Otto-von-Guericke Universität Magdeburg
Fakultät für Elektro- und Informationstechnik
Institut für Automatisierungstechnik

Inertial sensors for Functional Electrical Stimulation in the gait therapy

Pre-diploma Thesis

Submitted at 19.08.2004
by

Sabine Haumann
(* 12.05.1979 in Iserlohn)

Supervisors: Dipl.-Ing. Nils-Otto Negård
Dipl.-Ing. Thomas Schauer
Max-Planck-Institut für Dynamik komplexer technischer Systeme Magdeburg
Examiner: Prof. Dr.-Ing Jörg Raisch
Lehrstuhl für Systemtheorie technischer Prozesse
Otto-von-Guericke Universität Magdeburg

Abstract

Functional Electrical Stimulation (FES) makes it possible to accelerate the reactivation of the partially paraplegic leg muscles of stroke patients. To allow functional movements, e.g. the gait, sophisticated stimulation patterns are needed. Usually the stimulation patterns are controlled by a finite automaton. With closed-loop controls, the stimulation patterns can be adapted to the patients. To close the loop, gait parameters like step length and step height must be estimated as simple as possible.
Chapter 1 of this pre-diploma thesis introduces into the basic principles of FES, normal gait and estimation of gait parameters. Chapter 2 deals with the state of art. A physiological stimulation pattern is developed and implemented as finite automaton in Matlab / Simulink in Chapter 3. In Chapter 4 a stimulation pattern of the company 'Krauth + Timmermann', Hamburg, is introduced further. For identifying gait parameters, an algorithm which estimates the passed 3D-trajectory by the use of inertial sensored acceleration and orientation data is developed in chapter 5. The algorithm is applied to some test data. The results are validated by accomplishing some reference measurements with a 3D-motion analysis equipment. With this comparison data the accuracy of the inertial sensor is preliminary investigated. Chapter 6 resumes the results of this pre-diploma thesis and gives a short outlook to some further topics.

Kurzfassung

Mittels Funktioneller Elektrischer Stimulation (FES) kann die Reaktivierung der teilweise gelähmten Beinmuskulatur bei Schlaganfallspatienten beschleunigt werden. Um funktionelle Bewegungen wie z.B. den Gang zu ermöglichen, sind komplexe Stimulationsmuster notwendig. In der Regel erfolgt die Steuerung dieser Stimulationsmuster über einen Zustandsautomaten. Mittels einer Closed-loop-Regelung sollen die Stimulationsmuster an den jeweiligen Patienten angepaßt werden. Um den Regelkreis schließen zu können, müssen Gangparameter wie Schritthöhe und -länge möglichst einfach erfaßt werden.
Die Grundlagen der FES, des Ganges und der Erfassung der Gangparameter werden in Kapitel 1 dieser Studienarbeit eingeführt. Kapitel 2 beschäftigt sich mit dem Stand der Technik. In Kapitel 3 wird ein physiologisch sinnvolles Stimulationsmuster entwickelt und als Zustandsautomat für die geregelte Elektrostimulation beim Laufbandtraining in Matlab / Simulink implementiert. Kapitel 4 stellt ein Stimulationsmuster der Hamburger Firma 'Krauth + Timmermann' genauer vor. Zur Erfassung von Gangparametern wird in Kapitel 5 ein Algorithmus entwickelt, der aus Beschleunigungen und Orientierungen, die mit einem Inertialsensor gemessen werden, die durchlaufene Raumkurve bestimmt. Der Algorithmus wird auf Testdaten angewendet. Zur Validierung der Ergebnisse werden Vergleichsmessungen mit einem Referenzsystem (3D-Bewegungsanalysesystem) durchgeführt. Anhand dieser Vergleichsdaten wird die Genauigkeit des Inertialsensors voruntersucht. Kapitel 6 faßt die Ergebnisse dieser Studienarbeit zusammen und gibt einen kurzen Ausblick auf weitere Fragestellungen.

Contents

Chapter 1

Introduction

1.1 Functional Electrical Stimulation (FES)

The World Health Organization [17] estimates that in 2001 there were over 20.5 million strokes worldwide. 5.5 million of these were lethal. Many of the patients have got serious damages in their central nervous system (CNS). Their gait pattern is badly damaged and their muscles are weakened due to prolonged lying in bed. Often they are hemiplegic. Also incomplete spinal cord injury and multiple sklerosis can lead to upper motor neurone lesion (UMNL) and hemiplegia. Any incomplete lesion has got a good chance of recovery, so one key issue for these patients is the rehabilitation of gait. Independent ambulation means more indepency in everyday life and speeds up the social reintegration.

One problem of the gait rehabilitation is that the muscles needed for carrying the patient's own weight have to be strong enough before the gait training can start. There must be enought muscle bulk, and the patient must be able to activate the needed muscles. This leads to a very late rehabilitation of gait. This problem can be solved by the use of a treadmill with upper body weight support, where the patient is supported by electrical stimulation. In the convential rehabilitation two therapists move the feet of the patient equally. This method is physically hard work for the therapists, and one patient needs two therapists, so other methods of therapy are searched.

One method with good prospects is the use of Functional Electrical Stimulation (FES). Depending on the pathology that caused the hemiplegia the damage is normally located in the CNS or in the spinal chord. That means that the muscles theirselves are healthy, only their voluntary control does not work any more. In the FES the muscles are controlled from outside by putting currents on the nerves which innervate the muscles. The muscles contract and the treatened limb moves. If the nerves are damaged, it would also be possible to stimulate the muscles directly. The disadvantage of this is that the needed current is much higher than for stimulating the nerves, and the high currents might lead to burnings on the skin.

There are two different methods for putting the current onto the nerves: Surface arrangements and implantable arrangements. A surface arrangement means surface electrodes fixed on the

skin and connected with a stimulator which sends the currents to the electrodes. Disadvantages of the method are the needed exact placement on the skin and a possible irritation of the skin. Even if the electrodes were placed exactly, they did not allow selective stimulation of particular muscles. Because of these disadvantages implantable stimulators were developed, but patients with incomplete lesion shall be cured, so for them it is better not to implant something.

Another problem is the spasticity of many hemiplegic patients. There are some studies dealing with reduction of spasticity with FES. Alternative, the spasticity can be reduced manually by the therapist (one therapist is still necessary for supervising) before FES starts.

The aim of the rehabilitation is that the gait of the patient develops as normal as possible, and the basic idea is to re-teach the central nervous system the gait pattern. It is essential for the artificially induced gait pattern to be as similar as possible to the normal (physiological) gait pattern. On the other hand in the normal gait many muscles are activated, and it is difficult to select too many muscles with surface electrodes. There are also technical limits in the stimulators. A stimulation pattern that meets the requirements has to be developed and realised.

For patients with chronic hemiplegia the FES training improves the cardiovaskular fitness.

1.2 Physiological gait

The normal gait is introduced as described by Perry [11] to compare it with the electrically induced gait.

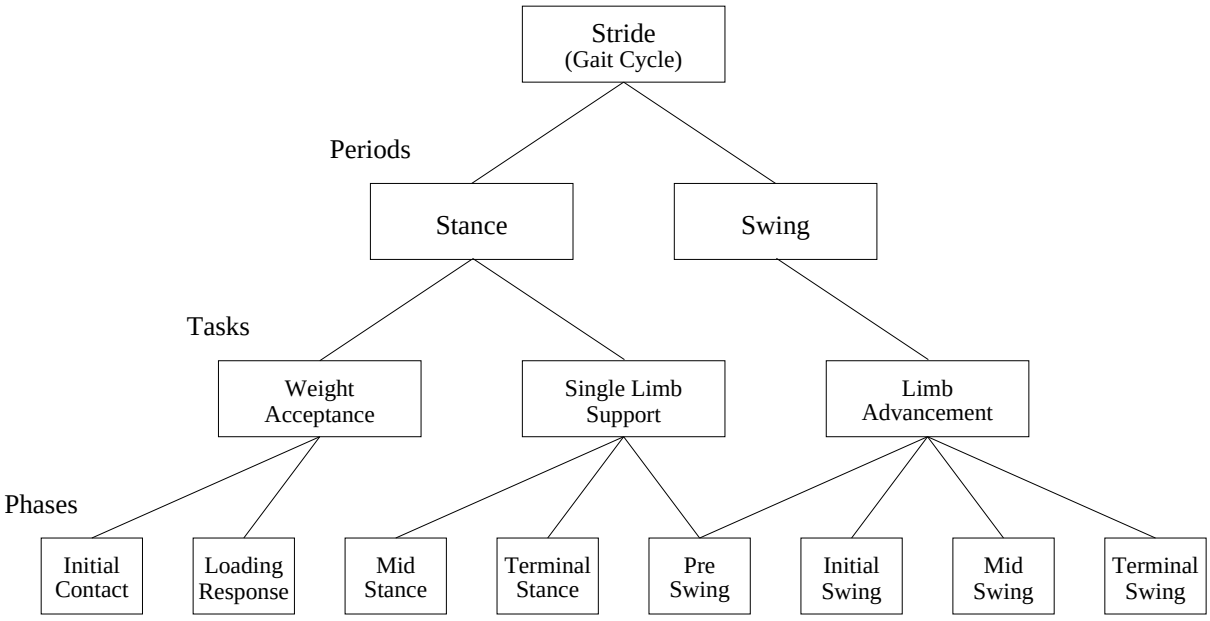


Figure 1.1: Physiological gait cycle (GC), adapted from Perry [11]

The gait cycle (GC) is shown in Figure 1.1. It is basically divided into the stance period (60% of GC) and the swing period (40% of GC). The gait can also be subdivided into the tasks weight acceptance (10% of GC), single limb support (40% of GC) and limb advancement (50% of GC). The three tasks contain different phases.

After valid nomenclature the gait starts with the initial contact, which is the first phase of the weight acceptance. Here the foot hits the ground first, normally with the heel. The second phase of the weight acceptance is the loading response, where the weight is shifted to the leg that hit the ground.

The single limb support is subdivided into two phases with the same length, mid stance and terminal stance. Their main difference lies in the mechanism of progression. That task is finished by the initial contact of the other leg.

The swing period has to be prepared at the end of the stance phase, in the phase pre swing. Here the weight is shifted to the other leg, and the heel is lifted. This phase belongs to the limb advancement, but it is still part of the stance period. At the moment the foot has left the ground the phase initial swing begins. It goes on until the swinging foot is opposite the stance foot, and normally it takes around a third of the swing period. The next third of the period is called mid swing, here the knee is extended until the tibia is vertical with respect to the ground. The last third of the swing is the terminal swing, where the leg is extended. This phase is finished by the initial contact that starts the next gait cycle.

1.3 Estimation of gait parameters

For the symmetry of the rehabilitated gait it is also important that gait parameters like the step length and the step height are equal. In conventional rehabilitation, the therapists care about this, but in FES it must be done by a closed loop control system. For generating equal steps, it is necessary to measure step length and height.

One possible method is the use of an optical or ultrasonic gait analysis system. In this pre-diploma thesis an ultrasonic equipment is used. It consists of a transmitter and some markers. The transmitter emits ultrasonic waves. The markers which are fixed to the interesting subject, here the foot, receive this waves. Based on the running times of the waves the position of the markers can be calculated. The problem of this measurement: the transmitter needs a free sight to the markers, but the therapist would stand between them.

An alternative is the use of micro-mechanical inertial sensors. These sensors measure angular velocities around all three axes and accelerations in all three directions. They are small and can be fixed to the foot. Another important advantage is that they do not require any reference system. The three-dimensional trajectory of the inertial sensor and following the trajectory of the foot can be estimated with the sensor-data. It is important to know that the estimated distances are always relative, that means only the change in respect to a start position can be obtained. In

the desired application the absolute data is not needed, so the start data is voluntary set to zero to simplify matters.

One problem of these sensors is that the acceleration signal includes the gravitation, and in the practical use it is difficult to exclude it from the signal. Another big problem is the bias of all signals. The bias is assumed to be constant, but the disturbances due to it increases during the necessary integrations.

In the second part of this pre-diploma thesis an algorithm is developed which estimates the three-dimensional trajectory of the sensor and displays the horizontal displacement and the height of the observed movement. Here the algorithm works offline, that means that after all movements the gait parameters of the single movements are estimated.

This algorithm is tested on simulated data and on simple movements performed with a measurement object. This measurement object consists of ultrasonic markers fixed on an inertial sensor. The estimated 3D-trajectory based on the sensor data is compared to the measured 3D-trajectory of the reference system. With this the accuracy of the inertial sensor system is preliminary investigated.

Chapter 2

State of the art

2.1 Stimulation programs

This chapter gives a short introduction into the state of the art this pre-diploma thesis is based on. The first section deals with existent research concerning artificially induced gait patterns and their implementations. In the second section the work of two other research groups who estimate 3D-trajectories by the help of inertial sensors is introduced.

The main approaches for artificially induced gait patterns are summarized by Lyons et al. [6]. This article gives an overview about the history of the use of FES in the gait. There are several different requirements and possibilities to fulfil them. Before developing an artificially induced gait pattern, the requirements concerning the number of stimulation channels must be decided. If only the dorsiflexor of the foot is too weak (this symptom is called 'drop foot'), it is enough to stimulate with one channel, if other muscles also need support, more channels are used.

Another important topic is the detection of start and stop of the stimulation. The most simple method was published by Liberson et al. [3]. A switch is fixed under the heel of the foot. When the patient lifts his heel, the switch opens and the stimulation is started. At the moment the heel hits the ground again, the stimulation is stopped. The advantage of this solution is that the patient has got maximum control, he can start and stop the step arbitrary and can also rest between the steps. But this method only works for the drop foot. If the m. quadriceps which stabilises the knee and the m. gluteus maximus which stabilises the hip are affected too, their stimulation has to be active longer time. Otherwise, the patient would fall down.

Pappas et al. present a more sophisticated method for gait detection ([8],[10]). By the use of a miniature gyroscope and three force sensors they detect four gait phases (stance, heel off, swing and heel strike). They validate their system with ten able-bodied subjects and six subjects with gait deficiencies, and the result is very satisfying [9].

One difficulty in the multichannel stimulation is that the hip flexor cannot be stimulated with surface electrodes, it lies to deep inside the body. A possible solution is to elicit the withdrawal reflex by stimulating the n. peroneus. This reflex causes the flexion of the hip, the knee and the ankle, but it is also difficult to elicit.

One multichannel stimulation pattern was realised by Mokrusch and Busch [14]. Their stimulation program is called TransStim©. In Chapter 4 their artificially induced gait pattern and its realisation are described more detailed. It was developed especially for walking with a treadmill. In TransStim© the m. quadriceps and the m. gluteus are stimulated during the whole gait, so the knees and the hip are always unbowed. In the stimulation pattern the withdrawal reflex is not used. Instead, the leg is lifted by contracting the contralateral waist flexor (m. quadratus lumborum). The forward motion takes place by shifting the weight voluntary and by the treadmill that moves the contralateral leg backwards. The toe is lifted by stimulating the dorsiflexor (m. tibialis anterior).

