AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Praca dyplomowa

Walking Control Algorithms of Bipedal Robot - Case Study
Algorytmy sterowania chodem dwunożnego robota kroczącego - studium przypadku

Autor: *Hubert Milanowski*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr hab. inż. Adam Piłat, prof. uczelni*

Kraków, 2021

Contents

1. Introduction

1.1. Legged Robotic

Service robots are likely to take over a significant part of human activities soon. This facility is mainly addressed to the ageing society issue - for instance, Toyota Research Institute on MIT works intensively on that [1]. To enable the elderly or people with disabilities to live independently, service robots must be fully integrated into the human environment. They must have the ability of robust gait. Walking on uneven terrain is critical here. Despite many advancements in humanoid robotics in recent years, walking is a significant bottleneck for the robotics industry. A look at the recent DARPA Robotics Challenge shows how humanoid robots are still incapable of walking robustly even on flat terrains [2].

Bipedal walking is one of the complex control problem posed in recent years. In general, there is four crucial factors which makes it such a challenge:

- **Non-linear dynamics:** The gait of the biped is globally unstable and non-linear. Linear control techniques do not guarantee stability.
- **Hybrid dynamics:** Each gait cycle consists of at least two phases: ***double support phase***, when the robot touches the ground with both feet and ***single support phase***, when only one foot touches the ground, and the other is swung. The transition between states is tough to model mathematically.
- **Multivariable system:** The state-space model of the object is of a high order.
- **Model uncertainty:** Model convergence with the real object is challenging. Changing operating environment hinders the design of the robust controller.

1.2. Problem Statement

As mentioned in the previous section, listed obstacles impose that the bipedal control issue is still an unresolved topic. There are promising studies related to the intelligent control algorithms, such as imitation learning [3] or reinforcement learning [4]. However, these results are mainly simulation ones, which are not validated with actual experiments. In Reinforcement Learning, when an agent learns from a simulation, we revisit the initial problem of the mathematical model accuracy and uncertainty. In addition, in machine learning methods used for control, the problem is their explainability, making it challenging to study stability. Therefore, according to the author, it is reasonable to consider synthesising both the classical and the intelligent control methods to develop a novel approach. Therefore, this work will introduce both classical control theory and intelligent control concepts. Presented thesis shall be treated as preliminary, which may hopefully be extended in the future.

As part of the research on walking robotics, Student Scientific Association Focus members, operating at the Robotic, Photovoltaic, and Magnetic Levitation Laboratory, have designed and manufactured the prototype of a planar bipedal robot. To the best of author's knowledge, AGH University of Technology in Cracow is the first research centre in Poland that undertakes topic.

1.3. Research Objectives

The following research topics are addressed:

- If the MRAC reference model inspired by DC motor system is sufficient to provide satisfactory control of bipedal robot leg?
- What constraints have to be given to the MPC optimiser to minimise the duration of the time horizon for the gait trajectory generation task?
- Can a highly unstable walking robot be taught to walk using Reinforcement Learning?

1.4. Thesis Outline

The remainder of the thesis is structured as follows:

- Chapter 2 presents results for implementing the Simulink model of bipedal robot. The chapter also discuss conducted identification experiments.
- Chapter 3 introduces model-reference adaptive control approach utilized to rotary position servo control task of bipedal leg's joints.
- Chapter 4 reviews Model Predictive Control approach used for gait pattern generation.
- Chapter 5 provides a formal introduction to RL by explaining the mathematical preliminaries associated with it. It presents results for implementing the particular DRL algorithm.
- Chapter 6 summarize results of all presented approaches.

2. Bipedal Robot Simulation Model

2.1. Mechanical Design of Bipedal Robot

This section describes the mechanical design of the bipedal robot, which is considered in the remainder of the thesis. The CAD model and the photography of the robot prototype are shown in Figure 2.1.

Fig. 2.1. On the left CAD model, on the right manufactured prototype

The following assumptions were made during the development of the project concept:

- The design provides for the weight of the robot to be concentrated at the top of the platform - similar to the design of the Cassie robot by Agility Robotics [5]. The actuators responsible for joint movement are mounted to the platform.
- The dimensions of the structure were determined on the basis of the endoskeleton of a 6-8 year old child.
- Each leg has 2 degrees of freedom - joints responsible for flexion/extension move of hip and knee joint.
- Brushless direct current motors - Gyems X8 Pro - actuates joints.
- Motor torque transmitted to the knee joint via pulley belt drive.
- Designed prototype components made using cavity technology and 3D printing.

Although, the whole design is conceptual, the Student Research Group Focus at AGH has already developed a prototype of design. The project has been built as part of the engineering projects of students from the Faculty of Mechanical Engineering [6], [7].

2.2. Simulation Model Implementation

The simulation model of the robot used in the following work is implemented using the rigid body dynamics simulator - Simulink and SimScape MultiBody Toolbox. In order to reach high convergence of the simulation model with the object, geometries modelled with the CAD software were used. A simplified simulation model of a bipedal robot has been implemented as part of the undergraduate thesis. The results of the study may be found at [8]. The exact information about the dynamics of the system based on the real geometrical dimensions, masses and moments of inertia of individual structural elements makes it possible to reach high convergence of the simulation model with the physical object response.

Figure 2.2 depicts the top-level simulation block diagram. ***Bipedal Robot*** subsystem consists of three submodules: ***Legs*** and ***Platform*** subsystem, ***Foot Contact Model*** subsystem and ***Sensing Tool*** subsystem. At the bottom, three blocks are in charge of the overall configuration of the simulation model. The ***Solver Configuration*** blocks defines numerical solver options. The ***World Frame*** is the initial frame in a MATLAB Simcsape Multibody mechanical model. ***Mechanism Configuration*** sets parameter like gravity. Figures 2.3 and 2.4 present block diagram view of ***Bipedal Robot*** and ***Leg*** subsystem respectively.

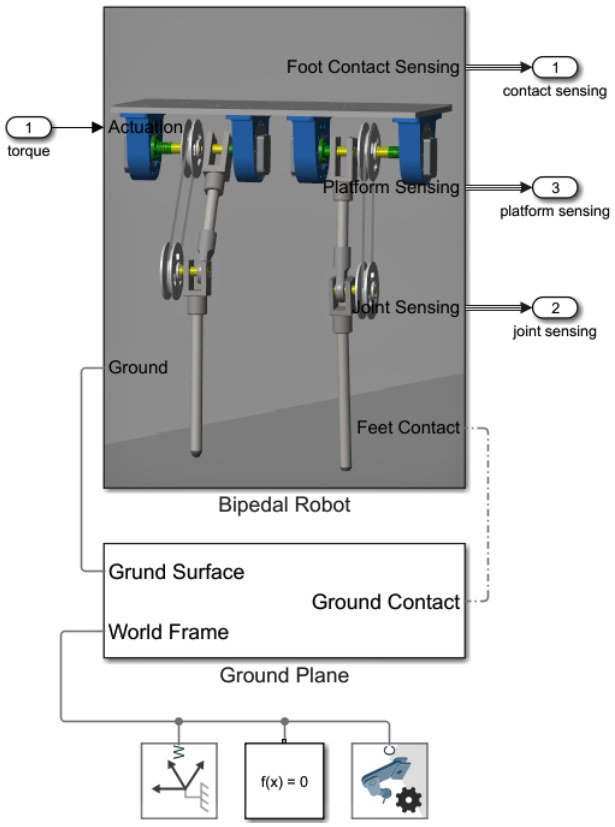

Fig. 2.2. MATLAB Simscape block diagram of Bipedal Robot Simulation - top level view

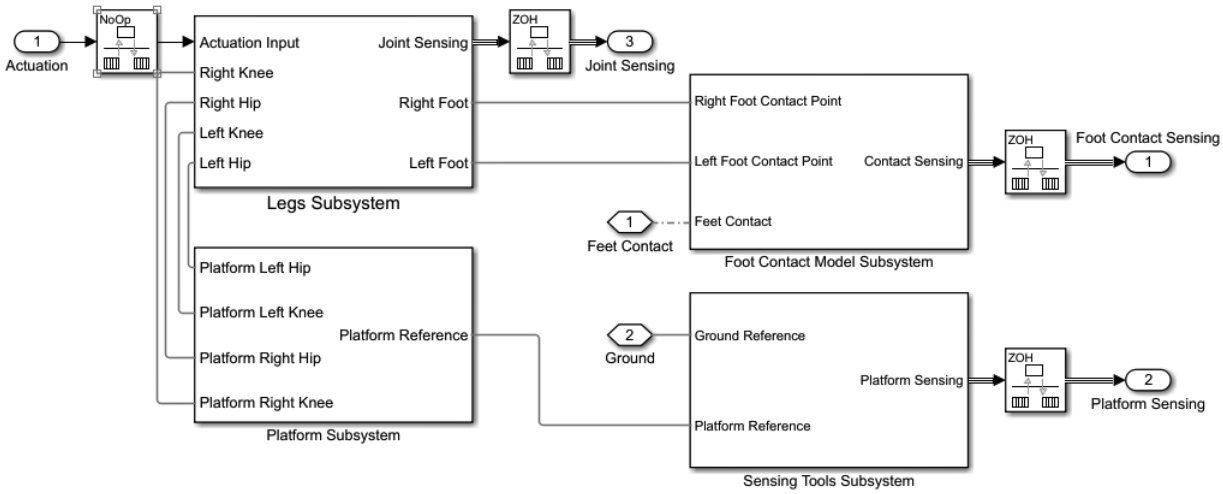

Fig. 2.3. MATLAB Simscape block diagram of Bipedal Robot subsystem

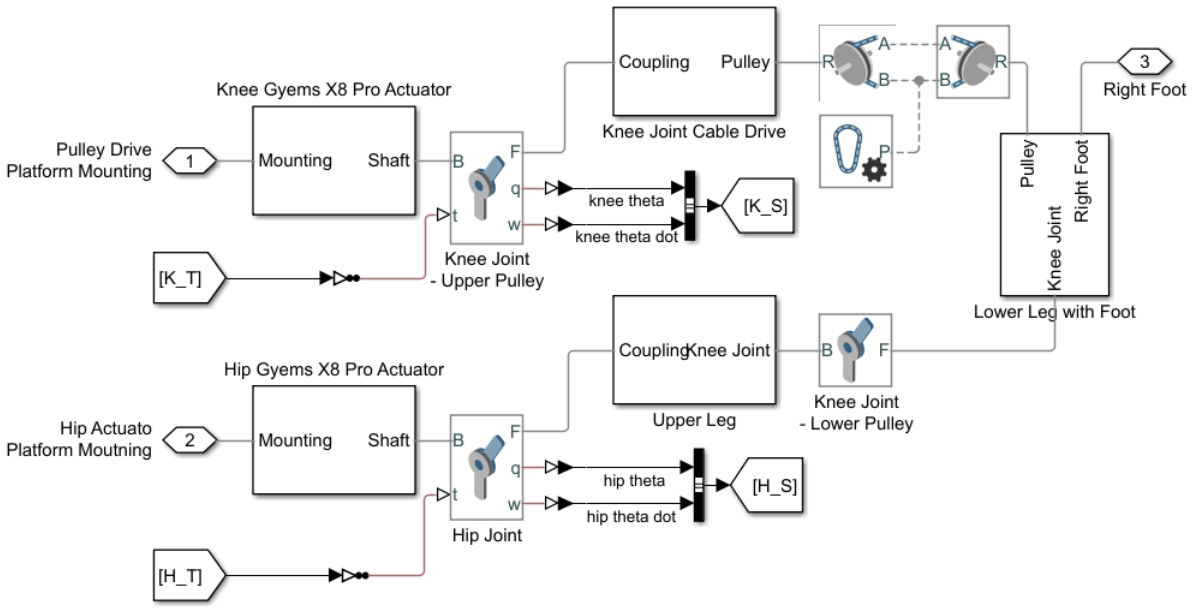

Fig. 2.4. MATLAB Simscape block diagram of Bipedal Robot subsystem

Simulink blocks, such as ***File Solid*** and ***Rigid Transform***, are used to determine the geometry of the bipedal leg like the real one. To model the biped leg, we use two ***Revolute Joint*** blocks, which provide one rotational degree of freedom. By tuning the joint stiffness and damping coefficient for the energy dissipation, we improve the convergence of the simulation model with the physical one. The parameter estimation process will be discussed in the next section.

Regarding the actuation option of ***Revolute Joint*** blocks, there are two configuration modes - providing torques as input and computing motion or providing the motion as input and computing torque. For this simulation, we choose to apply the torque signal as the input. In a later stage of the work, we will control the angular position of the joint using a torque signal. Consequently, at this stage of the work, we abandon the dynamics of the actuator. In the real case, the control signal would not be given as torque but rather as a voltage or PWM signal. In order to compensate for the lack of consideration of the BLDC motor dynamic, the damping coefficient of the joint block has been selected based on an identification experiment waveforms.The control signal applied to the BLDC motor controller is the torque value and the output signal is the angular position measured by the encoder. In order to prevent the simulation model from unphysical behaviour, torque signal has been constrained using the saturation block up to the value specified in the manufacturer's technical documentation [9].

The belt-cable drive is modelled using ***Pulley*** and ***Belt-Cable Properties*** blocks. The assumed model of movement is linear. Identification experiments prove that such a V-belt cable drive introduces nonlinearities that can be a source of uncertainty in the simulation model.

To model the contact between the ground and the foot, we use a ***Spatial Contact Force*** block - as Fig. 2.5 depicts. ***Spatial Contact Force*** block configuration requires properties such as normal and friction force coefficient, stiffness and damping. The stiffness parameter defines

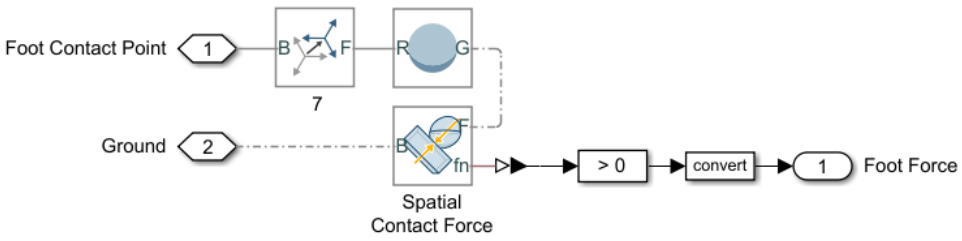

Fig. 2.5. MATLAB Simscape block diagram of Foot Contact Model subsystem

the range to which an object resists deformation in response to an applied force. The damping parameter expresses to oppose the relative motion between two interacting surfaces. A static friction coefficient, a dynamic friction coefficient, and a critical velocity must be defined within the frictional force. The static friction coefficient depends on the materials in contact. For the dynamic friction coefficient lower value than the static coefficient must be used. The critical velocity works as a threshold determining whether the static or dynamic friction coefficient we should use. Since there is no research facility to determine mentioned above parameters of the prototype under consideration, we decided to apply values from the exemplary MathWorks simulation - *Running Robot Model in Simscape* [10], [11].

