Master of Science (MSc)

Concise Modeling of Humanoid Dynamics

Florian Joachimbauer

Embedded and Communication Systems, 30 credit

Halmstad University, June 26, 2017

Florian Joachimbauer: *Concise Modeling of Humanoid Dynamics,*
©June 26, 2017

SUPERVISORS:
Walid Taha

EXAMINERS:
Alexey Vinel
Mohammedreza Mousavi

LOCATION:
Halmstad, Sweden

ABSTRACT

Simulation of mechanical systems like walking robots, is an essential part in developing new and more applicable solutions in robotics. The increasing complexity of methods and technologies is a key challenge for common languages. That problem creates a need for flexible and scalable languages. The thesis concludes that an equation-based tool using the Euler-Lagrange can simplify the process cycle of modeling and simulation. It can minimize the development effort, if the tool supports derivatives. Regretfully, it is not common to use equation-based tools with this ability for simulation of humanoid robots.
The research in this thesis illustrates the comparison of equation-based tools to common used tools. The implementation uses the Euler-Lagrange method to model and simulate nonlinear mechanical systems. The focus of this work is the comparison of different tools, respectively the development of a humanoid robot in a stepwise manner based on the principle of passive walking. Additionally, each developed model has given an informal argument to its stability. To prove the correctness of the thesis statement the equation-based tool called Acumen is evaluated in contrast to a common used tool, MATLAB.
Based on the achieved results, it can be concluded that the use of equation-based tools using Euler-Lagrange formalism is convenient and scalable for humanoid robots. Additionally, the development process is significantly simplified by the advantages of such tools. Due to the experimental nature of Acumen further research could investigate the possibilities for different mechanical systems as well as other techniques.

The future depends on what we do in the present.

— Mahatma Gandhi

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Prof. Walid Mohamed Taha, PhD from the Halmstad University. The door to Prof. Taha's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I must express my very profound gratitude to my parents and siblings for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

I would also like to acknowledge Veronika Haaf as the second reader of this thesis, and I am gratefully indebted to her for her very valuable comments on this thesis.

Finally, I want to thank my friends for awesome years during my studies in Halmstad, respectively in Salzburg.

Thank you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

DOF	Degree of Freedom
2D	Two Dimensional
3D	Three Dimensional
Q	Conservation of Momentum Matrices
q	Generalized Coordinates
EL	Euler-Lagrange
M	Inertia Matrix
C	Coriolis and Centrifugal Matrix
G	Gravity Matrix
K	Kinetic Energy
V	Potential Energy
B	Input Torque Matrix
J	Reset Map
T	Transformation Matrix
H	Impact Map
S	Event Definition
SSP	Single-Support Phase
DSP	Double-Support Phase
ZMP	Zero Moment Point
HZD	Hybrid Zero Dynamics
PWD	Passive Dynamic Walking
DAE	Differential Algebraic Equations
RGB	Color model

1 INTRODUCTION

The first control theory developments led to various technological solutions, such as feedback amplifier and state-space applications. Robustness and uncertainty were covered by multiple practical experiments where frequency response models were used to model such controllers. Further research and mathematical advantages of state-space applications resulted in new and more applicable concepts with different benefits in controllability and optimization. Practical methods like adaptive and digital controllers were developed by additional research. However, dynamical systems with nonlinear components can not be handled by simple controller designs. It is well established that such systems can result in highly complicated and unstable dynamics [26]. Therefore, more applicable control theories like the Euler-Lagrange formalism were developed to capture this nonlinear behaviour. The Euler-Lagrange formalism is predestined for walking robots due to its capability to handle highly unstable dynamics. Studies on biped walking robots have been performed since 1970 [23]. Since then, biped robots and humanoid robots have been advanced through the technological development [17, 37, 16]. Humanoid robots are widely used in various areas, such as military usage, medical behavior and the entertainment world. The increasing complexity of components and the use of more advanced technologies challenge the need for flexible and scalable languages, respectively development tools. The ability to simulate and visualize the behaviour opens a wide range of possibilities. A simulation, simple 2 dimensional as well as more advanced 3 dimensional, gives a more accurate illustration of performance and correctness than numbers or plots. For a better understanding of the concepts of these humanoid robots, different tools were designed to facilitate and simplify the process of development. Furthermore, the research on humanoid robots has become one of the most representative topics in the area of intelligent robotics [39].

1.1 PROBLEM DEFINITION

In the field of robotics there are various methods and tools to express the motion of mechanical system, such as humanoid robots. In terms of methods a naive approach is to apply the Euler-Newton formalism, which applies the forces and torques of a rigid body. In case the system needs to be adapted and extended due to changes in the modeling process this method reveals flaws with respect to scalability and convenience. A more convenient method to express the model of a mechanical system is the Euler-Lagrange formalism, which uses the potential and kinetic energy of the rigid body. Therefore, the thesis of this dissertation is

> A equation-based tool that supports Euler-Lagrange modeling can simplify the design of humanoid robots.

In this context the decision of the required tools is a fundamental one for a successful development. The modeling process of humanoid robots includes various steps whereby the calculation of the mathematical formalism, the modeling process and the simulation are commonly done in different tools like MATLAB. The semantics of tools used in this area are unlike the natural mathematical formalism and the usage of multiple tools leads to additional overhead in development. An equation-based modeling tool like Acumen can combine two or more phases into one tool. The complete potential of simplifications of such semantics is strongly linked to its similarity to the mathematical formalism. Therefore, the research question of this work can be stated as

> Can the Euler-Lagrange formalism be used in an equation-based tool to reduce the modelling effort of humanoid robots ?

In particular, this thesis will focus on the simplification potential of the modelling and simulation process for bipedal and humanoid robots using an equation-based tool.

1.2 CONTRIBUTION

As mentioned before bipedal and humanoid robots have been a representative research topic for many years. The common cycle of such developments includes various phases. This phase-based development comes with a high overhead. One has to know to use each tool and has to compensate for non-supported but needed features. Tools, which are used in the modeling domain usually focus on modeling a specific behaviour. Thus, the process requires the use of different tools in order to create a model with an appropriate simulation that covers the mathematical idea. In the specific case of Euler-Lagrange formalism the model is described by numerous continuous and discrete equations. On the other hand the Lagrangian involves the calculation of complicated derivatives. This results in a high overhead work whereas equation-based tools are different to the common tools. This type of tools can be used to express the mathematical formalism in a convenient and natural form. This work is based on a tool called Acumen [35, 1]. Using the tool the process of modeling and simulating can be combined. Thus, the previously described overhead introduced by using different tools can be avoided.

The main contributions of this work are:

1. Showing the scalability of a complex model with Lagrangian in a step-by-step implementation, focusing on stability.

2. Identifying a number of important aspects in order to develop a convenient model of a humanoid robot.

3. Outlining the advantages of a equation-based tool in contrast to other tools by means of a concrete example.

1.3 OUTLINE

This thesis is organized as follows. Chapter 2 gives a short overview about the state-of-the-art technology. This chapter also introduce the exact declaration of this project-relevant robots and used tools. After the introduction and description of various tools, the most important terms in the syntax of Acumen are explained.
Since the definitions and understanding of bipedal and humanoid robots are mixed in the literature, Chapter 3 discusses the mathematical fundamentals, notations and concepts, which are relevant to solve the tasks. Based on the fundamentals presented in this chapter, the general concept of walking robots is explained in Chapter 4. This concept starts with a two Degree of Freedom (DOF) passive bipedal robot, which movements are restricted to the sagittal plane. Considering these results, this bipedal is extended into a more complex bipedal robot with knees (also restricted to the sagittal plane). In a third step the upper body is added as another link, which transforms the bipedal into a humanoid robot. The transformation of the previous concept into a Three Dimensional (3D) approach completes the humanoid robot. The stability of the implemented models is covered by limit cycles and Poincare map. Chapter 5 presents the most important parts and steps in the integration of the tool Acumen. Similar to the previous chapter the explanation begins with a simple model followed by the humanoid robot.
In Chapter 6 aspects of expressivity and scalability are discussed. Furthermore, this chapter includes the direct comparison of the languages MATLAB and Acumen in terms of used techniques and especially the Euler-Lagrange equations. In the final Chapter 7, a complete conclusion and note directions for future work are given.

2 RELATED WORK

This chapter contains common methods and techniques for modeling humanoid robots are discussed. Furthermore, various tools are described that are widely used in evaluation of cyber-physical systems.

2.1 MOTIVATION

Legged robots have occupied many researchers for decades. Wheeled locomotion is one of the most efficient and energy-saving type of movement. This kind of movement is in contrast to legged locomotion limited to paved surfaces. Legged ones on the other hand have the possibility to navigate through difficult and uneven surfaces. The study of passive locomotion is first introduced by McGeer[24]. The term passive walking describes the behaviour of a robot which has no active control or energy input. This leads to a extremely high energy efficiency with a lack in flexibility on different surfaces. McGeer explained [24], that a rolling motion through a slope can be transformed into a walking motion by removing the rim of the wheel and keep only 2 spokes as legs [19]. This sort of robot was the first and the simplest kind of passive walking. Additionally, researchers from INRIA [14, 15, 11] designed unpowered biped robots based on passive walking and analyzed their gravity-induced passive motion through a slope. Since that time a various number of different passive walking robots are introduced and designed. In addition the principle of passive walking with knees was introduced[23] and the three dimensional approach of passive walking have been developed by Collins [7]. Researchers[13, 18, 2] reported results on the stability of passive control models based on limit cycles [26].
Another extreme in control of walking robots is the full-actuated bipedal robot. This means, each joint of the robot is controlled by a controlling technique. Honda's ASIMO robot [17] is an example for a fully controlled bipedal robot and the possibilities in motion with it. This robot can monitor the Zero Moment Point (ZMP) and plan the trajectories of his legs in advance. He is capable to move upwards, downwards and even running is possible. The downside of this controlling approach is the massive energy consumption. Therefore, the robot has to carry a heavy battery or pull a heavy cable. It is estimated, that the ASIMO robot needs 32 times more energy than a human would need with the same weight and size [6].
Another approach for controlling a walking robot, is the hybrid zero dynamics from Westervelt [37]. This method approximates a set of

functions for each joint. The functions, which are applied to the joints, are forming closed orbits to the walking behaviour and define the stability of walking. Therefore all joints need an actuation that increases the energy consumption. It is important to mention that this method is mostly used for two dimensional walking robots. Ames and Greg [16] developed a 3D version with Hybrid Zero Dynamics (HZD).
In contrast with the previous methods M. Spong [34] published a principle of a simple energy-shaping method. Generally, this method makes it possible for a higher DOF passive-dynamic walking robot to walk on any slope. A complete energy-shaping control law were introduced by Bhatia [33]. Also the energy consumption based control introduced by Asada [38] is a framework between the passive walking and the full-actuated approach. This method tries to minimize the control input by actuating only few joints. Passive walking is energy efficient, but it is really sensitive to parameters and has a limitation in applicability. On the other side, the full-actuated controller with a powerful actuation has a wide range of possibilities but is inefficient as described above (HZD, ZMP)[33]. The direct comparison of control techniques for different robots needs a practicable evaluation strategy like simulations. Evaluating and comparing such approaches on different kind of models benefits highly from having an effective technology for developing simulation of different kind of robots.
This thesis introduces a step-by-step implementation of a 6 joint 3 dimensional walking robot model which is focused on stability and energy efficiency. The following section covers an overview of various tools in this research area. Those tools show the applicability for modeling and simulation of humanoid robots.

2.2 SUPPORTING TOOLS

There are many tools available for designing mechanical systems in academic and commercial usage, ranging from simulation tools to verification tools. This section will cover most common tools and there modeling formalism.
A hybrid system is a system which contains continuous and discrete behaviour. Choosing the right tool for modeling such systems is a critical decision. A model of a hybrid system consists of discrete and continuous variables where the continuous variables can be described as differential equations. The more types of equations the modeling tool supports the less manual work needs to be done by the developer. Therefore, Table 1 shows common tools in terms of supported features. The evaluated properties used in Table 1, are:

- Partial derivatives: This feature is defined by the ability to support the partial derivative of more than one expression directly.

- Compact derivatives: This feature calculates the derivative in a compact form without producing large expressions
- Support for equations: This feature is defined by the possibility to use equations as well as assignments in the specific tool.
- Support for hybrid behaviour: The ability to use discrete events in combination with continuous behaviour.

	Matlab/ Simulink	SpaceEx	Open Modellica	Mathematica	Acumen
Partial Derivatives	Yes	Yes	Limited	Yes	Yes
Compact Derivatives	No	No	No	No	Yes
Support for Equations	No	No	Yes	Yes	Yes
Support for hybrid behaviour	Yes	Yes	Yes	Yes	Yes

Table 1: Important tools in the modeling domain [40]

Simulink/MATLAB : One of the most widely used industrial formalism is MATLAB with the block diagram and Stateflow extension called Simulink. After 20 years of releases, Simulink/MATLAB gained a huge user base and many libraries are developed. This formalism is a graphical language and uses nodes and edges for describing the behaviour of the system. The nodes (blocks) can transform the input signals (edges) based on the block function and pass the result to other blocks. This graphical extension of MATLAB is convenient for electrical solutions because of its similarities to the modeled system. With increased complexity the formalism becomes complicated and difficult to read. MATLAB and also Simulink are not designed for catching mathematical expressions directly [22, 3].

OpenModelica : Another widely used tool for hybrid systems is Modellica. The language includes textual, programmatic and graphical notations. Due to the fact that Modellica supports Differential Algebraic Equations (DAE) solvers, equations can be used. Because of this DAE solvers, equations left-hand side is not restricted to a single variable. Equations like $m * c^2 = E$ are supported and will be transformed in the needed form automatically. Nevertheless, partial derivatives are not supported in a direct and convenient form. This deficit in functionality does not allow using Euler-Lagrange formalism in this tool at all [25].

Mathematica : Tools, like Mathematica are based on symbolic algebra. Symbolic algebra can be used for analytical behavior as well as simple physical systems. The downside of symbolic algebra is the exponentially growing computing time which comes with the increased size of the modeled system. The computed symbolic result can be exponentially bigger in such systems than in other solvers. In view of applying Euler-Lagrange tools that supports partial derivatives can outperform Mathematica in computation time and result size as well [28].

SpaceEx : SpaceEx is a tool for modeling and verifying hybrid systems. The graphical language is based on the commonly used principle of hybrid automata and will be written in a XML based format. This hybrid automata defines the change of continuous variables with respect to time by using a set of differential equations. It also consists of different locations (modes), which can switch instantaneously. Each mode in SpaceEx has its own set of equations. [10].