Bogataj et al. [1] realise their stimulation pattern by dividing both gait periods, stance and swing period, into respectively eight equidistant subphases. For each subphase they decide which muscles are stimulated. The first step is started by pressing a footswitch; it can be chosen under which foot the switch is placed. The time of the subphases is scaled to the whole stride time that is measured, separating the stance and the swing period.
Bogataj et al. stimulate the n. peroneus to elicit the withdrawal reflex that lifts the foot and the leg. They also stimulate the hamstring muscle group to support this effect and the m. soleus. Bogataj et al. introduce the stimulation slowly, because they do not want to disturb the patient with full stimulation on all channels from the beginning. They start with stimulating only the n. peroneus, and after several steps they begin stimulating first the m. quadriceps, second the m. gluteus maximus, third the m. soleus and finally the hamstring muscle.
In the referenced article [1] the matter of one patient is presented in detail. Here the patient is stimulated on five channels, but in general there are six channels used. The stimulation pattern is adapted for every patient and also during the therapy.

In Figure 2.1 the stimulation patterns of the presented approaches with multichannel stimulation (own stimulation program, TransStim©, Bogataj et al.) and the normal activation of the muscles described by Perry [11] are assorted and compared. Every bar represents a muscle, and muscles which are used for the same motions are summarized and zoned by dark grey bars. The muscles as well as the approach that uses them are listed at the right edge of the figure. At the top the gait phases are given. The coloured sections of the muscle bars represent the times of the gait cycle at which the concerning muscles are stimulated.

One main difference lies in the number and the identity of the recruited muscles. In TransStim© four channels per leg are stimulated and Bogataj et al. stimulate on five channels per leg. Also, different muscles are stimulated. Bogataj et al. lift the leg and the foot by eliciting the withdrawal reflex by stimulating the n. peroneus, and in TransStim© stimulate the m. tibialis anterior and the m. quadratus lumborum are stimulated. Normally the m. tibialis anterior and the ischiocrural and some other muscles are used.
Another important difference lies in the time of the stimulation. In TransStim© the m. quadriceps and the m. gluteus medius are stimulated during the whole gait, Bogataj et al. does not do this. Normally, both muscles are not activated in this way.
Also the m. gluteus maximus is stimulated in a different way. Normally, it is activated only in the first half of the stance period, but Bogataj et al. stimulate it until the end of the stance. In

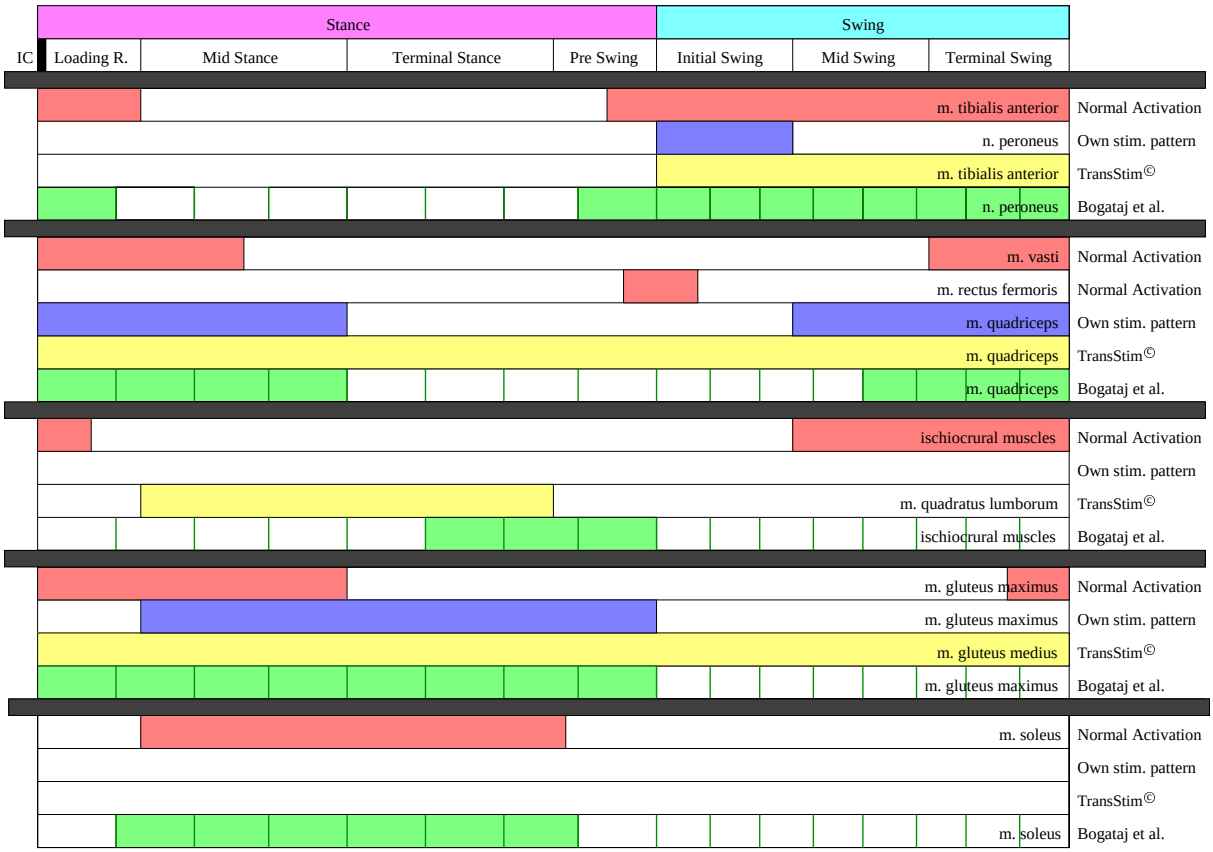


Figure 2.1: Stimulation on multiple channels

TransStim©, the similar m. gluteus medius is stimulated during the whole gait.

Compared to the normal gait, the stimulation pattern of TransStim© of seems to be quite unphysiological. The stimulation pattern of Bogataj et al. [1] is much more physiological, but it is doubtful if the n. peroneus should be stimulated during the whole swing phase.

In the first part of this pre-diploma thesis a physiological stimulation pattern is developed and implemented in Matlab / Simulink. The stimulation pattern is based on the pattern of Bogataj et al. [1]. The Implementation is designed multifunctional. It can be chosen if the patient is walking on a treadmill, and if both legs or only the paretic leg are stimulated. The additional stimulation of the healthy leg helps the patient to learn the timing of the gait and supports the symmetry of the gait.

2.2 Estimation of three-dimensional trajectories

There are several research groups who also work with inertial sensors, but not many who try to estimate gait parameters with them.
The main problem is still the accuracy of the estimation. With inertial sensors, angular velocity and acceleration are measured. In general, the acceleration data has to be integrated two times to obtain the path. Before integrating the acceleration data it has to be transformed into the earth fixed inertial coordinate system, because the sensor has got its own body-fixed sensor coordinate system which moves with the sensor. For this transformation it is necessary to know the orientation of the sensor. Without offsets and non-linearities of the sensors the angular velocity measured by the gyroscopes could be integrated to obtain the orientation of the sensor, and with the orientation data the acceleration data could be transformed and integrated two times. Also the gravitation which is included in the acceleration data could be removed easily. The problem is that the accelerometers and the gyroscopes suffer from drifts, offsets, non-linearities, noise and other disturbant factors. During the integration the influences of these aberrations increase.

Veltink et al., a research group from Twente, The Netherlands, has improved the orientation estimation by additionally using the acceleration data [5]. The acceleration data also contains the gravitation. While the sensor is not moving, the measured acceleration data is assumed to contain only the gravitation. According to the splitting of the gravitation with respect to the three measured directions the tilt of the sensor can be calculated. This calculation is compared to the tilt of the orientation calculated from the gyroscope data. The orientation from the gyroscope is splitted into a tilt and the rotation around the z-axis. The difference of the tilts is fed into a Kalman filter and fused with the rotation around the z-axis. In article [5] it is worked out that the error in the orientation estimation is much lower than without using acceleration data and Kalman filter.
Two years later the same research group has presented one single triaxial sensor that contains all three accelerometers and all three gyroscopes [16]. They still only estimated the orientation of the sensor without estimating the 3D-trajectories or the covered distances.
Again two years later the same research group estimates the path [15] and applicates the estimation for stroke patients. In this article a study is mentioned but not referenced exactly. This study supposes that the estimation error of the distance can be reduced to less than 3%. Veltink et al. also did not compare the derived foot orientations and positions to a reference 3D movement analysis system. However, they point out that in principle the estimation of positions with inertial sensors is possible. The PhD-Thesis of Luinge [4] also deals with inertial sensors, but does not present solutions for the estimation of path.

An other research group with J. Quintern, Hauske, Hagenauer which has applied inertial sensors for human movement is working in Munich, Germany. There is only the master's thesis of C. Schlossbauer available [12]. Schlossbauer uses a Kalman-Bucy-filter to estimate the gravitation vector. For the calculation of the orientation this gravitation vector is used as z-axis after normalising it. The x- and the y-axis are calculated by using vector cross products to ensure the orthogonality of the coordinate system. The calculation of the exact direction (the rotation around the z-axis) is not possible and also not necessary, because the movement of the walking subject will take place in all directions of the x-y-plane and has to be added up vectorial at all.

To obtain the path Schlossbauer integrates the acceleration data two times. Because of the large drift he substracts the constant component before integrating the data. This constant component is obtained by filtering the data with a low pass filter. This method assumes that in the resting state the acceleration and velocity are zero, so that the displayed data of these states must correspond to the bias. Schlossbauer also compares the estimated 3D-trajectory with the trajectory calculated from a reference measurement system. In the step length (0.6 m) Schlossbauer obtains a difference of 13% and in the step height (0.31 m) a difference of 2.6%.

The second part of this pre-diploma thesis preliminary investigates if it is possible to estimate 3D-trajectories more accurately.

Chapter 3

Stimulation program

3.1 Development of the stimulation pattern

Definition of the gait phases

In this chapter the developed stimulation pattern and its implementation is described. The first section presents the development of the stimulation pattern, firstly the definition of the gait phases and their comparison to the introduced physiological gait (see Section 1.2). Next the gait phases are formed to a gait program and described by a petri net. The third step is the selection of the stimulated muscles. After that the motion sequence of a stride is drawn physically, and at last the intrinsic stimulation pattern is developed and summarized in a table. Now the stimulation pattern must be implemented. This is described in the second section of this chapter.

The gait phases illustrated in Figure 3.1 are adapted from Perry [11]. In the mid swing and the terminal swing phase there is the same stimulation pattern (extension of the knee), so both phases are put together. The transitions (conditions for changing to the next phase) between initial swing and mid swing resp. mid stance and terminal stance are triggered by time. This transition time shall be scaled to the time of the whole swing period, so it is necessary to measure the needed swing time. That's why the swing period, the mid stance and the terminal stance of the contralateral leg are summarised to the phase 2 resp. 4. The combination of both legs with different transition times leads to the subdividing of both phases into the three subphases 2.a, 2.b and 2.c resp. 4.a, 4.b and 4.c.
The phase '1. double support' is not a part of the gait cycle, so it does not occur in Figure 3.1. The patient starts the gait from this phase and returns there after the gait.

There are 5 phases defined:
0. Initialisation
1. Double Support
2. Swing Paretic Leg, Stance Healthy Leg
3. Double Limb Stance 1
4. Stance Paretic Leg, Swing Healthy Leg
5. Double Limb Stance 2

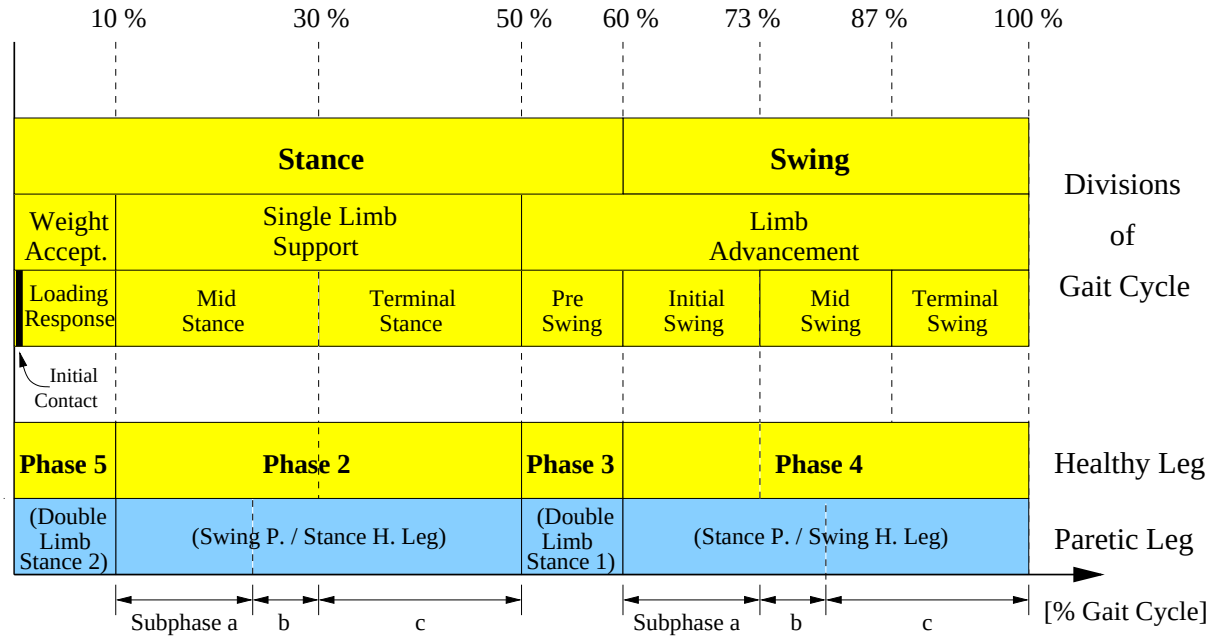


Figure 3.1: Gait phases

Description by a petri net

In Figure 3.2 the gait program is described by a petri net as done by Fuhr and Schmidt [2]. Each circle of the petri net represents a gait phase and each cross line a transition.
The different subphases of the phases 2 and 4 are realised by changing the stimulation patterns inside the phases, so they don't occur in the petri net of the gait cycle.
The (drawn) transitions between the phases are triggered by footswitches and a handswitch. The function of these switches is the gait phase detection.

The transitions inside the gait phases are triggered by time and scaled to the needed swing time. Before transition times can be scaled, the swing time must be measured. This is done in the implementation described in Section 3.2, but before the values are available, the transition between mid swing and terminal swing (phase 2.a to 2.b resp. phase 4.a to 4.b) must be triggered by a preset time value. If not, the knee would only be lifted, but not extended. That would mean that the swing phase would not be finished. The transition between mid stance and terminal stance (phase 2.b to 2.c resp. phase 4.b to 4.c) is not too critical. The difference between their stimulation patterns is that in the mid stance the m. quadriceps which extends the knee is stimulated and in the terminal stance it is released. Without the transition the m. quadriceps is stimulated during both phases and the gait is more stiffly. This is undesired, but not dangerous. So for the first two strides the transitions from the subphases a to b are triggered by a preset time, and the subphase c is not passed through. It should be noticed that the preset time represents the swing time, so the transition takes place after one third of the preset value. From the third stride on the transition a to b takes place after one third of the whole swing time and the transition b to c takes place after half of the whole swing time.

