Czech Technical University in Prague
Faculty of Electrical Engineering

Rigid body dynamics model of bipedal wheeled robot

Bacherol thesis

Radomír Macíček

Bacherol program: Cybernetics and robotics
Supervisor: Ing. Krištof Pučejdl

Prague, January 2022

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Macíček Radomír** Personal ID number: **474418**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Rigid body dynamics model of bipedal wheeled robot

Bachelor's thesis title in Czech:

Matematický model dynamiky dvounohého kolového robotu

Guidelines:

Using a suitable method derive a full 3D rigid body dynamics model of a bipedal wheeled robot SK8O, including reaction forces from the ground. Resulting model should be in some standard format for dynamical system models allowing simulation in Matlab, and suitable for model-based control (e.g., MPC).
(1) Review the literature and select a method for model derivation (Newton-Euler algorithm / Featherstone articulated body method...).
(2) Compile a mathematical model based on the physical parameters of the SK8O robot.
(3) Verify the model by comparing the simulation results with a high-fidelity rigid body dynamics simulator.

Bibliography / sources:

[1] Featherstone, Roy. Rigid Body Dynamics Algorithms. Springer, 2014.
[2] Klemm, Victor, et al. LQR-Assisted Whole-Body Control of a Wheeled Bipedal Robot with Kinematic Loops. IEEE Robotics and Automation Letters, 2020, 5.2: 3745-3752.
[3] Kim, Donghyun, et al. Computationally-Robust and Efficient Prioritized Whole-Body Controller with Contact Constraints. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 1-8.
[4] Kim, Donghyun, et al. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control. Preprint, 2019.

Name and workplace of bachelor's thesis supervisor:

Ing. Krištof Pučejdl, Department of Control Engineering, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **14.02.2021** Deadline for bachelor thesis submission: **04.01.2022**

Assignment valid until:
by the end of winter semester 2022/2023

Ing. Krištof Pučejdl	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I declere that this work is all my own work and I have cited all sources I have used in the bibliography. This Thesis has also not been submitted for any degree in previosly.

Radomír Macíček
January 2022

Abstract

This thesis focuses on the modelling full rigid body of a bipedal robot. The main objective is to find the equations of our robot's motion. First, we describe the mathematical knowledge. Then we describe our modelling tool. At the end, we describe our modelling approach. We make the kinematic loop. We create it on the simplified model. We use the force constraints to reach it. We finish the thesis with a summary of our results and suggestions for future work.

Keywords: equation of motion (EoM), center of gravity(CoG), center of mass (CoM), jacobian

Abstrakt

Tato práce se zabývá modelováním plného matematického modelu dvounohého robota. Hlavním úkolem je najít rovnice pohybu našeho robota. Nejříve popíšeme matematické znalosti. Následně popíšeme naše modelovací nástroje. Nakonec popíšeme náš přístup k modelování. Vytvoříme kinematickou smyčku na zjednodušeném modelu. Aplikujeme ji pomocí silových omezeních. Práci zakončíme shrnutím našich výsledků a návrhem budoucí práce.

Keywords: rovnice pohybu (EoM), centrum gravitace (CoG), těžiště (CoM), jakobián

Acknowledgements

I want to thank my supervisor Krištof Pučejdl for the patience, guidance and educational approach. I am grateful that he gives me a chance to work on this topic. I also want to thank my family for their support and the text correction.

List of Figures

Contents

Chapter 1

Introduction

This thesis is about how to model the two-legged wheeled robot. We have to deal with a new type of modeling of a floating body. The main task was to achieve a full mathematical model. We use the projected Newton-Euler method to calculate the motion equations. We can only verify our results by the dimension control because it is just a mathematical model. The result of our work is stored on the GitLab. The GitLab contains all scripts with our calculations, which corresponds to our task.

1.1 Problem statement

The objective of this thesis is to create and calculate the full model. Firstly, we have to find a library that helps us to calculate the kinematics and dynamics of our robot. Secondly, we have to describe our robot in the library. Finally, we have to add features, which are not supported by the library. We talk about force constraints, which are used to deal with kinematic loops. The real testing was not done due to COVID 19 pandemic state.

1.2 Related work

Robots became more popular in recent years. The increasing popularity of robotics also rises the popularity of legged robots. One of the most famous leg robots is Spot made by Boston Dynamics. The source of our inspiration is much geographically closer. It is a robot Ascento made at the ETH in Zurich [1]. Sk8o shares the same topology but construction is different. The other thesis was written on this type of robot which contains the driving method of the linear pendulum [2]. We are focusing just on the robot's model. That means we are not able to control this robot, we want to get his moving equations, which have high complexity. We followed the whole body control approach that was published

for the Ascento robot and tried to apply it to our robot, even though the robots are not same.

1.3 Robot description

Even though I did not participate in the robot's construction, we consider it essential to mention the basis of its construction . We have to describe how this robot looks like for better imagination and better orientation in this thesis. As we already mentioned the robot has two legs with a closed kinematic chain. Both of them have a torsion spring placed in the knee. The legs are ending with wheels. We have to work with the real dimensions of the robot because we want to get as close to reality as we can. The main task of the thesis was to create a kinematic and dynamic model, thus we don´t care about robot electronics. The robot's real dimensions we can see at Figure 1.1.

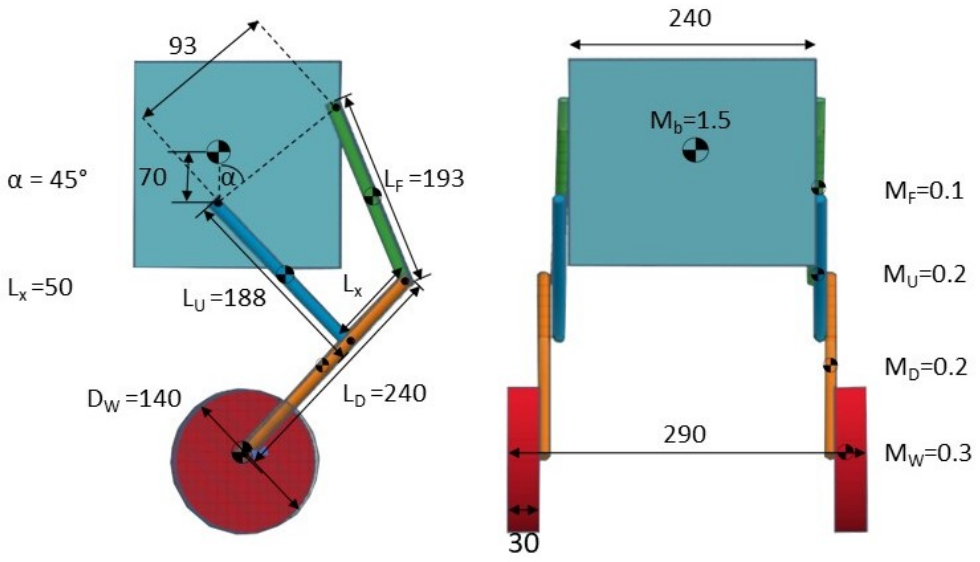

Figure 1.1: The robot's dimensions

Chapter 2

Methods

2.1 Dynamics

This section will describe the mathematical and physical background behind our program and calculations. It also explains what we will be looking for and why it is not easy to find a solution [3].