Acumen : The equation-based language Acumen is small language with expressive notations. This notations and statements are defined to simplify the modeling for large systems. This includes the supported behavior of derivatives and equations. Furthermore, systems with a dynamically changing number of variables can be modeled in Acumen. The semantics of this tool will be described in more detail [40, 35, 39]. A common approach to model the continuous change over time is to use differential equations. The notation for differential equation in Acumen will be shown by means of a simple clock. The $'$-operator can be defined as the derivative with respect to the time [9]. A concrete example in Acumen is

$$x' = 1$$

This simple system is modeled in Acumen with a variable whose derivative is one. In case of an instantaneous event Acumen uses discrete equations in form of

$$x+ = x - 1$$

This line of code changes the next value of x which is equal to the current value of x minus one. The same behaviour can be achieved in C with $x = x - 1$. Sequences of equations is a common way to express a system with dimension higher than one. Acumen uses for this methodology the composition vector " , ". A simple sequence is shown as

$$y' = x, x' = -u$$

This example is expressed internally as a higher-order differential equation and is equal to $p'' = -u$. A higher-order differential equation can also be written directly with multiple $'$ - operators. To use the code examples shown before a model must be defined where the code can be inserted. The syntax of the model always consists of the same fragments with fixed names [9]. The following model explains its definition

```
model P(simulator)=
initially
    x = 0, x' = 0
always
    x' = 1
```

The fundamental components which every model need, are name of the model (P above), zero or more input parameters (one above) and the two sections `initially` and `always`. Each variable used in the model must be defined with an initial value in the `initially`-section. The `always`-section on the other hand contains the behaviour of the model. An executable model needs a simulator parameter input for the execution. The interpreter needs the simulator parameter to build the model. The model shown above is such a model [9]. The tool MATLAB is only used for comparison and will not be discussed further. Respectively, more details can be found in [22].

3 BACKGROUND

This chapter gives a short introduction to some important basics of classical mechanics, differential calculus and control theory. The mathematical basics, introduced in this chapter, build a framework for modeling walking of a general n-Degree of Freedom (DOF) bipedal robot. The walking dynamics of a bipedal robot are composed by a Single-Support Phase (SSP) and a Double-Support Phase (DSP). In the single-support phase, one leg has always contact with the surface and the second leg is free to swing. The double-support phase is defined as both legs contacting the ground and transferring the impact from one to the other leg. This behaviour leads to a hybrid dynamic model where the single-support phase is implemented as a differential equation and the double-support phase is modeled as a instantaneous reset map of angles and velocities [11]. This chapter will start with an introduction in humanoid mechanics and its limitations, followed by the techniques of modeling the dynamics. In order to provide an informal argument for stability of the robots (which will be developed in this work), the method for stability will be introduced.

3.1 DEFINITIONS

In this work various terms are used, like bipedal or sagittal plane. In this section exact definitions are presented, explaining the terms and providing an overview about the differences.

3.1.1 *Bipedal and Humanoid robot*

The terms bipedal and humanoid robot are widely used in the robotics literature. Furthermore, the terms have in different papers different definitions and meanings. In this work these terms will refer to the definitions below.

The term bipedal robot describes, a robot which walks on exact two legs (without any statement about the rest of the robot). This means that each robot with two legs is a bipedal robot. For this thesis, a robot having two legs and the main mass on the hip is specified as bipedal robot.
A humanoid robot on the other hand is defined as a robot with some human features. There is no specification of the shape of the robot. For example a robot with only an upper body and no legs is also a humanoid robot. In this thesis, a humanoid robot is defined as a robot

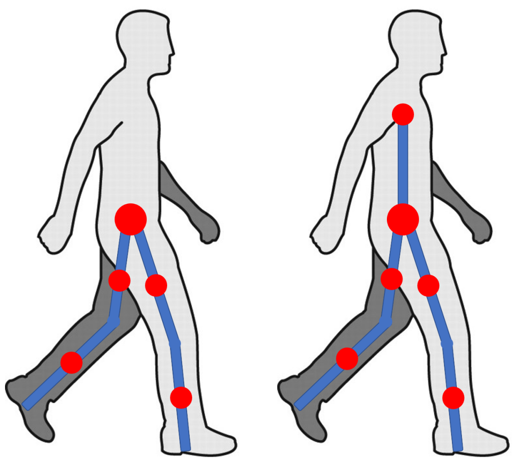


Figure 1: Bipedal robot vs. Humanoid robot

with 2 legs, a hip mass and a link, which represents the motions and the mass of the upper body (see Figure 1)[36].

3.1.2 *Sagittal and Coronal Plane*

The terms of sagittal and frontal plane describe the limitation to the specified motions. Figure 2 shows the planes of the example of a human. A robot, which is only capable of moving in the sagittal plane, is two dimensional (see Chapter 4) and can only move straight forward or backwards. There is no movement in the coronal plane. A 3 dimensional walking robot is not limitated in the coronal plane, so it can also move sideways (see Chapter 4).

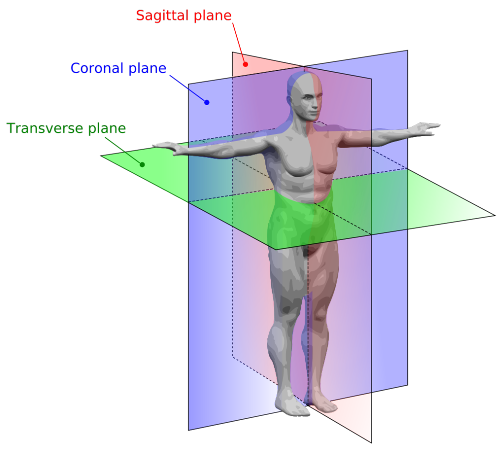


Figure 2: Anatomy Planes of a human [29]

3.2 HUMANOID MECHANICS

Due to the complexity of the real humanoid mechanics, the high number of Degree of Freedom (DOF) and the limited time of this work, some fundamental assumptions are needed to underline the thesis. The following assumptions are used and named as followed:

- The robot has point feet without any sliding and slipping.
- One support leg is always rigidly connected to the ground surface, this leg switches after each step.
- The robot is constructed with point masses with constraints relative to each other (described in Section 3.3.2)
- The impacts, which influence the robot through a collision, are modeled as impulses. This impulses happen in infinitesimally small time period and results in a discontinuity in the velocities.
- The motion of the robot is defined in each model and is separated in sagittal and coronal plane.

Based on the assumptions, we can introduce the techniques and methods for determining the dynamics as a hybrid system. This hybrid system is described in more detail in the following sections and a stepwise implementation is shown in Chapter 4[16].

3.3 EULER-LAGRANGE FORMALISM

The Euler-Lagrange formalism also known as the Lagrangian formalism, is a differential way to describe the evolution of a physical system with respect to time. This method describes the relationship between the accelerations and the motion of their structure. The Lagrangian equation includes all physical information about the system and the forces acting on it. In order to use this method for a mechanical system, the generalized coordinates must fulfill a number of requirements, that are explained in the following section [20].

3.3.1 *Constraints, Independence, and Completeness*

Consider a system consists of n independent masses, which are able to move freely and unconstrained from each other. The motion of each mass i can be described with the Newton's third law. If the particles are connected to each other as a kinematic chain, the particles are restricted in movement relative to each other. These constraints can be implemented as a mass-less rod between two connected masses.
Coordinates are used to describe the position and orientation of the masses, . Let $q(t) = [q_1, ..., q_n]^T$ be a n dimensional column vector of

coordinates. A coordinate q_n of the vector $q(t)$ is independent from the other ones, if it has a continuous range of movement, while all others are fixed.
The completeness of a system on the other hand is fulfilled , only if the locations of the masses can be described at all parts and all times with the given vector $q(t)$. This aspects of completeness, independency and constraints motivates the use of generalized coordinates for modeling robots [30].

3.3.2 *Generalized coordinates*

If there exists a n-dimensional vector of the form $q(t)$ based on the requirements presented in Section 3.3.1, the vector $q(t)$ is called generalized coordinates of the system.
The restriction of motion through the holonomic constraints reduces the DOF relative to the unrestricted system. To maintain the correctness of this reduction, the dynamics of Newton's law must be modified to introduce constraint forces. Without this constraint forces, the rod between two masses could get compressed or stretched freely by the motion of the masses. Therefore, the constraint forces are added. This modification increases the complexity of the dynamics as well as the representation of r and their coordinates. If the positions of the masses are represented as generalized coordinates, the additional complexity can be avoided by applying the Principle of Virtual Work and d'Alembert's Principle. The use of n-generalized coordinates for a mechanical system is the first step to describe the dynamics of the system [8].

3.3.3 *Lagrangian formalism*

The Lagrangian formalism uses the potential and kinetic energy to derive the evolution of motion for a kinematic chain. A kinematic chain is a set of links, where each link is defined as a particle with a length l and a mass m located at the center of the link. The orientation of all links is measured though the generalized coordinates. In the specific case of humanoid robots, each link is a revolute joint and has only one DOF. There are different possibilities to define the generalized coordinates. One possibility is to use the absolute angles of the links relative to the original frame. Another way is to use the kinematic approach to describe the geometry and use the relative angles between the links for the generalized coordinates. Given a set of generalized coordinates, the Lagrangian equation can be derived by the kinetic and potential energy. The Kinetic energy $K \in \mathbb{R}$ is expressed by the standard quadratic form of the velocities $\dot{q}$ as

$$K = \frac{1}{2} m_i \dot{q}_i^2 \tag{1}$$

where m_i defines the masses and v_i^2 describes the quadratic velocitiy of link i. Let $q \in \mathbb{R}^n$ and $\dot{q} \in \mathbb{R}^n$ be the vectors of the generalized coordinates and their velocities. The potential energy $V \in \mathbb{R}$ can be assumed similar in the form of

$$V = g m_i h_i \tag{2}$$

where h_i determine the position of the mass in vertical direction. After defining the potential (Equation 2) and kinetic energy (Equation 1), the Euler-Lagrange $L \in \mathbb{R}$ of a mechanical system can be expressed as followed

$$L(q, \dot{q}) = K(q, \dot{q}) - V(q) \tag{3}$$

This formalism can be used to describe the time evolution of a mechanical system by substituting Equation 3 into the differential equation

$$\frac{d}{dt} \nabla_{\dot{q}}^T L(q, \dot{q}) - \nabla_q^T L(q, \dot{q}) = B(q) u \tag{4}$$

$u \in \mathbb{R}$ is a vector of control inputs. If y is a vector, let ∇_y be the row vector of partial derivatives with respect to the vector y. Let $B(q) \in \mathbb{R}^{n*m}$ be the input map for the control vector u. This map assigns the input torques to the different joints. In case of passive robots, the input vector u is zero in all components. Based on this substitution of Equation 3 into Equation 4, the equation can be written as a compact matrix, which is commonly known in the robotic literature as the robot equation of motion

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B(q) u \tag{5}$$

with $M(q)$ as the positive finite inertia matrix, $G(q) \in \mathbb{R}^n$ as the gravity vector and $C(q, \dot{q}) \in \mathbb{R}^{n*n}$ is the matrix for centrifugal and Coriolis parts [31]. The gravity vector G is determined through the previous definition in Equation 4 and Equation 5 as $G = \nabla_q^T V$ [26].

3.4 ROTATION VECTOR

This section covers the possibilities of rotation in a kinematic chain of a bipedal or humanoid robot. The common way to describe the rotation of a joint in a kinematic chain is to use rotation matrices. The size of this matrices depends on the number of rotations each joint can perform [21] . This work covers two different sizes of matrices/vectors to describe the rotations. For the Two Dimensional (2D) approach of humanoid robots a rotation vector $R \in \mathbb{R}^{2x1}$ suffices as shown as followed

$$R = \begin{pmatrix} sin(\Theta) \\ cos(\Theta) \end{pmatrix} \tag{6}$$

The rotation vector shown above is based on the assumption that the angles are measured with respect to the global base frame. The positive rotation direction is clockwise. Such an angle is displayed for a simple inverted pendulum and a double inverted pendulum in Figure 3. In order to achieve a complete description in a 3 dimensional

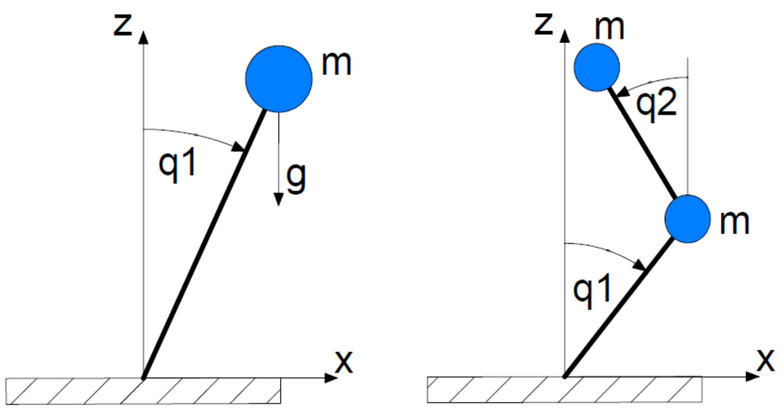


Figure 3: Example of an inverted pendulum

rotation different aspects, like the order of the rotation, must be considered. The common way to address this problem, with respect to position and orientation of different joints, is the forward kinematics. This approach determines the position and orientation of the tool or end-effector, given the base frame and the single values of the joints [8]. Therefore, a matrix $T \in \mathbb{R}^{4x4}$ is used, which is structured as follows

$$T^i_j = \begin{pmatrix} R^i_j & p^i_j \\ 0^{1x3} & 1 \end{pmatrix} \tag{7}$$

The $R^i_j \in \mathbb{R}^{3x3}$ determines the rotation matrix from frame i to frame j and $p^i_j \in \mathbb{R}^3$ determines the position of the frame i relative to frame j. The one in the transformation matrix T in Equation 7 defines the scale of the joint. For robotic behavior, such as humanoid robots, this value is one by default. The rest of the matrix is filled with zeros [31]. Based on the definitions in Section 3.2, the matrix in Equation 7 can be simplified into a vector $R \in \mathbb{R}^{3x1}$ of the form of

$$R = \begin{pmatrix} sin(\Theta) * cos(\Phi) \\ sin(\Phi) \\ cos(\Theta) * cos(\Phi) \end{pmatrix} \tag{8}$$

The angle Θ describes the rotation along the x-axes and Φ along the y-axes. For this work are the Roll, Pitch, Yaw-Angles with x-y-z order used. Because of the restrictions of the human motions, a rotation along the z-axes is not possible. After the definition of rotation of each link, the conservation law of angular momentum can be defined [31, 32].

