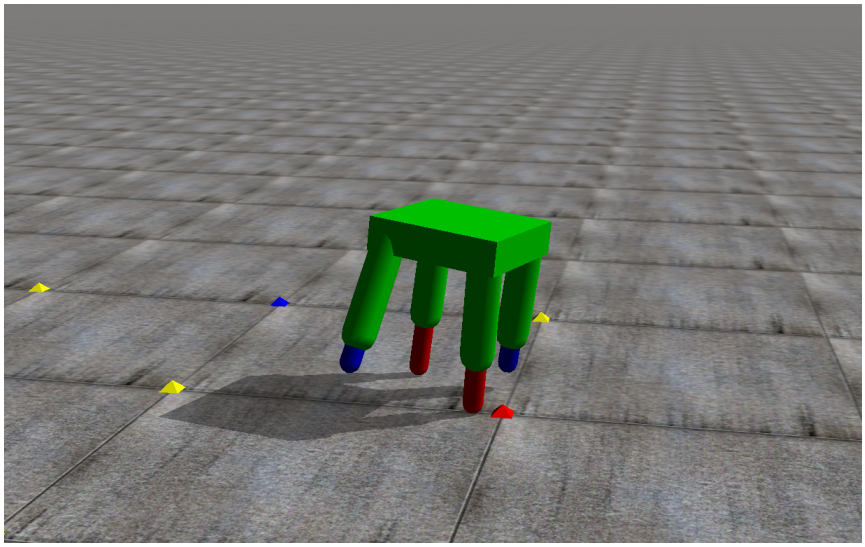

Figure 5.2: Illustration of the four-legged robot

5.2 Virtual Legs

In a quadruped robot, two legs can work in unison. In this case, this pair of legs can be thought to work as one single virtual leg. The control algorithms can be applied as if these two legs are replaced by one leg at the center (Sutherland, 1983).

The gait for quadruped robots can be of three types (refer 5.1):

1. Trot: Legs 1,4 and legs 2,3 form two pairs
2. Pace: Legs 1,2 and legs 3,4 form two pairs
3. Bound: Legs 1,3 and legs 2,4 form two pairs

5.3 2-Dimensional Motion

Initially, the motion of the quadruped robot was taken to be in the x-z plane. In this scenario, the position of the legs had to be taken such that the hinge joint

connecting the legs to the body had its center at y = 0. Using the control algorithm for the biped robot along with the concept of virtual legs, a stable translation along the x axis and hopping along the z-axis was achieved.

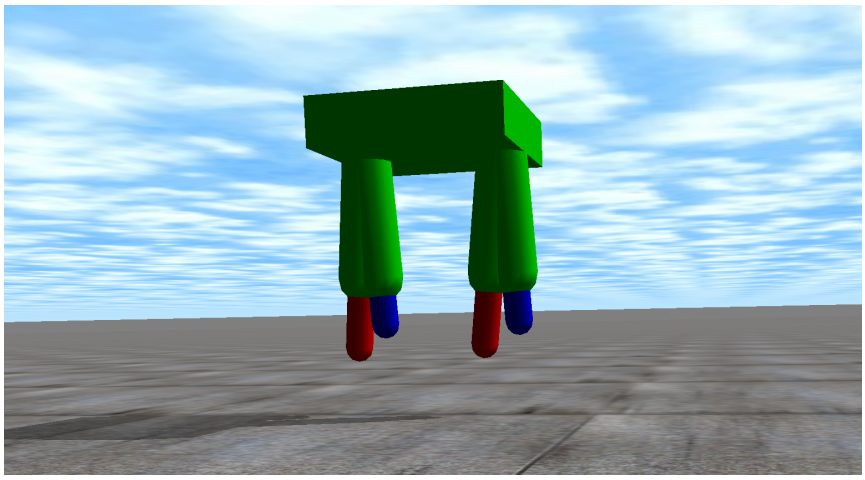

Figure 5.3: Illustration of the four-legged robot having 2D motion

When active leg is in the air:

$$p_{x,des} = \frac{T_g v_{Bx}}{2} + k_6(v_{Bx} - v_{x,des}) \tag{5.1}$$

$$\theta_{Ly,des} = \theta_{By} - \sin^{-1}(\frac{p_{x,des}}{r}) \tag{5.2}$$

$$T_{Ly} = -k_1(\theta_{Ly} - \theta_{Ly,des}) - k_2\omega_{Ly} \tag{5.3}$$

When active leg is in contact with the ground:

$$T_{Ly} = k_3\theta_{By} + k_4\omega_{By} + k_5 v_{Bx} \tag{5.4}$$

$$F_s = k_s l_s \tag{5.5}$$

For the resting leg:

$$\theta_{L2,des} = -\theta_{L1} \tag{5.6}$$

$$T_{L2} = k_1(\theta_{L2,des} - \theta_{L2}) - k_2\omega_{L2} \tag{5.7}$$

$$F_{s2} = constant \tag{5.8}$$

5.4 3-Dimensional Motion

Since the legs are placed in the x-y plane, rather than at the origin, an additional axis needs to be taken care of, for stability. Previously, the control algorithm was implemented for x-axis motion only. The same algorithm can be implemented for y-axis motion as well, as has been illustrated in the one-legged robot.