2.1.1 State of the art

The main goal of this work is to find the Equation of the motion (EoM) for our robot. The multibody description will be in formulated as equation (2.1) .

$$M(q)\ddot{q} + b(q, \dot{q}) + g(q) = \tau + J_c(q)^T F_c \tag{2.1}$$

It consists of the following components.

$M(q) \in R^{n \times n}$	is generalised mass matrix
q, $\dot{q}$, $\ddot{q} \in R^n$	are generalised coordinates, velocities and accelerations
$b(q, \dot{q}) \in R^n$	Coriolis and centrifugal component
$g(q) \in R^n$	Gravitational terms
$\tau \in R^n$	External generalised coordinates
$F_c \in R^n$	External Cartasian forces
$J_c(q) \in R^{n \times n}$	Geometric Jacobian corresponding to external forces
$n \in N$	Number of degrees of the freedom

We use formulation for fixed-base systems, which is not commonly used for floating base systems. We can use three methods, but the results will be equivalent. We are using the projected Newton-Euler, which is based on the other two methods.

The Newton-Euler method uses the principle of conservation of angular and linear

momentum for all links. Then It applied it in the Cartesian space. The second is a method known as the Lagrange method. It depends on the generalised coordinates. The coordinates create scalar energy-based functions. The coordinates will also adhere to the minimalisation principle. This means that the kinematic constraints of the system are satisfied by the trajectories. This brings us to the method we will be using for our computation. It combines the advantages of the Newton-Euler and Lagrange methods. It uses the reformulation Cartesian coordinates in terms of generalised coordinates from Newton-Euler. It satisfies the applicable kinematic constraints due to directly feasible motions of the system.

To make the mathematics behind the following calculations of EoM explicit, we need to describe our mathematical methods, conventions, and terms. We will start with classical mechanics.

2.1.1.1 Newton's law for particles

We describe the motion of point masses as mass with an infinitely small dimension. That means all mass is concentrated in one point. The point is defined by position vector r. Newton‘s second law is as follows.

$$\ddot{r}m = F \tag{2.2}$$

We can think of infinitesimal mass dm to infinite small forces concentrated as at the position of particles because we have defined:

$$\ddot{r}\dot{m} = dF \tag{2.3}$$

We will use this knowledge as the basis for our calculation. They will have an essential role in the derivative method we described. We will use them for a case of the Center of Mass of rigid bodies.They describe the effects of the internal forces.

2.1.1.2 Virtual displacement

To simplify our thoughts, we use the concept of variable notation δ. It behaves the same as the differential d but describes something different. The displacement describes the infinitesimal change of the coordinates without changing time. It allows us to compute all time-dependent quantities independently of the time. This is the reason why it is called virtual. It has the following relation:

$$\delta r(q,t) = \sum_{i=1}^{n} \frac{\partial r}{\partial q_k} \delta q_k \tag{2.4}$$

2.1.1.3 Virtual displacement of the single rigid body

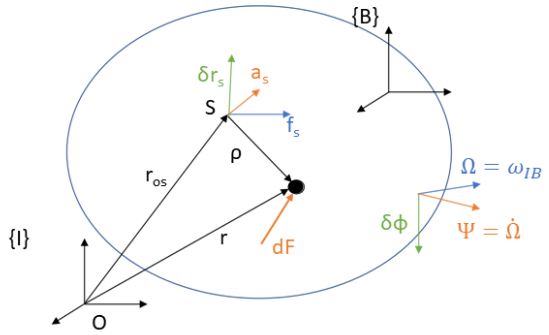

Figure 2.1: Virtual displacement of single body

A body with mass in 3D Cartesian space is represented by many particles placed together to form our rigid body. We must compute each infinitesimal point-mass dm. Each mass point is a subject motion of a total body. Now we can assign an absolute position and velocity in instant time to them. We consider another point S on the body. It has a relative position ρ of dm written in the point S. We use the rigid body kinematics to describe the motion f an arbitrary point-mass dm. It is defined body b through another point S:

$$r = r_{os} + \rho \tag{2.5}$$

$$\dot{r} = v_s + \Omega \times \rho \tag{2.6}$$

$$\ddot{r} = a_s + \Psi \times \rho + \Omega \times (\Omega \times \rho) \tag{2.7}$$

The r_{Os} is the absolute position of point S. The ρ is the relative position of dm in S coordinates. V_s,a_s,Psi and Ω are the final velocities and acceleration point S. Then we applied the virtual displacement,

We describe the motion of point masses as mass with an infinitely small dimension. That means all mass is concentrated in one point. The point is defined by position vector r. The Newton‘s second law is as follows.

$$\delta r = \delta r_S + \delta\Phi \times \rho \tag{2.8}$$

2.1.1.4 Virtual displacement of Multi-Body Systems

The multi-body system can only exhibit motion which is limited by the joints. This limits the relative motion between links. We describe it using generalized coordinates. The previous concepts have now become relevant to our multi-body system. We are now using the general coordinates q.

$$\begin{pmatrix} V_s \\ \Omega \end{pmatrix} = \begin{bmatrix} J_p \\ J_R \end{bmatrix} \dot{q} \tag{2.9}$$

$$\begin{pmatrix} a_s \\ \Psi \end{pmatrix} = \begin{bmatrix} J_p \\ J_R \end{bmatrix} \ddot{q} + \begin{bmatrix} \dot{J}_p \\ \dot{J}_R \end{bmatrix} \dot{q} \tag{2.10}$$

We can now use the displacement system to the multi-body system,consistent with the joints.

$$\begin{pmatrix} \delta r_s \\ \delta \Phi \end{pmatrix} = \begin{bmatrix} J_p \\ J_R \end{bmatrix} \delta q \tag{2.11}$$