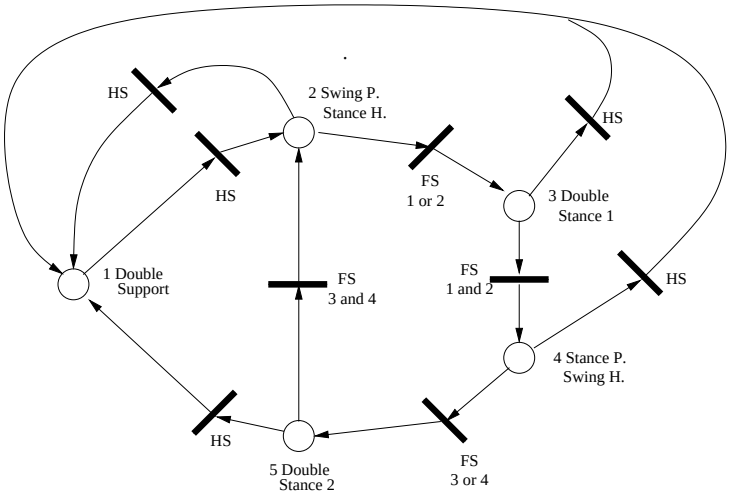


Figure 3.2: Petri net of the gait cycle

There are four foot switches (FS 1-4). FS 1 is placed under the heel and FS 2 under the toe of the paretic leg. Corresponding, FS 3 is placed under the heel and FS 4 under the toe of the healthy leg. HS is a hand switch.

At the beginning of the gait the patient stands on the ground (circle 1: 'double support'). With pressing the handswitch the gait is started (transition from 'double support' to 'swing paretic, stance healthy leg'). The paretic leg is brought to the swing phase, while the healthy leg remains in the stance. After FS 1 or FS 2 is closed, which means that the paretic heel has hit the ground (it is irrelevant for the transition if the toe or the heel hits the ground first) the phase 'double stance 1' is reached. This phase goes on until both footswitches (1 and 2) are closed. If the heel and the toe hit the ground at the same time, this phase is skipped. The closing of both footswitches is the transition to the phase 'stance paretic, swing healthy leg', that means that the swing of the healthy leg and the stance of the paretic leg is started.
The gait goes on cyclically until the patient closes the handswitch. This can happen during every phase, and in that case the patient returns to the 'double support' immediately.

Two problems with the foot sensors are how to define the switching threshold and their short life-time. The sensors measure pressure which has to be translated to an on-off signal, and the switching thresholds depend on the exact position of the sensors under the foot. This placement is difficult to reproduce. So it remains to replace the foot sensors, e. g. by an inertial sensor which is introduced further in Chapter 5. Among other things, this sensor measures accelerations in all three directions. From this acceleration data it should be possible to detect the gait phase, but this is not a part of this pre-diploma thesis.

Another disadvantage of the foot sensors is the difficulty to separate voluntary starts of the gait and unvoluntary bounces. In this implementation the separation is done by an additional time condition at the end of the swing phase. If the 'step' takes a shorter time than the condition, it is interpreted as a bounce, if it takes more time, the 'step' is identified as a real step.

Stimulated muscles

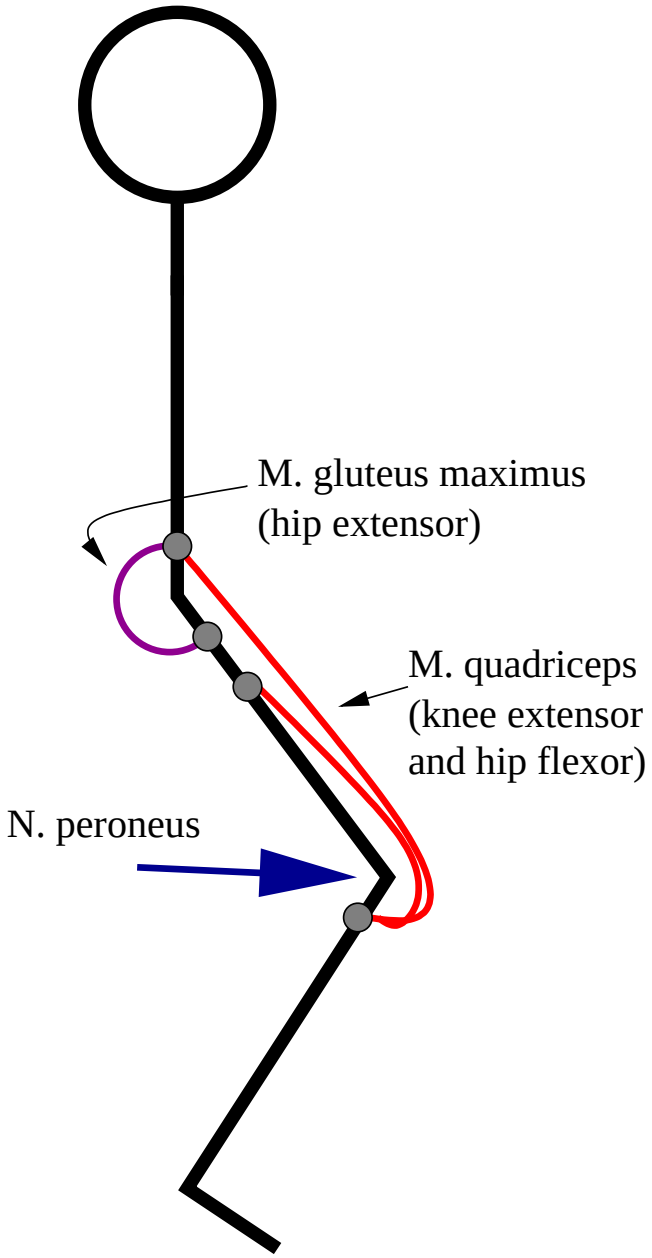


Figure 3.3: Stimulated muscles

Walking with the developed stimulation pattern

Figure 3.4 shows the motion sequence of a stride with all subphases and transitions (exept the transition 'close HS' that aborts the gait).
The treadmill is preset to run with constant velocity during the phases 2 to 5, that means it runs during the whole gait. This can be changed for every phase individual.

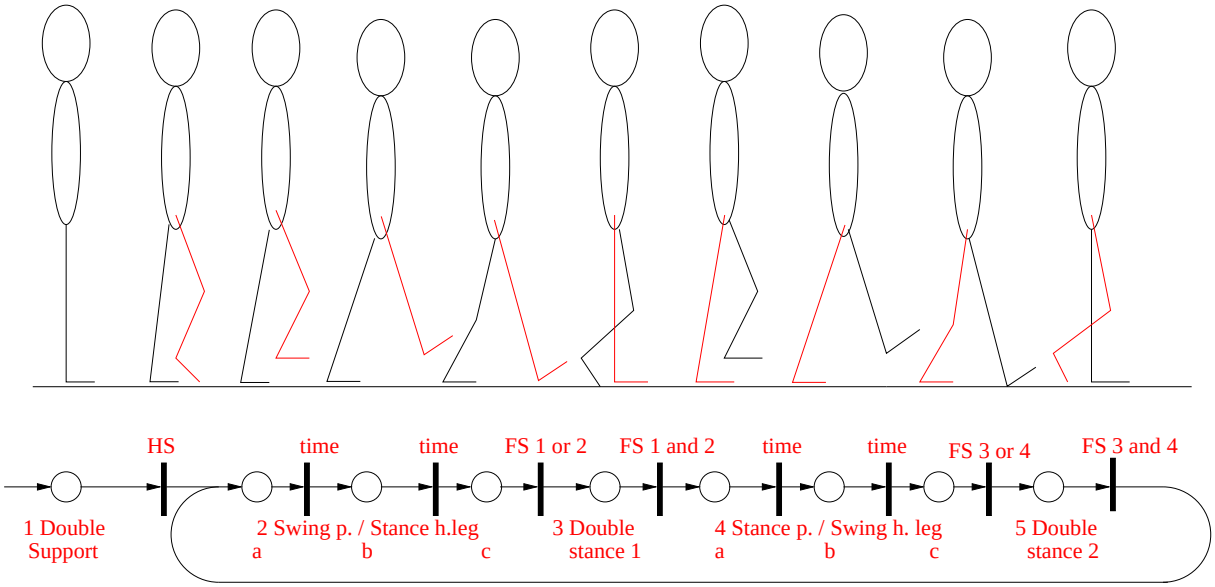


Figure 3.4: Motion sequence of a stride

Stimulation pattern

Table 3.1: Stimulation pattern

phase	paretic leg			healthy leg		
	n. peroneus	m. quadriceps	m. gluteus max.	n. peroneus	m. quadriceps	m. gluteus max.
1. double support		x	x		x	x
2. swing p. / stance h. a	x				x	x
2. swing p. / stance h. b		x			x	x
2. swing p. / stance h. c		x				x
3. double stance 1		x				x
4. stance p. / swing h. a		x	x	x		
4. stance p. / swing h. b		x	x		x	
4. stance p. / swing h. c			x		x	
5. double stance 2			x		x	

At the beginning of the gait (phase ’double support’) the patient is standing with parallel legs; his weight is distributed to both legs. Both m. quadricepes and m. glutei maximi are stimulated. The gait starts with lifting the paretic leg by stimulating its n. peroneus. Normally the heel is lifted before swing period starts. This detail is disregarded in the first step. From the second step on, the heel is automatically raised due to the weight shifting and the velocity of the gait.

After around one third of the swing period the n. peroneus is released, and the knee is extended by stimulating the m. quadriceps. The m. quadriceps is stimulated during the rest of the swing period, the shifting of the weight to the paretic leg in the phase ’double stance 1’ and during the first half of the swing period of the healthy leg. From the beginning of the swing period of the healthy leg also the m. gluteus maximus that extends the hip is stimulated. The m. gluteus maximus is released after the phase ’double stance 2’, where the next stride of this leg starts with stimulating the n. peroneus. In the stance period the m. quadriceps and the m. gluteus maximus are stimulated to stabilise the patient.

The stimulation pattern of the healthy leg is analogical with lower stimulation intensities; here the gait starts with the stance period.

3.2 Description of the implementation

The stimulation pattern is implemented in Matlab / Simulink. The best implementation is to design an automaton with finite states, one state for every gait phase resp. every circle in the petri net. Figure 3.5 shows the first level of the whole automaton. In the following, the automaton and the used Simulink blocks are described in detail.

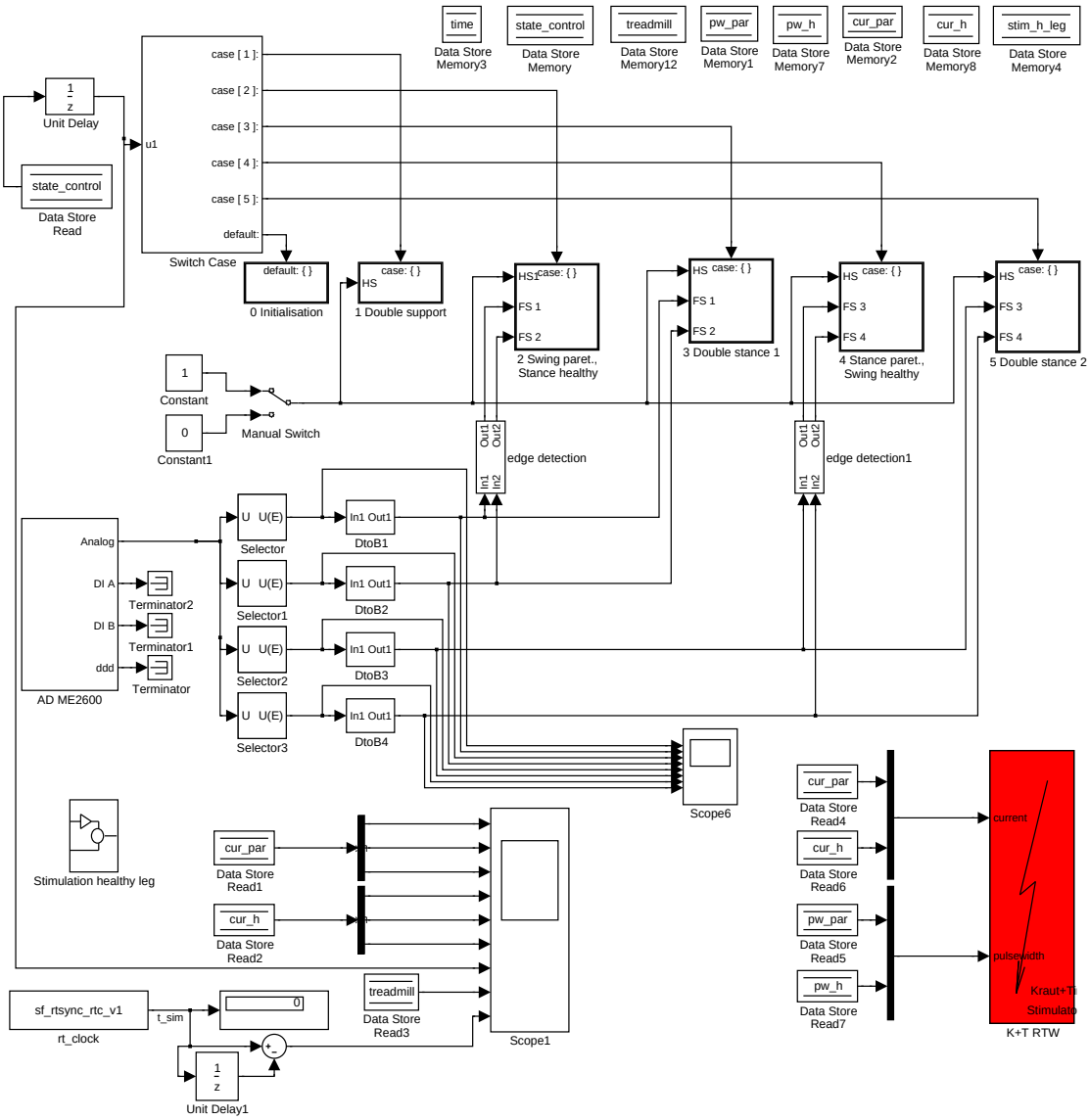
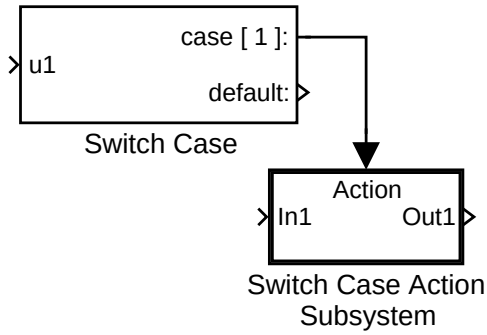


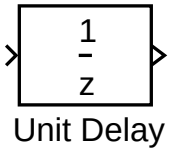
Figure 3.5: The whole automaton

Basic structure



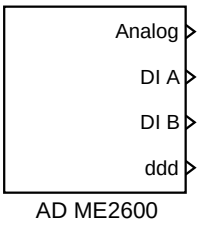
The basic structure of the automaton is a ’switch case’ block with 5 action subsystems and the default case. Every ’action subsystem’ is a finite state and represents a gait phase and so a circle in the petri net (Figure 3.2). At first it has to be controlled which state is executed at which time. This state control is done with the ’switch case’ block. The input of this block can only be a number, and according to this number the block decides about the signal way. That number (resp. the number of the next state) is generated inside the state subsystems. In the state subsystems also the stimulation pattern is implemented.
The state subsystems are described more detailed on the pages 23 to 26.