At this stage of the research, a single plane motion model - the sagittal one - is assumed. For this purpose, the Planar Joint block is used, which combines the ***World Frame*** and the ***Bipedal Platform Frame***. This configuration raises the issue of modelling robotic structures to a floating-base approach. Solver called ***daessc*** was used for the simulation [12]. The ***daessc*** variable-step Simulink solver is designed specifically for physical modeling. Table 2.1 presents adopted solver configuration parameters.

Table 2.1. Solver Configuration Parameters

Solver Configuration Parameter	Value
Max step size	0.05 [s]
Min step size	0.01 [s]
Initial step size	0.025 [s]
Relative tolerance	1e-3

2.3. Identification of Parameters of the Bipedal Robot Simulation Model

The identification experiment has been carried out for a biped's leg moving without contact with the ground. The choice of the experimental methodology is due to the lack of hardware resources enabling the identification of a robot with a floating base. As an excitation trajectory, the waveforms recorded as part of the work described in the BSc thesis [13] were used. The trajectory in question is a man's gait on a treadmill, recorded with a high-speed camera. It is worth nothing that the indicated trajectory enabled a successful identification of parameters for the current configuration of the robot - a planar one. When thinking about configurations with a more significant number of degrees of freedom - the transition from a two-dimensional into a three-dimensional workspace - it is necessary to determine a trajectory that allows the parameters of the new system to be fully identified. The subject is well covered in [14].

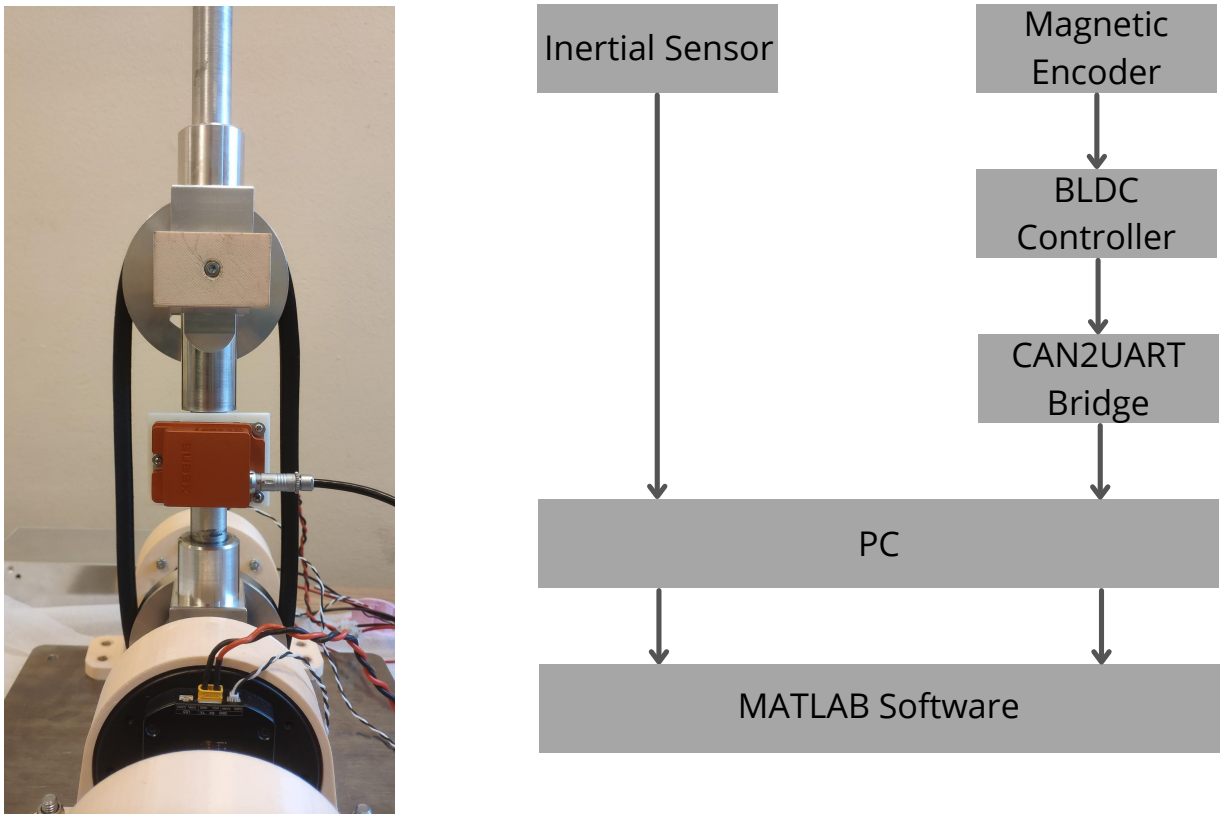

Fig. 2.6. On the left a block diagram of the test stand, on the right a photo during the experiments

Figure 2.6 shows the schematic and photo of the test rig set up for the identification experiment, respectively. A Magnetic encoder mounted on the BLDC motor shaft and an inertial sensor - Xsens MTi-7 were used for data acquisition.

The Xsens MTi-7 sensor was provided by the EAIiIB Faculty Interdisciplinary Research Laboratory [15]. The BLDC motor controller records parameters such as motor shaft angular position, angular velocity and torque current. The controller has been communicated with a PC using CAN2Serial interface bridge. The availability of only one piece of inertial sensor forced to repeat the experiment twice - for the case with the sensor mounted in the upper and lower position of the leg. The sensor mounting points were set as the leg centre of gravity calculated with CAD software. The experiments were performed for the position control configuration.

Figure 2.7 shows a graph of the recorded experimental waveforms for the hip joint. The plot at the top compiles the angular position values recorded with the encoder and the inertial sensor, respectively. The bottom chart shows the torque produced by the motor. Comparable data are shown in graph 2.8 of the lower part of the leg measurements. In the case of the waveforms for the upper part of the leg, the convergence of the signal measured by the encoder and inertial sensor indicates good stiffness of the system. For the lower part, significant discrepancies in the angular position signal are observed, which indicates the occurrence of belt slippage.

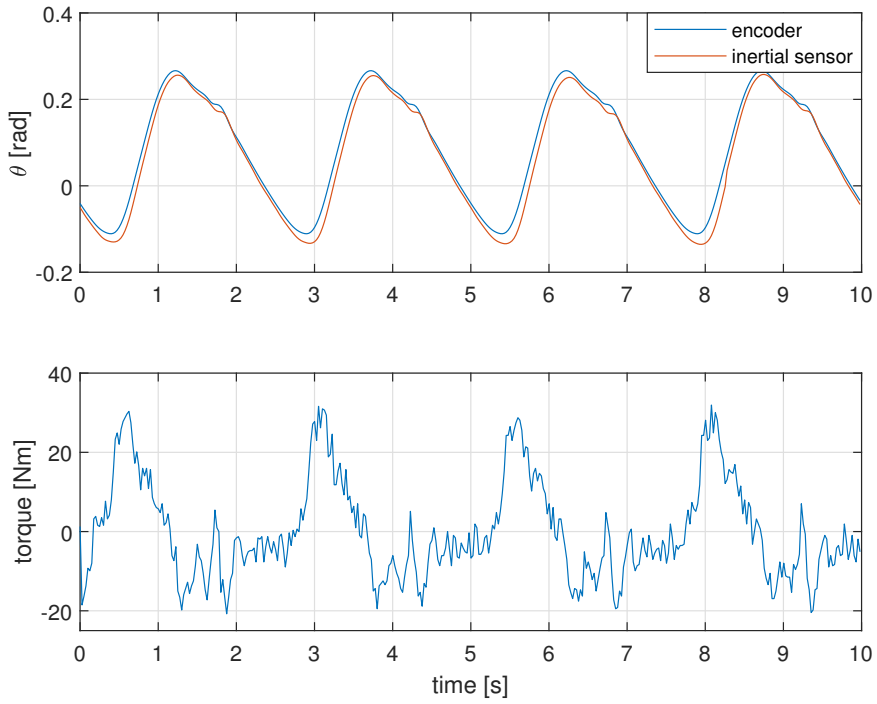

Fig. 2.7. MATLAB Simscape block diagram of Foot Contact Model subsystem

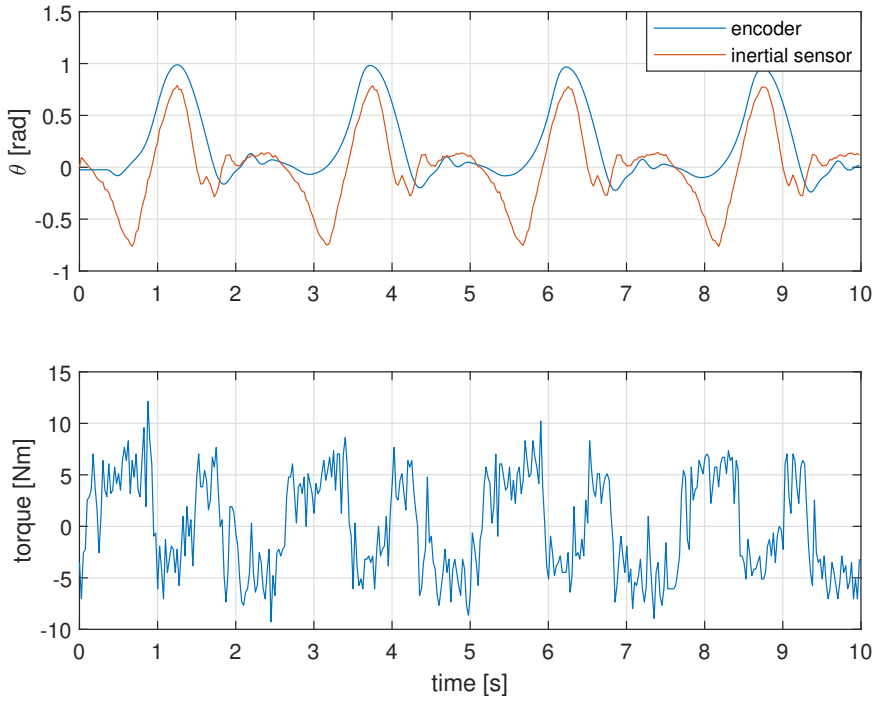

Fig. 2.8. MATLAB Simscape block diagram of Foot Contact Model subsystem

During the experiment, there has been a malfunction. Element transmitting torque from the motor shaft to the pulley was damaged - Figure 2.9 shows a preview of the destroyed element.

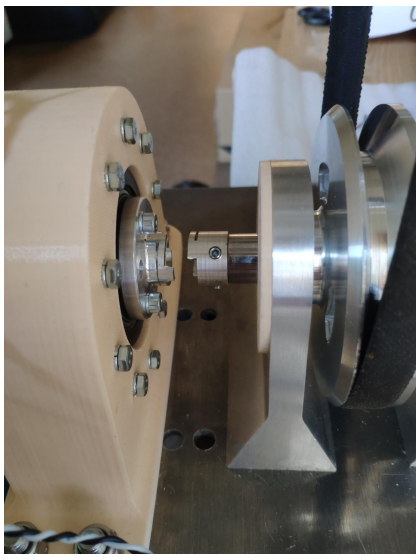

Fig. 2.9. The damaged torque transmission part

2.4. Summary

The simulation model will be used for the research on the adaptive and intelligent control algorithms. It is also a good basis for the further development of the issue of walking robotics. Regarding the recorded waveforms, it is found that the methodology proposed for identification experiment is acceptable at this point. On the other hand, the flaw noted on this basis - belt slippage - indicates faulty design assumptions that should be reconsidered.

3. Adaptive Control Problem

3.1. Introduction

Considering the control of physical systems, engineers utilize model-based design in most cases. They need to meet the expectation to derive such a mathematical model of the system that it will be converged with the response of the real one. It raises the question, whether it is possible to provide the required information in the mathematical model. There is always a risk that some model uncertainty may occur.

For this reason, control scientists investigate techniques that can adjust control design parameters such as online adaptation of control gain based on the inputs received by the plant to accommodate system uncertainty. The adjustable parameters are called adaptive parameters. The adjusting mechanism, which is described by a set of mathematical equations, is called an adaptive law [16].

This section will discuss one of the adaptive control techniques - model reference adaptive control (MRAC). Using this approach, we shall control the joints' rotary position of the 2-link serial robot - freely moving bipedal robot leg. In the first part, we will discuss the MRAC approach. Next, the implementation of the control algorithm will be presented. Finally, simulation experiments will be discussed.

3.2. Model Reference Adaptive Control

The model-reference adaptive control is one of the adaptive control technique. Desired performance is expressed in respect of a reference model, which gives the desired response to the command signal.

Figure 3.1 presents a typical MRAC block diagram. The system has a conventional feedback loop composed of the controller and process - called inner loop. The second loop - the outer loop - adapts the controller parameters. The reference model is essentially a command shaping filter to achieve the desired command following. Parameter adjustment is based on feedback from the error signal, which is the difference between the output of the process and the reference

system. The adjustment mechanism in a model-reference adaptive control can be introduced in two ways: using a gradient method or following stability theory. In this thesis, we focus on the gradient method, especially the MIT rule [17].