2.1.1.5 Principle of the virtual work

The following mathematical fundamental knowledge is the principle of virtual work. It says to us how big will be Work W if we move with our body in any direction δr with force F, known as the virtual displacement. It must be applied for each constraint force F_c in their points r_c. That means for the stable body following:

$$\delta W = \delta r_c^T \cdot F_c = 0 \tag{2.12}$$

We can use this definition if we extend this formulation with infinitesimal dm and consider d'Alambert's principle describing dynamic equilibrium for particles.

$$\delta W = \int_B \delta r^T \cdot (\ddot{r} - dF_{ext}) = 0 \tag{2.13}$$

dm	infinitemisal mass
dF_{ext}	external forces acting on dm
$\ddot{r}$	acceleration of element dm
δr	virtual displacement of dm
B	body system

2.1.2 Newton-Euler method

The first step is to describe the method on the simple body. We start with evaluating the principle of virtual work and simplify extract to a bare minimum.

$$0 = \delta W = \begin{pmatrix} \delta r_s \\ \delta \Phi \end{pmatrix}^T (\begin{bmatrix} I_{3\times3} & 0 \\ 0 & \Theta_s \end{bmatrix} \begin{pmatrix} a_s \\ \Omega \end{pmatrix} + \begin{pmatrix} 0 \\ \Omega \times \Theta_s \cdot \Omega \end{pmatrix} - \begin{pmatrix} F_{ext} \\ T_e xt \end{pmatrix}) \forall \begin{pmatrix} \delta r_s \\ \delta Phi \end{pmatrix} \tag{2.14}$$

We use these definitions to define laws of conservation and angular momentum:

$p_s = mv_s$	linear momentum
$N_s = \Theta_s \cdot \Omega$	angular momentum around center of gravity (CoG)
$\dot{p}_s = ma_s$	change in linear momentum
$\dot{N}_s = \Theta_s \cdot \Phi + \Omega \times \Theta_s \cdot \Omega$	change in angular momentum

We can not forget that a free moving body must fulfil the change in linear momentum equal to the sum of all external forces.

$$0 = \delta W = \begin{pmatrix} \delta r_s \\ \delta \Phi \end{pmatrix}^T (\begin{pmatrix} \dot{p_s} \\ \dot{N}_s \end{pmatrix} - \begin{pmatrix} F_{ext} \\ T_e xt \end{pmatrix}) \forall \begin{pmatrix} \delta r_s \\ \delta Phi \end{pmatrix} \tag{2.15}$$

From this interference, we get Newton-Euler formulations.

$$\dot{p_s} = F_{ext} \tag{2.16}$$

The external forces F_{ext} are acting through the COG.

$$\dot{N}_s = T_{ext} \tag{2.17}$$

The T_{ext} is the result of the external torque. We can not forget that numerical calculation must be expressed in the same coordinate system. It applies to the external torque and force. If the external forces do not act through the CoG, we must shift the equivalent pair to the CoG. Then we use this method for the multi body system.
The first step is to divide multi-body system into separate bodies at all joints. We consider every body as a single unit. The first step is to divide multi-body system into separate bodies. For all these bodies, We have to change the force constraint of F_i at the joints to external forces. For everybody, we apply the conservation of linear and angular momentum. We add 5 degrees of freedom if the connection is with the ideal joint. We must be sure that two bodies are connected correctly and do not move in any other direction. Many packages work with this method. We can apply if it is a hard or soft

constraint.

2.1.3 Lagrange method

It is another common approach for deriving the motion equation of the system. The system uses the so-called Lagrange method. It has origin in analytical mechanics. It is very close to d'Alambert and Hamilton's principle. The method centres around three concepts. Define generalized coordinates q and velocities u, which can or can not encode the information about constraints. The Lagrangian function $\mathcal{L}$ is the difference between total kinematic energy $\mathcal{T}$, and total potential energy $\mathcal{U}$.

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{2.18}$$

The Euler-Lagrange equation is applied to Lagrangian $\mathcal{L}$ and to all external generalized forces τ:

$$\frac{d}{dt}(\frac{\mathcal{L}}{\partial \dot{q}}) - (\frac{\mathcal{L}}{\partial q}) = \tau \tag{2.19}$$

The Lagrangian is the function of generalised coordinates q and velocities $\dot{q}$. It is also time-dependent on the time t. Hence we redefine the scalar energy functions $\mathcal{T} = \mathcal{T}(t, q, \dot{q})$ and $\mathcal{U} = \mathcal{U}(t, q)$, thus $\mathcal{L} = \mathcal{L}(t, q, \dot{q})$.

The best feature of this formulation is the elimination of all internal reaction forces of the system from EoM. On the other hand, Newton -Euler formulation explicitly account for them. To apply derivation of the EoM, we have to consider additional aspects before we use the three aforementioned concepts.

2.1.3.1 Kinetic energy

We compute kinematic energy through this equation:

$$\mathcal{T} = \sum_{i=1}^{n} (\frac{1}{2} m_i \dot{r}_{Si}^T \dot{r}_{Si} + \frac{1}{2} \Omega_{Si}^T \cdot \Theta_S i \cdot \Omega_{Si}) \tag{2.20}$$

We need to express the kinetic energy as a function of general quantities. We get this thanks to jacobian matrices, which are calculated for each body instead of the end effector. We use the following relationship:

$$\dot{r}_{Si} = J_s \dot{q} \tag{2.21}$$

$$\Omega_{Si} = J_{Ri} \dot{q} \tag{2.22}$$

We can fill it to the previous definition and get this simplified form:

$$\mathcal{T}(q,\dot{q}) = \frac{1}{2}\dot{q}^T(\sum_{i=1}^{n}(J_{Si}^T + J_{Ri}^T\Theta_{Si}J_{Ri}))\dot{q} \tag{2.23}$$

$$M(q) = \sum_{i=1}^{n}(J_{Si}^T + J_{Ri}^T\Theta_{Si}J_{Ri}) \tag{2.24}$$

The quantity M(q) is defined as the generalised mass matrix or the generalised inertia matrix. It is responsible for generating the inertial, centrifugal and Coriolis force in the final EoM.

2.1.3.2 Potential Energy

The potential energy has two fundamental contributions to the mechanical system. Firstly, the mass contribution act via gravitational potential energy. Secondly, the elastic element contribution is caused by the deflection energy. Each body bi has potential energy caused by the gravitational field potential field of the earth. We approximate it as linear and nondirectional, despite non-linear on large scales. The unit vector e_g acts through center of the mass (CoM) of each body. Because we know the position r_si of at the CoM of each body, we can compute the potential energy of each body as:

$$F_{gi} = m_i g e_g \tag{2.25}$$

$$\mathcal{U} = -\sum_{i=1}^{n} R_{Si}^T F_{gi} \tag{2.26}$$

Note that we can choose the zero energy level appropriately. We can not forget that many applications use elastic contributions such as springs or the other nonlinear components. We try to reasonably approximate the component to have a linear deflection to force or deflection-to-torque relationship. Then the potential energy contributions are described as:

$$\mathcal{U}_{E_j} = \frac{1}{2}k_j(d(q) - d_0)^2 \tag{2.27}$$

The d(q) is a function of generalised coordinates, which describe the deflection of the elastic element. The d_0 is the configuration during the rest state without any forces.