The number of the next state to be executed has to be put into the ’switch case’ block. The easiest way to do that is to use the ’data store’ blocks. Here the number of the next gait phase is put into a ‘data store write‘ block. This block writes its data into the ‘data store memory‘ block with the same name. The ‘switch case‘ block polls the data from the fitting ’data store read’ block.
The ’data store’ blocks have got several advantages compared to normal signal wires: They can save data, they are more clearly arranged, and it is more comfortable to change levels with them. It is only important that the ’data store memory’ block is placed in the highest level where the according signal appears.

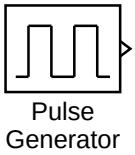


The number of the state that is generated inside the state subsystems is the number of the next state. That means that in the actual sample the ’switch case’ block has to read the number of the old sample. For this a ’unit delay’ block is interposed.
The ‘unit delay‘ block is also useful to avoid algebraic loops.

Including of the sensor signals

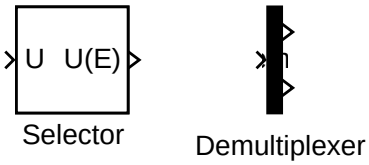


This is the block for the data acquisition card. It contains an user-defined s-function and four outputs for the different natures of the signals. With this block Simulink communicates with the foot pressure sensors and the handswitch.

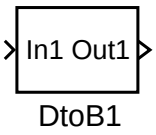


If the footswitches have to be simulated, it can be done with four ’pulse generator’ blocks, one for each footswitch. They can generate digital on/off signals.

The ‘terminator‘ block terminates signals and so it avoids warnings from Simulink about e.g. unconnected ports.



The ‘selector‘ block selects one or more signals from a vectorial signal. The output is also vectorial, but smaller sized. Here every selector selects one footswitch signal out of the 16 possible analog outputs of the s-function block. The ’demultiplexer’ doesn’t really select signals, but splits a vector signal into its single signals.



The next important step is to generate a boolean switch signal from the decimal foot pressure sensor data. For this purpose a decimal-to-boolean converter (DtoB) was created.

Figure 3.6 shows the inside of this subsystem ’DtoB’. The analog signal enters the converter at ‘in1‘. In the ‘if‘-subsystem, which works similar to the ‘switch case‘-subsystem, its value is compared to a lower and an upper threshold. If the value is between both thresholds, the old boolean value is used. For this hysteresis the old boolean value is returned to the action subsystem ’old value’. In the action subsystems the according boolean value is put into the data store. It must be noticed that the signal becomes inversed: Decimal 5 V means boolean 0, and decimal 0 V means boolean 1. At ‘out1‘ the boolean value leaves the DtoB block.

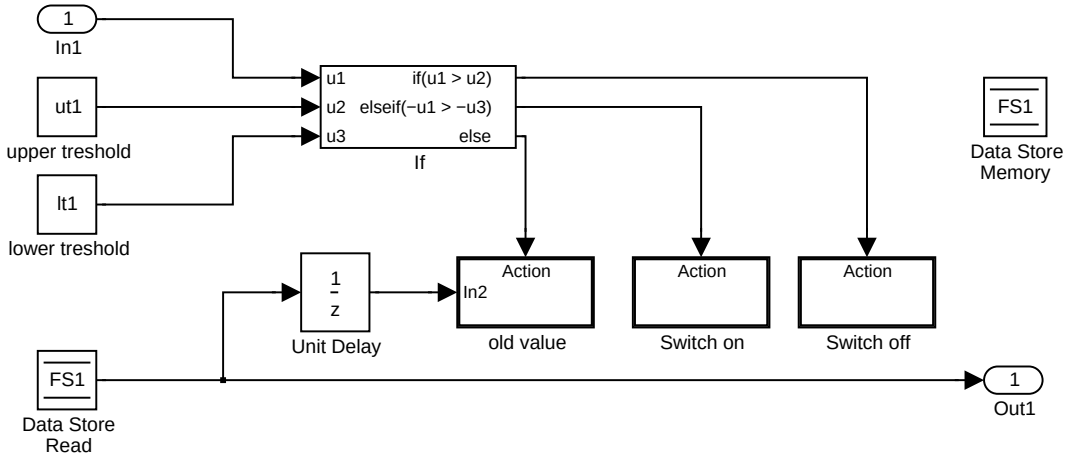
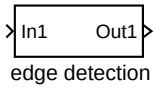
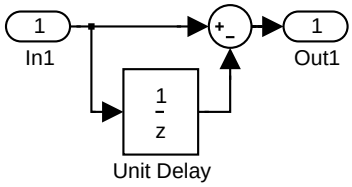
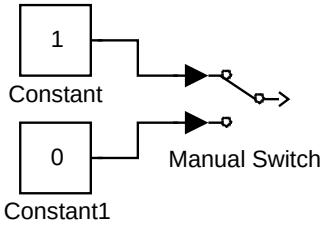


Figure 3.6: The inside of the decimal-to-boolean converter ’DtoB’





The ’swing / stance’ phases contain the whole swing period of one leg. The transitions from the ’swing / stance’ phases to the next phases are triggered by the footswitches of the swinging leg. The problem is that one of the relevant footswitches is still pressed for one sample time at the beginning of the phase. On the other hand, it should be unimportant which of the two footswitches is released resp. pressed first, so the state control cannot differentiate between the beginning and the end of the phase by regarding both relevant foot switches differently.
This problem can be avoided if not the signals but their changes are used, so it can be detected wether the switches are pressed, released or if nothing happens. Therefore the edge detection is interposed.



With this constellation the digital handswitch can be simulated. The block ’manual switch’ has got the advantage that it can be pressed during the simulation. While stimulating a person the handswitch is also included by the described s-function for the data acquisition card (see page 20), but here a digital port can be used. Furthermore the edge detection is not necessary, because the handswitch controls by an on/off-signal whether the whole gait is going on or aborted.

Examples of the ’switch case’ action subsystems

Figure 3.7 shows the inside of the state subsystem 1 (’double support’), Figure 3.8 shows the more complex inside of state 2 (’swing paretic leg, stance healthy leg’) as example for the states. All state subsystems contain mainly the state control group that detects the number of the next state, the intrinsic implementation of the stimulation pattern and an item about the treadmill.

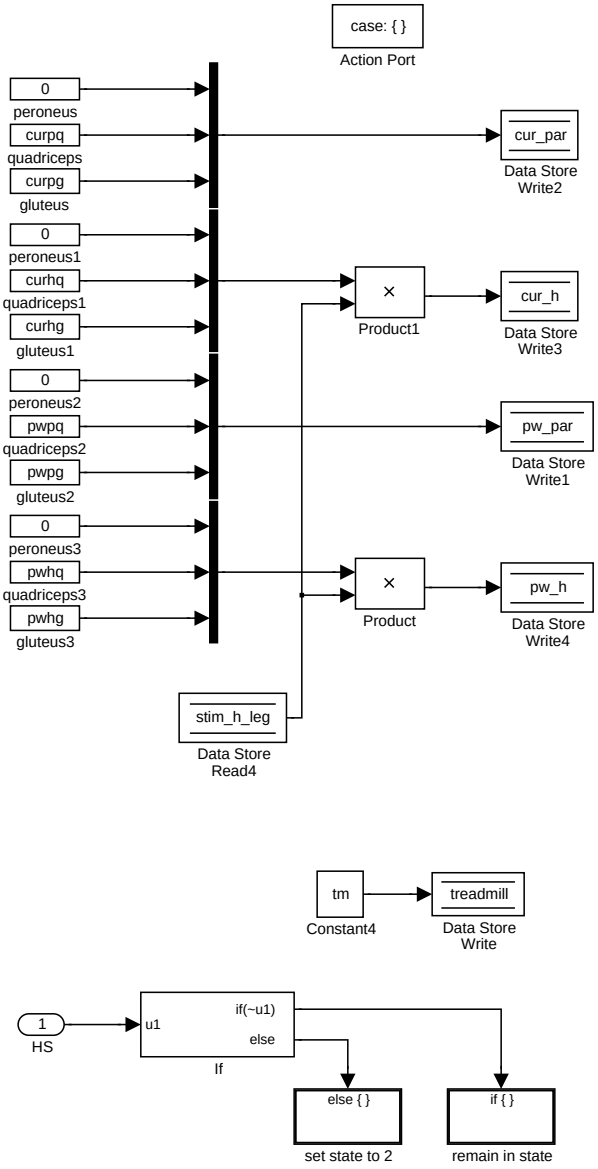


Figure 3.7: State subsystem ’Double Support’

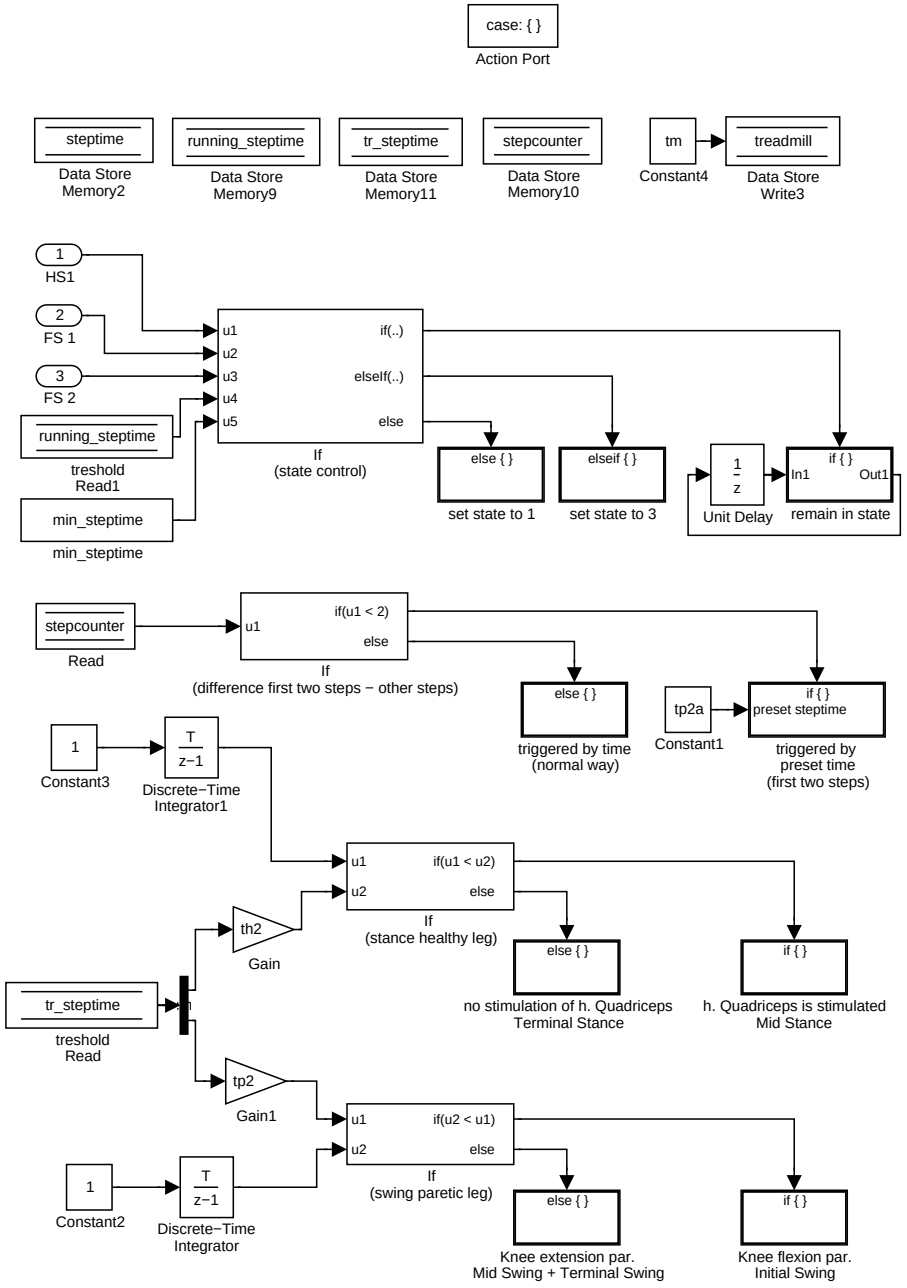
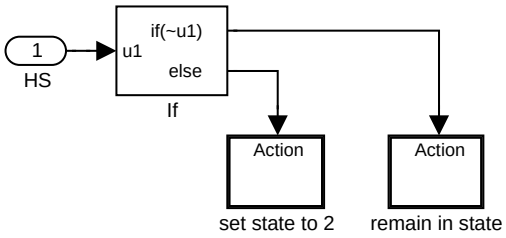


Figure 3.8: Subsystem ’Swing Paretic Leg, Stance Healthy Leg’

State Control

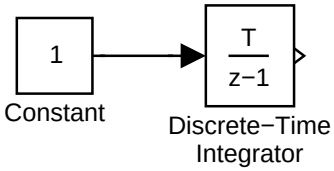


This is the state control group of phase 1 (’double support’). It generates the number of the next state. The ’If’-Block checks the incoming signal of the handswitch and decides which ’If-Action-Subsystem’ is executed. That means it decides if the patient stays in the actual gait phase or if he advances to the next phase. Inside these if-action subsystems the number of the next state (1 or 2) is written in the ‘Memory Store Write‘-block for the state control.
The state control groups of the other phases are little more complicated; their ‘If‘-block checks the handswitch and the interesting footswitches. There are also three possible next states: The actual one, the next state and state no. 1, which the patient is able to return to at every time by pressing the handswitch.

Implementation of the stimulation pattern

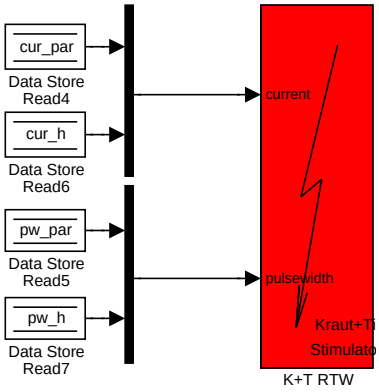
There is no stimulation at all in the Initialisation phase, which is the default case. The introduced stimulation pattern is implemented in the other phases. In the phases with constant stimulation, which are the phases 0, 1, 3 and 5, the data is simply put into the ‘Memory Store Write‘ blocks. The values of the currents and the pulse widths can be defined on the mask of the subsystem, also the value for the treadmill (whether it runs in that phase or not).

In the phases 2 and 4 the stimulation pattern changes during the phase (compare section 3.1). The swinging leg is at first lifted by stimulating the n. peroneus and then extended by stimulating the m. quadriceps. The supporting leg is at first stabilised by stimulating the m. quadriceps and the m. gluteus maximus and then the stimulation of the m. quadriceps is stopped.
For these time transitions inside the stance / swing phases it is necessary to measure the time which is needed for the whole phase. To measure the time each sample the block ’remain in state’ is executed, the sample time is added up. At the end of the phase (as soon as the ’set state to 3 resp. 5’ block is executed), the measured swing time is written into a memory store block, and the counter loop is set to zero for the next step. The needed swing time could also be measured once and then be used for the whole gait, but the speed of the patient can change during the gait. This is not too bad for the swinging leg, and also not for the supporting leg if the quadriceps is released later, but if the quadriceps of the supporting leg is released too early (this would happen if the patient would move slowlier, e.g. due to muscle fatigue) the patient could fall down.