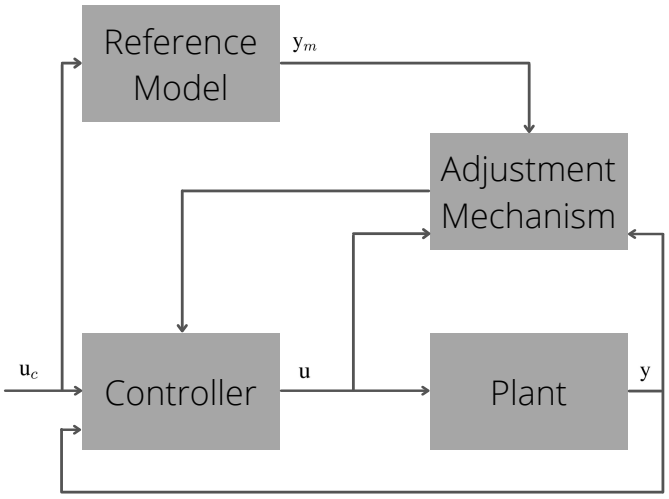

Fig. 3.1. Block diagram of a model-reference adaptive controller

The MIT rule utilizes error feedback based on the difference between system output and the reference model output. Let us assume that our adjustment law consists of one adaptation parameter θ, responsible for adapting the control signal. In that case, we form the loss function describing the error concerning the parameter.

$$J(\theta) = \frac{1}{2}e^2 \tag{3.1}$$

We adjust the parameter in such a way that the loss function is minimized. To do that, we ought to change the parameter in the direction of the negative gradient of a loss function.

$$\frac{d\theta}{dt} = -\gamma\frac{\partial J}{\partial \theta} = -\gamma e\frac{\partial e}{\partial \theta} \tag{3.2}$$

The partial derivative $\frac{\partial e}{\partial \theta}$ expresses how the adjustable parameter influences the error.

3.3. Controller Design

To implement the model-reference adaptive controller for a 2-link serial robot, we utilize the control structure as presented in Figure 3.2. Furthermore, by considering signal filtering in the controller, we protect the system from sensitivity derivatives and limit cycle behaviour.

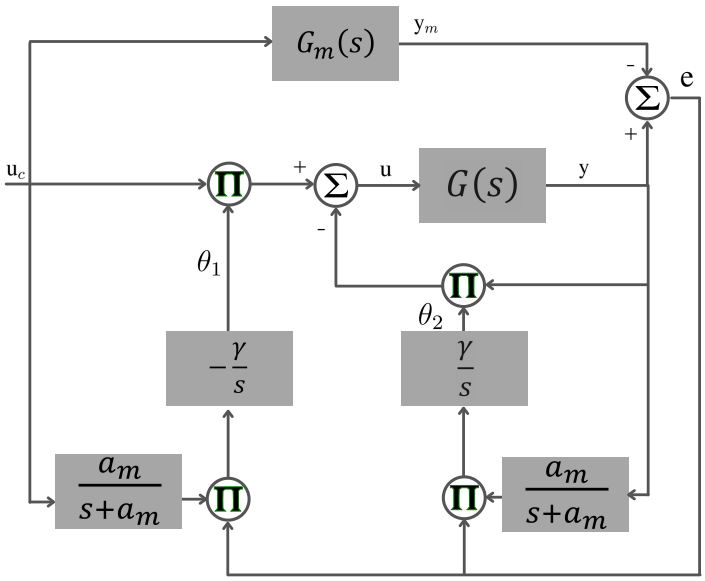

Fig. 3.2. Control structure of 1st order MRAC system, graphics based on [16]

A reference model is used to specify the ideal response of the adaptive control system to the external command. It provides the ideal plant response, which the adaptation mechanism should seek in adjusting the parameters.

The reference model was selected based on identification experiments of an unloaded BLDC motor. For the task of controlling the angular position of the motor shaft, the response waveforms of the real object, respectively both angular position and speed of the motor shaft, and torque current were recorded. The experiment was carried out for a step signal with a value of one radian. Figure 3.3 shows the recorded response of the real system.

Using MATLAB Identification Toolbox, a transmittance model of the object was determined. The transmittance model with two poles and one obtained the best fit with the response of the real system - a fit of 98%. In theory, we can consider that this object acts like a DC motor operating in closed-loop with a PI-type compensator for the velocity control task. Figure 3.4 shows the response of the identified and the real object. The transmittance equation of the system is as follows

$$G(s) = \frac{7.254s + 821}{s^2 + 47.76s + 818} \tag{3.3}$$

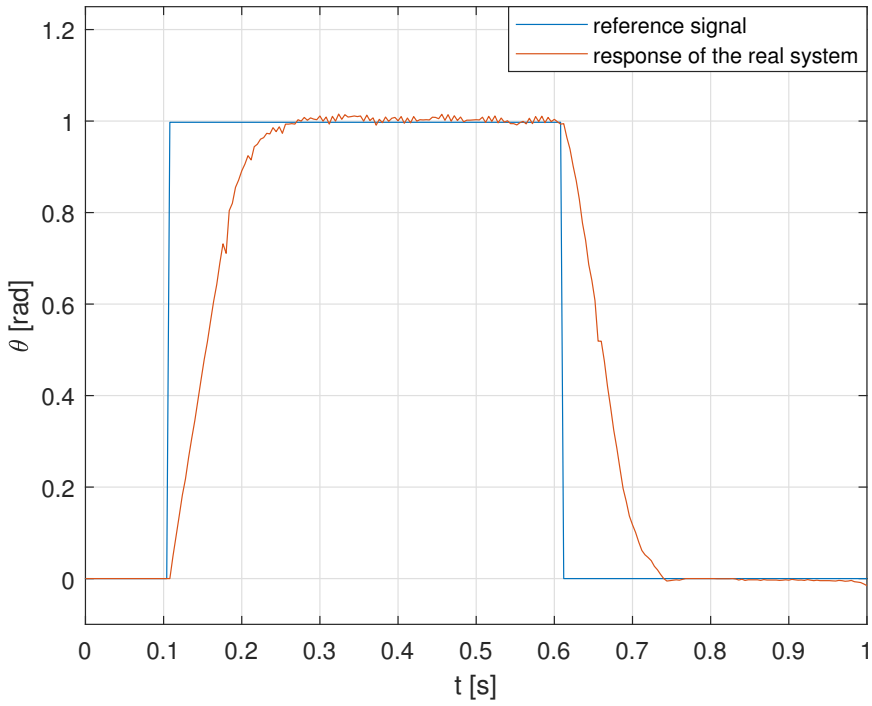

Fig. 3.3. Response of the real object recorded during the identification experiment

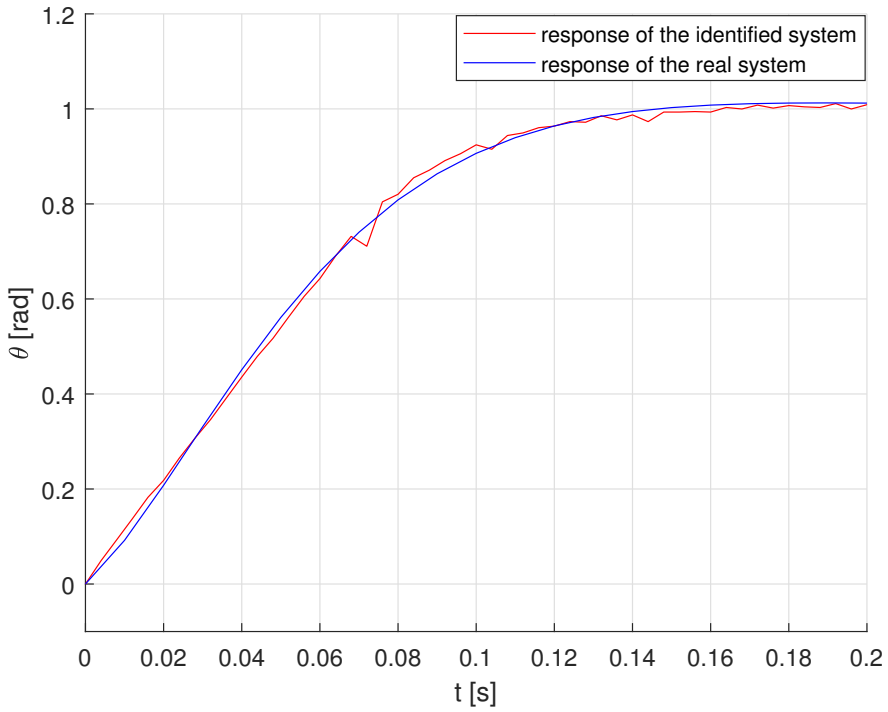

Fig. 3.4. Comparison of the responses of the identified and the real object

The Table 3.1 summarises information on the actual gain parameters set in the BLDC motor controller and step response characteristic parameters. Figures 3.5 and 3.6 show the details of the model-reference adaptive controller implementation in the Simulink.

Table 3.1. Value of controller gains and the desired response of the reference system for bipedal leg hip joint

Parameters		Value
Controller	P	120
	I	30
Performance	Rise time	0.08 [s]
	Settling time	0.13 [s]
	Overshoot	0.05

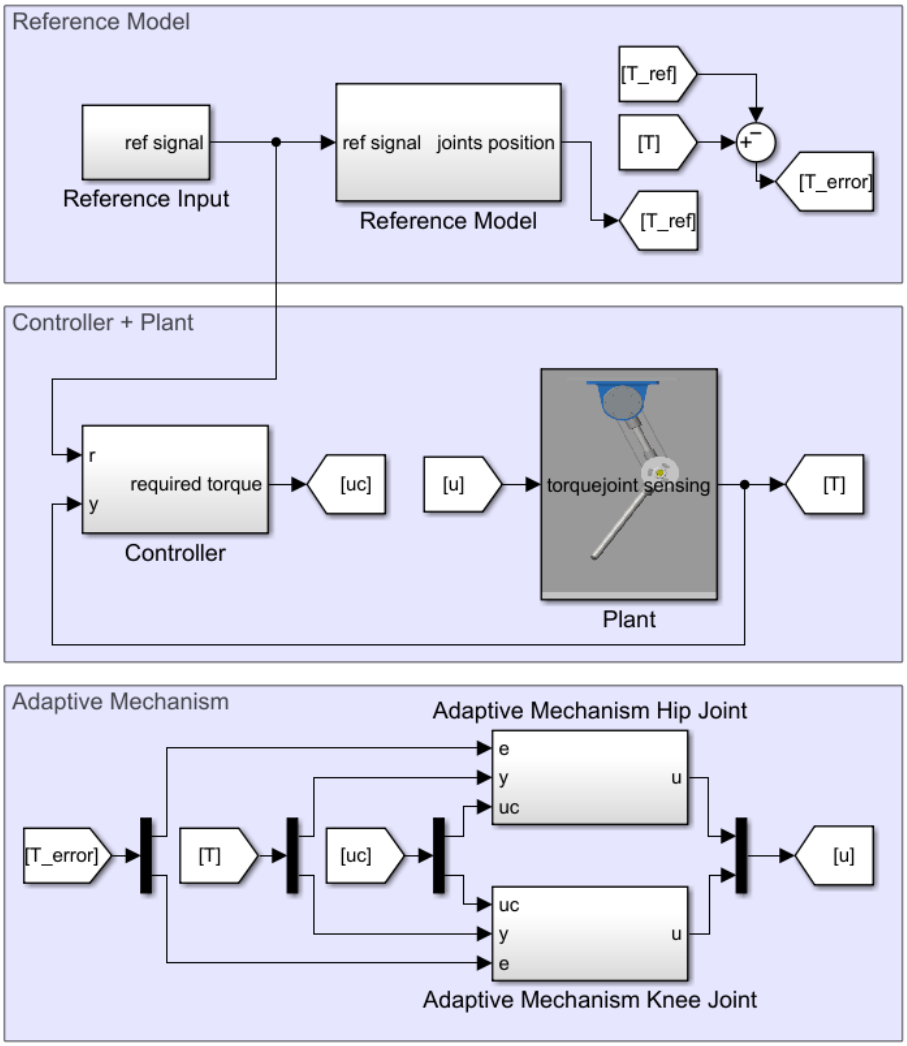

Fig. 3.5. Simulink block diagram of model-reference adaptive controller

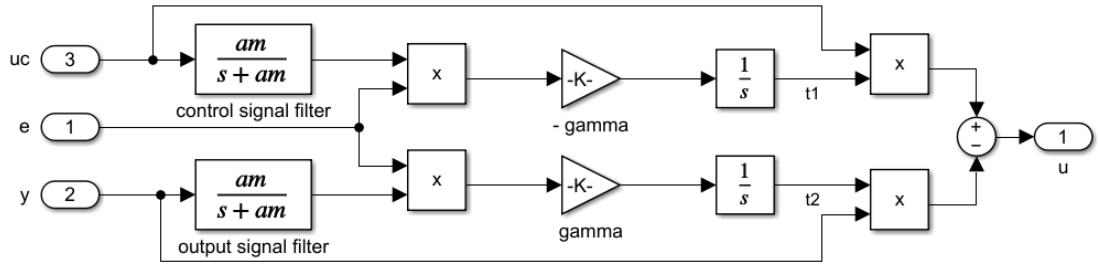

Fig. 3.6. Simulink block diagram of an adaptation mechanism

3.4. Simulation Experiment Results

A series of numerical experiments are carried out to investigate the influence of the learning parameter on the control quality. The reference signal is the trajectory of the angular position of the joints during human gait recorded as part of the research for the B.Sc. thesis [13], Figure 3.7 shows one gait cycle. Figures 3.8 and 3.11 shows the system's responses for the hip and knee joint, respectively, depending on the value of the learning rate γ. Graphs 3.9 and 3.12 shows the error occurring between the response of the real object and the reference model depending on the γ parameter. The 3.10 and 3.10 diagrams compare the adaptive controller's calculation of the desired control signal value for the hip and knee joint case under consideration, respectively. The convergence rate increases with increasing γ, as is shown in Figure 3.8. Increasing the value of gamma causes the system to adapt more quickly, but if the γ is too large, the system might be very sensitive and unstable.