2.1.4 Projected Newton-Euler method

The last method is the Projected Newton-Euler method. It describes the deriving EoM. This method combines the classical Newton-Euler equations for dynamic equilibrium in

the Cartesian coordinates and Lagrange generalised coordinates with constraint complaints, as was said earlier. This allows us to use both derived methods. The proNEu result will be:

$$M(q)\ddot{q} + b(q,\dot{q}) + g(q) = \tau + J_c^T F_c \tag{2.28}$$

Because we will use this method as the main method of calculation in our library, we will perform a complete derivation from the basis of the remaining two. We directly applied it to the multi-body system.

$$\begin{aligned} \begin{pmatrix} \dot{p}_{Si} \\ \dot{N}_{Si} \end{pmatrix} &= \begin{pmatrix} ma_{Si} \\ \Theta_{Si}\Psi Si + \Omega Si \times \Theta_{Si} \cdot \Omega Si \end{pmatrix} \\ &= \begin{pmatrix} mJ_{Si} \\ \Theta_{Si} J_{Ri} \end{pmatrix} \ddot{q} + \begin{pmatrix} m\dot{J}_{Si}\dot{q} \\ \Theta_{Si} J_{Ri}\dot{q} + J_{Ri}\dot{q} \times \Theta_{Si} J_{Ri}\dot{q} \end{pmatrix} \end{aligned} \tag{2.29}$$

We get this definition after we connect the Newton-Euler with Lagrange definitions. We will continue developing this concept.We combine patterns (2.11) and (2.15) with multi-body system, which gives us the summation.

$$0 = \delta W = \delta q^T \sum_{i=1}^{n} (\begin{pmatrix} \dot{p}_{Si} \\ \dot{N}_{Si} \end{pmatrix} - \begin{pmatrix} F_{Ext,i} \\ T_{Ext,i} \end{pmatrix}) \forall q \tag{2.30}$$

We can combine the two concepts, which we calculated in (2.29) and (2.30)

$$0 = \sum_{i=1}^{n} \begin{pmatrix} J_{Si} \\ J_{Ri} \end{pmatrix}^T \begin{pmatrix} mJ_{Si} \\ \Theta_{Si} J_{Ri} \end{pmatrix} \ddot{q} + \begin{pmatrix} J_{Si} \\ J_{Ri} \end{pmatrix}^T \begin{pmatrix} m\dot{J}_{Si}\dot{q} \\ \Theta_{Si} J_{Ri}\dot{q} + J_{Ri}\dot{q} \times \Theta_{Si} J_{Ri}\dot{q} \end{pmatrix} - \begin{pmatrix} J_{Pi} \\ J_{Ri} \end{pmatrix}^T \begin{pmatrix} F_{Ext,i} \\ T_{Ext,i} \end{pmatrix} \tag{2.31}$$

Our results is different from the solution we are looking for (2.28), because we need to express our mass matrix M, Coriolis and centrifugal b and gravitational terms g.

$$M = \sum_{i=1}^{n} (J_{Si}^T m J_{Si} + J_{Ri}^T \Theta_{Si} J_{Ri}) \tag{2.32}$$

$$b = \sum_{i=1}^{n} (J_{Si}^T m \dot{J}_{Si} + J_{Ri}^T (\Theta_{Si}\dot{J}_{Ri}\dot{q} + \Omega_{Si} \times \Theta_{Si}\Omega_{Si})) \tag{2.33}$$

$$g = \sum_{i=1}^{n} -J_{Si}^T F_{g,i} \tag{2.34}$$

We can not forget to choose the correct coordinate base because the members with J_{Ri} jacobian applies to the different body than with J_{Si}.

The left of the equation is now computed. Now we look at the right side with external forces generalised by the force vector τ. We can express the external forces and torques to

generalised coordinates, but we must transform them from Cartesian coordinates through the appropriate jacobian matrices. We use the external forces as the F_j and external torque as T_k. The force F_j acts on point P_j, so we need the translational jacobian of that point J_{P_j}.

$$\tau_{F,ext} = \sum_{j=1}^{n} J_{Pj}^{T} F_j \tag{2.35}$$

$$\tau_{T,ext} = \sum_{k=1}^{n} J_{Rg}^{T} Tk \tag{2.36}$$

We use a similar method to the torque. In the end, we need one vector τ. We can easily add torque and force tau together because they are in the same coordinate system.

Chapter 3

Results

3.1 ProNEu Library

We use the Matlab proNeu library, which was created at E.T.H. university. The proNeu name means the projected Newton-Euler equations used to calculate our equation of motion. That means the library is a potent tool to find our sought solution. Unfortunately, it is not described enough to be user friendly[4]. So this chapter will be about its description.

3.1.1 The description of the library

The library works with three essential components: simulation script, dynamic script, and world script. The most interesting component for us is the dynamic component. This component calculates the rigid body dynamics as the EoM equations, then is stored as the data file. The world component computes and stores information about the simulation world and the world's limitations. The last one is the simulation script. It takes data from the other two and creates the visualised 3D simulation of our model.

3.1.2 The dynamics script

To compute the dynamics with this library, we have to simplify the robot into geometrical shapes with the Center of Mass (CoM). We give each entity a kinematic chain number. Then we can start filling body structure. The `body` structure is one of two input structures. We use numbers from the kinematic chain and connect each part by its parent. The variable `x` is the body number from the kinematic tree. Then we have to identify transition and rotation against parent. The joints are applied through `body(x).cs`. It has two options. The first one is `body(x).cs.P_r_PB`, which secures transition between child and parent. The second one is `body(x).cs.C_PB`, which secures rotational matrix. To change these variables, we have to define them as thy degree of freedom with a symbolic variable.