3.5 CONSERVATION OF ANGULAR MOMENTUM

The conservation law of angular momentum is one of several methods to determine the discrete velocity change for an impact, such as the impact of the swing leg with the ground. Although there are various method like the change in mechanical energy for the impact, described in [13]. For this work is only the conservation of angular momentum method covered.
The angular momentum itself, is defined as the cross product of the linear momentum $\vec{p} = m\vec{v}$ and the displacement $\vec{r}$ from the pivot.

$$\vec{L} = \vec{r} \times m\vec{v} \tag{9}$$

If no forces are acting at the moment of the impact, the angular momentum is conserved before and after the impact [27]. Based on the definition, that the humanoid robot consists of several point masses with fixed restrictions towards each other, the angular momentum equation (Equation 9) can be written as

$$L^O = \sum_i r_i^{O+} \times m_i v_i^+ = \sum_i r_i^{O-} \times m_i v_i^- \tag{10}$$

where + and - subscript the times right before and after the impact. The sum variable i is equal to the number of masses in the system and the subscript O defines the point of impact. With regard to the correctness of the calculations, this law works with generalized coordinates, as well as the Lagrangian formalism. Therefore, the statement of the pre-impact and post-impact velocities as mentioned in Equation 10 can be rewritten as a function in the following way

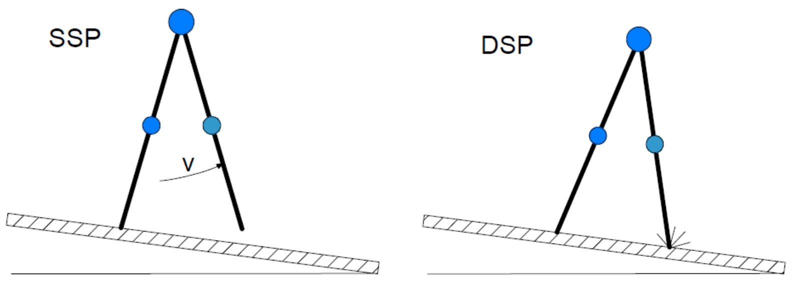
$$L^O = Q_+ * \begin{bmatrix} \dot{q}_1^+ \\ \vdots \\ \dot{q}_n^+ \end{bmatrix} = Q_- * \begin{bmatrix} \dot{q}_1^- \\ \vdots \\ \dot{q}_n^- \end{bmatrix} \tag{11}$$

where the $Q_{+/-} \in \mathbb{R}^{n \times m}$ are the matrices for a specific impact. The matrix sizes n and m depend on the scale of the system and the configuration of the system. This system can be solved for $\dot{q}^+$ by multiplying the right-hand side with the inverse matrix Q_+^{-1} as shown as in Equation 12 [31, 27].

$$\dot{q}^+ = [Q_- * \dot{q}^-] * Q_+^{-1} \tag{12}$$

3.6 HYBRID DYNAMICS

The Euler-Lagrange equation (Equation 4) describes the unrestricted continuous dynamic motion of a mechanical system. However, a mechanical system such as a bipedal or humanoid robot can not be modeled by Lagrangian only. The Lagrangian does not consider discrete events, which emerge through interaction with the environment. The moment of interaction between the leg of the robot and the ground surface is one of this impacts. During this phase, the system behaviour changes, such that the support leg and swing leg are switched. Furthermore, the velocities of the generalized coordinates, which are essential for both dynamics, will experience a transformation at this moment of impact. Therefore, a combined mathematical description, a hybrid system, is needed. Based on the humanoid mechanics discussed in Section 3.2, this hybrid system will be described by the following two phases



(a) Single-Support phase of walking (b) Double-Support phase of walking

Figure 4: Walking phases for humanoid robots

3.6.1 *Dynamics of single-support phase*

The single-support phase shown in Figure 4, can be expressed by the differential equation of motion. This phase is based on the Lagrangian (Equation 13) and is therefore bordered by the impacts with the environment. Due to the fact, that the Lagrangian can not handle discrete events, the system has to switch to the double-support phase.

$$\frac{d}{dt}\nabla_{\dot{q}}^{T}L(q,\dot{q}) - \nabla_{q}^{T}L(q,\dot{q}) = B(q)u + \Gamma(q,\dot{q}) \tag{13}$$

3.6.2 *Dynamics of double-support phase*

The double-support phase is a instantaneous dynamic event, which transfers the bipedal robot from one leg to the other leg. This means, that both legs switches their roles. The roles of the legs are defined in Section 3.2. Furthermore, the kinetic energy emerges a discontinuity, while the potential energy stays constant. To calculate the exact moment of impact a summary of conditions, a guard, is needed [34]. Considering the following assumptions, the DSP can be calculated using the conservation law of angular momentum

1. the impact is perfectly plastic (no bounce)
2. there is no slip along the ground surface
3. the impact from one to the other leg is instantaneously transferred
4. the impact is defined as a guard S_g

Firstly, the guard for the ground impact must be defined. Let $h(q)$ be the step height, which depends on the generalized coordinates. Due to this definition, the guard of the impact can be assumed as

$$S_g = \{(q, \dot{q}) | h(q) <= 0, \dot{h}(q) <= 0, \ddot{h}(q) <= 0\} \tag{14}$$

3.6.3 *Summary of dynamics*

After the definition of SSP and DSP, the dynamics of such a system can be summarized into a hybrid system of the following form

$$\begin{aligned} &\frac{d}{dt}\nabla_{\dot{q}}^T L(q,\dot{q}) - \nabla_q^T L(q,\dot{q}) = 0 && x(t) \notin S_g \\ &q^+(T) = J(q)q^-(T) && x(t) \in S_g \\ &\dot{q}^+(T) = H_g \dot{q}^-(T) && x(t) \in S_g \end{aligned}$$

where the guard S_g defines the moment of the impact in which the angles and velocities will experience a instantaneous transformation. The matrix $J(q)$ is defined as the reset map for the angles and H_g as the impact map for the velocities. For each moment, where S_g is false, the Lagrangian formalism describes the dynamics [5].

3.7 PERIODIC CYCLES

After the definition of the hybrid dynamics, necessary aspect for stability needs to be defined. Due to the hybrid dynamics and the higher number of DOF, such systems become non-linear and also highly unstable in various different directions. There are different possibilities to give an informal argument for the stability of such unstable behaviour. As mentioned before, a bipedal (or humanoid) robot is a dynamical system, where the system refers to a periodic solution with an equilibrium point. Periodic cycles, respectively limit cycle called, is a common technique in the robotic literature to measure highly complex periodic systems. In this dissertation, every model have given an informal argument by using limit cycles.

First, let $\Phi(x_0, t)$ be the flow of the solution trajectory with the property x_0 for the initial conditions and the time t. A basic definition of periodic systems is described in [30] as

$$\Phi(x^*, T) = x^* \tag{15}$$

where T describes the time of the periodic cycle. If for every moment each multiple of T results in the same x-value, then Equation 15 defines a periodic orbit. For more complicated periodic orbits in higher dimensions, it can be really tough to find a visualization for showing the trajectory. A system in less or equal 3D, a periodic orbit would look like a loop, that is closed to itself. In contrast to linear systems, where infinite periodic cycles are possible (highly depending on the initial conditions), nonlinear systems mostly have only one isolated periodic cycle. A cycle is isolated, if there is no other periodic solution in the neighborhood around it. In this context, such a cycle is then called limit cycle [12]. Figure 5 shows such a limit cycle with var-

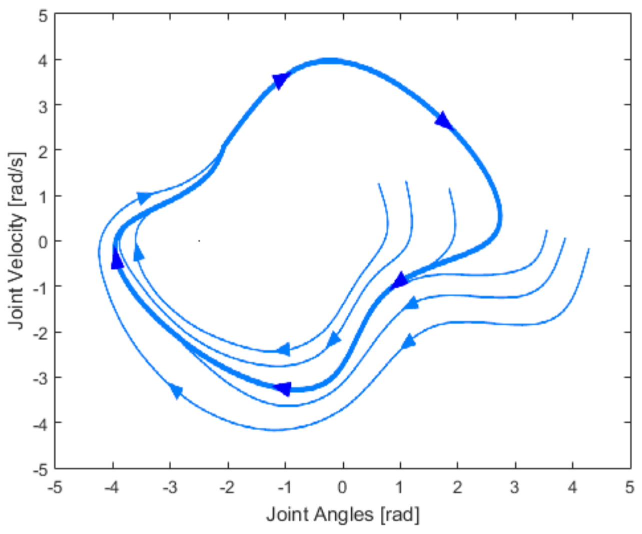


Figure 5: Phase portrait of nonlinear system

ious initial conditions x_0. Each of this lines represent a flow Φ with different x_0, where each of them converge to a closed limit cycle. The arrows monitor the direction of the trajectory. If all trajectories in the contiguity of the limit cycle converges to the cycle at $t \to \infty$, it can be state that it is stable. In reality it is a time consuming and tricky task to find a limit cycle - also the stability - for highly nonlinear mechanical system, such as humanoid robots. To use limit cycles for stability is common in the literature [18] [13], because the behaviour of walking is periodical trough the time between two steps. This kind of limit cycle can be found with the generalized coordinates, their velocities and a given set of initial configurations. Therefore, the flow $\Phi(x, t)$ must be specified in more detail as

$$\Phi(x_0, t) = [q(t), \dot{q}(t)]^T \tag{16}$$

where $q(t)$ is the vector of generalized coordinates. The complexity to find a limit cycle analytically is proportional to the increasing set of $q(t)$. Furthermore, the stability is one of the key problems in the robotic literature. Different researchers developed a various number of definitions for stability. A basic approach of determining the stability is to look at the behaviour at time $t \to \infty$ as with the Poincare Map, described in [13, 12].

3.8 STABILITY OF PERIODIC CYCLES - POINCARE MAP

The Poincare map, respectively surface called, is a convenient method to determine the dynamic stability of a hybrid system based on limit cycles. The Poincare surface is a plane normal to the limit cycle, which samples a single point of the cycle at each period of time. This point is the intersection point between the limit cycle and the Poincare surface. If this intersection point converges to a single point gradually, then the limit cycle is stable (see Figure 6). In order to use this method, the

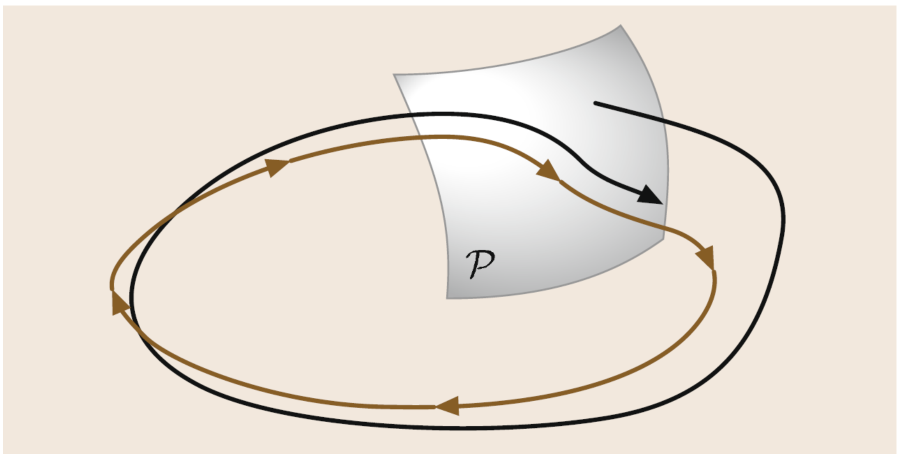


Figure 6: Poincare Map [31]

surface P , which will be used for sampling the intersections, must be defined. Theoretically each plane can be used for this method as long as it intersects with the limit cycle. In case of walking robots, the ground event (defined as guard S_g) is necessary for walking, furthermore it is applicable, because each model requires this event. Based on the assumption that the guard S_g is used, this means the generalized coordinates and velocities (summarized in $\Phi(x,t)$) are sampled at every DSP. To be more specific, for this method the surface P must be (n-1)-dimensional hypersurface, which is transversal to the flow of the limit cycle γ at all points. To achieve such a plane, P is defined by a vector $\nabla_x \gamma(x)$, which is normal to the surface and each point x_n

on the surface. The numerical solution of this definition is shown as followed

$$\begin{aligned} \gamma(x) &= 0 && x \in P \\ \gamma(x) &\neq 0 && others \end{aligned}$$

That implies all trajectories $\Phi(x, t)$, which go trough P must fulfill the statement $\gamma(x) = 0$. The correctness of stability for a model of a walking robot is given, if fixed points x^* can be found on the Poincare map, which are placed at the Double-Support Phase (DSP) of the hybrid behaviour. This section covers a short introduction in the stability theory of humanoid robots, more detailed descriptions of this technique are shown in [31, 16, 13].

After these theoretical definitions of the mathematical fundamentals, which will be used in this work. The following chapter will cover the concept based on this essentials. This chapter will also illustrate a practical application for the explained techniques.

4 CONCEPT

Bipedal and humanoid robot are mechanical systems, hybrid systems. Modeling and simulation of such systems is a highly complex task. Furthermore, a humanoid robots with a higher number of DOF is also highly unstable in their dynamics. Therefore, the previously described hybrid model (Section 3.6) is realized by improving a simple 2 dimensional passive walker with 2 links in a stepwise manner. The basic principle of the passive walker is the transformation of potential energy into kinetic energy, which is done by a walking gait along a slope. This method of passive dynamics is well known in the field of robotic [18] [19] [23], but will be not discussed in more details. This chapter covers all aspects, which are needed to model such robots.

4.1 COMPASS-GAIT BIPEDAL ROBOT (2D, 2 LINKS)

The compass-gait bipedal robot as shown in Figure 7, is one of the simplest bipedal robots. This bipedal robot with 2 links and 2 DOF is limited in motion at the sagittal plane and consists of 3 masses, which are necessary to define for the energy calculations. Let m_i with $i \in \{1, 2, H\}$ be the masses of the robot, then m_1 and m_2 describe the mass of the left, respectively the right leg, while m_H describes the main mass at the hips. The chosen parameters are shown in Table 2. Based on the 2 dimensional behaviour, each point can be represented with positions. These positions can be defined as $p_i = (x_i, z_i)^T$.