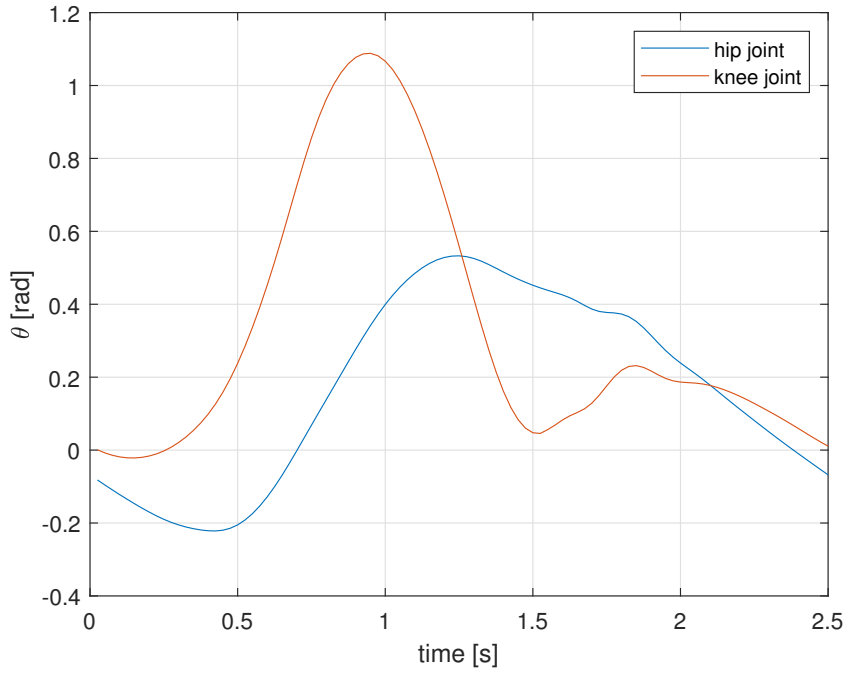

Fig. 3.7. Reference trajectory

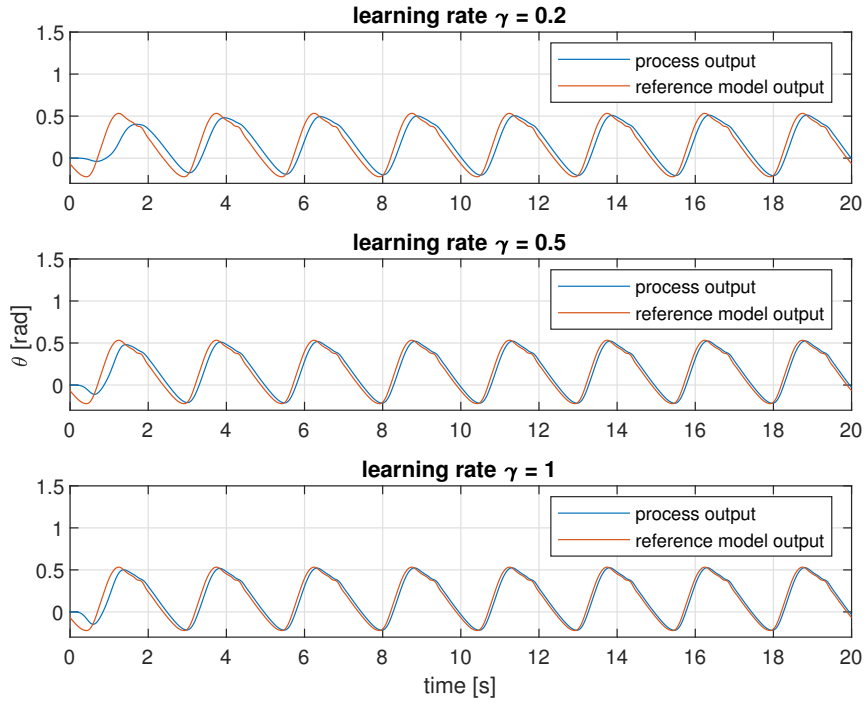

Fig. 3.8. Comparison of the system response concerning the reference trajectory depending on the learning parameter γ - hip joint

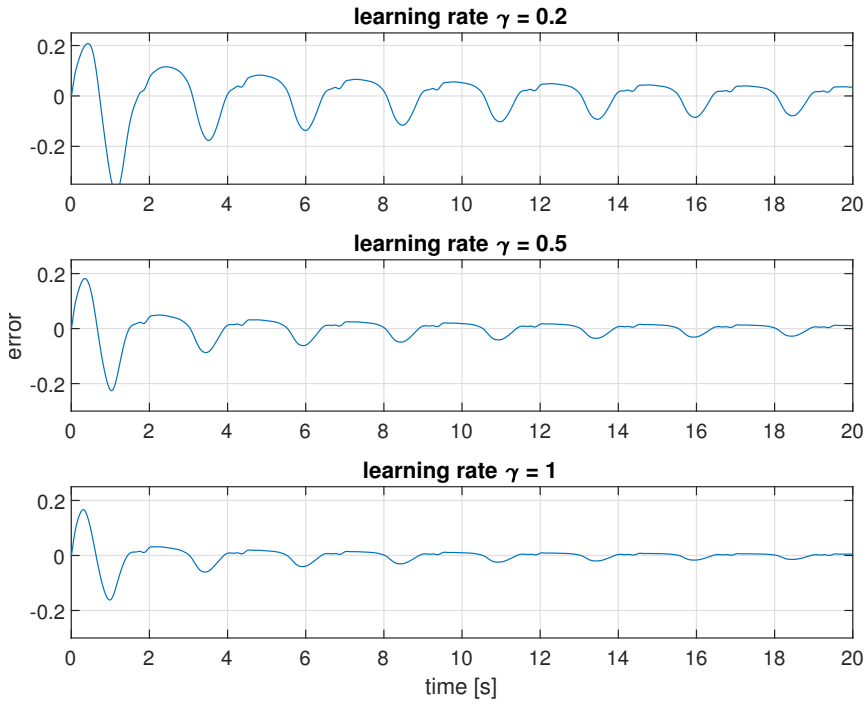

Fig. 3.9. Comparison of the difference of the output signal of the plant and the reference model depending on the the learning parameter γ - hip joint

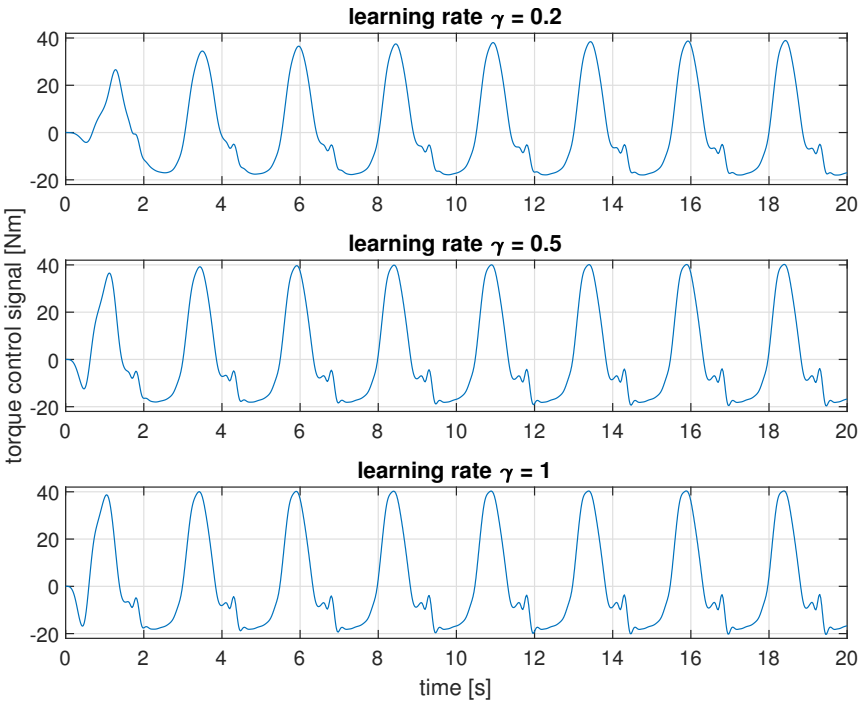

Fig. 3.10. Comparison of the control signal value of the object and the reference model depending on the learning parameter γ - hip joint

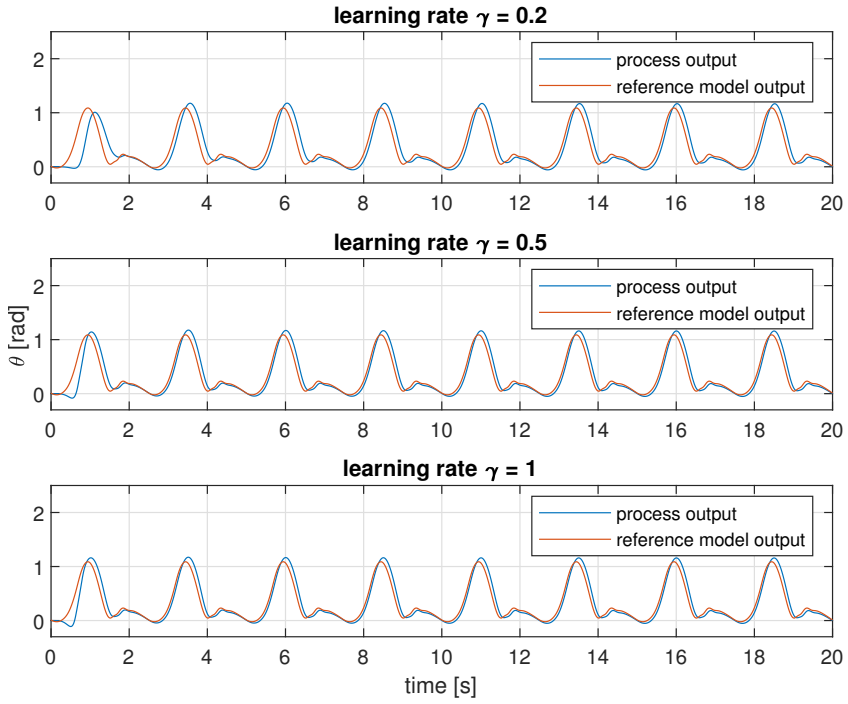

Fig. 3.11. Comparison of the system response concerning the reference trajectory depending on the learning parameter γ - knee joint

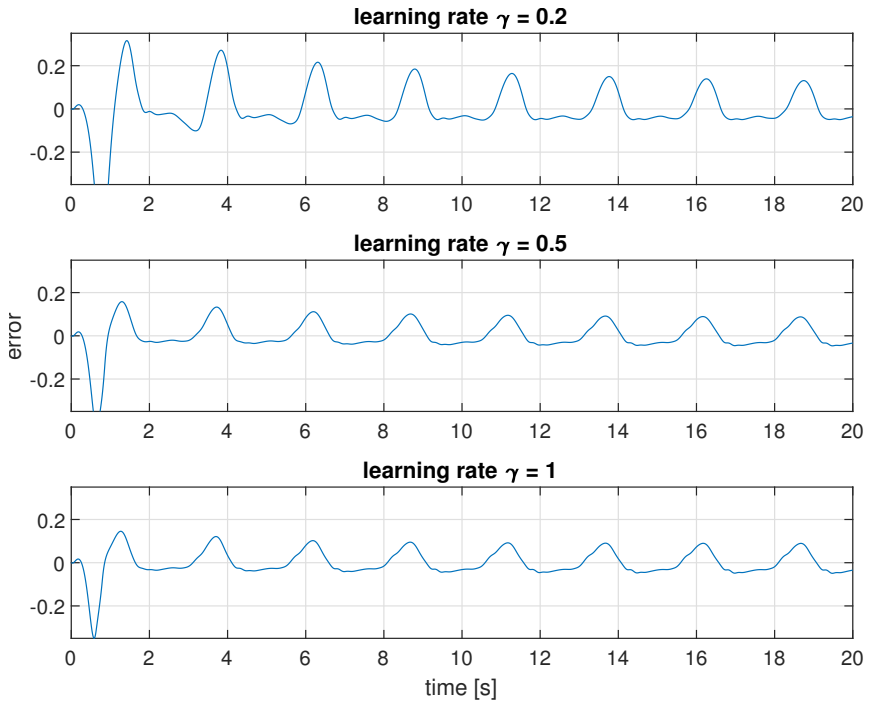

Fig. 3.12. Comparison of the difference of the output signal of the plant and the reference model depending on the the learning parameter γ - knee joint

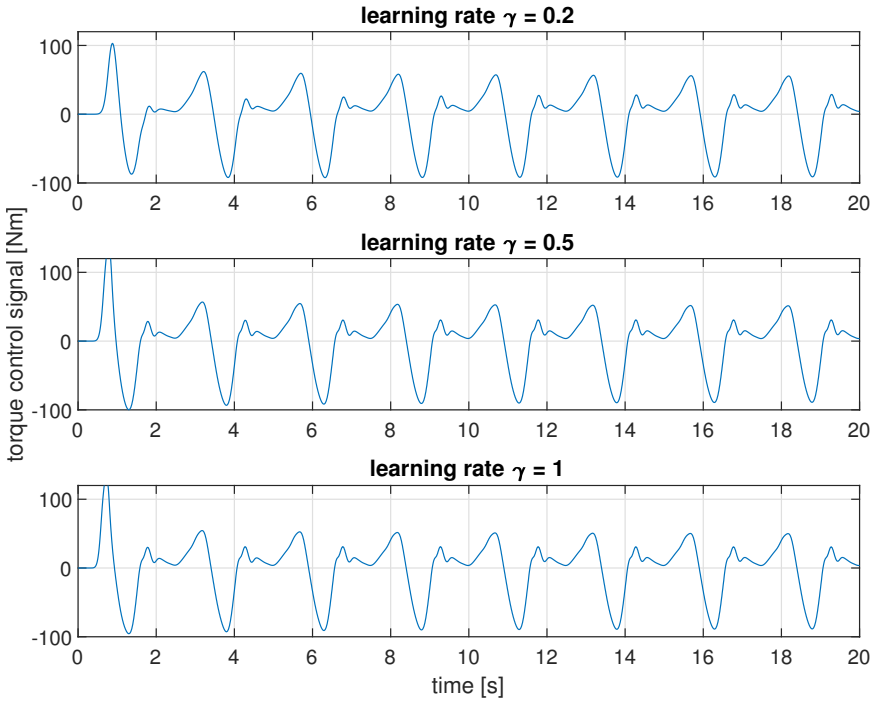

Fig. 3.13. Comparison of the control signal value of the object and the reference model depending on the learning parameter γ - knee joint

4. Model Predictive Control for Bipedal Gait Generation Problem

4.1. Introduction

The previous chapter deals with the adaptive control concept for the servo control task applied to control the bipedal robot leg position. Having the known leg's joints trajectory, the controller adjusts the control parameters to achieve the desired performance. Let us now consider the generation of trajectories for a biped in the joint or operational space. If these trajectories are generated offline, the motion cannot adapt to the walking conditions. Therefore, a stable gait can be hindered. A promising approach is thus to generate a motion trajectory online. This can be achieved using the Model Predictive Controller technique (MPC) [18], [19]. As pointed out in the previous chapter, applying a complete model of a bipedal robot gives rise to a highly nonlinear dynamic model. Real-time control based on Nonlinear Model Predictive Control (NMPC) is not always feasible since its application is a computationally expensive task. These limitations motivate researchers to develop a controller other than NMPC, use simple Linear Model Predictive Control (LMPC) instead, that still maintain satisfactory control quality.
This chapter reviews the gait generation approach called Intrinsically Stable MPC [20], [21]. MATLAB Implementation of MPC controller will be considered. Further, the impact of constraints on the quality of control will be discussed.