Then we use sub-body geometry `body(x).geometry`, which stores information about geometry `.params`, colour `.color`, type `.type` and offset `.offset`. The last subclass under is `body(x).param`. It stores information aboute position of the CoM `.B_r_BCoM`, rotations about the CoM `.C_BCoM`, the mass `.m` and the inertia vector `.B_theta`.

```
%Body
i=1;
body(i)                     = RigidBodyDescription_v2;
body(i).name                = 'B';
body(i).ktree.nodeid        = i;            % Body number from kinematic
    tree
body(i).ktree.parents       = i-1;          % Body parent's number from
   kinematic tree
body(i).cs.P_r_PB           = [0 0 0];      % Vector 3x1
body(i).cs.C_PB             = [];           % Rotation against parent
body(i).param.m             = m_b;          % Positive Real number
body(i).param.B_Theta_B     = [];
body(i).param.B_r_BCoM      = [0 0 0.7].';  % Vector 3x1
body(i).param.C_BCoM        = sym(eye(3));  % Rotational Matrix 3x3
body(i).geometry.type       = 'cuboid';     % other options are Sphere
   or Cylinder
body(i).geometry.issolid    = true;
body(i).geometry.params     = [body_l body_w body_h]; % Body dimension
   vector
body(i).geometry.values     = [body_l body_w body_h]; % Body dimension
   vector
body(i).geometry.offsets.r  = [0 0 0].';    % Body offset vector
body(i).geometry.offsets.C  = eye(3);       % Rotational Matrix 3x3
body(i).geometry.color      = [197 172 202]/255; %RGB color definintion
```

We can apply forces and torques through class `ftell`, which is the second input function. We have to specify the joint type `ftell.type` as rotational, linear and wrench. The example explains how to apply every type of joint.

```
j=1;                    %Example for rotational joint with torque
ftel(j) = ForceTorqueDescription_v2;
ftel(j).name    = 'W_R1';           %Joint name
ftel(j).type    = 'rotational';     %other options are linear and wrench
ftel(j).body_P  = 1;                %Parent number from kinematic
ftel(j).body_B  = 2;
ftel(j).P_r_R   = [];
ftel(j).B_r_A   = sym([0 0 0].');
ftel(j).B_T     = [0  T_Q1 0].';

j=2;                    %Example for linear joint with force
```

```
ftel(j) = ForceTorqueDescription_v2;
ftel(j).name    = 'F_Sbf';
ftel(j).type    = 'linear';
ftel(j).body_P  = 2;
ftel(j).body_B  = 3;
ftel(j).P_r_R   = sym([0 0 0].');
ftel(j).B_r_A   = sym([0 0 0].');
ftel(j).B_F     = [0 0 F_Sbf].';

j=1;   % Example of Environmental wrench
ftel(j) = ForceTorqueDescription_v2;
ftel(j).name    = 'W_E';
ftel(j).type    = 'wrench';
ftel(j).body_P  = 0;
ftel(j).body_B  = 1;
ftel(j).P_r_R   = [];
ftel(j).B_r_A   = sym([0 0 0].');
ftel(j).I_F     = [F_Ex F_Ey F_Ez].';  %Force to inertial system
ftel(j).I_T     = [T_Ex T_Ey T_Ez].';  %%Torque to inertial system
```

The penultimate thing which we have to define is system definitions. They tell us which variable is a generalised coordinate `q_j`. We also specify external forces and torques `tau_ext`. We can not forget their internal counterparts `tau_j`. Those are symbolic variables used in previous joint and body descriptions. In the end, we fill the `RobotModel`, which generate the model object.

```
%% System Definitions

% Definition of the joint DoFs of 2-link system
q_j  = [zeta_S].';

% Controllable joint forces/torques
tau_j = [F_Sbf T_Bx T_By].';

% External forces and torques
tau_env = [F_Cx F_Cy F_Cz T_Cx T_Cy T_Cz].';

%% Generate Full System Model using proNEu.v2

robot_name = 'Sk80Model';

%Generate the model object
robotmdl = RobotModel(body, ftel, q_j, tau_j, tau_env, I_a_g, 'name',
   robot_name, 'type', 'floating', 'orientation', 'cardanxyz', 'method',
    'proneu', 'symsteps', 100);
```

3.1.2.1 The World script

The world represents simulation conditions and stores information about the simulation space. They are stored in the `element` class. It is defined as the WorldElementDescripion. The `element` variable consists of `name` and `geometry` subclasses. The geometry saves the same information as the ones mentioned previously.

```
%% Set up the World

k=1;
element(k) = WorldElementDescription;
element(k).name = 'floor';
element(k).geometry.type = 'plane';
element(k).geometry.issolid = false;
element(k).geometry.params = [w_floor l_floor];
element(k).geometry.values = [10.0 10.0];
element(k).geometry.offsets.r = [0 0 1].';
element(k).geometry.offsets.C = eye(3);
element(k).geometry.color = [0.96 0.96 0.96];

%% Generate a Model of the World

% Set the world's name
world_name = 'Sk80World';

% Generate the world model object
worldmdl = WorldModel(element, @sk80_world_environemnt, 'name',
    world_name);
```

3.1.2.2 The Simulation script

This script loads data from the previous two saved data models. It allows us to simulate calculated models. Whenever we load data, we define the initial system state as `xinit` variable. Then we define `params`, which are static parameters from the model, which was expressed explicitly through symbols. After that, we set up the actuator controller as `controller` from function `RobotController`. Then we generate the simulation environment through function `RobotSimulator()`, which we save as a robot sim. The last part is visualisation. To visualise, we use all variables defined earlier. We can see how the simulation looks like at picture Figure 3.1,

```

% Define the internal actuator forces callback function.
controller = RobotController(@sk80_controller);