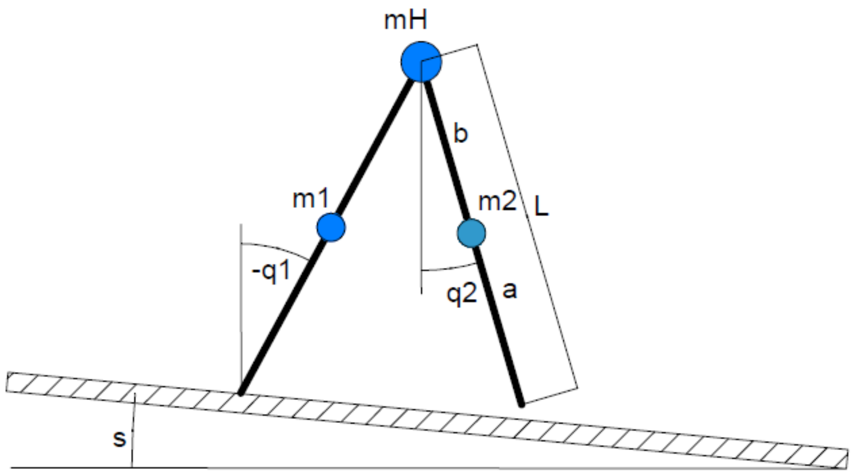


Figure 7: Bipedal robot with 2 joints, 3 masses in 2 dimensions

Table 2: physical parameter of bipedal robot

Parameter	Description	Value
m	leg mass	5
m_H	hip mass	10
g	gravity	9.81
a	length shank	0.50
b	length thigh	0.50
l = a + b	length leg	1

As mentioned in Section 3.6, a system like a bipedal robot, can be described with two different phases. The SSP defined by the Lagrangian formalism on the one hand and the DSP formulated using conservation of angular momentum on the other hand. In terms of generalized coordinates, $q(t) = [\theta_1, \theta_2]^T$ is defined as shown above. Due to the Lagrangian, the potential and kinetic energy are needed for the calculations of the continuous behaviour. The potential energy defined in Equation 2 and the kinetic energy described in Equation 1, can be applied to this model as follows

$$P = g(m_1 z_1 + m_H z_H + m_2 z_2)$$

$$K = \frac{1}{2}(m_1 v_1^2 + m_H v_H^2 + m_2 v_2^2)$$

Since this bipedal robot is modeled as passive walker, there is no need for a controller or any other external input. So, the whole right-side of the Lagrangian equation (Equation 4) is equal to zero. Depending on the programming language and the used tool, this equation can either be used directly or needs to be derived manually. In case of manual differentiation, the equation of motion is expressed in the following way

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = 0$$

For that reason, the matrices $M(q)$, $C(q,\dot{q})$, and $G(q)$ must be derived from the Lagrangian. The resulting matrices are based on the equations explained before and must fulfill the requirements from Chapter 3. For the evaluation of the languages, this matrices are adopted for this model from [5, 18] as

$$M(q) = \begin{pmatrix} m_H l^2 + m_1 b^2 + m_2 l^2 & -m_2 l a cos(\theta_1 - \theta_2) \\ -m_2 l a cos(\theta_1 - \theta_2) & m_2 a^2 \end{pmatrix} \tag{17}$$

$$C(q,\dot{q}) = \begin{pmatrix} 0 & -m_2 l a sin(\theta_1 - \theta_2) \\ m_2 l a sin(\theta_1 - \theta_2) & 0 \end{pmatrix} \tag{18}$$

$$G(q) = \begin{pmatrix} -g(m_1b + m_2l + m_Hl)sin(\theta_1) \\ m_2agsin(\theta_2) \end{pmatrix} \tag{19}$$

In contrast to the continuous part, the discrete impact is calculated through a reset map. Therefore, the moment of the impact is defined as a guard S_g in Equation 14. This guard uses the step height h_g of the swing leg and its derivatives to approximate the moment of the impact. The step height can be measured with the knowledge of the angles and the geometric description as followed

$$h_g(q) = l(cos(\theta_1 + s) - cos(\theta_2 + s)) \tag{20}$$

In order to accomplish a stable cyclic movement, the reset map has to cover two cases. The first case describes the change of the angles. At each ground impact, the support leg switches with the swing leg, thus the angles of the legs switched, too. This change of the angles is described with a reset matrix $J(q)$, which ensures the correctness of the definitions, that have been made before. After this instantaneous transition, the robot will be in the same configuration as depicted in Figure 7. In case of the compass-gait biped, let $J(q)$ be

$$J(q) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{21}$$

The exchange of the generalized coordinates requires an instantaneous transition for the velocities, as well. Therefore, the second case of the reset map is introduced by the impact map, which uses the conservation law of angular momentum (Section 3.5) to calculate the velocities after the impact. Based on this assumption, the conservation law for this model can be written as

$$H_g = Q_+^{-1} * Q_- = \begin{pmatrix} h_{11}^+ & h_{12}^+ \\ h_{21}^+ & h_{22}^+ \end{pmatrix}^{-1} * \begin{pmatrix} h_{11}^- & h_{12}^- \\ h_{21}^- & h_{22}^- \end{pmatrix} \tag{22}$$

whereas the single components of Q_+ und Q_- are defined as follows:

$$\begin{aligned} h_{11}^+ &= m_1a^2 - (m_1al)cos(\theta_1 - \theta_2) \\ h_{12}^+ &= m_1l^2 + m_Hl^2 + m_2b^2 - (m_1al)cos(\theta_1 - \theta_2) \\ h_{21}^+ &= m_1a^2 \\ h_{22}^+ &= -(m_1al)cos(\theta_1 - \theta_2) \\ h_{11}^- &= (m_2lb + m_Hl^2 + m_1lb)cos(\theta_1 - \theta_2) \\ h_{12}^- &= -m_2ab \\ h_{21}^- &= -m_1ab \\ h_{22}^- &= 0 \end{aligned}$$

$$\dot{q}^{+} = H_g * \dot{q}^{-} \tag{23}$$

Equation 23 multiplies the velocities $\dot{q}^{-}$ before the impact with the impact map h_g to get $\dot{q}^{+}$. Based on the continuous Lagrange formalism, the discrete reset map and the definition of the surface, the hybrid model can be summarized as

$$\begin{aligned} &\frac{d}{dt}\nabla_{\dot{q}}^{T} L(q,\dot{q}) - \nabla_{q}^{T} L(q,\dot{q}) = 0 && x(t) \notin S_g \\ &q^{+}(T) = J(q)q^{-}(T) && x(t) \in S_g \\ &\dot{q}^{+}(T) = H_g \dot{q}^{-}(T) && x(t) \in S_g \end{aligned}$$

STABILITY

Due to the method described in Section 3.8, an informal argument for stability can be given through a Poincare map. This stability argument is necessary to achieve a valid comparison to existing developments [13, 5]. In case of a walking robots with hybrid dynamics, the surface is applicable to be set on the DSP, where the swing leg hits the ground. The environment (the ground) is set to $\Psi = 0.05 rad$ and the compass-gait biped experience no control input $u(q) = 0$. After various iterations with different properties of the robot, the following limit cycle was found as shown in Figure 8. In this figure the velocities

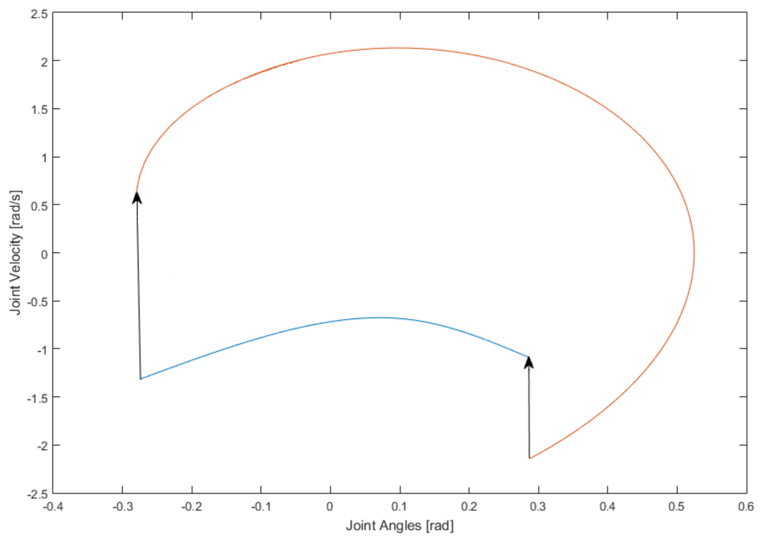


Figure 8: Stability prove for Compass-Gait biped

and angles of Θ_1 and Θ_2 are shown, the orange curve visualizes Θ_2 and the blue one displays the change of Θ_1. The arrows between the curves displays the discontinuity through the ground impact as well as the direction in which the trajectories changes. In order to give an informal argument for the stability of the limit cycle the intersection point must converge to a fixed point.For this model, the following fixed point were measured

$$x^* = \begin{bmatrix} -0.28 & 0.27 & 0.51 & -1.1 \end{bmatrix}^T \tag{24}$$

After this model is stable with the given fixed point, the model can be extended with new links and additional behaviour. The next model will show a more realistic robot with knees.

4.2 THE BIPEDAL ROBOT WITH KNEES (3 JOINTS, 2D)

The bipedal robot with knees is more realistic in case of anthropomorphic than the compass-gait biped from above. The kneed biped is shown in Figure 9 and the used parameters are listed in Table 3. Similar to the last one, this model has no controller and is limited to the sagittal plane. For a complete description the generalized coordinates have enlarged to $q(t) = [\theta_1, \theta_2, \theta_3]^T$ where θ_3 defines the angle of the thigh of the swing leg, θ_2 the angle of the swing leg shank and θ_1 the angle of the support leg with respect to the vertical axis of the base frame. Due to this changes the potential and kinetic energy has slightly expanded to

$$P = g(m_1 z_1 + m_H z_H + m_2 z_2 + m_3 z_3) \tag{25}$$

$$K = \frac{1}{2}(m_1 v_1^2 + m_H v_H^2 + m_2 v_2^2 + m_3 v_3^2) \tag{26}$$

The used masses for this calculations are shown in the figure below and are defined as m_i with $i \in \{1, 2, 3, H\}$. Regarding the limitation through the sagittal plane, the position of each mass can be identically described than the compass-gait bipedal robot as $p_i = (x_i, y_i)^T$.

Similarly to the first model, the biped can be expressed with a single-support and a double-support phase. Because of the separation of the thigh and the shank of the swing leg, the SSP has to be divided into two sub-phases, which are separated by the knee event. In the first phase of the SSP, the shank and the thigh can move freely and independently to each other. Based on the humanoid mechanics, the shank can't move further than the thigh and needs therefore an additional limitation in the movement [4]. Similar to the ground impact, the knee event can be modeled through the conservation law of angular momentum. The equation for the knee event is shown in Equation 28, given by the guard S_k from Equation 27.

$$S_k = \{(q, \dot{q}) | \theta_2 - \theta_3 <= 0, \dot{\theta}_2 - \dot{\theta}_3 <= 0\} \tag{27}$$

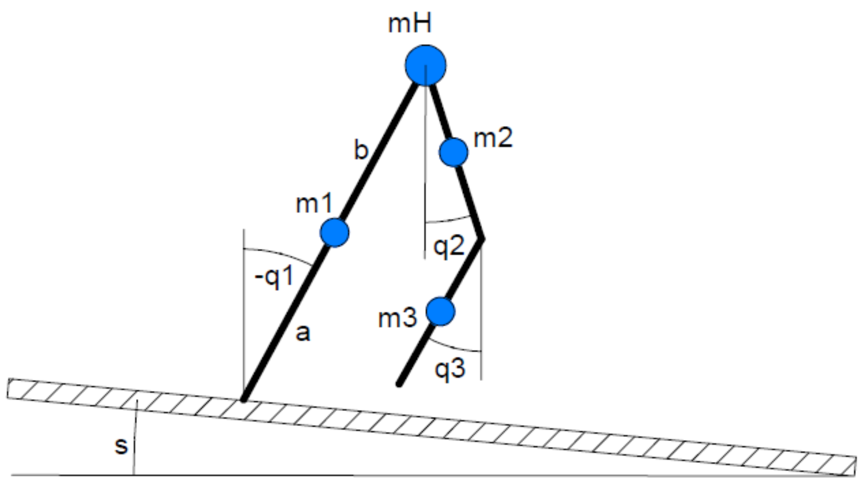


Figure 9: Bipedal robot with 3 joints in 2D

Table 3: physical parameters of knee biped [18]

Parameter	Description	Value
g	gravity	9.81
m_H	hip mass	10
m_1	gravity	5
m_2	length shank	3.5
m_3	length thigh	1.5
l	length leg	1
a_1	leg mass	0.53
b_1	hip mass	0.47
a_2	gravity	0.35
b_2	length shank	0.15
a_3	length thigh	0.25
b_3	length leg	0.25

$$H_k = \begin{pmatrix} h_{11}^{+} & h_{12}^{+} \\ h_{21}^{+} & h_{22}^{+} \end{pmatrix}^{-1} * \begin{pmatrix} h_{11}^{-} & h_{12}^{-} & h_{13}^{-} \\ h_{21}^{-} & h_{22}^{-} & h_{13}^{-} \end{pmatrix} \tag{28}$$

The single values of the matrices Q_- and Q_+ are listed as followed

$$\begin{aligned}
h_{11}^+ &= (m_1 + m_H + m_2)l^2 + m_1 b_1^2 + m_2(l_1 + b_2)^2 \\
&\quad - (m_2 l a_2 + m_1 l(l_2 + a_1)c_{12} \\
h_{12}^+ &= m_2 a_2^2 + m_1(l_2 + a_1)^2 - (m_2 l a_2 + m_1 l(l_2 + a1)c_{12} \\
h_{21}^+ &= -m_2 l a_2 c_{12} - m_1 l(l_2 + a_1)c_{12} \\
h_{22}^+ &= m_2 a_2^2 + m_1(l_2 + a_1)^2
\end{aligned}$$

$$\begin{aligned}
h_{11}^- &= m_1 b_1^2 + (m_H + m_1 + m_2)l^2 + m_2(l + b_2)^2 \\
&\quad - (m_1 l l_2 + m_2 l a_2)c_{12} - (m_1 l a_1)c_{13} \\
h_{12}^- &= m_2 a_2^2 + m_1 l_2^2 - (m_2 l a_2 + m_1 l l_2)c_{12} \\
&\quad + m_1 a_1 l_2 c_{23} \\
h_{13}^- &= m_1 a_1^2 - m_1 l a_1 c_{13} + m_1 l_2 a_1 c_{23} \\
h_{21}^- &= (m_2 l a_2 - m_1 l l_2)c_{12} - m_1 l a_1 c_{13} \\
h_{22}^- &= m_2 a_2^2 + m_1 l_2^2 + m_1 l_2 a_1 c_{23} \\
h_{23}^- &= m_1 a_1^2 + m_1 l_2 a_1 c_{23}
\end{aligned}$$

$c_{12} = cos(\theta_1 - \theta_2)$, $c_{13} = cos(\theta_1 - \theta_3)$ and $c_{23} = cos(\theta_2 - \theta_3)$ define the used cosines of Equation 28. After the knee lock event, the second sub-phase of the SSP starts. In the second phase the knee stays locked and keeps the constraint $\theta_2 = \theta_3$, which means the shank and thigh move as a single link. Therefore, the mechanics in this phase can be simplified to a two-link system (identically with the first model) until the tip of the swing leg hits the ground. Regarding the definition, that both legs are fully extended at the ground impact, the reset map can be adopted from the knee-less model. This assumption of fully extended knees also affects the exchange of angles for the impact in the following form

$$J(q) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \tag{29}$$

With the description of the walking phases, the continuous and discrete equations and the reset maps, the hybrid system can be summarized into the following lines as

$$\begin{aligned}
&\frac{d}{dt}\nabla_{\dot{q}}^T L(q,\dot{q}) - \nabla_q^T L(q,\dot{q}) = 0 && x(t) \notin S_g \\
&q^+(T) = J(q)q^-(T) && x(t) \in S_g \\
&\dot{q}^+(T) = H_g \dot{q}^-(T) && x(t) \in S_g \\
&q^+(T) = q^-(T) && x(t) \in S_k \\
&\dot{q}^+(T) = H_k \dot{q}^-(T) && x(t) \in S_k
\end{aligned}$$

S_g and S_k are the guards for the ground and knee event, which are defined above. If one of this events are triggered, the velocities experience a discontinuity and the angles switch depending on the reset map. At each point where no event needs to be handled, the Lagrangian formalism calculates the dynamics in a continuous manner.