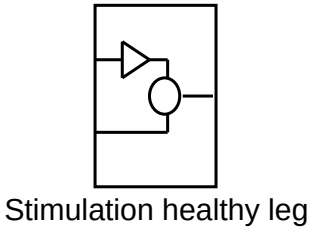
To obtain initial values the stride is performed two times with the same stimulation pattern in the subphases ’mid stance’ and ’terminal stance’. To count this two strides, a phase counter loop is attached in the ’set state to 3’ resp. ’set state to 5’ block.
From the third step on after partial times of the measured step time that can be chosen in the block ’gain’ resp. on the mask, the stimulation pattern is different for the three subphases.
The ‘Discrete-Time Integrator‘ block is useful for time-counters.

Excluding of the Currents

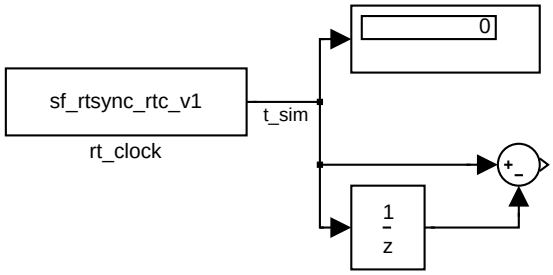


The current and the pulse width for each stimulated channel can be transmitted to the stimulator with this subsystem. It contains an user-defined S-Function.

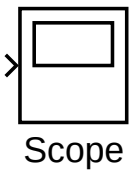
Miscellaneous



For some hemiplegic patients only the paretic leg has to be stimulated, for some patients it is better to stimulate both legs. The change between both modes should be very simple. The stimulation mode can be chosen easily inside the subsystem ’stimulation healthy leg’. For the stimulation of both legs the boolean value ’1’ is put into the data store ‘stim_h_leg‘, for the stimulation of only one side the boolean value is ’0’. All currents and pulse widths for the healthy limb are multiplicated by this boolean value.



This is the real-time clock, also written as S-Function. The running time is shown in the display; the signal leaving the sum is the exact length of every time step, so the accuracy of the clock can be watched.
Simulink executes all blocks of the model in a determined order. This order can be displayed in the menu 'Format' → 'Execution order'. The order can be changed by setting priorities for the block. This is done by clicking the block with the right mouse button, choosing the the dialog box 'block properties' and opening the general panel. In the property 'Priority' the execution priority of this block relative to other blocks in the model can be set. High numbers stand for low priorities.
The real-time clock waits until the general sample time is over when it is executed. That means it must be executed at last. If another block would be executed after the clock, the real-time would fail.



The signals can be plotted with the scope block. If a vectorial signal is put in, the different signals are plotted one upon the other. If the signals shall be plotted in different subplots, the 'Number of axes' in the menu 'Parameters' can be changed.

Simulated gait phases and the stimulation currents

To simulate the whole automaton the footswiches are simulated with pulse generators instead of including the footsensor signals. Figure 3.9 shows the resulting gait phases, the treadmill (wether it runs or not) and the currents belonging to the gait phases. The values of the currents are chosen arbitrary.

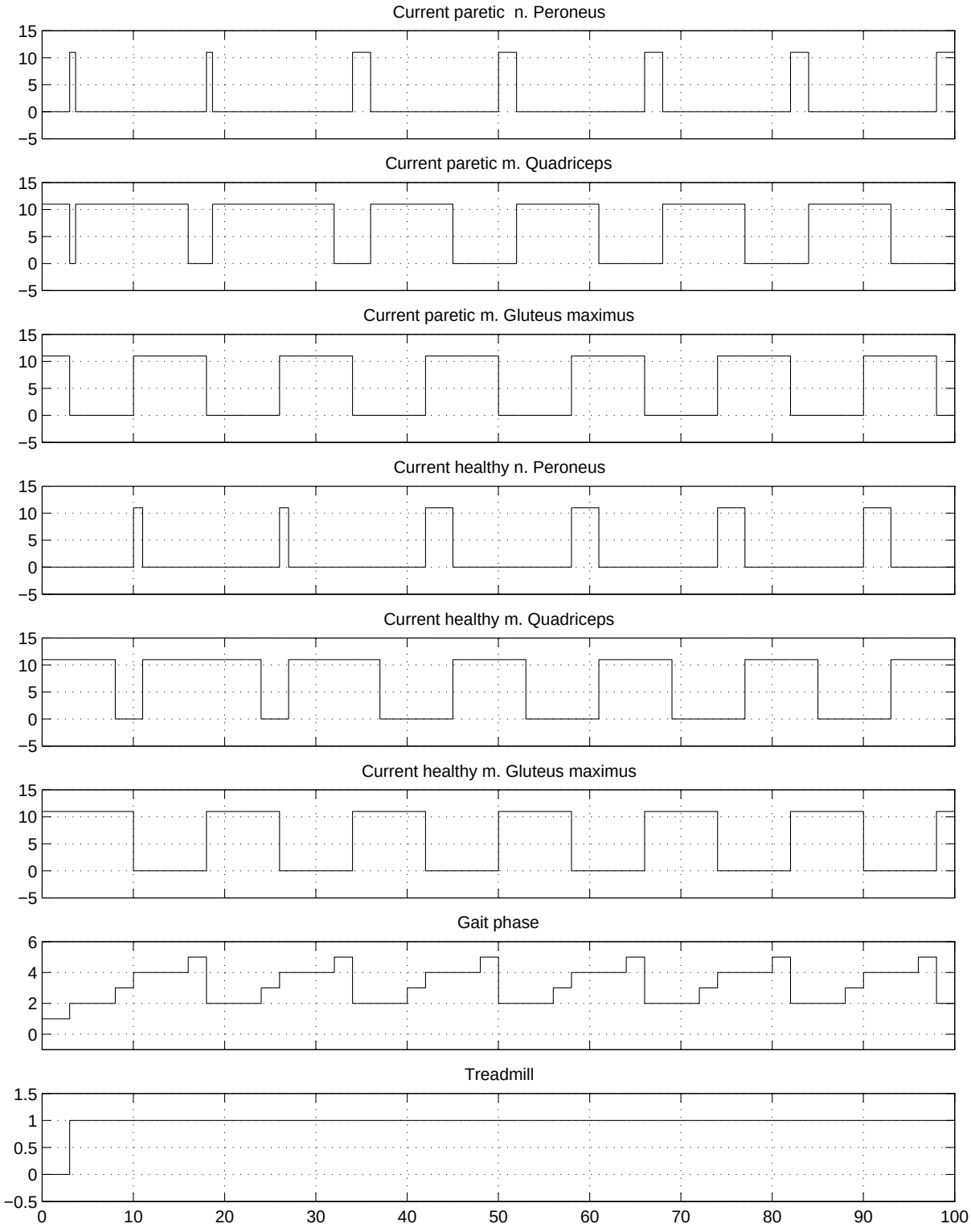


Figure 3.9: Gait phases and stimulation currents with simulated switches

Chapter 4

Stimulation program 'TransStim©'

4.1 Identifying of the gait phases

In Section 2.1 the stimulation program TransStim©, created by Mokrusch and Busch of the company Krauth + Timmermann [14], is introduced shortly. Here it is analysed more detailed. This description concentrates on the developement of the stimulation pattern. Similar to Chapter 3.1, this chapter starts with the selection of the gait phases. Next, the stimulation program is described by a petri net. In the third section the stimulated muscles are illustrated. The motion sequence of the stride is analysed and drawn physically in the forth section. The stance period is dealt with in the next section, and last the intrinsic stimulation is summarized in a table.

The stimulation program contains 16 gait phases.

01. Sitting	(Sitzen)
02. Standing up	(Aufstehen)
03. Stabilising 1	(Stabilisieren 1)
04. Stabilising 2	(Stabilisieren 2)
05. Double stance 1	(Doppelstand 1)
06. Double stance 2	(Doppelstand 2)
07. S3-Start	(S3-Los)
08. Snapping off 1	(Abknicken 1)
09. Snapping on 2	(Anknicken 2)
10. Sitting down	(Hinsetzen)
11. Begin step R	(Beginn Schritt R)
12. Continue step R	(Weiter Schritt R)
13. Finish step R	(Ende Schritt R)
14. Begin step L	(Beginn Schritt L)
15. Continue step L	(Weiter Schritt L)
16. Finish step L	(Ende Schritt L)

4.2 Description by petri net

The petri net of the stimulation program TransStim© [14] is shown in Figure 4.1. The switches S1 and S2 are foot switches. S1 is placed under the right toe, S2 under the left toe, and S3 is a handswitch.

In the following, only the stance and the walking will be discussed, not the standing up and the sitting down.