```

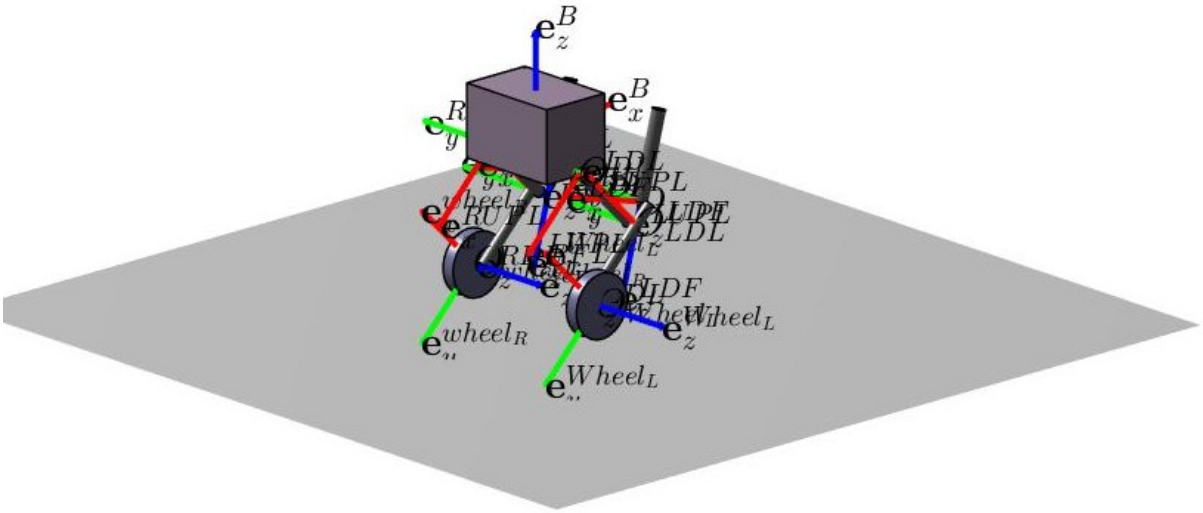

Figure 3.1: The robot simulation

```
% Create the robot simulation engine
robotsim = RobotSimulator(robotmdl, controller, worldmdl, 'solver', '
    fsfb');

% Generate 3D Visualization instance
robotviz = RobotVisualization(robotmdl,worldmdl);
robotviz.open();
robotviz.load();

% Give the simulator access to the 3D visualization
robotsim.setvisualizer(robotviz);

% Execute the simlation engine
robotsim.run(simconf);
```

3.1.2.3 Summary

The library is not only able to simulate and create our robot model. It also generates all matrices of a simplified model. This gives us a very powerful modelling tool because the complexity of this calculation is very high. The result will have the following equation.

$$M(q)\ddot{q} + b(q,\dot{q}) + g(q) = S_e^T \tau_{ext} + S_a^T \tau_{act} \tag{3.1}$$

We can see that the selection matrix S and τ is divided into two parts. The first part of the selection matrix S_{ext} takes care of the external forces of the τ_{ext}. The second part of the selection matrix shows us the actuator forces τ_{act}. The general tau we get by adding these two together.

The library also stores information about each rotational body matrices the rotation of each body to the inertial I and base B coordinates. We use this feature in our calculation of the algebraic loop closure.

On the other hand, the library has one significant disadvantage: the computation time. To calculate our Sk8o robot EoM, it takes two hours to calculate them. It is a pretty significant time due to insufficient description of library.

3.2 Modeling

This chapter is about the methods and mathematical statements used to get our full rigid body model [1]. We are focusing on how we can obtain our solution and how we create the model of the robot. Our main task is to find the motion equation of our robot in the following shape.

$$M(q)\ddot{q} + b(q, \dot{q}) + g(q) = \tau \tag{3.2}$$

3.2.1 The coordinates and convention

We defined the generalised coordinates, velocities, accelerations, and actuation torque followingly:

$$r = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \phi = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} q = \begin{bmatrix} r \\ \phi \\ \Phi_{1-8} \end{bmatrix} u = \begin{bmatrix} v \\ \omega_{\alpha\beta\gamma} \\ \omega_{1-8} \end{bmatrix} \dot{u} = \begin{bmatrix} a \\ \dot{\omega_{xyz}} \\ \dot{\omega_{1-8}} \end{bmatrix} \tag{3.3}$$

The variables meaning are shown in the following pictures:

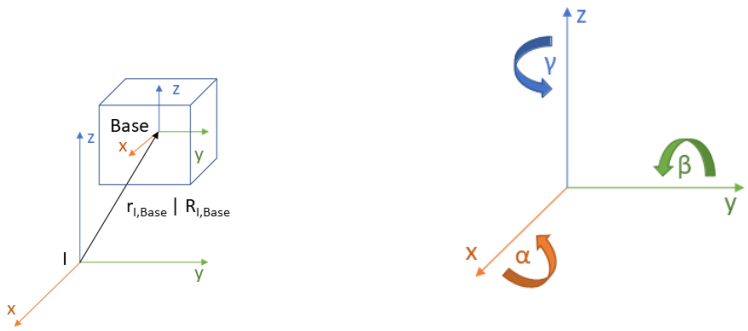

Figure 3.2: The variables description

The vector $\mathbf{q} \in R^{14}$ describes the whole body position of the robot. It consists of the position vector $\mathbf{r}$, the rotational vector ϕ and the vector of robot generalised coordinates Φ_{1-8}.The $\mathbf{r}$ vector describes the relative location of our body in the inertial frame I to base robot frame $\mathbf{B}$. The rotational vector ϕ describes rotation from the inertial coordinates to our robot base coordinates. The vector Φ_{1-8} describes the generalised coordinates of the robot's body, precisely the rotation of each connection. Then we calculate the vector of the speeds $u \in R^{14}$ as the first derivation of the vector q. The second derivation on the same vector gives us the vector of the acceleration $\dot{u} \in R^{14}$. Each subfolder of the vector q also has its new sign. The specific symbol $[\cdot]_x$ means the skew-symmetric cross-product matrix. We will use it with the closure of the kinematic loop in chapter .

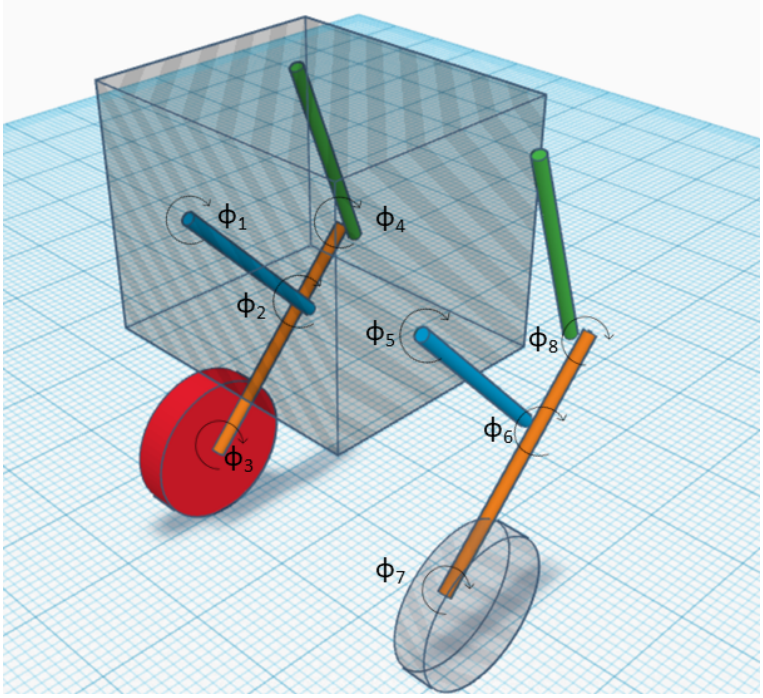

Figure 3.3: The robot's joints and

3.2.2 Kinematic tree

We use for our notation kinematic tree[5]. Its work is to simplify body structure. This allows us to use two principal coordinates, inertial I and body Base. All other coordinates are dependent on the body coordinates and its degrees of freedom. That means if we know the exact location of the robot and all degrees of freedom coordinates. We can compute the exact position and rotation of the body in generalised coordinates. We use the kinematic chain in the description of our robot in the proNEu library. Each part of the model has its number, which corresponds to the number in the kinematic tree.