When leg is in the air:

$$p_{x,des} = \frac{T_g v_{Bx}}{2} + k_6(v_{Bx} - v_{x,des}) \tag{5.9}$$

$$p_{y,des} = \frac{T_g v_{By}}{2} + k_6(v_{By} - v_{y,des}) \tag{5.10}$$

$$\theta_{Ly,des} = \theta_{By} - \sin^{-1}(\frac{p_{x,des}}{r}) \tag{5.11}$$

$$\theta_{Lx,des} = \theta_{Bx} - \sin^{-1}(\frac{p_{y,des}}{r}) \tag{5.12}$$

$$T_{Lx} = -k_x 1(\theta_{Lx} - \theta_{Lx,des}) - k_{x2}\omega_{Lx} \tag{5.13}$$

$$T_{Ly} = -k_{y1}(\theta_{Ly} - \theta_{Ly,des}) - k_{y2}\omega_{Ly} \tag{5.14}$$

When leg is in contact with the ground:

$$T_{Lx} = k_{x3}\theta_{Bx} + k_{x4}\omega_{Bx} + k_{x5}v_{By} \tag{5.15}$$

$$T_{Ly} = k_{y3}\theta_{By} + k_{y4}\omega_{By} + k_{y5}v_{Bx} \tag{5.16}$$

$$F_s = k_s l_s \tag{5.17}$$

There is also a need to synchronise ground contact between the two legs that are part of a virtual leg. The calculations for this are explained in detail in (Raibert, 1986).

Chapter 6

Conclusion and Future Work

The work presented here provides a sneak peek into how the hopping robot would behave when it is actually developed. The algorithms have been tested in a simulated environment, and the motion of the robots is as expected. The project's purpose has been fulfilled and the outcomes of this project can be carried forward to optimize the motion, enable path planning and motion in uneven terrains, and initiate more types of gait, especially in the quadruped robot.

In the future, the code can be used to program a control unit that can actuate the legs and the slider to the desired position. If a hopping robot is developed in the future, the position data can be acquired by sensors or an inertial measurement unit (IMU). The control unit can calculate and accordingly actuate the joints to achieve a stable motion of the robot.

Hopping robots may not just include an actuated slider. A hinge joint in between can act like a knee to resemble the gait of any four-legged animal. There are some benefits of hopping robots over others, like wheeled robots. Unstructured terrains can be better explored and gait is very versatile. However, in soft terrains, problems may arise. There is potential to improve hopping robots such that they can traverse even marshy or sandy land.

Appendix A

Code

The codes for simulating one-legged, biped and quadruped robots are available on `https://github.com/kannykanishk/robot_ode`. Simulation videos are available on `https://youtube.com/playlist?list=PLnj7y7LL42Gt6VvOh8YzYUvMQzWtFuMrF`.

Follow these steps if you wish to run the code in your PC:

1. Download ODE by following the steps on their official webpage (Smith, 2001b). You will find a folder called "ode-(version name)" in the directory where you downloaded it. For example, it may be named "ode-0.16.2".

2. Make a new folder inside the ode folder and rename it however you like. This is where you'll be adding your codes.

3. Download any one of the folders given on GitHub into your new folder.

4. Find the "textures" folder in ode, copy the path, paste it in "texturepath.h".

5. Check the file paths in "Makefile", ensure that they match those in your PC.

6. In the terminal, go to the directory/folder where all the code files and "Makefile" is present. This is the folder you downloaded from GitHub. Run the commands "make" and then "./main".

References

Arslan, E. and Yıldırım, Ş. (2018). "ODE (Open Dynamics Engine) based walking control algorithm for six legged robot". *Journal of New Results in Science*, 7:35 – 46.

Demura, K. (2008). "ODE Elementary Lecture". `https://demura.net/tutorials`. Accessed August 2, 2021.

Kooijman, R. (2010). "Evaluation of open dynamics engine software". *Internship report, Eindhoven University of Technology*, 3:1. Accessed December 15, 2021.

Raibert, M. H. (1986). "Legged Robots That Balance". *Massachusetts Institute of Technology*.

Sato, M., Kanda, A., and Ishii, K. (2009). "Simultaneous optimization of a robot structure and a control parameter using evolutionary algorithm and dynamics engine". *IEICE Proceedings Series*, 43(A4L-D3).

Smith, R. (2001a). "Manual - ODE". `http://ode.org/wiki/index.php?title=Manual`. Accessed August 2, 2021.

Smith, R. (2001b). "Open Dynamics Engine". `http://ode.org`. Accessed August 2, 2021.

Sutherland, I. E. (1983). "A Walking Robot". *Pittsburgh: The Marcian Chronicles, Inc.*

Yıldırım, Ş. and Arslan, E. (2018). "ODE (Open Dynamics Engine) based stability control algorithm for six legged robot". *Measurement*, 124:367–377.