STABILITY

Similarly to the first argument of stability this model can also be identified by a limit cycle. Due to the assumption of fully extended knees of this model, the Poincare map can be adopted from the previous model. Also the slope angle $\Psi = 0.05$ rad and the passivity of the model are equal to the first model. The resulting limit cycle is shown in Figure 10, where the yellow curve visualize the dynamic change of the non-supporting thigh. As before, the trajectories of the links

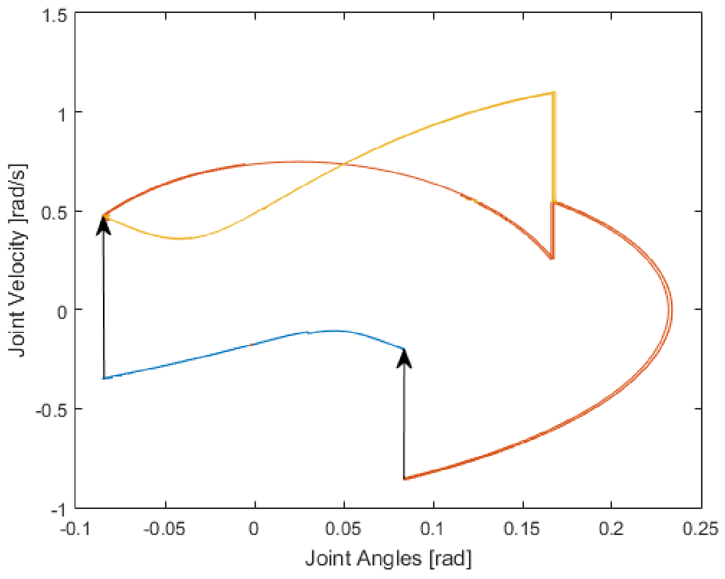


Figure 10: Stability prove for kneed biped robot

are converging into a fixed point on the surface. The arrows show the direction of the trajectories as well as the discontinuity in terms of the ground impact. The resulting fixed points are shown in Equation 30.

$$x^* = \begin{bmatrix} 0.075 & -0.081 & -0.081 & -0.26 & 0.49 & 0.49 \end{bmatrix}^T \qquad (30)$$

After fixed points are determined by the limit cycle for this model, the model can be extended with new links and additional behaviour. The next model will show a humanoid robot in 2D with the same conditions for the lower body as before.

4.3 HUMANOID ROBOT

This model is in contrast to the previously discussed models different in its definitions, it has transformed into a humanoid robot. This means an upper body has been added above the hips. Due to this definitions in Section 3.1.1 only 1 link need to be extended for the humanoid robot. This extension will be modeled as an inverted pendulum above the hip mass, which also extends the potential and kinetic energy. The potential and kinetic energy for this model are shown in Equation 31, respectively in Equation 32. In order to calculate the positions and angles of this link, additional parameters are needed. In Table 4 shows the parameters for this model.

$$P = g(m_1 z_1 + m_H z_H + m_2 z_2 + m_3 z_3) \tag{31}$$

$$K = \frac{1}{2}(m_1 v_1^2 + m_H v_H^2 + m_2 v_2^2 + m_3 v_3^2) \tag{32}$$

Table 4: Additional parameters for the humanoid robot

Parameter	Description	Value
m_T	torso mass	3
l_T	length torso	5

The number of the generalized coordinates also increased to $q(t) = [\theta_1, \theta_2, \theta_3, \theta_4]^T$. These coordinates are defined in Figure 11. This new link in the kinematic chain can move independent to the other joints and needs therefore a control behaviour. The continuous behaviour of the humanoid robot is given by

$$\frac{d}{dt}\nabla_{\dot{q}}^T L(q,\dot{q}) - \nabla_q^T L(q,\dot{q}) = \Gamma(q,\dot{q}) \tag{33}$$

where the $\Gamma(q,\dot{q})$ describes the control behaviour.

The controller explained in [38], uses an energy based approach to keep the upper body balanced. The Equation 34 shows such a controller.

$$\tau_5 = k_5(\theta_{5d} - \theta_5) - k_{5v}\dot{\theta}_5 \tag{34}$$

where k_5 and k_{5v} are the control parameters and θ_{5d} is the desired angle of the upper body. The additional energy, which is consumed by the control behaviour for the upper body must be added externally to the system. This form of energy can be added through a torque to the hip joint as followed

$$\tau_2 = -k_{leg} L \qquad (L > 0)$$
$$\tau_2 = 0 \qquad (L < 0)$$

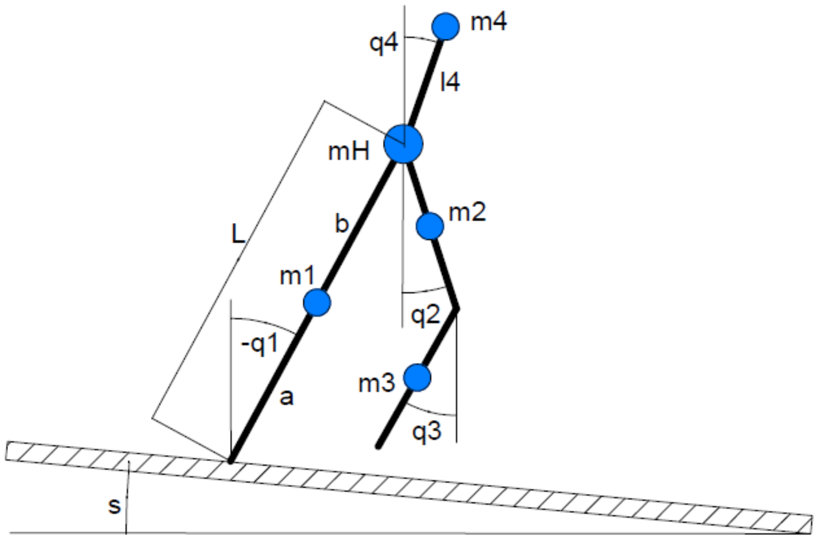


Figure 11: Humanoid robot in 2D

L determines the step length and k_{leg} is defined as the torque gain. The used step length is calculated through the angles and the geometric description, it is positive as long as the swing tip is behind the support tip [3].

$$L = -l_1 sin(\theta_1) + l_2 sin(\theta_2) + l_3 sin(\theta_3) \tag{35}$$

Contrary to the previous extension this link is changing the impact maps, which are used for the ground event as well as the knee event. For the last model the additional knee for the swing leg did not affect the impact with the ground, because the knee is completely stretched when the leg hits the ground. Therefore the swing leg can be assumed as one rigid link. The enlarged matrices for this model can be calculated equally to the previous ones and will not be discussed in more detail. After the advanced definitions of the continuous and discrete behaviours as well as the new introduced controller, the overview of the dynamics can be summarized into the following form

$$\begin{aligned}
&\frac{d}{dt}\nabla_{\dot{q}}^T L(q,\dot{q}) - \nabla_q^T L(q,\dot{q}) = \Gamma(q,\dot{q}) && x(t) \notin S_i \\
&q^+(T) = q^-(T) && x(t) \in S_g \\
&\dot{q}^+(T) = \dot{H}_g q^-(T) && x(t) \in S_g \\
&q^+(T) = q^-(T) && x(t) \in S_k \\
&\dot{q}^+(T) = \dot{H}_k q^-(T) && x(t) \in S_k
\end{aligned}$$

STABILITY

Due to the complexity and higher number of DOF, the limit cycle has increased by the angle and velocity of the torso (upper body). In contrast to the other links, the torso can move independently from the legs and has therefore no direct impact in the stability. In order to give an informal argument for the stability, the velocity and the angle of the upper body are not needed, but shown for a complete visualization of the system in the limit cycle. The limit cycle of the humanoid robot in 2D is shown in Figure 12, where the separated violet cycle show the change of the trajectory of the torso. The rest of the limit cycle has the same behaviour as the kneed robot, except of the fixed point. The fixed point has slightly changed through the

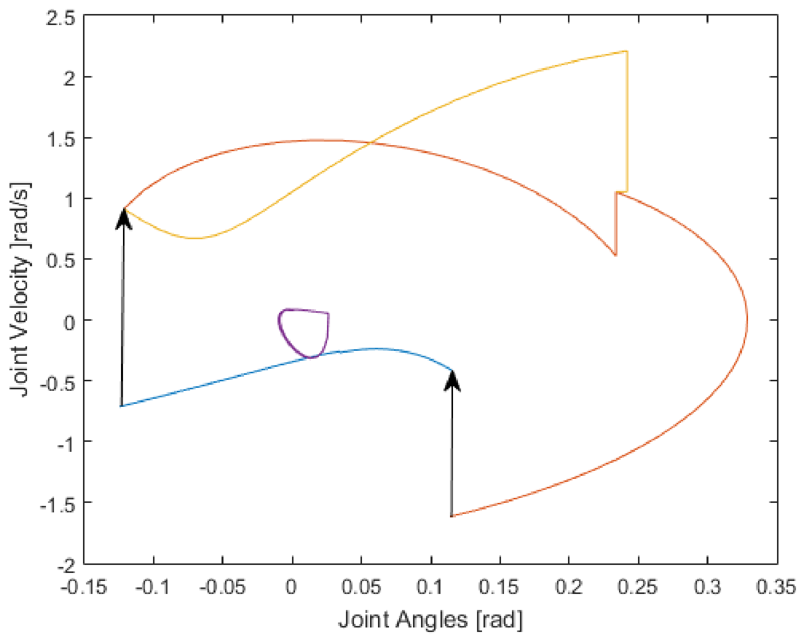


Figure 12: Stability prove for humanoid robot in 2D

enlarged matrices for the discrete events, respectively the extension in the kinematic chain. The resulting vector for the fixed points of the limit cycle is displayed in Equation 36.

$$x^* = \begin{bmatrix} 0.075 & -0.081 & -0.081 & -0.26 & 0.49 & 0.49 & 0.03 & 0.05 \end{bmatrix}^T \tag{36}$$

For the next and final model, the concept of the humanoid robot in 2D has been used for the development of an 3 dimensional approach. This final model will be discussed in the following section and will illustrate all necessary aspects.

4.4 HUMANOID ROBOT IN 3D

The last model in this series of developments is the humanoid robot in 3D as shown in Figure 13. Therefore, the previous model from Section 4.3 will be extended from 2D into 3D. Based on the definitions in Section 3.1, the 3 dimensional approach is not limited in the sagittal plane anymore and can move in the coronal plane too. In order to guarantee the completeness of the movements in both planes, this change results in a higher number of generalized coordinates. The resulting vector of the coordinates is $q(t) = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7]^T$. Contrary to the other extensions, this model needs a different geometric definition of the positions and rotations. This 3 dimensional approach of rotations and measurement of the positions is described in Section 3.4 and shown in

$$R = \begin{pmatrix} sin(\Theta) * cos(\Phi) \\ sin(\Phi) \\ cos(\Theta) * cos(\Phi) \end{pmatrix} \tag{37}$$

Φ and Θ are corresponding to the angles of Figure 13. In the concrete example of the support leg, Φ and Θ are expressing the angle θ_1, respectively θ_2. Also the measurement of each point has extended to a 3 dimensional vector. Each point is now expressed as a vector $p_i = (x_i, y_i, z_i)^T \in \mathbb{R}$. Due to the calculations of potential (Equation 38) and kinetic energy (Equation 39), let m_i with $i \in \{1, 2, 3, 4, H, T\}$ be the masses of the robot.

$$P = g(m_1 z_1 + m_H z_H + m_2 z_2 + m_3 z_3 + m_T z_T) \tag{38}$$

$$K = \frac{1}{2}(m_1 v_1^2 + m_H v_H^2 + m_2 v_2^2 + m_3 v_3^2 + m_T v_T^2) \tag{39}$$

The movements of this model can be divided into 2 different types, sagittal and coronal movement. The sagittal movement on the one hand, is cyclic and need therefore only a control behaviour to keep a constant step length and for balancing the upper body. The coronal movement on the other hand is an additional factor of instability. This instability can be handled through a spring-damper behaviour, which represents the functionality of muscles [7]. This functionality is needed for the sideways movement of the legs as well as for the upper body. The controller for the step length stays the same as for the 2 dimensional approach. Let $\Gamma(q, \dot{q})$ be the vector of control forces, applied to the system. This vector Γ from Equation 33 is defined for this robot as followed

$$\Gamma(q, \dot{q}) = (0, \tau_2, 0, 0, \tau_5, \tau_6, \tau_7)^T \tag{40}$$

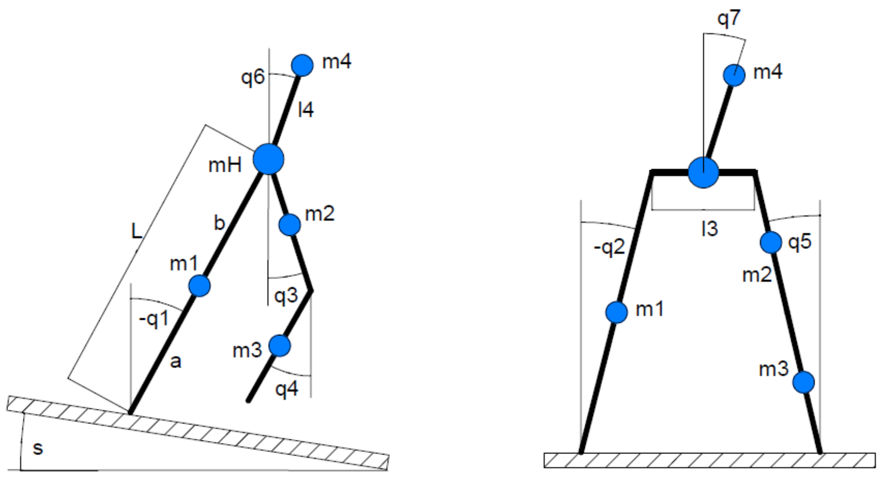


Figure 13: Humanoid robot in 3D

where $\tau_i = k_i\theta_i + k_{vi}\dot{\theta}_i$ with $i \in \{2, 5, 7\}$ and τ_6 stays the same as for the 2 dimensional approach. Continuing with the discrete behaviours, the knee calculation is not affected by the additional dimension. Hence, the calculation from the previous model can be adopted for this approach. The ground event which needs to handle a 3 dimensional momentum is a complex and time consuming task. Regarding the limited time of this work, this improved behaviour of the robot is out of scope for illustrating the research question. Therefore, the 3D humanoid robot will only be evaluated in terms of sagittal movements. The additional coronal movements are not discussed in this work and will also not considered for the stability argumentation.

STABILITY

As mentioned before, this model only consider movements in the sagittal plane. Thus, equally to the last model this one consists of 5 links with 5 regarded angles and velocities. As an informal argument for the correctness of the kinematic descriptions, the limit cycle for the 3 dimensional humanoid robot is identical to the 2D approach that means the Figure 12 also refers to this model. Also the fixed points of the previous limit cycles are equivalent to this approach.

This chapter covered the concept of a humanoid robot in 3D in a stepwise manner. This includes all used nontrivial descriptions and equations. The following chapter will illustrate the implementation of the concepts described in the previous sections. It starts with the Compass-Gait biped, followed by the humanoid robot.

5 IMPLEMENTATION

Various tools are used for modeling and simulating humanoid robots. Section 2.2 refers to such tools with the capabilities to use the Lagrangian formalism for complicated mechanical systems. Due to the benefits of equation-based tools like Acumen, the model of humanoid robots can be done by an implementation in a stepwise manner. This chapter will cover the most important aspects of the implementations of the models in Acumen. The implementation is based on the concepts and definitions of Chapter 4. Thus, this chapter starts with the explanation of the compass-gait bipedal robot from Section 4.1, then continues with the humanoid robot from Section 4.4. The implementation of kneed biped and humanoid in 2D will not be discussed in this chapter, due to the similarities of those methods. These codes can be concluded out of the concepts of Chapter 4 and the explained implementations of this chapter.