3.2.3 Simplified 3D model

In the beginning, we must simplify the robot body to the geometrical shape, as is stated in the section with proNEu library. Then, we give each body number from the kinematic chain. We take the body as the root. It has simple reasons. The robot is axe symmetrical through that part because this feature will help us with the algebraic loop closure. Then, we divide the body into two sides, right and left. The sides are chosen from the robotic view. We use sites in body parts marking. The leg is made of the upper leg, lower leg, free leg, and wheel. We can see the numbers in Figure 3.4. The simplified model has no joint between the body and the free leg. This joint creates a kinematic loop, and that is why it is not applied in a simplified model, and we call it the free leg. The other joints are rotational. The joints Φ_2 and Φ_6 between the upper leg and lower leg have the torsion

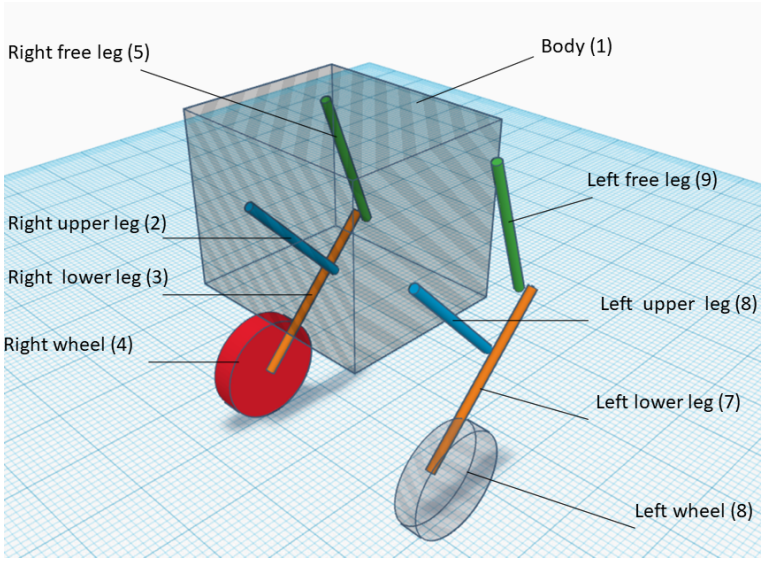

Figure 3.4: The robot's kinematic tree

spring. The joints are described here . We have to make a model with an open kinematic tree.

3.2.4 The open kinematic loop

All of this data helps us to create the open kinematic loop model[6]. The equation will have the following equation shape.

$$M(q)\dot{u} + b(q,u) + g(q) + s(q) = S^T\tau \tag{3.4}$$

We changed a marking in the equation to be closer to our library. The $S \in R^{14\times14}$ describes us which generalised coordinates is τ acting. We also add member $s(q) \in R^{14}$, which express our torsion spring in joints Φ_2 and Φ_6. All other equation members are described previously in the projected Newton-Euler chapter (2.1). But in this case, their dimensions are $M(q) \in R^{14\times14}$, $b(q,u) \in R^{14}$ and $g(q) \in R^{14}$. The motors are in joints Φ_1, Φ_3, Φ_5 and Φ_7. It means that we need to simulate the algebraic loop to be able to move with legs[7]. The model will collapse without this loop. Another option is to change rotational links to one translational joint in each leg. Nevertheless, we stayed with the difficult one. We create these simple dynamics with the `sk80_simulation` and `sk80_dynamic scripts`. The model with the open kinematic loop is on this Figure 3.1. We can see the picture from the simulation at the left side. It is not very clear. This is a reason why we use an external model for better readability.

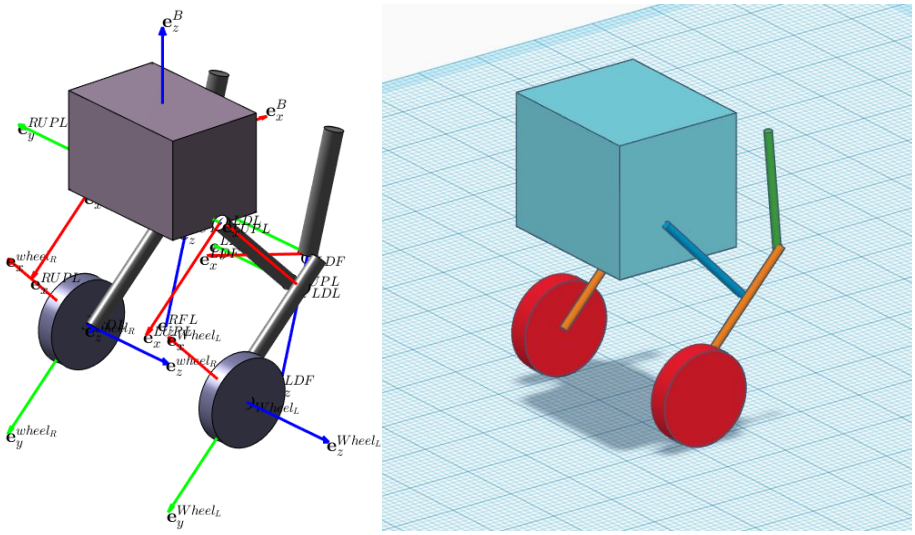

Figure 3.5: The simplified robot model

3.2.5 The loop closure

The key to calculate the closure is our disconnected joint[8]. We apply a loop closure through force limitations in this joint. This loop closure force is F_L[1]. It is reacting on the hinge points on the opened loop. We give hinge points the names A, B on the right site and C, D on the left side as describe in the picture. We add the force directly to our equation.

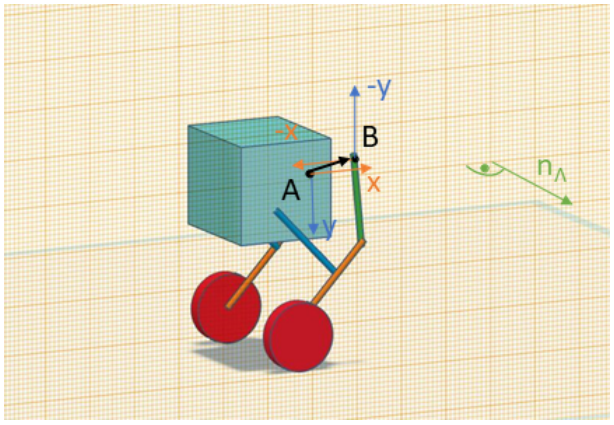

Figure 3.6: Model projection with dimensions in 2D [mm,kg]

$$m\dot{u} + b + g + s + J_{I,A,B}^T F_L - J_{I,B,A} F_L = S^T \tau \tag{3.5}$$

The $J_{I,B,A}$ means the jacobian in I as inertial coordinates. It is evident that loop closure forces are acting in the loop plane Λ with normal direction n_λ.