4.2. 3D Linear Inverted Pendulum

The dynamics of a bipedal robot walking on flat ground can be approximated by a linear model, called Three-dimensional Linear Inverted Pendulum Model (3D-LIPM) [22]. The 3D-LIPM is a simplified model expressing the motion of the CoM of a biped, assuming its height is kept constant, and it is no rotational effects. An additional advantage of the 3D-LIPM model is the decoupling and identicality of the differential equations governing the dynamics along with each ax. Numerous studies confirm that gait trajectory solved by the LIP model is efficient despite the simplicity.

Let us derive the dynamic equation for the single-axis pendulum as an illustration. The LIPM dynamics is expressed as

$$\ddot{x}_c = \omega^2(x_c - x_z) \tag{4.1}$$

x_c and x_z are the coordinates of the *CoM* and *ZMP* respectively, $\omega = \sqrt{g/h_{CoM}}$, h_{CoM} is the constant height of the *CoM*. Figure 4.1 illustrates mathematical equation parameters.
A system with eigenvalues located on both sides of the complex plane can be decomposed into two subsystems, one stable and the other unstable.

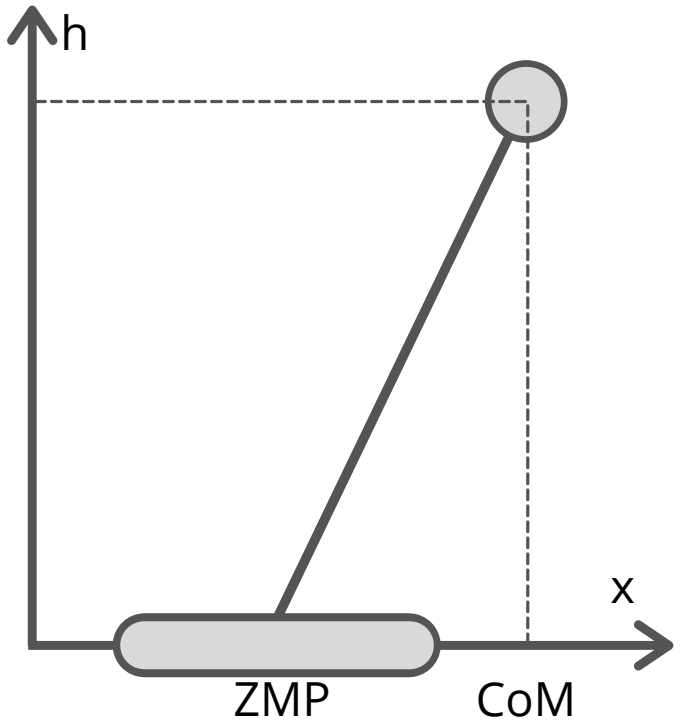

Fig. 4.1. The Linear Inverted Pendulum in the X direction

"Even if the system is unstable, for any input(i.e., for each ZMP trajectory $x_z(t)$) there exists a special initialization of x_u such that the resulting CoM trajectory is stable, in the sense that it does not diverge w.r.t the input itself. This initialization is expressed as

$$x_u(t_{in}) = x_c(t_{in}) + \dot{x}_c(t_{in})/\omega = x_u^*(t_{in}) \tag{4.2}$$

where t_{in} is the initial instant and

$$x_u^*(t_{in}) = \omega \int_{t_{in}}^{\infty} e^{-\omega(\tau - t_{in})} x_z(\tau) d\tau \tag{4.3}$$

Note that $x_u^*(t_{in})$ depends on the whole history of the control input, i.e., on the whole ZMP trajectory. The initialization of x_s is obviously irrelevant. In the following, we shall refer to (4.2-4.3) as the stability constraint." [20]

4.3. Model Predictive Control Problem Formulation

MPC is a collection of control algorithms widely used in industry. With respect to other control strategies, it is distinguished by two key features: it is based on the knowledge of the model of the process, and the behaviour of the system is predicted over some preview horizon. The goal of MPC is to choose a sequence of control inputs for the system over prediction horizon with respect to given objective function and constraints. The problem is resolved periodically in order to accommodate for the current situation - control algorithm operates in the open-loop. The preview window is usually shifted in time on each iteration of MPC.

This chapter reviews some novel approach discussed by Nicola Scianca et. all called Intrinsically Stable MPC (IS-MPC).
The mathematical model of considered process takes the following equation

$$\begin{pmatrix} \dot{x}_c \\ \ddot{x}_c \\ \dot{x}_z \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ \omega^2 & 0 & -\omega^2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ \dot{x}_c \\ x_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{x}_z \tag{4.4}$$

The authors of the paper assumed two constraints to be taken into account resolving quadratic programming optimization problem. *"The first concerns the ZMP position, which must be at all times within the support polygon defined by the footstep sequence and the associated timing. The second is the stability constraint needed to guarantee that the CoM trajectory generated by our MPC scheme will be stable; this constraint must be satisfied at the start of the prediction horizon."* [20]

In discussed MPC framework, ZMP and stability constraints act on both x_z and y_z coordinate of the ZMP. As a cost function, the sum of the squared norms of the control variable $\dot{x}_z$ and $\dot{y}_z$ is used. The Quadratic Programming (QP) problem is given by

$$\min_{\dot{x}_z, \dot{y}_z} ||\dot{x}_z||^2 + ||\dot{y}_z||^2 \tag{4.5}$$

subject to ZMP and stability constraints.

4.4. Experimental Results

In this section, we present some MATLAB simulations on the LIP that highlights the key features of the reviewed approach. In particular, we compare the intrinsically stable MPC technique and the standard MPC of [23]. The latter uses the CoM jerk as a control variable (and in the cost function) and does not include stability constraints, while the ZMP constraints are the same. The simulated model is with h_{CoM} = 0.6 m. A sequence of evenly spaced footsteps is assigned to be attained every 0.5 s, of which 0.2 s in double support and 0.3 s in single support. The foot is represented as a square with a side of 0.1 m. In MATLAB implementation, we

assume that discretization quanta is equal 0.01 s. The MPC optimization QP problem is solved with the *quadprog* function, which uses an interior-point algorithm.

Figure 4.2 shows results of both MPC approaches. The prediction horizon time is equal 2 seconds, i.e., four steps ahead. For such configuration only IS-MPC produces stable CoM trajectories. Increasing the time horizon to 10 gait cycles made it feasible to find a stable trajectory. For the control of real-time systems, this is a high computational overhead that may not be achievable. Here the advantage of the presented IS-MPC approach is apparent.

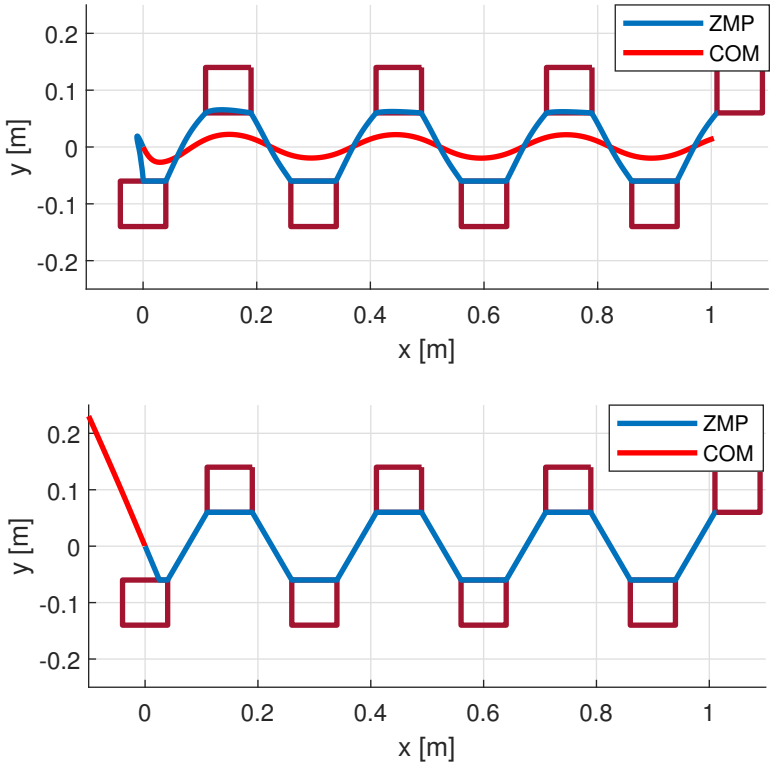

Fig. 4.2. Intrinsically stable (top) vs. standard MPC (bottom).

To fully realize the introduced concept on a real robot it is necessary to further consider the inverse kinematic task. Planning the implementation of the control on the hardware platform, the topic of the implementation of the optimisation solver should be extended. The discussion of both topics is well-formed in [24].

5. Reinforcement Learning Problem

5.1. Introduction

Reinforcement Learning (RL) is a branch of machine learning which deals with sequential decision making. A RL problem consists of an agent and an environment. The RL agent acts on the environment and gets a reward. The reward lets the agent know how good the action was at that particular time. The task of the RL agent is to learn an optimal policy - a mapping from states to action, which maximizes the expected sum of rewards. In RL the agent does not know what the right action is at the particular instant and must figure that out based on trial and error. The chapter is based on [25], [26].

5.2. Markov Decision Process

A Markov Decision Process (MDP) is defined by:

- A set of states, $s \in S$, where S denotes the state space
- A set of actions, $a \in A$, when A denotes the action space
- A scalar reward, r
- Transition probability

$$p(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{5.1}$$

- The reward function, $R : S \text{ x } S \text{ x } A \to \mathbb{R}$

$$R(s, a, s') = E(r_t|s_t = s, a_t = a, s_{t+1} = s') \tag{5.2}$$

- Discount factor γ

5.3. Policy

The policy is defined as mapping from states to probabilities of selecting each possible action. Reinforcement learning methods specify how the agent's policy is changed as a result of its experience. The policy could be either deterministic - depends only on the state - or stochastic - defines a probability distribution for each action, given a state.

5.4. Reward

The goal of the agent is formalized in terms of a signal, called the reward, passing from the environment to the agent. At each time step, the reward is a scalar. Informally, the agent's goal is to maximize the total amount of reward it receives. The sum of rewards is called as the return, G_t, which is given as,

$$G_t = r_t + t_{t+1} + r_{t+2} + ... + r_{N-1} \tag{5.3}$$

where, N denotes the end of an episode, t is the time index.

5.5. Value Function

The agent in order to be able to decide what action to take in a specific moment needs knowledge how good it is for it to be in a particular state. Value function is in a certain way a measure of suitability to the state. It is defined as the expected sum of rewards that the agent will receive while following a particular policy π from a state s. The value function, $V_\pi(s)$ for policy π is given by

$$V_\pi(s) = E_\pi(G_t|s_t = s) = E_\pi(\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t = s) \tag{5.4}$$

5.6. Model-free Methods

Model-free methods can be applied to any reinforcement learning problem since they do not require any environment model. Most model-free approaches either try to learn a value function and infer an optimal policy from it (Value function-based methods) or directly search in the space of the policy, parameters to find an optimal policy (Policy search methods).

Model-free approaches can also be classified as being either on-policy or off-policy. On-policy methods use the current policy to generate actions and use it to update the current policy, while off-policy methods use a different exploratory policy to generate actions as compared to the policy which is being updated.

5.7. Model-based Methods

Model-based methods try to utilize some information of the environment for planning. Planning with a model can substantially improve the sample efficiency of the algorithm which makes it popular in robotic applications. There are two approaches to model-based learning: one is to build a model of the environment from first principles and use it for learning the policy or value function. The other method is to learn a model of the environment by performing experiments on the system.

The problem with starting with a first principles model is that if the model is not accurate, the policy learned is likely to be sub optimal when applied on the actual setup. It might also happen that the policy just out right fails to do anything. Thus, most successful model-based approaches try learning a model and then using the model to accelerate the learning.

5.8. DDPG Simulink Implementation

The previous section introduced the Deep Deterministic Policy Gradient (DDPG) algorithm. The algorithm is completely model-free and requires no knowledge of the environment and has the ability to learn from off-policy data. This section looks at the Simulink implementation of the DDPG algorithm for learning a walking task for bipedal robot in a simulated environment. The implementation of the ***Learning Environment*** subsystem is analogous to that described in chapter 2. Its implementation has been extended to the calculation of the reward function and flag checking if the current training should be stopped prematurely. Simulink block diagram of described system is shown in Figure 5.1.