5.1 COMPASS-GAIT BIPED IN 2D AND ACUMEN

Mechanical systems, like a humanoid or bipedal robot, can be described by hybrid systems. Such systems are defined trough a continuous (SSP) and a discrete (DSP) component. As discussed in Chapter 4, the compass-gait bipedal robot can be modeled by a hybrid system. The general overview of the behaviour is shown in Figure 14. This flow chart shows the different tasks, the model has to handle in order to implement a stable and cyclic movement. Based on the shown concept, the structure of the model is defined as follows

```
model Main(simulator)=
initially
 //Definitions
 ...
always
 simulator.endTime+ = 20,
 simulator.timeStep+ = 0.001,
 //Static definitions
 ...
 //Geometric description
 ...
 //Continuous equations (Lagrange)
 ...
 //Discrete mapping (angular momentum)
 ...
```

```
  //Simulation
  ...
```

`model` defines the creation of the model with a name and its properties. The comments in the model relate to the steps shown in Figure 14. Their order is based on the behaviour of this model. Basically,

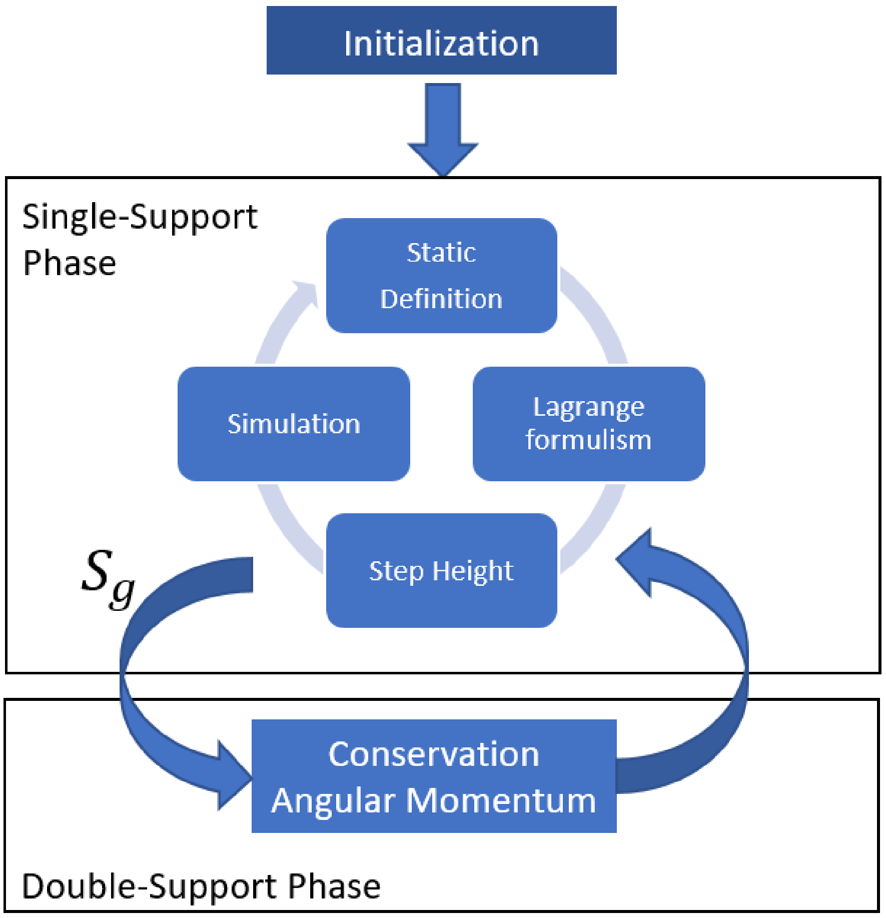


Figure 14: Flow chart of compass-gait biped

the model consists of two main parts, the `initially`- and `always`-section. In the `initially` part each of the used variables need to be defined with respect to its desired type. There are four variable types:

- numerical number [1 , 0.3 , 10.04]
- vector/matrix [(1,1),((1,2.2),(3.1,0))]
- constant ["Hello", "World"]
- boolean ["True", "False"]

Due to the supported equations, it is also possible to define a derivative of a variable in this section. A more detailed description is given in Section 2.2. The following snippet shows the possibilities of definitions in the `initially`-section

```
m = 1, d = 1, M = 10, g = 9.81,
t1 = 0, t1'=0, t1'' = 0,
mode = "ThreeLinks",
H = (0,0),
_3D=(), _3DView = (0,0), _Plot=(t1,t1'),
```

`_3DView`, `_3D` and `_Plot` are specific operators used for visualization of the data or for simulation of the model. The operator `_Plot` is used for selecting the variables that will be shown in the plot window. `_3DView` and `_3D` on the other hand specify the simulation, which view and components should be used for the simulation. After every variable is defined the `always`-section can be programmed. At first all static variables have to be defined which can be done by setting a static value for the variables in the `always`-section. Regarding the behaviour of the robot, it is important to ensure that the support leg does not experience a slip or any other movement. Therefore, the acceleration of the position variables x'' and y'' must be defined to zero. Also the generalized coordinates need to be defined as a vector based on the concept presented in Section 4.1 as

```
q = (t1,t2),
```

In the next step the geometries of the robot have to be defined. The foundation of the kinematic approach has been described in Section 3.4 (see Equation 6). Due to those definitions, the positions can be calculated considering the leg length `l`

```
p2 = pt + l*R1,
p4 = p2 + l*R2,
```

whereas `R1` and `R2` labels the rotation vector of the support leg, respectively the swinging leg. After calculating the positions of the masses, the potential and kinetic energy as well as the Lagrange equation can be computed. The potential and kinetic energy as defined in Equation 1 and Equation 2 are adapted to this model in form of

```
V = g*(m*y1 + M*y2 + m*y3),
T = 0.5*(m*((x1)'^2 + (y1)'^2) + M*((x2)'^2 + (y2)'^2) +
    m*((x3)'^2 + (y3)'^2)),
```

The primary behaviour of the model is described in Equation 4 and can be expressed using the following lines of code. The first line in this example subtracts the potential energy from the kinetic energy, followed by the differentiation of the equation for each dimension of the generalized coordinates.

```
L = T - V,
foreach i in 0 : (length(q) -1) do
   L'[(q(i))']' - L'[q(i)]=0,
```

In contrast to the continuous part, for the discrete event of hitting the ground with the swinging leg, the angles and velocities are calculated by applying the angular momentum (Equation 22). Specifically the step height (Equation 20) has to be computed, followed by the calculation of the change in angular momentum and the event itself. The guard S_g triggers this event based on the step height and its derivatives. If the tip of the swing leg finally hits the ground, the velocities after the impact are equal to the product of the change in angular momentum and the velocities before the impact. The structure of this behaviour is shown below.

```
h = l*(cos(t1+s)-cos(t2+s)),
Qp= inv(((h11p , h12p),(h21p , h22p))),
Qm = ((h11m , h12m), (h21m , 0)),
H =  trans((Qp * Qm) * trans((t1p,t2p))),
if (h <= 0 &&  (h)' < 0 && (h)'' < 0) then
    //switching angles
    ...
    //discontinuity of velocities based on H
    ...
noelse,
```

In order to complete the modeling and simulation of the compass-gait bipedal robot, a simulation of the biped is needed. The simulation is done by the operator `_3D`. The components to visualize are listed in

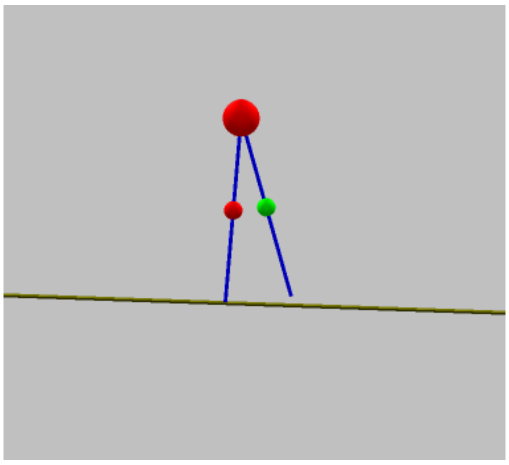


Figure 15: Simulation output of compass-gait biped in 2D

this operator. Each component of the simulation has also a specific set of properties which define the position, angle and other features. Acumen provides a limited set of objects for the internal simulation. In this case only the following 3 types are needed

- Sphere
- Cylinder
- Box

A concrete example of such a definition is shown as followed

```
_3D=(Sphere center=(x1,0,y1) size=0.05 color=(1,0,0))
```

where center and size define the geometric properties, the color property needs a Color model (RGB)- code. The resulting simulation of the model is shown in Figure 15.

All masses in the model are monitored as red spheres, except of the mass of the swing leg, which is a green sphere. The legs of the robot are displayed as blue cylinders with a small radius.

5.2 HUMANOID ROBOT IN 3D AND ACUMEN

Due to the similarities between the models and the fact, that each model can be derived from the model implementation shown in Section 5.1, the implementation of the kneed and the humanoid robot in 2 dimensions are not discussed further. The code can be derived from the detailed concept and the implementation from the previous section. Moreover, this section will cover only the parts of the code, which have changed and are not trivial to conclude out of the concept explanation.
To visualize the differences of the model in contrast to the first one, the flow chart shown in Figure 16 illustrates the most important sections of this model. The initialization part, which is programmed in the `initially`-section, is similar to the previous model. Only the number and size of the variables have changed. In terms of the vector of generalized coordinates, it expanded to a longer vector.

```
q = (t1,t2,t3,t4,t5,t6,t7),
```

Also, the section of static definitions is a trivial extension of the first implementation. All static variables have to be redefined in the `always`-section of the model. Based on the assumption that all masses and lengths are defined as static, the positions can be computed using Equation 37 and directly transferred to code as

```
R1 = (sin(-t1)*cos(t4),sin(t4),cos(-t1)*cos(t4)),
p1 = px + (b1)*R1,
p2 = px + (l2+b2)*R1,
```

where p1 and p2 are the positions of the thigh and shank masses of the support leg. Both masses are connected to the same link (support leg) and therefore, the rotation vector is applicable to both. Based on the positions, the kinematic energy (Equation 39) and the potential energy (Equation 38) can be directly used for the calculations. The Lagrangian differentiation stays basically the same, except for the right-hand side of the equation. The compass-gait robot has no need for a controller, therefore the right-hand side is zero. In this model the robot has a higher number of DOF and is also more unstable in

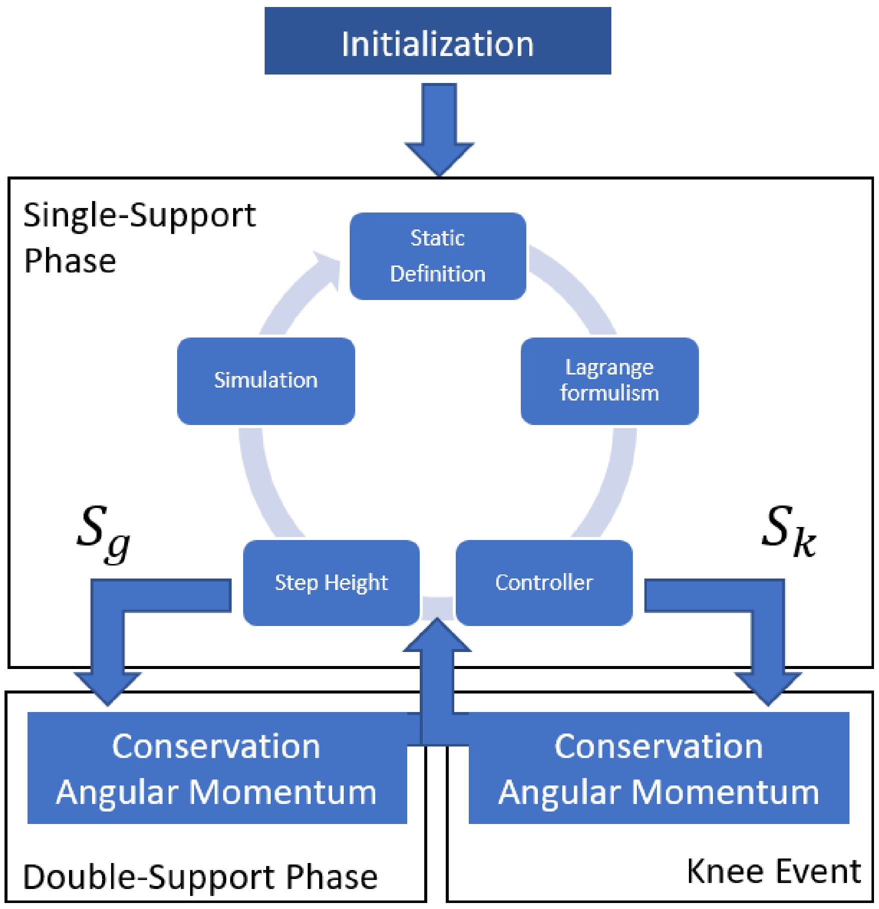


Figure 16: Flow chart of 3 dimensional humanoid robot

the movements. A controller can not be avoided in order to achieve a stable movement.

```
foreach i in 0 : (length(q3)-1) do
    L'[(q3(i))']' - L'[q3(i)]= tau(i)
```

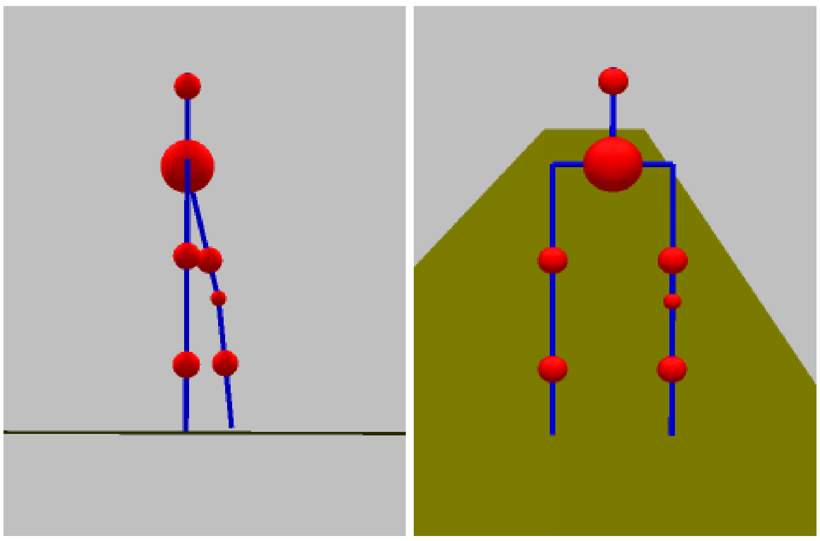
As described in Equation 40 the right-hand side of the equation is equal to the controller input and given by a vector based on single controller equations. The foundation of this control behaviour is the spring-damper principle [6]. The control to keep the movement stable is based on proportional controller, which uses the current step length as input as long as it is negative. The used code is displayed in the following snippet.

```
stepL = (xp - xt),
tau7 = -kp*(t7) - kd*t7',
tau2 = k2*stepL,
```

Another extension in the model of the robot is the second discrete event, that is used for the movement limitation of the knee. The event is triggered by the guard S_k (Equation 27) and is similar to the ground event. It involves the calculating of the matrices for the discontinuity of the velocities and the event itself. If the event is triggered, the restriction of the stretched leg simplifies the model from n-DOF to (n-1)-DOF. It is assumed that the leg is locked after the knee event until the ground impact, so the swinging leg is determined by a single link as well as the support leg, shown in Section 4.2. If this event happens, the following steps must be completed

- discontinuity of velocities (angular momentum)
- knee locking (Θ_3 stays equal to Θ_2)

The last section of this model is again the simulation, implemented by the operator `_3D`. The extension with respect to the previous model is trivial and will not be discussed in more detail here. The final simulation of the model of the 3D robot is shown in Figure 17. On the left the robot is visualized sideways, the right image shows the same robot from a frontal perspective.