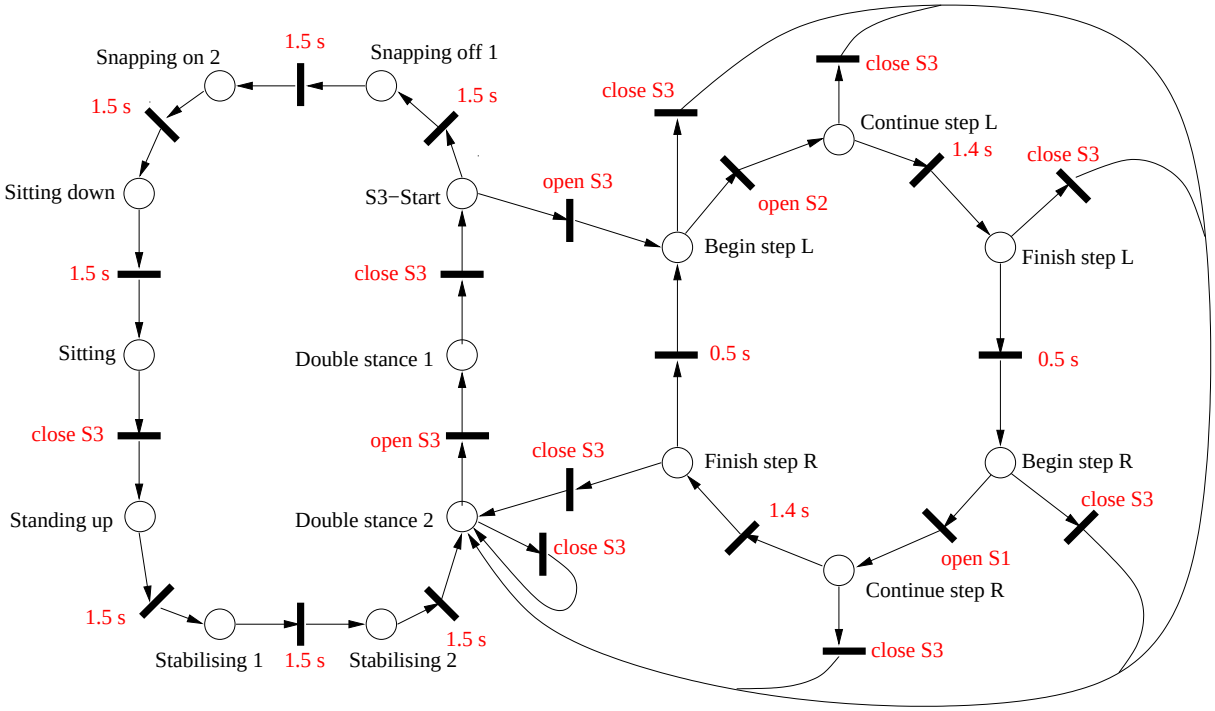


Figure 4.1: Petri net of the 3-switch-controlled walking cycle

4.3 Stimulated muscles

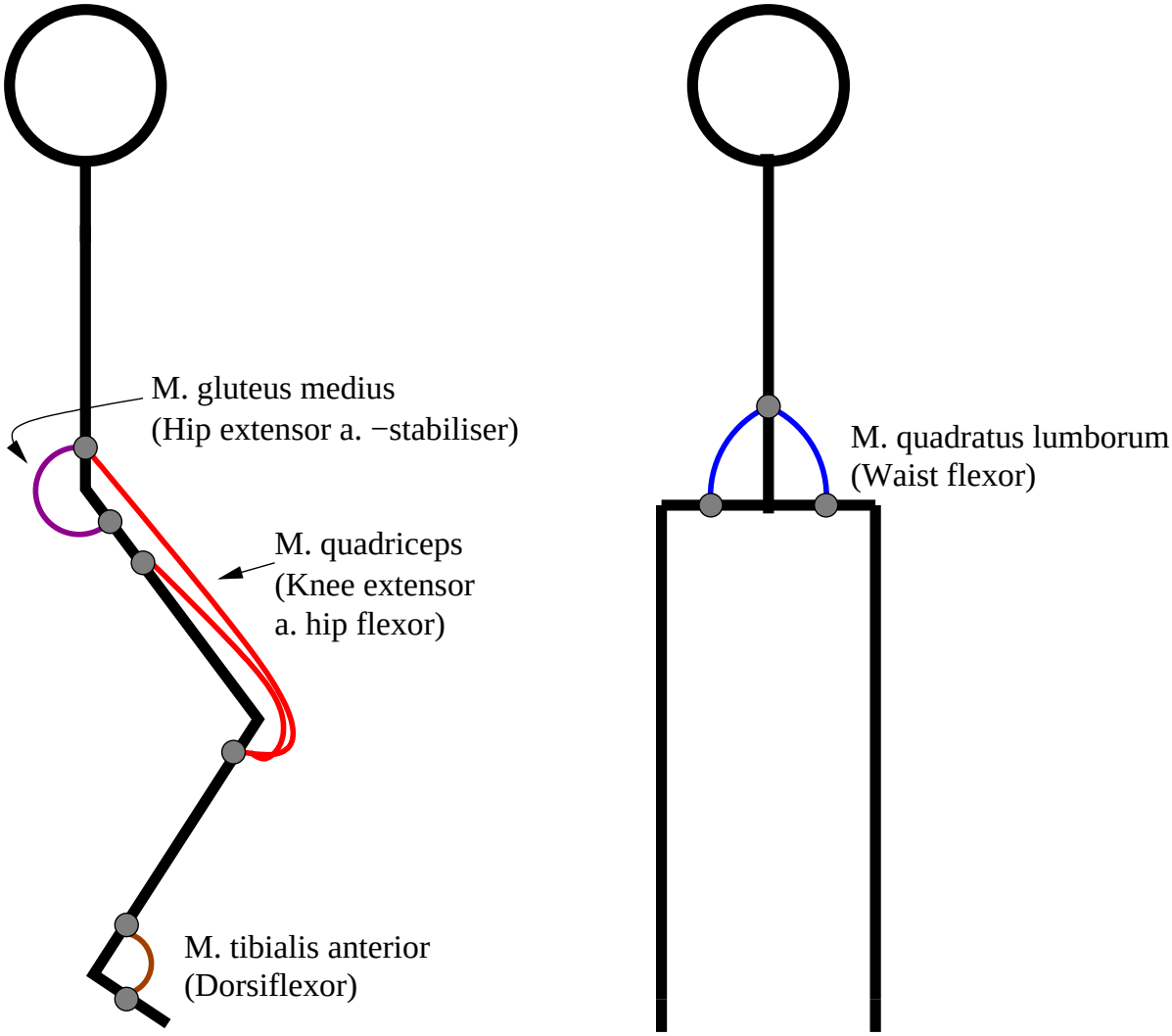


Figure 4.2: Stimulated muscles

4.4 Walking with the stimulation program TransStim©

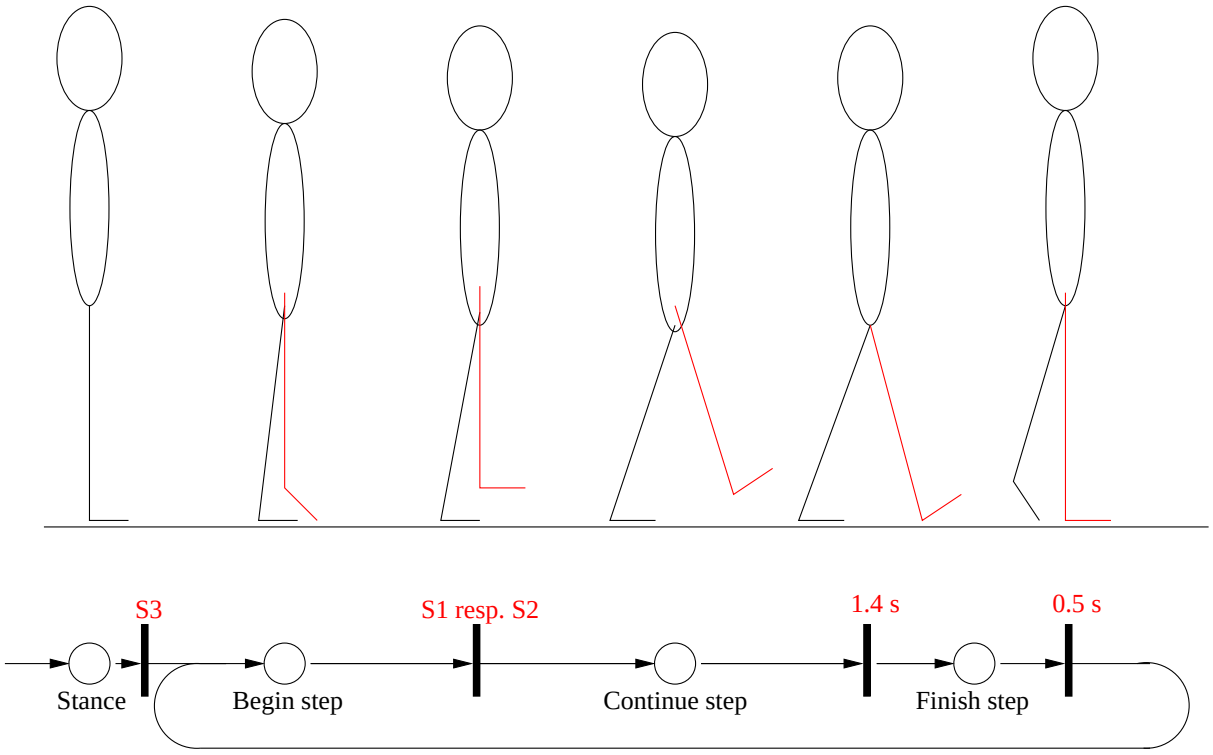


Figure 4.3: Sequence of a step

Basically, the gait starts with the left leg. During the whole gait the knee extensor and the hip extensor of both legs are stimulated with a pulse width of 500 us. In the phase ‘Begin step‘ that is started by opening the hand switch the waist flexor of the supporting leg and the dorsiflexor of the free leg are additionally stimulated. The former causes a laterally inclination and thereon the lifting of the free leg. The pulse width ramp is 0.1 s for all channels.

Under each toe there is one foot switch. As soon as the toe and with it the foot leaves the ground, the foot switch is opened and the phase ‘Continue step‘ is started. The pulse width ramp is changed to 0.5 s; there are no further changes in the stimulation. The intrinsic forward motion takes place by the treadmill which is running in this phase, so the supporting leg is pulled backwards. Alternatively, the patient can move forward by shifting his weight voluntary. The phase ‘Finish step‘ is reached 1.4 s later. The treadmill is stopped, and the dorsiflexor is nearly released with a pulse width of 10 us. The pulse width ramp is shortened to 0.2 s for all channels. In this phase the waist flexor is still fully stimulated, so the toe might hit the ground first; this still has to be clarified. The closing of the foot switches is ignored completely.

The next step starts 0.5 s later. From the second step on this happens automatically, and the walking has to be aborted by pressing the handswitch S3 again. This can be done during every phase. In that case, the patient returns to the stance immediately.

4.5 Stance

The single stance phases cannot be separated, because there is no visible difference in the stimulation pattern. Both knee and hip extensors are stimulated consistently with a pulse width of 500 us and a pulse width ramp of 0.5 s. So the accordant shiftings of weight have to be produced voluntary by the patient.

4.6 Stimulation pattern

Table 4.1: Stimulation pattern of TransStim© [14]

phase	left leg				right leg			
	m. gluteus medius	m. quadriceps	m. tibialis anterior	m. quadratus lumborum	m. gluteus medius	m. quadriceps	m. tibialis anterior	m. quadratus lumborum
05. Double Stance 1	x	x			x	x		
06. Double Stance 2	x	x			x	x		
07. S3 - Start	x	x			x	x		
11. Begin step R	x	x		x	x	x	x	
12. Continue step R	x	x		x	x	x	x	
13. Finish step R	x	x		x	x	x	(x)	
14. Begin step L	x	x	x		x	x		x
15. Continue step L	x	x	x		x	x		x
16. Finish step L	x	x	(x)		x	x		x

Chapter 5

Estimation of the three-dimensional gait trajectory

5.1 Design and functionality of the inertial sensor

The hemiplegic patient shall learn to walk as normal as possible. One issue of this normality is the requirement of equal lengths and heights of the step. To reach the equality in a closed loop control, these parameters must be measured resp. estimated. The gait parameters of the paretic leg shall be estimated as actual value, and the gait parameters of the healthy leg shall be used as desired value.

This chapter presents one method of estimating these gait parameters. Basically, the acceleration of the foot is measured by an inertial sensor and the gait parameters are estimated by double integration of the acceleration data after removing several disturbances. In this first section the sensor is introduced. In Section 5.2 the algorithm which estimates the gait parameters is presented. How the algorithm was verificated is described in Section 5.3. In this pre-diploma thesis the accuracy of the sensor is preliminary investigated. Following, the results are compared to an optical or acoustical reference system. In Section 5.4 the acoustical reference system is introduced. The setup of the experiments is illustrated in Section 5.5, and in Section 5.6 the results are discussed.

The inertial sensor used in this pre-diploma thesis is the MT9, created by Xsens Technologies B.V., The Netherlands[1]. It measures acceleration all in three directions, angular velocity around all three axes and additionally the heading of the sensor in respect of the earth magnetic field.

The description of the measurement principles is adapted from Luinge [4], Schlossbauer [12] and the technical documentation of Xsens [18]. Figure 5.1 illustrates the principle of a single-axial accelerometer. The accelerometer consists of a mass held by a spring. The unit vector $\vec{n}$ represents the sensitive axis of the sensor, that means the mass is allowed to move only in this direction. The gravity $\vec{g}$ and the acceleration $\vec{a}$ move the mass from its rest position (the rest position of the mass is drawn dashed). The distance d between the mass and the housing is

[1] www.xsens.com

measured. From this distance data the affected acceleration can be calculated.
The electrical signal $s_{A,n}$ is modelled as

$$s_{A,n} = k_{A,n} \cdot (\vec{a} - \vec{g}) * \vec{n} + \vec{bias}_{A,n} \tag{5.1}$$

$k_{A,n}$ - scaling factor
$\vec{bias}_{A,n}$ - bias

For a tri-axial accelerometer three single-axis accelerometers can be mounted together. It is also possible to use one single mass with three translation degrees of freedom. Here the distances are measured capacitively.

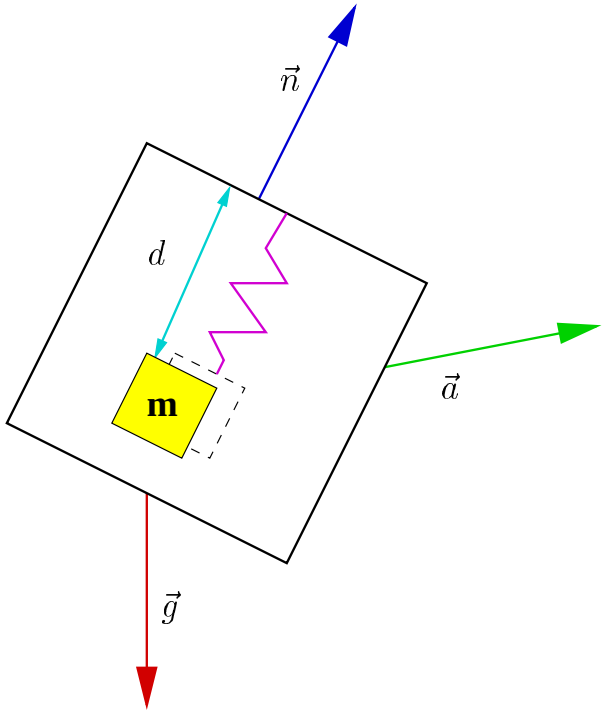


Figure 5.1: Measuring principle of a single axis accelerometer, adapted from Luinge [4]

The measuring principle of the gyroscopes is illustrated in Figure 5.2. One single gyroscope consists of a mass which is brought into vibration by an actuator in direction given by $\vec{r}_{act}$. If the housing is rotated, the mass experiences a force perpendicular to the angular velocity and to the direction of the actuation. This force is called coriolis force. It is only apparent in the sensor coordinate system (described in Section 5.2). It is proportional to the angular velocity and the momentary mass speed. The exact relation is given by

$$f_C = 2m \cdot v \cdot \omega \tag{5.2}$$

f_C	-	coriolis force
m	-	mass
v	-	momentary mass speed
ω	-	angular velocity

The coriolis force can be obtained by measuring the actual distance between the mass and the housing.

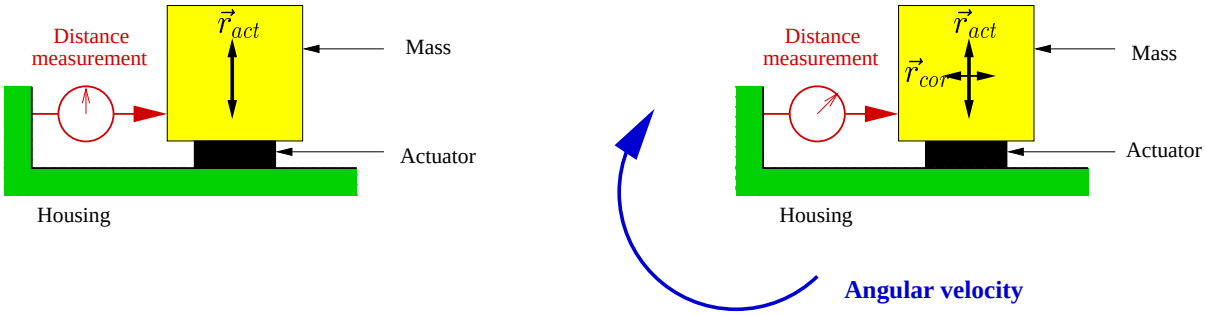


Figure 5.2: Measuring principle of a single axis gyroscope, adapted from Luinge [4]

Like the accelerometers, a tri-axial gyroscope can be designed by mounting three single-axis gyroscopes together.

The Xsens inertial sensor contains a tri-axial accelerometer, a tri-axial gyroscope and a tri-axial magnetometer that works like a compass and measures the heading of the sensor with respect to the magnetic earth field. The problem with the magnetometers is that they are disturbed if they are used near iron, which changes the magnetic field locally. Because of the fact that a treadmill, on which the patients should learn to walk again, contains a lot of iron, the magnetometers are not used in this pre-diploma thesis.

For miniaturising the sensor its mechanical components are etched in silicon, so the housing's dimensions (height x breadth x depth) can be small (28.73 x 53.5 x 38.5 mm).

The three axes of the accelerometer and the three axes of the gyroscope are put together to a right-handed coordinate system. Figure 5.3 shows a drawing of the sensor with its body fixed coordinate system.

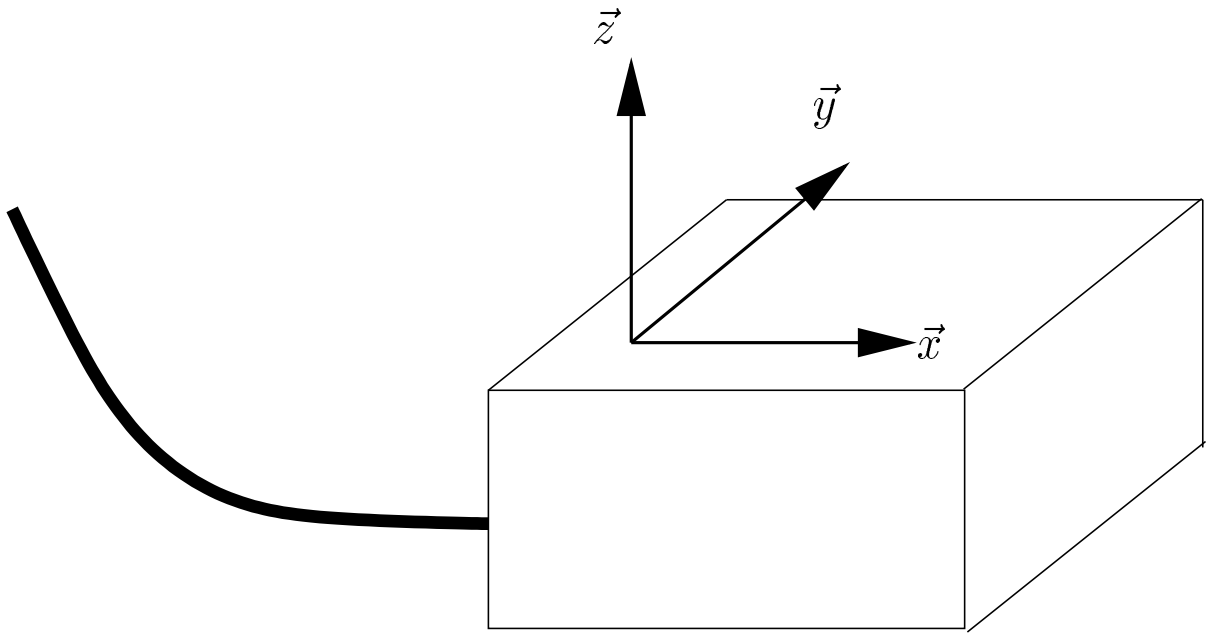


Figure 5.3: Sensor with body fixed coordinate system

5.2 Estimation of the 3D-trajectory

The sensor coordinate system

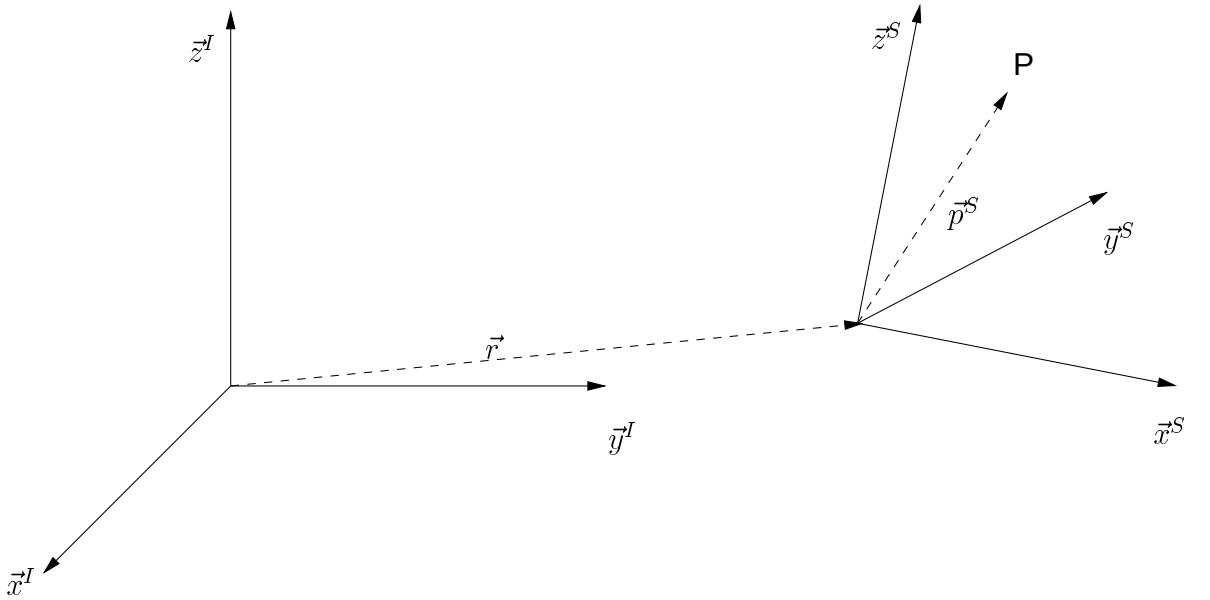


Figure 5.4: Inertial coordinate system and sensor coordinate system

The sensor measures its data in its own body fixed sensor coordinate system (S) which is also carthesian, but rotated in respect to the earth fixed inertial coordinate system (I). Figure 5.4 shows the inertial reference and the sensor coordinate system. The mathematical background is taken from N.-O. Negård [7]. The orientation of the sensor coordinate system with respect to the inertial coordinate system can be described with the rotation vector $\vec{r}$. For a better handling, this rotation vector is formed into the rotation matrix R. The point P, handled as vector from the point of origin to the point P is known in sensor coordinates. It can be expressed as components along the three coordinate axes of the coordinate system. These axes are representated by unit vectors.