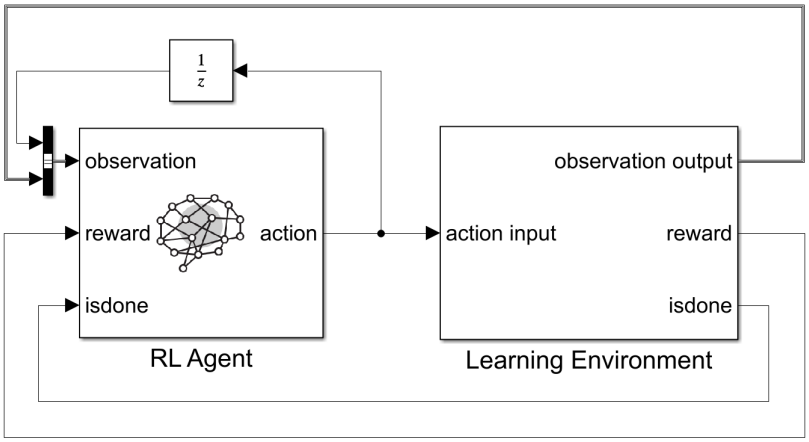

Fig. 5.1. Simulink block digram of reinforcement learning model

The state-space consists of 13 dimensions: the angles and the angular platform velocities, hips and knees rotary position and velocity as well. The angles for all the joints are defined with respect to the vertical position. The action space consists of torques applied to each actuator. The state space and action space can be written as

$$s = \begin{bmatrix} \theta_{platform} \\ \dot{\theta}_{platform} \\ x_{platform} \\ z_{platform} \\ v_{x_{platform}} \\ \theta_{l.hip} \\ \dot{\theta}_{l.hip} \\ \theta_{r.hip} \\ \dot{\theta}_{r.hip} \\ \theta_{l.knee} \\ \dot{\theta}_{l.knee} \\ \theta_{r.knee} \\ \dot{\theta}_{r.knee} \end{bmatrix} \qquad a = \begin{bmatrix} U_{l.hip} \\ U_{r.hip} \\ U_{l.knee} \\ U_{r.knee} \end{bmatrix} \tag{5.5}$$

where subscript l, r denote the left and right leg respectively, U is the torque to be applied to each actuator.

The agent's task is to walk forward at the highest possible velocity while keeping balance. The reward function shall thus be expressed as follows

$$r = k_{v_x} * v_x + T_d - k_z * \Delta z^2 - k_u * \sum_{i=1}^{4} \Delta T_i^2 \tag{5.6}$$

The reward function wards the agent for forward movement of the robot by taking into account the platform forward velocity. Author suggest that such a choice might not only improve efficiency of the task but also force the robot to move faster from point A to B. The second reward factor is the time of walking - T_d, the value is derived as

$$T_d = k_t * \frac{T_s}{T_f} \tag{5.7}$$

where T_s is the agent sample time, T_f is the total time of simulation, k_t is a time reward constant. Agent is penalized for not keeping constant height of the platform in the Z-axis by factor k_z - Δz is equal absolute difference between initial height and actual position of robot with reference to Z-axis. The purpose of such factor is to prevent the robot from falling down or jumping to move forward. To encourage the agent to generate a symmetric gait - using all actuators similarly - we penalize rapid change of torque value signal. It should prevent the robot from forward movement using only one limb while the other one is pulled and used only as stabilizing support.
Value of reward function factors are detailed in the Table 5.1. The reward function has been modelled in Simulink using the block diagram shown in Figure 5.2.

Table 5.1. Reward function coefficient value

Reward Constant	Value
k_{v_x}	5
k_t	25
k_z	50
k_u	2

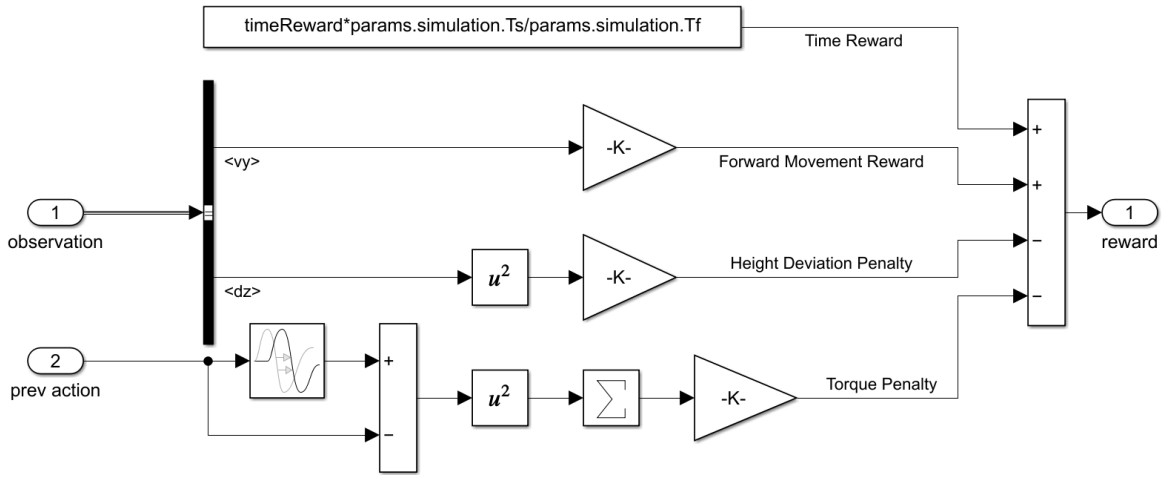

Fig. 5.2. Simulink block diagram of reward function

Each episode lasts for 15 seconds unless it is terminated prematurely. The termination condition of the episode is based on the angle of the torso, and the position of platform regarding Z-axis. The episode is terminated if the torso angle is less than 0.8 rad or if the difference between initial and current height of platform is less than 0.15 m. The termination condition for the hips and the torso denote the falling of the robot. The detailed implementation is shown in Figure 5.3.

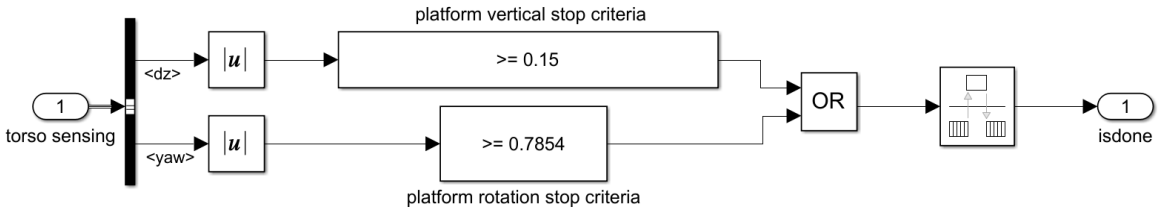

Fig. 5.3. Simulink block diagram of flag stopping current episode of training

At the start of every episode, the robot's left leg is completely stretched and the right leg is in a bend position. The initial conditions are as follow

$$s = \begin{bmatrix} \theta_{platform} \\ \dot{\theta}_{platform} \\ x_{platform} \\ z_{platform} \\ v_{x_{platform}} \\ \theta_{l.hip} \\ \dot{\theta}_{l.hip} \\ \theta_{r.hip} \\ \dot{\theta}_{r.hip} \\ \theta_{l.knee} \\ \dot{\theta}_{l.knee} \\ \theta_{r.knee} \\ \dot{\theta}_{r.knee} \end{bmatrix} = \begin{bmatrix} 0[rad] \\ 0[rad/s] \\ 0[m] \\ 0.6[m] \\ 0[m/s] \\ 0[rad] \\ 0[rad/s] \\ 0[rad] \\ 0[rad/s] \\ 0.17[rad] \\ 0[rad/s] \\ 0.17[rad] \\ 0[rad/s] \end{bmatrix} \tag{5.8}$$

DDPG algorithm initialization is based on the original paper [27]. The actor and critic networks are initialized with two neural networks having two hidden layers with 400 and 300 hidden neurons respectively. The training algorithm is chosen to be Adam optimization algorithm [28]. The exploration noise is modeled as an Ornstein-Uhlenbeck process with parameters, $\sigma = 0.3$, $\theta = 0.1$.

5.9. Experiment Results

Figures 5.4 and 5.5 show the performance of DDPG for presented model. Algorithm is run for 20000 episodes, the results are averaged out for 50 episodes. Figure 5.4 shows the episode reward as a function of conducted experiments. The shape of the learning curve initial part suggest that it takes 5000 episodes for the agent to learn how to move forward. Potentially, during this episodes agents fall down immediately.
For agents with a critic, Episode Q0 is the estimate of the discounted long-term reward at the start of each episode, given the initial observation of the environment. As training progresses, if the critic is well designed. Episode Q0 approaches the true discounted long-term reward, as shown in Figure 5.5.

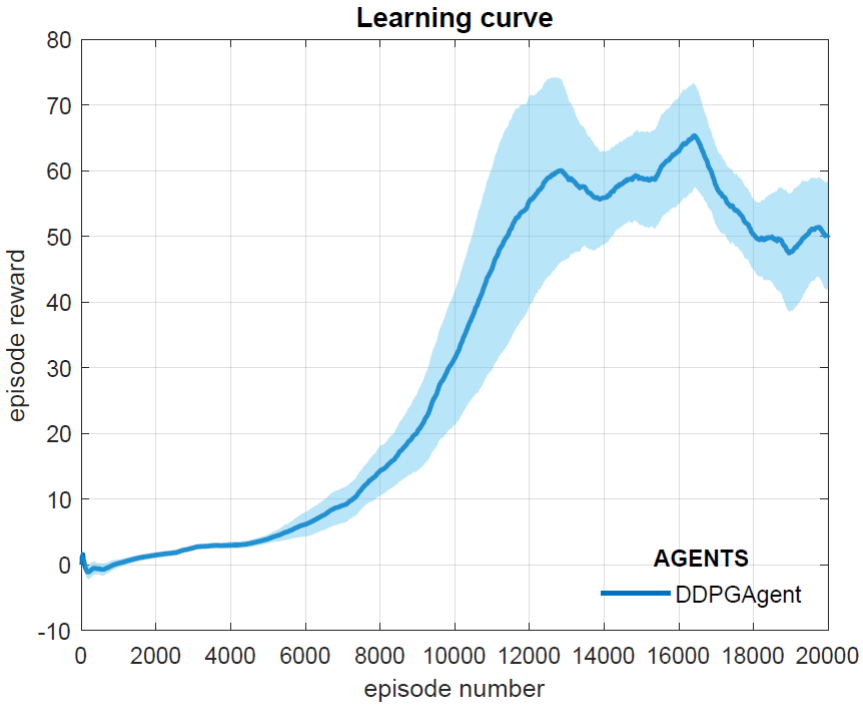

Fig. 5.4. Simulink block diagram of reward function

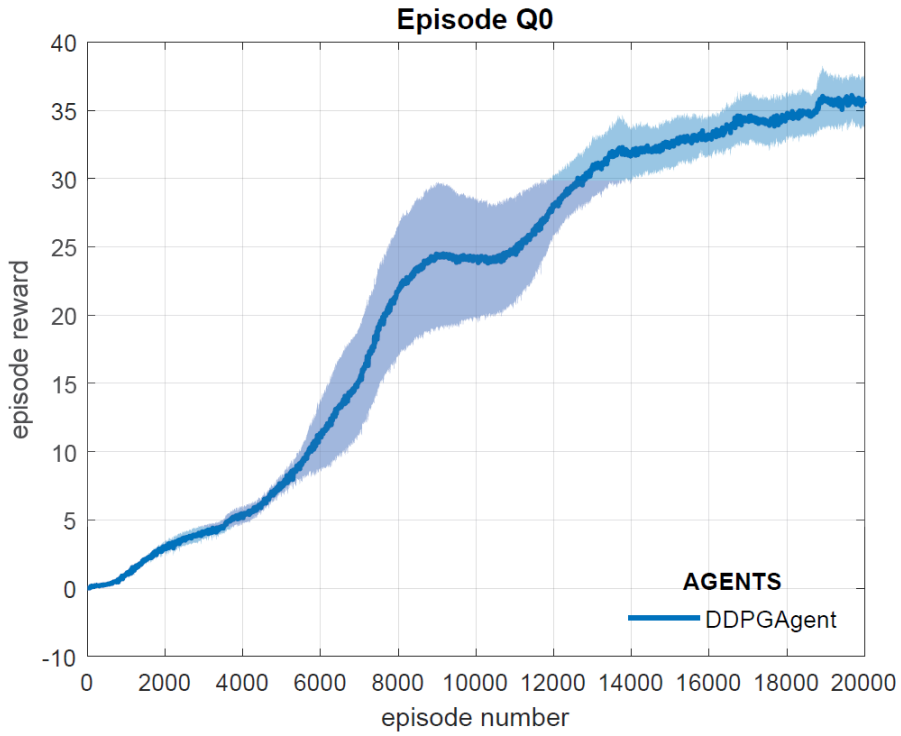

Fig. 5.5. Simulink block diagram of reward function

Comparing the results of the learning curve with the response of the learning environment simulation, one must, unfortunately, state that the agent did not learn the task thoroughly after 20 thousand episodes. Conducted simulation studies suggest that the agent has learned to make decisions leading to control joints so that the gait trajectory is concurrent with the nature of human gait. However, the agent has not learned the balancing subtask and can only take first few steps.

Figures 5.6 - 5.10 show the captured state trajectories of the simulated system. Figure 5.6 shows the relationship of the platform position to the zero reference point in the X and Z axes. Figures 5.7 and 5.8 summarize the relative angular position of each robot joint with respect to time. Comparing mentioned trajectories with the measured human gait trajectories of joints, one may conclude that the character of the function converges. However, there are deviations in the maximum amplitude of the signal.