(a) Side View 3D humanoid (b) Front View 3D

Figure 17: Different Views of the 3 dimensional humanoid robot simulation

This chapter covered all aspects, necessary for the modeling and simulation of simple as well as more complicated mechanical systems like humanoid robots. The implementation presented here is crucial for the next chapter which will illustrate the implementation between Acumen and Matlab based on expressivity as well as scalability. Additionally, this chapter will cover the differences in the process cycle introduced by the use of equation-based tools compared to common tool, called Matlab.

6 LANGUAGE EVALUATION

This chapter evaluates the expressivity and scalability of different languages. As mentioned in previous chapters the passive walking robot from Section 4.1 is a well known model in the robotics literature, due to the energy efficiency and the redundant controller. Therefore this robot is particularly suited to be compared in the context of implementations in different languages. The determined criteria for this implementation are stated in Section 6.2. The following languages are used for comparison

- Acumen [1, 40]
- MATLAB [22, 5]

The robots are modeled and designed for this work in Acumen and a similar model is done in MATLAB by [5]. Both models are using the same methods and techniques for modeling and simulating. This chapter will also present the advantages in the process of modeling, which goes in hand with the usage of equation-based tools.

6.1 PROCESS OF MODELING AND SIMULATION

In previous chapters the fundamental mathematical basics have been discussed (Chapter 3), the design and mathematical formalism of each single model including an informal argument for stability (Chapter 4) as well as a stepwise implementation (Chapter 5). Except of the fundamentals, each of these sections present a basic step in the process of modeling mechanical systems. Each step in the process is based on the previous parts and cannot be completed without the steps before.

The development of such a mechanical system as described in this work, starts with the design phase. This phase should cover all aspects that are necessary for the mathematical formalism in the next step. This section consists of the design of the robot as well as the definition of the angles and the environment. In case of the compass-gait biped, all aspects that are important for its design are summarized in Figure 7. After fixing all angles and measurements the mathematical formalism can be defined. A key challenge of describing the formalism is to kept it as simple as possible while aiming for the highest accuracy possible. The differentiation of the equations is a key challenge in this process, if the Euler-Lagrange equation (Equation 4) is used (key equation in the context of evolution of motions)[32]. Due to the definition, this differentiation needs to be done for each of the coordinates. Depending on the programming language this step can lead to an high overhead in manual work. In most of the cases the development process is not a waterfall one where the previous phase is completely done at the moment of starting with a new phase. It is in fact an cyclic process where the design or the formalism has to be extended and improved in order to achieve a convenient model. Such a cyclic process is shown in Figure 18, where the simulation phase is followed by the design phase. In the development of a complex

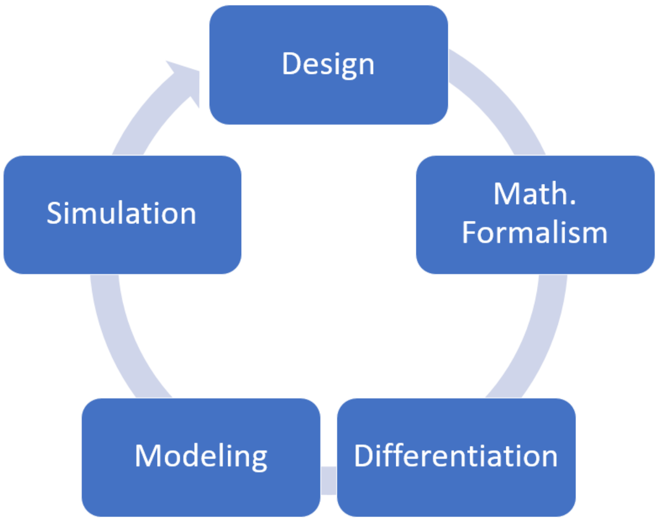


Figure 18: Typical Process of Modeling Robots

and highly unstable mechanical system it is common to start with a simple model. If the simple model is finished, it is extended one by one in order to raise its complexity. This is often easier than developing a complicated solution within one iteration. In case of the Euler-Lagrange formalism applied to humanoid robot, this iterative development process leads also to an increased calculation effort of the differentiation. This iterations of deriving are increasing the already existing overhead from the previous iteration. Additionally to this overhead, the efficiency of the simulation highly distinguishes by using different tools. Independent of the calculation method the goal of the differentiation process is to achieve the smallest result in terms of mathematical operations. The less mathematical operations are needed, the faster it can be calculated. This efficiency is compared for different tools in Table 1, called compact derivatives. Compact derivative is a feature, which guarantees a compact form without any large expressions. Most of the compared tools do not support this feature and even OpenModellica [25] is limited by producing a compact form for highly complex systems. In order to achieve a satisfying small simulation time with common tools, is supplying this tools with already calculated derivatives. Compact differentiation like this, can be also done by more intelligent programming tools like Acumen

[40]. Otherwise the simulation time is significantly higher, especially for more complicated systems like the 3 dimensional humanoid robot. This means the efficiency in terms of computation time highly depends on the form of the derivatives and can be optimized by using more advanced tools. Due to the comparison of Table 1 some tools supporting more features than other tools do. Equation-based tools like Acumen can handle and calculate these derivatives internally in a compact form. This leads to the conclusion, that with a equation-based tool can the whole differentiation phase from Figure 18 can be optimized, respectively eliminated. The next phase in the iteration is the modeling part, the equation from previous sections are combined and transformed into executable code. There are various tools available to model mechanical systems. A list of some tools can be seen in Section 2.2. The modeling phase is similar to the previous phase, since the amount of work that has to be spend in order to create a convenient model, highly depends on the used language. The expressivity of a language is a key characteristic for suitability, which will be discussed in more detail in the next section. The last step is the simulation of the model. Similar to the modeling phase, there exists various simulation tools. A key disadvantage of most tools is that they can not handle the modeling as well as the simulation. Furthermore, it has established to use more than one tool for modeling and simulation, this habit leads to additional overhead. Therefore, it would be more convenient, if the modeling and simulation part could be done in one step and in one tool. In contrast to other tools Acumen supports a simulation of models. This means, that a simulation does not require additional effort with respect to programming and can be done as one step of the modeling phase.

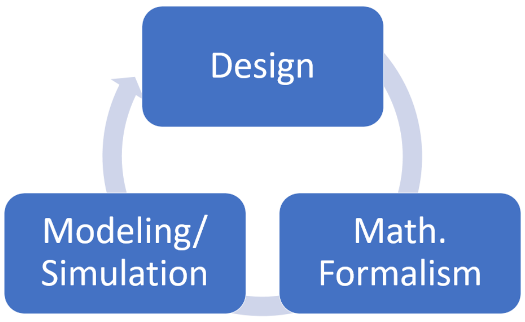


Figure 19: Modeling Process with Equation-based Tool Acumen

Summarizing, the process model can be simplified further from 5 into 3 phases as shown in Figure 19, because the differentiation phase is done automatically by Acumen, respectively the modeling and simulation phase are combined into one step.

6.2 EXPRESSIVITY/SCALABILITY OF USED LANGUAGES

The previous section explained the advantages in the development process using Acumen. Especially the omission of the differentiation phase using Acumen leads to an advanced development cycle. Such an improvement cannot be achieved with tools like MATLAB. The following chapter shows a short summary of the components and their implementation effort in Acumen, respectively MATLAB.

6.2.1 *Comparison of components*

Continuing with their differences, this chapter will discuss the expressivity and scalabilitiy of MATLAB and Acumen. In this context expressivity describes the similarity of the programming language to the natural mathematical formalism. The expressivity of a language is considered high, if the language can represent a equation as a minimum set of code lines. Thus, a common measure for expressivity is the number of needed code lines to achieve the same behaviour as the mathematical formalism. Scalability on the other hand defines the effort which is needed to change or enhance an existing model. It is concerned with the cost of adding a new link to the kinematic chain, for example. In order to keep the results comprehensible and comparable, the scalability will be measured by the number of changed code lines. For an overall evaluation the following aspects of a model implementation will be reviewed:

- The number of code lines
- Number of functions used, as well as their individual implementation effort in terms of code lines
- Scalability

As mentioned before, the languages Acumen and MATLAB which stand for different types of tools, respectively will be examplarily evaluated in the context of modeling and simulation. Each of the models described in this work, is implemented in Acumen. These models have also a informal argument for stability by using limit cycles with Poincare map. Apart from that, [5] developed a Compass-Gait robot using MATLAB. Due to the similarities in techniques, stability and methods, this comparison shown in Table 5 uses the following criteria for evaluation:

- Euler-Lagrange formalism: Summary of code lines which are needed to represent the desired behaviour.
- Angular momentum: Amount of code lines for calculating the impact using the angular momentum law
- Simulation: Number of code lines for the simulation of a Compass-Gait model
- Remaining code: Language specific code for connecting the used methods
- Complete code: Summary of all code lines

Table 5: Summary of code lines

Components	Acumen	MATLAB
Euler-Lagrange formalism	5	27
Angular momentum	16	20
Simulation	9	99
Remaining code	42	250
Complete code	72	400

Table 5 summarize the differences of Acumen and MATLAB in terms of needed code lines per component. The components for the comparison are the main techniques of the model. As shown in this table all rows in the table, except of Euler-Lagrange equation and the simulation are equally long. These criteria also grow due to extension the same amount of lines. A knee extension for example would lead to the same amount of additional lines in both languages, except for Euler-Lagrange. In the context of simulation, these implementations are not comparable, because of its differences. The simulation in Acumen is 3 dimensional, focusing on the visualization of the movement in a convenient way. The simulation in MATLAB focuses mostly on stability, evolution of movements over time and limit cycles. Therefore, both simulation techniques will not be discussed here in more detail. Finally, there is the implementation to the Euler-Lagrange formalism. Due to its significance for mechanical systems, it will be discussed in a very detailed manner over the next few pages.

6.2.2 *Euler-Lagrange formalism*

This section will cover the significance of the Euler-Lagrange formalism and the importance of the used language behind it. Due to the increasing complexity of components and new technologies, convenient languages with similarities to the natural formalism are needed in order to minimize the effort in the solution. Therefore, this section will also discuss the analogy to the natural mathematical formalism of Acumen in contrast to MATLAB. To refresh the principle, the Lagrangian uses the potential (Equation 41) and kinetic energy (Equation 42) to calculate the motions of the restricted masses. It is defined as the kinetic energy subtracted by the potential energy as shown in Equation 43. In order to determine the evolution of motion over time, it is necessary to derive the Lagrangian as shown in Equation 44. This behaviour results in four different equations, which involves different mathematical calculation techniques. As mentioned in the concept, a Compass-Gait robot is stable through the dynamics itself, meaning there is no need for a controller. Thus, the right-hand side of the Lagrangian is zero. Otherwise, this side of the equation needs to be derived as well.

$$P = g(m_1 z_1 + m_H z_H + m_2 z_2) \tag{41}$$

$$K = \frac{1}{2}(m_1 v_1^2 + m_H v_H^2 + m_2 v_2^2) \tag{42}$$

$$L = P - K \tag{43}$$

$$\frac{d}{dt}\nabla_{\dot{q}}^T L(q,\dot{q}) - \nabla_q^T L(q,\dot{q}) = 0 \tag{44}$$

If the model is extended with additional masses and links, only the terms for potential and kinetic energy will change. Thus, by means in terms of scalability, this formalism can be easily extended. An implementation of these mathematical formulas can be realized in Acumen the following way:

```
P = g*(m*y1 + M*y2 + m*y3),
K = 0.5*(m*((x1)'^2 + (y1)'^2) + M*((x2)'^2 + (y2)'^2) +
    m*((x3)'^2 + (y3)'^2)),
L = P - K,
// Euler-Lagrangian equation of motion
foreach i in 0 : (length(q) -1) do
  L'[(q(i))']' - L'[q(i)]= 0,
```

The formalism described in Equation 41 - Equation 44, can be implemented directly in Acumen. The possibility of partial derivatives in the language leads on the one hand, to a convenient and easy description of the Lagrangian formalism. On the other hand, the compact form of the derivatives in the background of Acumen results in a faster computation. The difference in computation time will increase noticeable in larger systems, that require more and higher differentiations. In terms of scalability, this language is identical with the natural formalism, since only the potential and kinetic energy needs to be adapted, while the Lagrangian itself remains the same. In contrast to Acumen, MATLAB does not support equations or derivatives in a natural way. This means Equation 44 can not be used directly in the programming language and it must be derived manually. As explained in Section 3.3, the final equation of the differentiation is the following

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = 0 \tag{45}$$

Equation 45 is equivalent to Equation 44 just after the differentiation. The following code was written by [5] and shows one way of implementing the Lagrangian in MATLAB.