Because the robot is a floating based system, the orientation of the plane change over time.We can stick to our plane. However, it helps us because we can calculate in 2D

instead of 3D. We can then change the equation following way:

$$m\dot{u} + b + g + s + (J^T_{Base,A,B,Lin} - J_{Base,B,A,Lin})F_{Base,L} = S^T\tau \tag{3.6}$$

$$J_L = J^T_{Base,A,B,Lin} - J_{Base,B,A,Lin} \tag{3.7}$$

We can divide the jacobian J_L to the left and right kinematic loop. We can transform the equation the following way:

$$m\dot{u} + b + g + s + \begin{bmatrix} J_{L_l} & J_{L_r} \end{bmatrix} \begin{bmatrix} F_{L_l} \\ F_{L_r} \end{bmatrix} = S^T\tau \tag{3.8}$$

We must compute a jacobians. This jacobian use the following marking:

$$\begin{bmatrix} v_{I,A,B} \\ \omega_{I,A,B} \end{bmatrix} = \begin{bmatrix} J_{I,A,B,Lin} \\ \omega_{Base,A,B,Rot} \end{bmatrix} u \tag{3.9}$$

The first index describes the acting coordinate system. The second and third describe the vector position. We have to determine the unknown loop closure forces. We know that the position of $r_{A,B} = 0$ at the acceleration level. We can perform two derivatives by the time on our $r_{A,B}$ to get our forces. After the second differentiation step, we check our method. We are acting in 2D, but we are using a 3D vector. The y coordinates equal 0 as it has according to the 2D theory. Than, we can simplify our solution in terms of the jacobians due to u and $\dot{u}$. We will obtain jacobian by deriving our force by the generalised coordinates. All things described in this article are applied in the script `loop_closure`. The closure must then fulfil existential constraints:

$$\tilde{X}u + \tilde{Y}\dot{u} = 0 \tag{3.10}$$

We use $\tilde{\text{X}}$ and $\tilde{\text{Y}}$ are used for the left and right sides. The constraints for the whole robot are:

$$\begin{bmatrix} \tilde{X}_L \\ \tilde{X}_R \end{bmatrix} u + \begin{bmatrix} \tilde{Y}_L \\ \tilde{Y}_R \end{bmatrix} \dot{u} = 0, \begin{bmatrix} \tilde{X}_L \\ \tilde{X}_R \end{bmatrix} = X, \begin{bmatrix} \tilde{Y}_L \\ \tilde{Y}_R \end{bmatrix} = Y \rightarrow Xu + Y\dot{u} = 0 \tag{3.11}$$

We can not forget to show how each constraint look like:

$$\begin{aligned} \tilde{X} = R_{Base,I}(&-[\omega_{I,I,Base}]_\times[r_{I,A,Base}]_\times J_{I,I,Base,Rot} + [r_{I,A,Base}]_\times \cdot J_{I,I,Base,Rot} \\ &-2[\omega_{I,I,Base}]_\times J_{I,A,B,Lin} + \cdot J_{I,A,B,Lin} \end{aligned} \tag{3.12}$$

$$\tilde{Y} = R_{Base,i}([r_{I,A,B}]_\times)J_{I,I,Base,Rot} + J_{I,A,B,lin}) \tag{3.13}$$

The definition of the kinematic loop is complete after calculation and implementation. We perform the calculation by first calculating $\dot{u}$ and then substituting it into the resulting conditions. The F is an unknown constraint force. The J is our calculated jacobian.

$$m\dot{u} + b + g + s + JF = S^T\tau \tag{3.14}$$

$$Xu + Y\dot{u} = 0 \tag{3.15}$$

The force from the previous two equations we calculate folowingly:

$$F = (YM^-1J^T)(Xu + YM^-1(S^T\tau - n - g - s)) \tag{3.16}$$

This gives us all information we are looking for in this thesis. This loop closure is simulated in the script `closed_loop.m`.

Chapter 4

Conclusion

Main goal of this thesis was to find the complete mathematical model of the sk8o robot. In particular, that means finding a suitable modelling tool.
We used a different modelling approach to modelling. We also described the mathematical and physical methods behind the library we used and behind our methods.

We found a tool that helped us to solve our complex equations in chapter 2. We examined several libraries, and the best was the Matlab proNeu library. We described it in chapter 3. Thanks to this library we calculated the EoM for the simplified model. We introduced the force limitations. These limitations helped us to create the model with the algebraic loop. We were able to find the EoM of the full model. Simulation not done due to Covid.

The future work can focus to ground contact. The ground contact is far more complex than the closure of the kinematic loop. However, it is based on the same principles as the loop. It brings new challenges notably to how to solve the wheel's rotation and now to deal with the additional dimension.

References

[1] Klemm, Victor, et al. LQR-Assisted Whole-Body Control of a Wheeled Bipedal Robot with Kinematic Loops. IEEE Robotics and Automation Letters, 2020, 5.2: 3745-3752.

[2] Kollarčík, Adam. Modeling and Control of Two-Legged Wheeled Robot. Prague, 2021. Master's thesis. CTU. Supervisor Ing. Martin Gurtner.

[3] "Robot Dynamics Lecture Notes." Accessed: Jan. 03, 2022. [Online]. Available: https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf.

[4] Marco, Hutter and Christian, Gehring, "proNEu Documentation",2012 , "Unpublished" https://bitbucket.org/leggedrobotics/proneu/src/master/documentation/manual/proNEu_documentation.pdf (accessed Jan. 03, 2022).

[5] Featherstone, Roy. Rigid Body Dynamics Algorithms. Springer, 2014.

[6] Kim, Donghyun, et al. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control. Preprint, 2019.

[7] Kim, Donghyun, et al. Computationally-Robust and Efficient Prioritized Whole-Body Controller with Contact Constraints. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 1-8.

[8] M. Iwamura and M. Nagao, "A method for computing the Hessian tensor of loop closing conditions in multibody systems," Multibody System Dynamics, vol. 30, no. 2, pp. 173–184, Dec. 2012, doi: 10.1007/s11044-012-9334-7.