$$\vec{p}^S = p_x^S \vec{x}^S + p_y^S \vec{y}^S + p_z^S \vec{z}^S \tag{5.3}$$

p_x^S, p_y^S, p_z^S - components along the according axes
$\vec{x}^S, \vec{y}^S, \vec{z}^S$ - unit vectors

This notation can be simplified by using one vector which contains only the components of the vector $\vec{p}$ along the axes.

$$\vec{p}^S = \begin{bmatrix} p_x^S \\ p_y^S \\ p_z^S \end{bmatrix} \tag{5.4}$$

In the inertial coordinate system, the vector $\vec{p}$ is described like this:

$$\vec{p^I} = p_x^I \vec{x}^I + p_y^I \vec{y}^I + p_z^I \vec{z}^I \tag{5.5}$$

The relation between both coordinate systems can be calculated as follows:

$$\begin{aligned} p_x^I &= (p_x^S \vec{x}^S + p_y^S \vec{y}^S + p_z^S \vec{z}^S)^T \vec{x}^I \\ p_y^I &= (p_x^S \vec{x}^S + p_y^S \vec{y}^S + p_z^S \vec{z}^S)^T \vec{y}^I \\ p_z^I &= (p_x^S \vec{x}^S + p_y^S \vec{y}^S + p_z^S \vec{z}^S)^T \vec{z}^I \end{aligned} \tag{5.6}$$

Now the rotation can be expressed as rotation matrix R. The described operation is linear.

$$\begin{bmatrix} p_x^I \\ p_y^I \\ p_z^I \end{bmatrix} = \begin{bmatrix} (\vec{x^S})^T \vec{x^I} & (\vec{y^S})^T \vec{x^I} & (\vec{z^S})^T \vec{x^I} \\ (\vec{x^S})^T \vec{y^I} & (\vec{y^S})^T \vec{y^I} & (\vec{z^S})^T \vec{y^I} \\ (\vec{x^S})^T \vec{z^I} & (\vec{y^S})^T \vec{z^I} & (\vec{z^S})^T \vec{z^I} \end{bmatrix} \begin{bmatrix} p_x^S \\ p_y^S \\ p_z^S \end{bmatrix} \tag{5.7}$$

This equation is written more compactly:

$$\vec{p^I} = R * \vec{p^S} \tag{5.8}$$

So the vector $\vec{p}$ can be transformed by multiplying it with the rotation matrix R to obtain its representation in inertial coordinates.

The algorithm in general

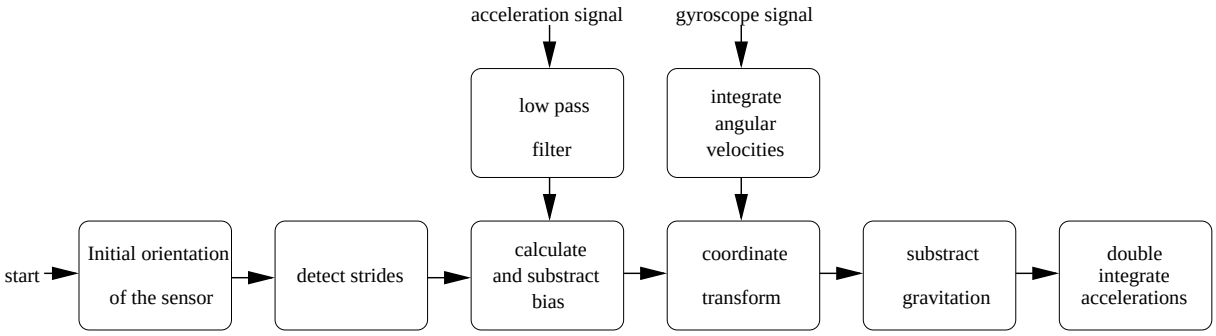


Figure 5.5: Assessment of the 3D-trajectory, adapted from Veltink at al. [15]

The acceleration signal of the sensor contains not only the real acceleration, but also other components like the gravitation and disturbances. For a better estimation, the signal is modelled as

$$\vec{y} = \vec{a} + \vec{g} + \vec{bias} \tag{5.9}$$

$\vec{y}$	-	acceleration signal
$\vec{a}$	-	real acceleration
$\vec{g}$	-	gravitation
$\vec{bias}$	-	disturbances, modelled as bias

Both, gravitation and bias have to be removed before the distance can be estimated by double integrating the acceleration.
Figure 5.5 shows the skeletal structure of the algorithm which estimates the three-dimensional trajectory of the sensor. At first the initial orientation is estimated. This is necessary, because normally the sensor is not placed exactly even on the foot. Afterwards the detection of the strides and the boundaries of the integration have to take place. The next step is the calculating and substracting of the bias from the low pass filtered acceleration data. After that the orientation of the sensor is estimated as rotation matrix by integrating the angular velocity data. Now for every sample the acceleration is transformed into the inertial coordinate system and the gravitation fraction can be removed. Next the obtained acceleration data can be integrated two times, and the gait parameters step length and step height can be calculated. The step length (referring to the swing period) and the whole stride length should be the same, because the foot does not move during the stance period. In reality, there are big differences between the step length and the stride length, so the used data for the integrations has to be limited to the steps.

In the following, the used formulae are developed. The index '$bias$' means that the variable still contains the bias. According to this, the index '$grav$' is used for variables containing the gravitation. The exponent describes the coordinate system the variable refers to. 'I' means the inertial and 'S' the sensor coordinate system. A lower case character with an arrow atop ($\vec{a}$) represents a vectorial variable, a capital letter (A) a matrix.

Calculation of the orientation

The gyroscopes suffer from bias gravely. This fact leads to complicated algorithms for calculating the orientation. Because of their complexity these algorithms are not part of this pre-diploma thesis but taken from N.-O. Negård [7]. Like Veltink et al. [5], Negård uses a Kalman filter. The filter is indirect, this means that the orientation errors are used as the states in the Kalman filter. The principle of this calculation is illustrated in Figure 5.6.

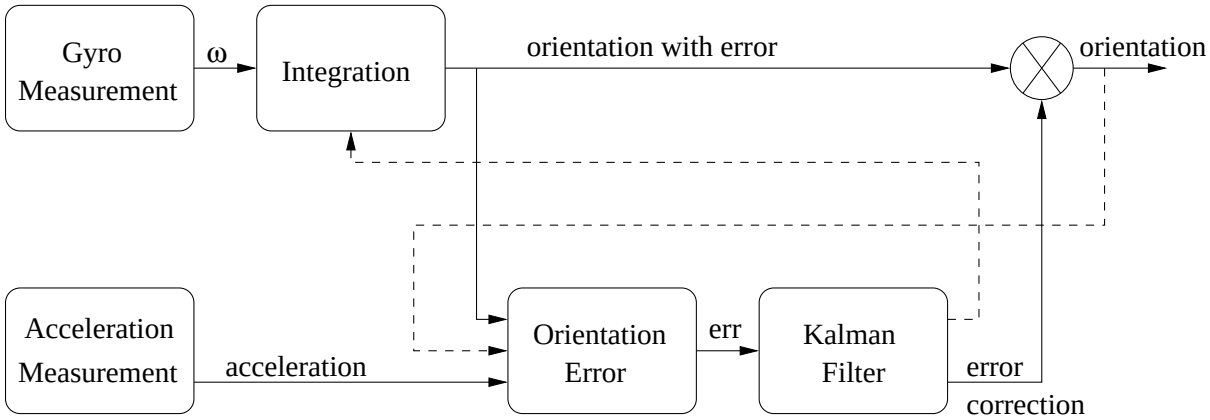


Figure 5.6: Calculation of the orientation, adapted from Negård [7]

First an error model for the orientation error is derived. In this model the acceleration data is also included. The error model assumes that the error is relatively small. Next the initial orientation is calculated. The assumption is made that there is no acceleration at the beginning of the movement, so the sensor measures only the gravitation. In this calculation the bias of the accelerometer is disregarded. From the distribution of the gravitation with respect to the axes, the orientation of the sensor can be calculated by finding the axis of rotation and the angle to rotate. Internally, this algorithm describes the rotation by using quaternions instead of rotation matrices. This is another mathematical representation of the same issue. Now the gyro and the acceleration data are feeded as illustrated into the Kalman filter. The Kalman filter distributes the total error into particular errors of the different components of the orientation signal. Last the particular errors are removed from the orientation signal.

Detection of the steps

One important issue of the estimation of gait parameters is the detection of the steps. If the beginning and the end of a step were unknown, there could be no estimation of the length. The steps are be detected by the use of the Xsens acceleration data. If the acceleration data would be perfect, there would be no change in the data between one step and the next, because during stance the foot should not move or accelerate. At the beginning of the stride there would be a peak pointing in positive z-direction, and the stride would stop after a peak in negative z-direction. The detection of start and stop would be very easy here.

The problem about real data is the noise in the signal and the vibration of the sensor during leaving the ground and during touching the ground again. Also the sensor cannot be placed exactly onto the foot, so the acceleration peaks at start and stop are distributed to all directions. So it is difficult to find the true values, and small errors in the bounderies of the steps lead to large errors in the estimated movement length.

First, the acceleration is summed to the total acceleration and the gravitation is removed.

$$a_{total} = a_x + a_y + a_z - |\vec{g}| \tag{5.10}$$

Now the data is filtered by a mean filter. For every value the mean and the standard deviation of the value and its neighbors are calculated. If both are small enough, the value is assumed not to be part of a step. If only the mean were small enough, the detected zero value could also be the zero-crossing of the acceleration between positive and negative acceleration (retardation). If only the standard deviation were small enough, phases of constant acceleration during the step could be detected wrong. By filtering the data again, this time with a median filter which removes outliers, little failures are corrected.

If the parameters are tuned well, the obtained data looks like pulses which are one during the steps and zero between the steps. With an edge detection the starts and stops are extracted from this pulselike data. Because of the fact that the mean filtering blurs the exact edges, the starts are shifted forward a few values and the stops are shifted backwards a few values. One problem is that the increases and the decreases of the pulses often takes two samples. To avoid this error, it is checked wether the value before is also a start resp. stop value or not before the value itself is declared as start or stop. Figure 5.7 shows the total acceleration and the detected starts as peaks of the magnitude ten pointing up and stops as peaks of the same magnitude pointing down. The experiment the data is taken from is described in Section 5.4. There were no real steps, but simple movements.

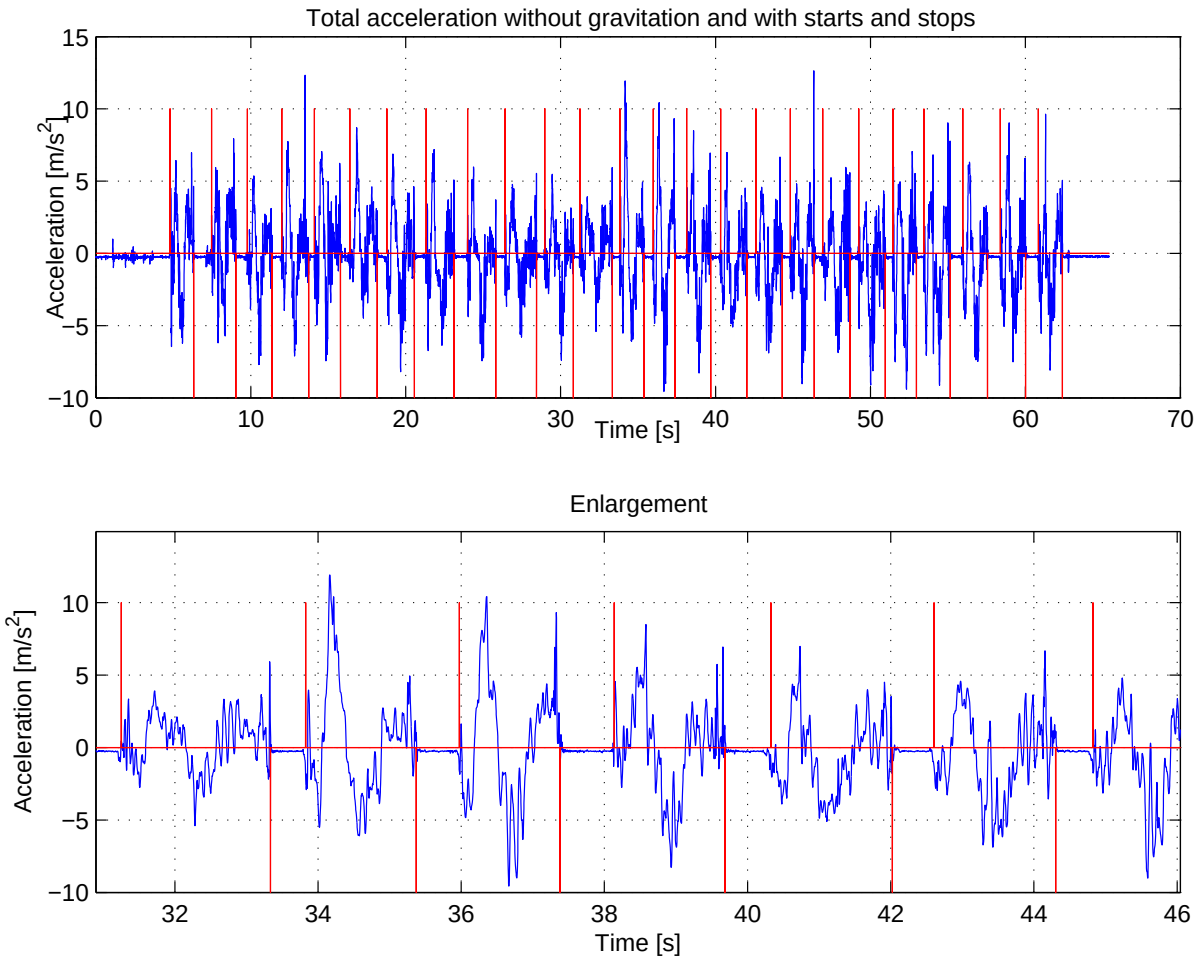


Figure 5.7: Detection of the strides

Estimation of the bias

In every sensor signal there is are disturbances included, here modelled as bias. This bias is assumed to be constant, but the error due to the bias increases during the integration of the data, so the bias has to be estimated and removed. The assumption is made that the real velocities of the foot at the begin and the end of a step are zero, so the calculated velocity at the end corresponds to the bias. The estimated velocity at the begin of the step is zero, because the influence of the bias in the acceleration signal on the velocity estimation increases during the integration. Therefore, the influence of the bias on the velocity is zero at the begin and maximum at the end of the step.