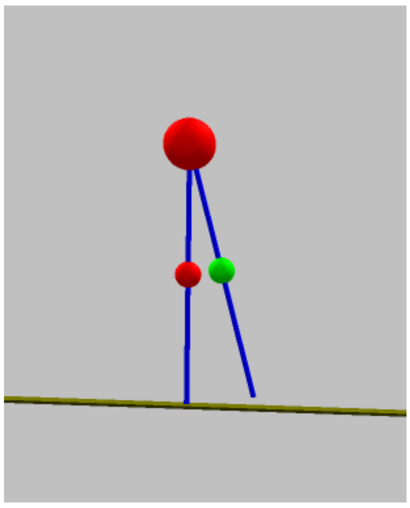
```
function [Dx] = equations of motion(t,x)
     m11 = p1;
     m12 = -p2 ∗ cos(q1 - q2);
     m21 = -p2 ∗ cos(q1 - q2);
     m22 = p3;
     c12 = -Dq2 ∗ sin(q1 - q2) ∗ p2;
     c21 = Dq1 ∗ sin(q1 - q2) ∗ p2;
     g1 = -sin(q1) ∗ p4;
     g2 = sin(q2) ∗ p5;
     M = [m11, m12; m21, m22];
     C = [0, c12; c21, 0];
     G = [g1; g2];
     DDq = M \ (-C∗Dq-G);
     Dx = [Dq; DDq];
end
```

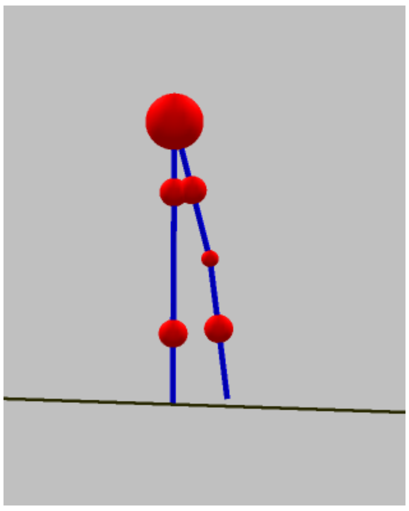
Due to the lack of supported features this implementation requires manual work before it can be used. For this implementation the matrices M, C and G are needed, as explained in Section 3.3. This also means every time the design or the mathematical equations change, the differentiation needs to be recalculated again. Furthermore, an iterative design process of a mechanical system using the Lagrangian increases the already existing overhead of manual work even more. For a simple passive walker the matrices are relatively small, but with increased complexity the order and intricacy of those matrices are also raising. Another simplicity factor of this approach is that it only consider two dimensions. If the model incorporated a higher number of DOF and three dimensions, the implementation in MATLAB would be even more complex. In contrast to these scalability difficulties in MATLAB, the implementation of Acumen can easily be extended to a higher number of DOF and three dimensions as shown here:

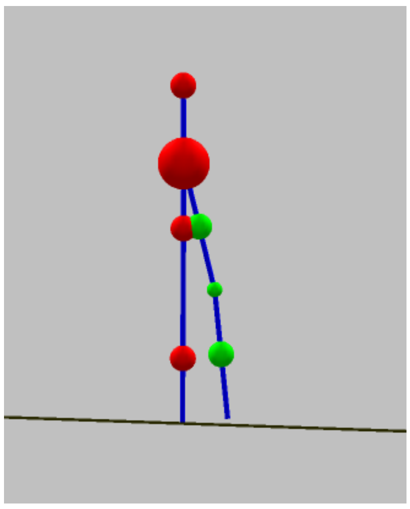
```
V = g*(m1*p1(2) + mh*pzh(2) + m2*p2(2) + m2*p3(2) + m1*p
    4(2) + m4*p5(2)),

T = 0.5*( m1*(  (p1(0))'^2 + (p1(1))'^2 + (p1(2))'^2 ) +
     mh*( ... ) + m2*(... ) + m2*( ... ) + m1*( ... ) +
    m4*( ... )),
L = T - V,
foreach i in 0 : (length(q)-1) do
 L'[(q(i))']' - L'[q(i)]= tau(i)
```

These lines of code implement a humanoid robot in three dimensions and 7 DOF. In comparison with to MATLAB example of Lagrangian representations, this extension in dimensionality and complexity has not really changed the length of the Acumen code. The only differences are the additional terms in the energy equations. This emphasizes the fact, that the implementation in Acumen is scalable as well as convenient for using the Lagrangian equations. Furthermore, the use of Lagrangian in Acumen can simplify the development of humanoid robot significantly. Considering the above mentioned aspects of implementation and their comparison in MATLAB and Acumen, a clear conclusion can be drawn. Based on the data obtained from the previous chapters it can be stated that the equation-based tool Acumen is not only appicable in the context of Lagrangian formalism, it is also conveniently applicable for bipedal and humanoid robots. The transformation of the natural mathematics to the final implementation can be simplified by using Acumen. The original cycle of modeling and simulation consists of five phases with a considerable work overhead during multiple phases. Due to equation-based tools that support differentiations, this cyclic process can be simplified and the overhead which is accompanied by the usage of different tools for modeling and simulation, can be minimized. Based on this simplified process introduced by Acumen, a three dimensional humanoid robot has been modeled and simulated in a stepwise manner. As described in Chapter 4 and shown in Chapter 5, the implementation starts with a simple passive walker with two dimensions. Leveraging the scalability of this expressive language, the model has been extended with respect to its complexity and dimensionality into a humanoid robot. In total there have been four different models been build along the way, whereas each model is build upon the previous ones. This development process is illustrated by the resulting simulations of the four models in Figure 20.


(a) Compass-Gait Robot


(b) Kneed Bipedal Robot


(c) 2D Humanoid Robot

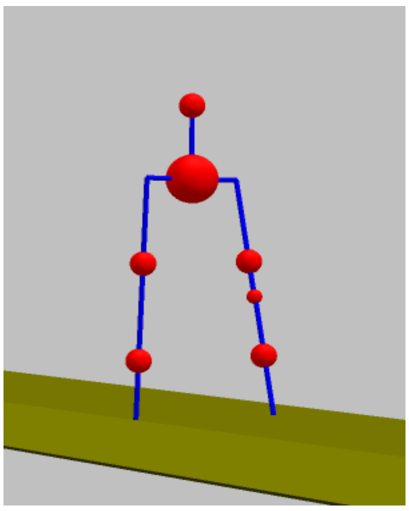

(d) 3D Humanoid Robot

Figure 20: Simulation views of all four implemented models

7 CONCLUSION AND FUTURE WORK

The increasing complexity of techniques and the use of new components challenge the need for flexible solutions. The walking cycle of a humanoid robot consists of two phases which can be realized as a hybrid behaviour. These phases are defined by the number of legs contacting the ground and both of them use another method for the calculation. The Single-Support Phase (SSP) is calculated through the continuous technique, Euler-Lagrange. The Euler-Lagrange formalism uses the potential and kinetic energy for calculating the evolution of the motions of a mechanical system (Section 3.3).The Double-Support Phase (DSP) on the other hand handles the discrete moment of interaction with the surface and determines the change of velocities and angles through the conservation law of angular momentum (explained in Section 3.5). Furthermore, all necessary mathematical fundamentals and background knowledge for designing complicated mechanical systems are explained and discussed in Chapter 3. Within this thesis, convenient models for bipedal as well as humanoid robots has been developed in Acumen. Based on the fundamentals, the concepts of the models are determined in Chapter 4, respectively the implementation of the models is shown in Chapter 5. Chapter 6 describes the benefits of the used tools and methods in contrast to each other.

Before the thesis statement can be discussed, the correctness of the models need to be shown by an informal argument of stability. This stability statement underlines the significance in what follows, respectively the similarity between the developed models and the compared models. Such a stability evidence is performed by using limit cycles. The principle of the limit cyle is explained in Section 3.7 and shown for each model separately. The thesis of this work investigates if a equation-based tool can be used for the Euler-Lagrange formalism in order to simplify the development process of humanoid robots. Equation-based tools like Acumen support equations and derivatives which lead to a solution closer to the natural mathematical formalism. This similarity can decrease the required work respectively the manual overhead for modeling. The thesis identifies necessary aspects influencing the required work for modeling. Due to the similarities of the natural mathematics the use of Acumen can lead to a convenient and scalable model for humanoid robots. To prove the correctness of this statement different factors like Euler-Lagrange, angular momentum and simulation, are compared to a equivalent implementation in

MATLAB. In order to use the Euler-Lagrange formalism in MATLAB, the equation needs to be derived multiple times before it can be used. This differentiation of the formalism can be described as an addition of different matrices (Section 3.3), called equation of motion. As explained in Chapter 6, this differentiation needs to be done manually or by an external tool which leads to an increased modeling effort by using naive tools. Moreover, manual differentiation doesn't guarantee the criteria of compactness resulting in insufficient computation time. Each change or extension of the design enlarge the equation of motion as well as the already existing overhead of modeling. The resulting code of the Lagrangian part of the model in Acumen sums up to 5 lines whereas the amount of lines in MATLAB is 27. The overall number of code in MATLAB amounts approximately to 400 lines, respectively 75 lines for Acumen. This difference in code lines as well as the compactness of the derivatives in Acumen leads to the conclusion that the usage of Acumen for Lagrangian is convenient and scalable.

Based on the implementations and results of this work it can be stated that the usage of Acumen brings some handy features and advantages. These advantages of Acumen are proving the correctness of the thesis considering the research question. On the other hand there is still place for improvements. Due to the limited time and the fact that Acumen is an experimental language the humanoid robot in 3 dimensions is only able to move forward on a slope and also the double-support phase is only capable of sagittal movements. Additionally, the humanoid robot is simplified on the upper body by using only a single inverted pendulum. A more realistic model could consist of a more complicated inverted pendulum with a higher order controller. Therefore, further research should include investigation into a complete concept for the 3 dimensional ground impact. This situation is necessary to calculate a complete model of humanoid robots. The upper body can be extended into a more complex one which is closer to the human nature. Besides that the already proven benefits of Acumen opens room for additional research. Due to the experimental nature this language can be adapted for other mechanical systems.

The conclusion of this thesis validates the statement that the used language Acumen can improve the handling of Euler-Lagrange formalism by reducing the distinction to the natural mathematical formalism. It also opens possibilities for additional research.

BIBLIOGRAPHY

[1] Acumen. http://www.acumen-language.org/, 25-03-2017.

[2] Jesper Adolfsson. Passive control of mechanical systems. *arXiv preprint arXiv:1303.2792*, 2001.

[3] Jesper Adolfsson, Harry Dankowicz, and Arne Nordmark. 3d passive walkers: Finding periodic gaits in the presence of discontinuities. *Nonlinear Dynamics*, 24(2):205–229, 2001.

[4] Aaron D Ames. Characterizing knee-bounce in bipedal robotic walking: A zeno behavior approach. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 163–172. ACM, 2011.

[5] Torleif Anstensrud. 2-d passive compass bipedal walker. Master's thesis, Norwegian University of Science and Technology, 2013.

[6] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[7] Steven H Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

[8] John J Craig. *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall, 2005.

[9] Adam Duracz. *Rigorous Simulation: Its Theory and Applications*. PhD thesis, Halmstad University Press, 2016.

[10] Goran Frehse. http://spaceex.imag.fr/sites/default/files/introduction_to_spaceex_0.pdf, 05-04-2017.

[11] Mariano Garcia, Anindya Chatterjee, Andy Ruina, Michael Coleman, et al. The simplest walking model: stability, complexity, and scaling. *J Biomech Eng Trans ASME*, 120(2):281–288, 1998.

[12] Ambarish Goswami, Bernard Espiau, and Ahmed Keramane. Limit cycles and their stability in a passive bipedal gait. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 246–251. IEEE, 1996.

[13] Ambarish Goswami, Benoit Thuilot, and Bernard Espiau. Compass-like biped robot part i: Stability and bifurcation of passive gaits. *INRIA Research Report No. 2996*, 1996.

[14] Ambarish Goswami, Bernard Espiau, and Ahmed Keramane. Limit cycles in a passive compass gait biped and passivity-mimicking control laws. *Autonomous Robots*, 4(3):273–286, 1997. ISSN 1573-7527. doi: 10.1023/A:1008844026298. URL http://dx.doi.org/10.1023/A:1008844026298.

[15] Ambarish Goswami, Benoit Thuilot, and Bernard Espiau. A study of the passive gait of a compass-like biped robot: Symmetry and chaos. *The International Journal of Robotics Research*, 17 (12):1282–1301, 1998.

[16] Jessy W Grizzle, Christine Chevallereau, Aaron D Ames, and Ryan W Sinnet. 3d bipedal robotic walking: models, feedback control, and open problems. *IFAC Proceedings Volumes*, 43(14): 505–532, 2010.

[17] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1321–1326. IEEE, 1998.

[18] Jonathan Karl Holm. *Gait regulation for bipedal locomotion*. University of Illinois at Urbana-Champaign, 2008.

[19] A. Ruina M.Coleman Mariano Garcia, A. Chatterjee. The simplest walking model: Stability, complexity, and scaling. *ASME Journal of Biomechanical Engineering*, 1997.

[20] M. Vidyasagar Mark W. Spong, S. Hutchinson. *Robot Dynamics and Control Second Edition*, chapter Dynamics, pages 187–221. John Wiley and Sons, 2004.

[21] M. Vidyasagar Mark W. Spong, S. Hutchinson. *Robot Dynamics and Control Second Edition*, chapter Rigid Motions and Homogeneous Transformations, pages 29–51. John Wiley and Sons, 2004.

[22] MATLAB & Simulink. https://se.mathworks.com/products/stateflow.html, 25-03-2017.

[23] Tad McGeer. Passive walking with knees. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1640–1645. IEEE, 1990.

[24] Tad McGeer et al. Passive dynamic walking. *I. J. Robotic Res.*, 9 (2):62–82, 1990.

[25] OpenModelica. https://openmodelica.org/useresresources/userdocumentation, 25-03-2017.

[26] R. Ortega, J.A.L. Perez, P.J. Nicklasson, and H. Sira-Ramirez. *Passivity-based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electromechanical Applications*. Communications and Control Engineering. Springer London, 2013. ISBN 9781447136033. URL https://books.google.at/books?id=jJzeBwAAQBAJ.

[27] Marko Popovic, Andreas Hofmann, and Hugh Herr. Angular momentum regulation during human walking: biomechanics and control. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2405–2411. IEEE, 2004.

[28] Wolfram Research. http://reference.wolfram.com/language/tutorial/TheSyntaxOfTheWolframLanguage.html, 17-05-2017.

[29] Cindy Schmidler. https://www.healthpages.org/anatomy-function/anatomy-terms/, 17-05-2017.

[30] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 354023957X.

[31] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2016. ISBN 354023957X.

[32] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[33] Mark W. Spong. Some new results in passivity based control of robots. *{IFAC} Proceedings Volumes*, 37(13):33 – 40, 2004. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)31197-7. URL http://www.sciencedirect.com/science/article/pii/S1474667017311977. 6th {IFAC} Symposium on Nonlinear Control Systems 2004 (NOLCOS 2004), Stuttgart, Germany, 1-3 September, 2004.

[34] Mark W Spong et al. Passivity based control of the compass gait biped. In *IFAC world congress*, volume 3, pages 19–23, 1999.

[35] Walid Taha and Roland Philippsen. Modeling basic aspects of cyber-physical systems. *arXiv preprint arXiv:1303.2792*, 2013.

[36] Mattias Wahde and Jimmy Pettersson. A brief review of bipedal robotics research. In *Proceedings of the 8th UK Mechatronics Forum International Conference (Mechatronics 2002)*, pages 480–488, 2002.

[37] E. R. Westervelt, J.W. Grizzle, and D. E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48:42–56, 2001.

[38] Fuminori Yamasaki, Koh Hosoda, and Minoru Asada. An energy consumption based control for humanoid walking. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2473–2477. IEEE, 2002.

[39] Yingfu Zeng, Chad Rose, Paul Brauner, Walid Taha, Jawad Masood, Roland Philippsen, Marcia OMalley, and Robert Cartwright. Modeling basic aspects of cyber-physical systems, part ii. In *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on*, pages 550–557. IEEE, 2014.

[40] Yingfu Zeng, Rose Chad, Walid Taha, Adam Duracz, Kevin Atkinson, Roland Philippsen, Robert Cartwright, and Marcia O'Malley. Modeling electromechanical aspects of cyber-physical systems. *Journal of Software Engineering for Robotics*, 7(1):100–119, 2016.

DECLARATION

Put your declaration here.

Halmstad, Sweden, Mai 2017

Florian Joachimbauer, June 26, 2017

.

HALMSTAD UNIVERSITY