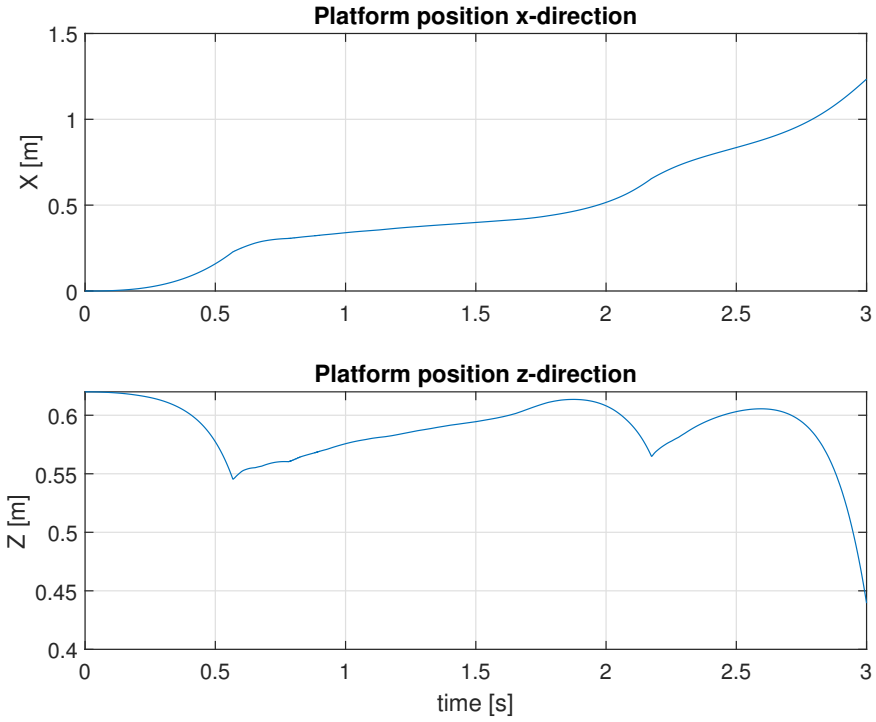

Fig. 5.6. Simulink block diagram of reward function

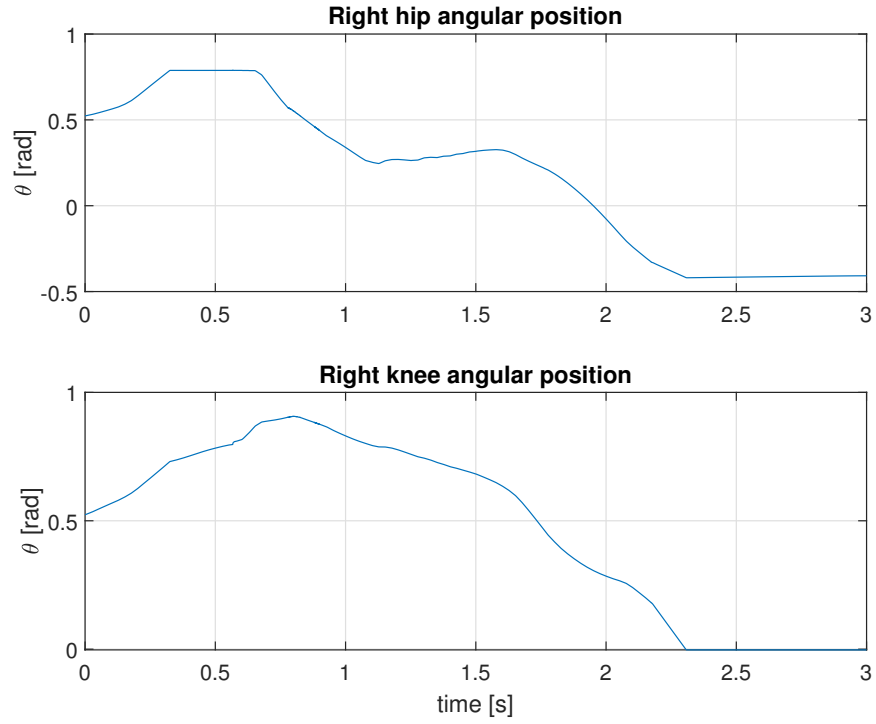

Fig. 5.7. Simulink block diagram of reward function

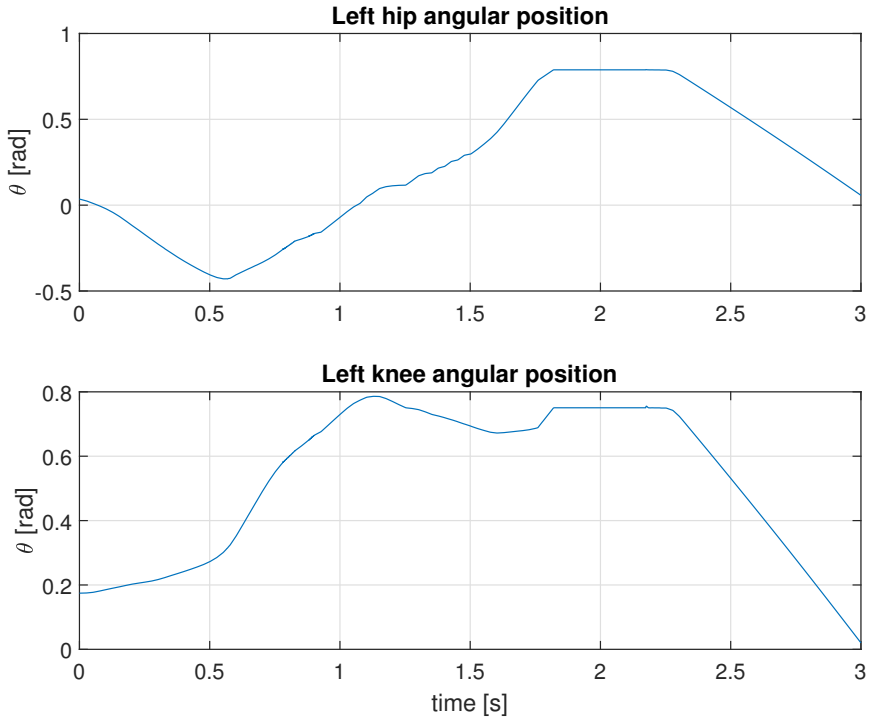

Fig. 5.8. Simulink block diagram of reward function

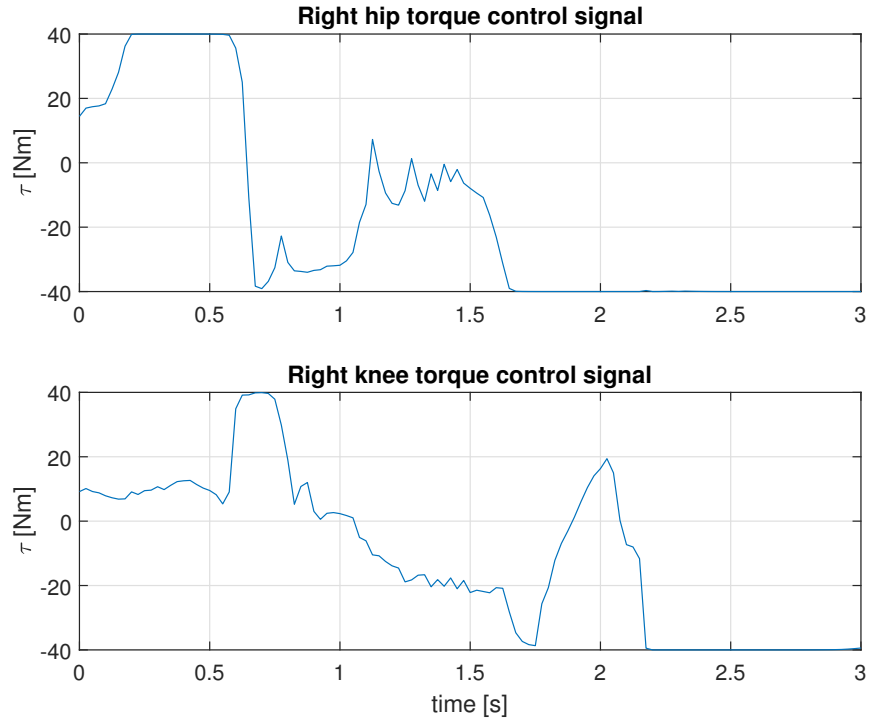

Fig. 5.9. Simulink block diagram of reward function

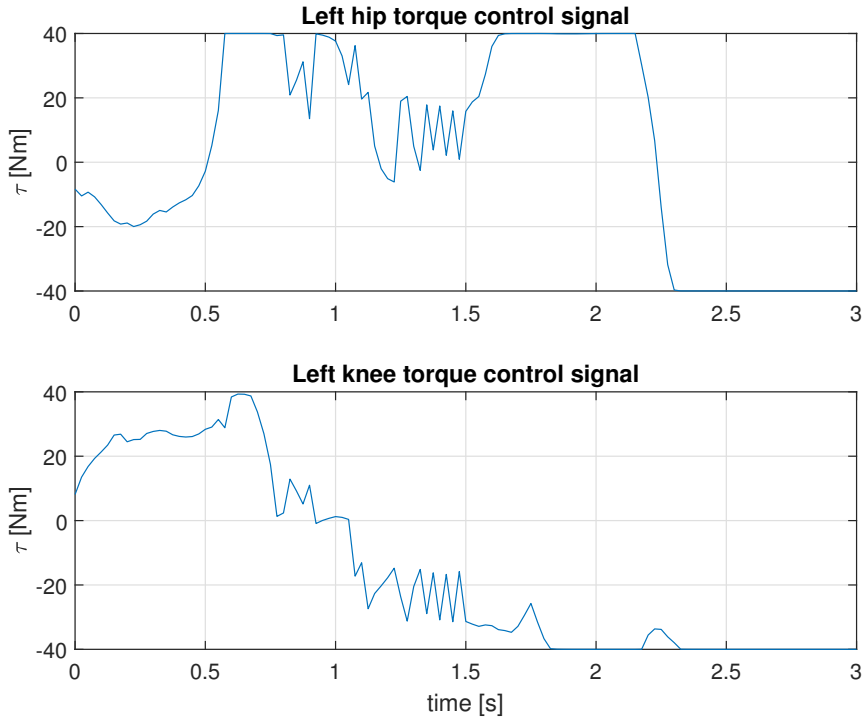

Fig. 5.10. Simulink block diagram of reward function

In order to visualize the state of the system, Figures 5.11 - 5.18 show snapshots of the simulation experiment conducted in the Simulink software from selected quanta of simulation time.

Figure 5.11 presents the initial conditions of executed simulation. At the time of 0.5 seconds of simulation, the robot is in the single contact phase, the right leg is moved, as seen in Figure 5.12. Then, the robot touches the ground with the transferred foot - it is in the double support phase as seen in Figure 5.13. Thus, the first gait cycle has been completed. The robot analogically moves to the single support phase in which the left leg is moved. Figures 5.14 - 5.16 show this phase in the simulation. It is visible that the transferred left leg is in very close proximity to the ground - even on the border of contact with the ground as if it was moving. At this time the robot is trying to maintain a stable position by balancing with the platform. The next two Figures 5.17 and 5.18present the phase in which the robot is not able to keep balance - the system loses stability.

Comparing the accurate decisions of agents in charge of bipedal gait control presented in [29], [26], one seek an explanation of the reason why discussed by us agent struggled to learn a fully stable gait. The learning process should likely be repeated for a more significant number of episodes conducted. However, there is a hypothesis that the problem posed in this task is much more complicated than described in the mentioned studies. A small-footed robot is much more unstable than a flat-footed structure. It can be seen that for the built environment model, the

agent has learned to make decisions through which the robot can perform movements similar to human gait. Running the learning process for a more significant number of episodes should potentially provide knowledge of the stability of robot to solve complex task of keeping the balance during walking.

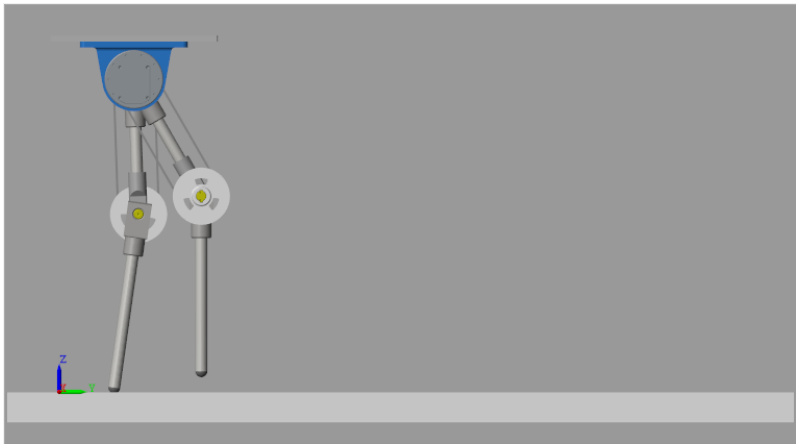

Fig. 5.11. Screenshot of the simulation experiment for the initial state of the simulation

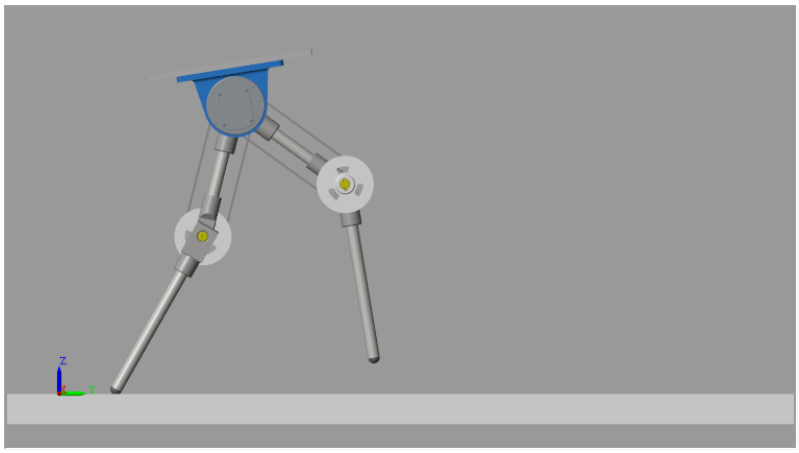

Fig. 5.12. Screenshot of the simulation experiment for a simulation time equal 0.5 seconds

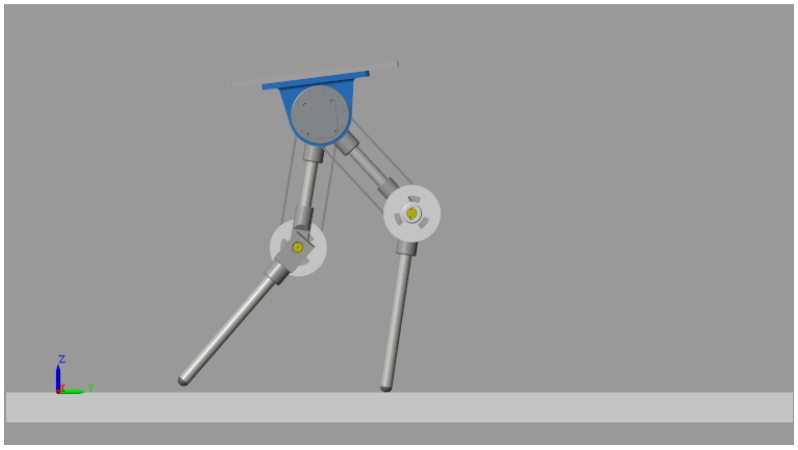

Fig. 5.13. Screenshot of the simulation experiment for a simulation time equal 0.75 seconds

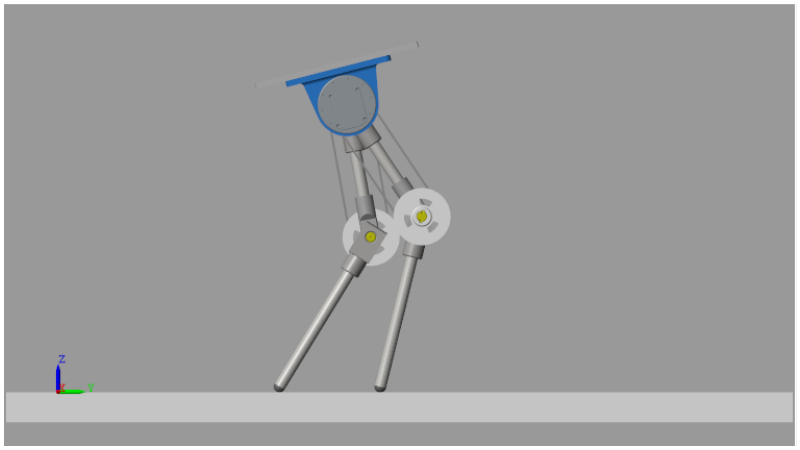

Fig. 5.14. Screenshot of the simulation experiment for a simulation time equal 1 second

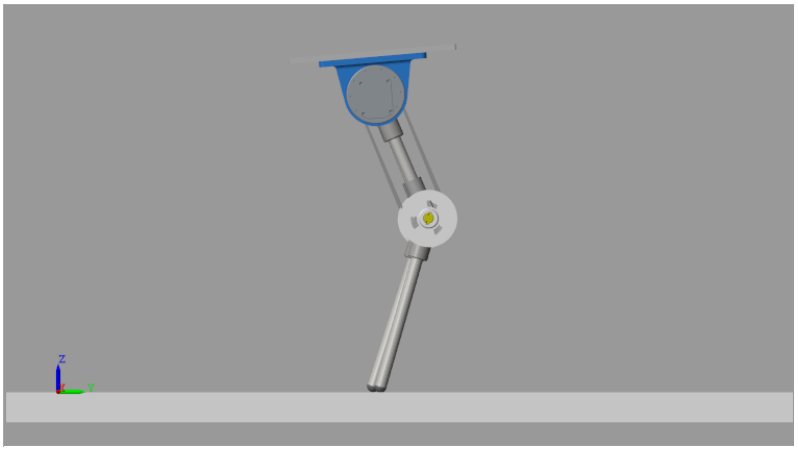

Fig. 5.15. Screenshot of the simulation experiment for a simulation time equal 1.5 seconds

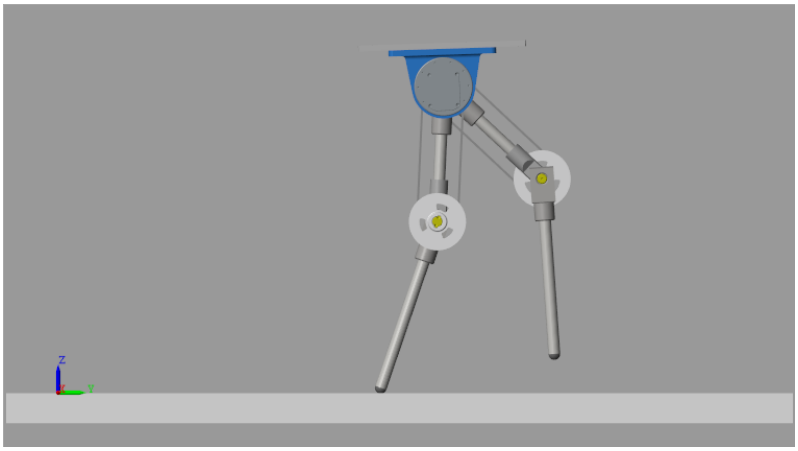

Fig. 5.16. Screenshot of the simulation experiment for a simulation time equal 2 seconds

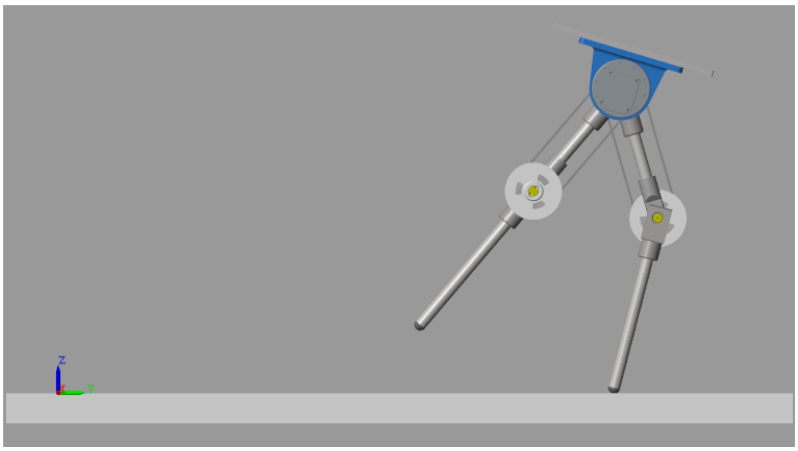

Fig. 5.17. Screenshot of the simulation experiment for a simulation time equal 2.5 seconds

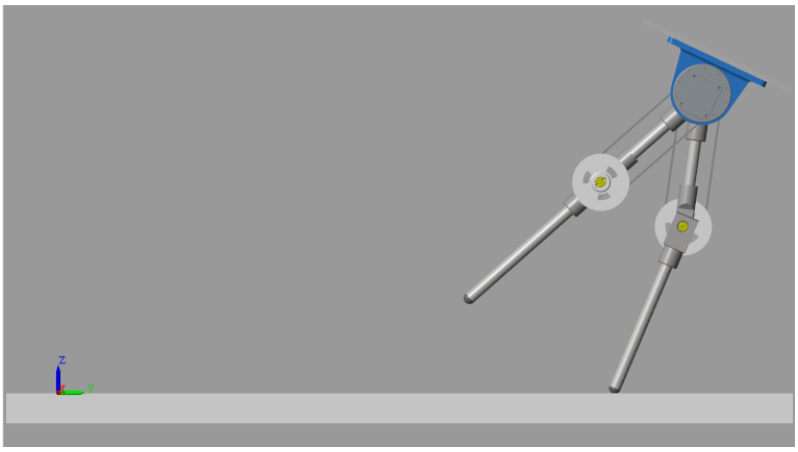

Fig. 5.18. Screenshot of the simulation experiment for a simulation time equal 3 seconds

6. Conclusion and future works

The following issues were addressed in this thesis:

- Implementation of the Simulation Model of a Walking Robot Prototype.
- Application of adaptive control techniques for leg position control.
- Investigating the impact of the stability-based constraints on the performance of MPC control.
- A feasibility study on the use of Reinforcement Learning approach for the bipedal robot walking.

The implementation of the bipedal robot prototype simulation model was accomplished with success using rigid-body simulator MATLAB Simscape Multibody. This allows control algorithms to be tested in the proof of concept phase without the risk of damaging the real object. The simulation was used for further work carried out as part of the thesis. It is necessary in the future to consider the dynamics of the actuator system in the simulation model and to extend the model of contact with the ground.

Identification experiments were carried out to improve the convergence of the model with the real object. During these experiments, the robot mechanical part failed. For this reason, the work had to be interrupted at this stage. As part of future work, it is worth considering extending the experimental equipment with sensors enabling even more precise measurement of the robot's dynamic parameters. The force sensors are essential for the further progress of the research, as they allow the acquisition of data related to the contact between the robot and the ground. They will contribute to forming a contact model, which is essential if hybrid systems are considered.

The first of discussed control approaches - model reference adaptive controller - has been implemented to follow the reference signal of the angular position of the leg's joints. Despite dividing the task into two sub-problems of actuator's position control, instead of considering the complete information about the leg's dynamics, the adaptive control reduced the significant error between the setpoint and the object response generated by the PD controller during the first seconds of the experiment.

We applied Model Predictive Controller to bipedal gait pattern generation. The MPC controller detailed in [20] has been implemented in the MATLAB environment. Experiments were carried out to investigate the impact of implemented constraints in the optimisation problem on the optimal time horizon for which the system is stable. The study shows that using the constraint related to the stability conditions of the pendulum, it was proved possible to generate a stable gait trajectory with a shorter time horizon compared to the MPC with ZMP constraint only. It is necessary to check how the inclusion of this constraint affects the computational complexity of the controller.

Finally, the hardware implementation of the control algorithms discussed in the thesis should be carried out. On this basis, the properties of the hardware implementation should be checked, such as: the maximum control frequency, ability to keep control in the real-time regime, volume of automatically generated code, ability to be deployed on heterogeneous platforms.

Bibliography

[1] Evan Ackerman. „*How Toyota Research Envisions the Future of Robots*". http://spectrum.ieee.org/automaton/robotics/home-robots/toyota-research-future-of-robots. [Online; accessed 28-June-2021]. 2020.

[2] Evan Ackerman and Erico Guizzo. „*DRC Finals: CMU's CHIMP Gets Up After Fall, Shows How Awesome Robots Can Be*". https://spectrum.ieee.org/automaton/robotics/humanoids/drc-finals-cmu-chimp-gets-up-after-fall-shows-how-awesome-robots-can-be. [Online; accessed 28-June-2021]. 2015.

[3] Jun Nakanishi et al. „*Learning from Demonstration and Adaptation of Biped Locomotion with Dynamical Movement Primitives*". In: *Robotics and Autonomous Systems - RaS* 2004 (Jan. 2003). DOI: *10.1299/jsmermd.2004.32_2*.

[4] Guillermo A. Castillo et al. „*Reinforcement Learning Meets Hybrid Zero Dynamics: A Case Study for RABBIT*". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 284–290. DOI: *10.1109/ICRA.2019.8793627*.

[5] Andrew M. Abate. „*Mechanical Design for Robot Locomotion*". PhD thesis. Oregon State University, 2018.

[6] Natalia Frankowska. „*Analiza wytrzymałościowa egzoszkieletu rehabilitacyjnego z wykorzystaniem pakietu ANSYS*". Bachelor Science Thesis. Thesis supervisor: dr inż. Krystian Szopa. AGH, 2020.

[7] Szymon Folek. „*Modelowanie i analiza kinematyczna egzoszkieletu kończyn dolnych do zastosowań medycznych*". Bachelor Science Thesis. Thesis supervisor: dr inż. Krystian Szopa. AGH, 2020.

[8] Hubert Milanowski and Adam Pilat. „*Comparison of Identified and SimScape Model ofHuman Leg Motion*". In: 2020. DOI: *10.1109/MSM49833.2020.9201736*.

[9] „*GYEMS X8 PRO - Technical Paper*". http://www.gyems.cn. [Online; accessed 28-June-2021].

[10] R. A. Hyde and J. Wendlandt. „*Tool-supported mechatronic system design*". In: *2008 34th Annual Conference of IEEE Industrial Electronics*. 2008, pp. 1674–1679. DOI: *10.1109/IECON.2008.4758205*.

[11] Steve Miller. „*Running Robot Model in Simscape - Github Repository*". http://github.com/mathworks/Simscape-Robot-4Legs. [Online; accessed 30-June-2021].

[12] „*Making Optimal Solver Choices for Physical Simulation*". http://www.mathworks.com/help/physmod/simscape/ug/making-optimal-solver-choices-for-physical-simulation.html. [Online; accessed 30-June-2021].

[13] Hubert Milanowski. „*Model dynamiczny egzoszkieletu*". Bachelor Science Thesis. Thesis supervisor: dr hab. inż. Adam Piłat. AGH, 2020.

[14] John Hollerbach, Wisama Khalil, and Maxime Gautier. „*Model Identification*". In: *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008, pp. 321–344. ISBN: 978-3-540-30301-5.

[15] „*EAIiIB Faculty Interdisciplinary Research Laboratory's website*". http://www.ilb.agh.edu.pl. [Online; accessed 28-June-2021].

[16] K.J. Åström and B. Wittenmark. „*Adaptive Control: Second Edition*". Dover Books on Electrical Engineering. Dover Publications, 2013. ISBN: 9780486319148.

[17] N.T. Nguyen. „*Model-Reference Adaptive Control: A Primer*". Advanced Textbooks in Control and Signal Processing. Springer International Publishing, 2018. ISBN: 9783319563930.

[18] S. Kajita et al. „*Biped walking pattern generation by using preview control of zero-moment point*". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1620–1626 vol.2. DOI: *10.1109/ROBOT.2003.1241826*.

[19] Nahuel A. Villa and Pierre-Brice Wieber. „*Model predictive control of biped walking with bounded uncertainties*". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 836–841. DOI: *10.1109/HUMANOIDS.2017.8246969*.

[20] Nicola Scianca et al. „*Intrinsically stable MPC for humanoid gait generation*". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 601–606. DOI: *10.1109/HUMANOIDS.2016.7803336*.

[21] Nicola Scianca et al. „*MPC for Humanoid Gait Generation: Stability and Feasibility*". In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1171–1188. DOI: *10.1109/TRO.2019.2958483*.

[22] S. Kajita et al. „*The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation*". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. Vol. 1. 2001, 239–246 vol.1. DOI: *10.1109/IROS.2001.973365*.

[23] Pierre-brice Wieber. „*Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations*". In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*. 2006, pp. 137–142. DOI: *10.1109/ICHR.2006.321375*.

[24] Alexander Sherikov. „*Model predictive control of a walking bipedal robot using online optimization*". MA thesis. Örebro University, 2012.

[25] Richard S. Sutton and Andrew G. Barto. „*Reinforcement Learning: An Introduction*". Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[26] Arun Kumar, Navneet Paul, and S Omkar. „*Bipedal Walking Robot using Deep Deterministic Policy Gradient*". In: (July 2018).

[27] Timothy P. Lillicrap et al. „*Continuous control with deep reinforcement learning*". 2019. arXiv: *1509.02971*.

[28] Diederik P. Kingma and Jimmy Ba. „*Adam: A Method for Stochastic Optimization*". 2017. arXiv: *1412.6980* [cs.LG].

[29] Divyam Rastogi. „*Deep Reinforcement Learning for Bipedal Robots*". Master Science Thesis. Thesis supervisor: Kober, Jens, Koryakovskiy, Ivan. TU Delft, 2017.