In the assumed model about the acceleration signal (see Equation 5.9: $\vec{y} = \vec{a} + \vec{g} + \vec{bias}$), the bias is caused inside the sensor. Therefore, it has to be estimated in sensor coordinates. The velocity and the path are estimated with respect to the (inertial) ground, so every integration must occur in inertial coordinates.

First, the acceleration is measured and low pass filtered in sensor coordinates. The data contains the gravitation and the bias. So the gravitation which is known in inertial coordinates is transformed to sensor coordinates and substracted from the acceleration data for every sample. Afterwards the data is transformed to inertial coordinates, because the integrations have to take place in inertial coordinates.

$$\vec{a}^I_{bias}(k) = R(k) * (\vec{a}^S_{bias,grav}(k) - R^{-1}(k) * \vec{g}^I) \tag{5.11}$$

$\vec{a}$ - acceleration
k - sample
R^{-1} - inverted rotation matrix
$\vec{g}$ - gravitation vector, $9.81 \frac{\mathrm{m}}{\mathrm{s}^2}$ in z-direction

To obtain the velocities, the data has to be integrated. The trapezoidal rule is used for this.

$$\vec{v}^I_{bias}(k) = \sum_{i=1}^{k} \frac{t_s \left(\vec{a}^I_{bias}(i-1) + \vec{a}^I_{bias}(i)\right)}{2} \tag{5.12}$$

$\vec{v}$ - velocity
t_s - sample time

The next step is the intrinsic calculation of the bias. The signal still contains the acceleration and the bias, and the equation is set up in inertial coordinates.

$$\vec{a}^I_{bias}(k) = \vec{a}^I(k) + \vec{bias}^I(k) \tag{5.13}$$

The bias is assumed to be constant for every step, but this assumption is only valid in the sensor coordinate system. So the bias is estimated in sensor coordinates and transformed into inertial coordinates for every sample.

$$\vec{a}^I_{bias}(k) = \vec{a}^I(k) + R(k) * \vec{bias}^S \tag{5.14}$$

Now the equation is integrated. For a better understanding, the integration is firstly described continuous.

$$\int_{t=start}^{stop} \vec{a}^I_{bias}(\tau)d\tau = \int_{t=start}^{stop} \vec{a}^I(\tau)d\tau + \int_{t=start}^{stop} (R(\tau) * \vec{bias}^S)d\tau \tag{5.15}$$

The first two terms are integrated discrete by the trapezoidal rule described in equation 5.12, the last term by the easier Forward Euler integration.

$$\vec{v}^I_{bias}(k) = \vec{v}^I(k) + t_s \sum_{i=start}^{stop} (R(i) * \vec{bias}^S) \tag{5.16}$$

Next the equation is applicated for the end of the movement. As assumed, the real velocity at the beginning and at the end of the movement is zero. Also, the velocity with bias at the start of the step is zero.

$$\vec{v}^I_{bias}(stop) - \underbrace{\vec{v}^I_{bias}(start)}_{=0} = \underbrace{\vec{v}^I(stop)}_{=0} - \underbrace{\vec{v}^I(start)}_{=0} + t_s \sum_{i=start}^{stop} (R(i) * \vec{bias}^S) \tag{5.17}$$

Now the equation is turned, and that the bias can be estimated.

$$\vec{bias}^S = \frac{\left(\sum_{i=start}^{stop} R(i)\right)^{-1} * \vec{v}^I_{bias}(stop)}{t_s} \tag{5.18}$$

Estimation of the 3D-trajectory

Now the bias can be removed from the low pass filtered acceleration signal by substracting it.

$$\vec{a}^S(k)_{grav} = \vec{a}^S_{bias,grav}(k) - \vec{bias}^S \tag{5.19}$$

After that the data is transformed into the inertial coordinate system.

$$\vec{a}^I(k)_{grav} = R(k) * \vec{a}^S(k)_{grav} \tag{5.20}$$

In the inertial coordinate system, the gravitation can be removed.

$$\vec{a}^I(k) = \vec{a}^I(k)_{grav} - \vec{g}^I \tag{5.21}$$

Now the data should only contain the real acceleration data in inertial coordinates. So it can be integrated two times in discrete time by using the trapezoidal rule to obtain the path.

$$\vec{v}^I(k) = \sum_{i=1}^{k} \frac{t_s \left(\vec{a}^I(i-1) + \vec{a}^I(i)\right)}{2} \tag{5.22}$$

$$\vec{s}^I(k) = \sum_{i=1}^{k} \frac{t_s \left(\vec{v}^I(i-1) + \vec{v}^I(i)\right)}{2} \tag{5.23}$$

$\vec{s}^I$ - path

At last the step parameters height and length can be calculated. The maximum of the z-values is used for the height. The rotation matrix refers to the earth coordinate system. Normally, the direction of the step does not correspond to the direction of the earth magnetic field, so the estimated step lengths in x- and y-direction have to be summed up vectorial. It is also noticeable that only the last values are used for these step parameters. As mentioned in Section 1.3, all start data is voluntary set to zero.

$$height = max(s_z^I) \tag{5.24}$$

$$distance = \sqrt{(s_{x,end}^I)^2 + (s_{y,end}^I)^2} \tag{5.25}$$

5.3 Verification of the algorithm

To verify the algorithm, it is tested with some simulated acceleration data. This data is generated in Matlab / Simulink. The used Simulink model is pictured in Figure 5.8. Figure 5.9 shows the simulated data in all three directions, the first three subplots show the data in x-direction, the last three subplots the data in y- and z-direction, which are identical. With four ramps a trapezoidal signal is generated which is plotted as second and fourth subplot of Figure 5.9. In x-direction this signal represents the velocity, and in y- and z-direction the path, all of them in inertial coordinates. The x-direction acceleration resp. the y- and z-direction velocity are calculated by differentiating the trapezoidal signal one time. The x-direction path is calculated by integrating the trapezoidal signal, for the accelerations in y- and z-direction the trapezoidal signal is differentiated a second time.

For the further calculations only the simulated acceleration signals are used. In z-direction the gravitation is added, and the three signals are multiplexed to one vectorial signal. The next step is the transformation of the vectorial acceleration signal into 'sensor' coordinates. The rotation matrices which represent the change in the orientation are taken from an arbitrary data record. After the transformation to 'sensor' coordinates, some constants which represent the bias are added. Now this signal is feeded into the algorithm.

The results of the algorithm are plotted in Figure 5.10. The first subplot shows the trajectory three-dimensional, in the second subplot the data is represented in three two-dimensional trajectories (the trajectories for the y- and the z-direction are identical). The step length is estimated as 2.0004 m instead of 2.0000 m, but this is a numerical error. The step height is estimated correctly as 1.0000 m. The bias is also identified correctly. That means that the algorithm works correct and can be applicated on real data.

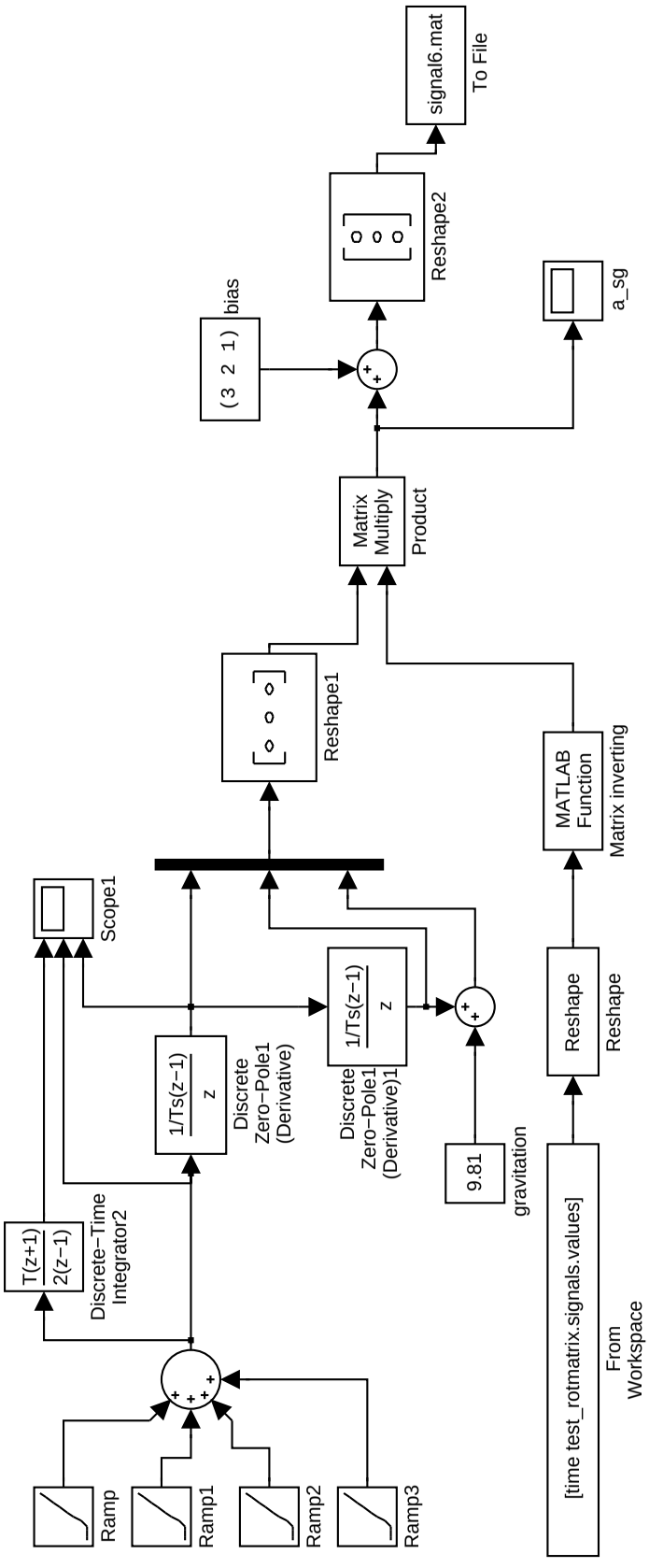


Figure 5.8: Simulink model for simulating the acceleration data

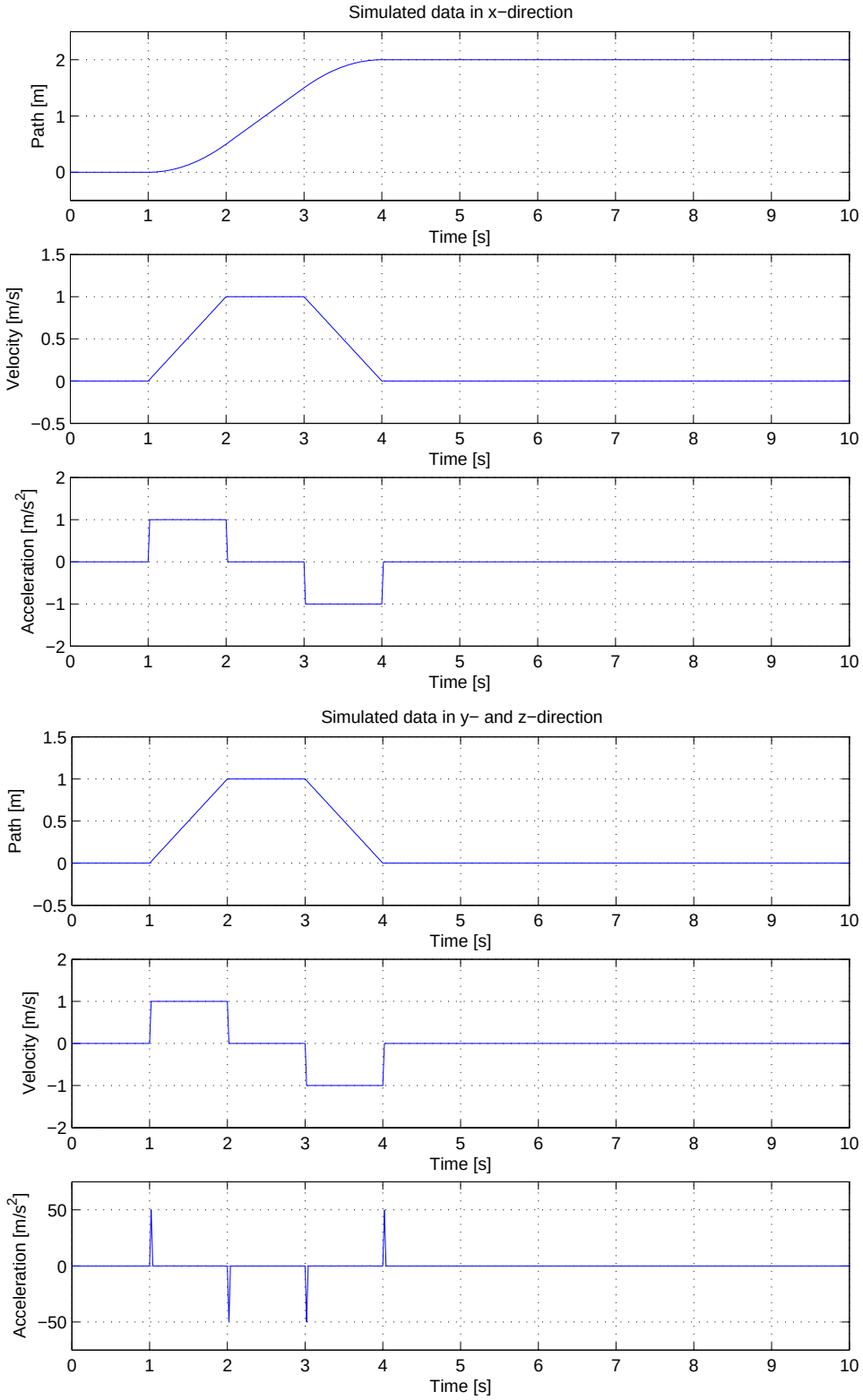


Figure 5.9: Simulated data in x-, y- and z-direction

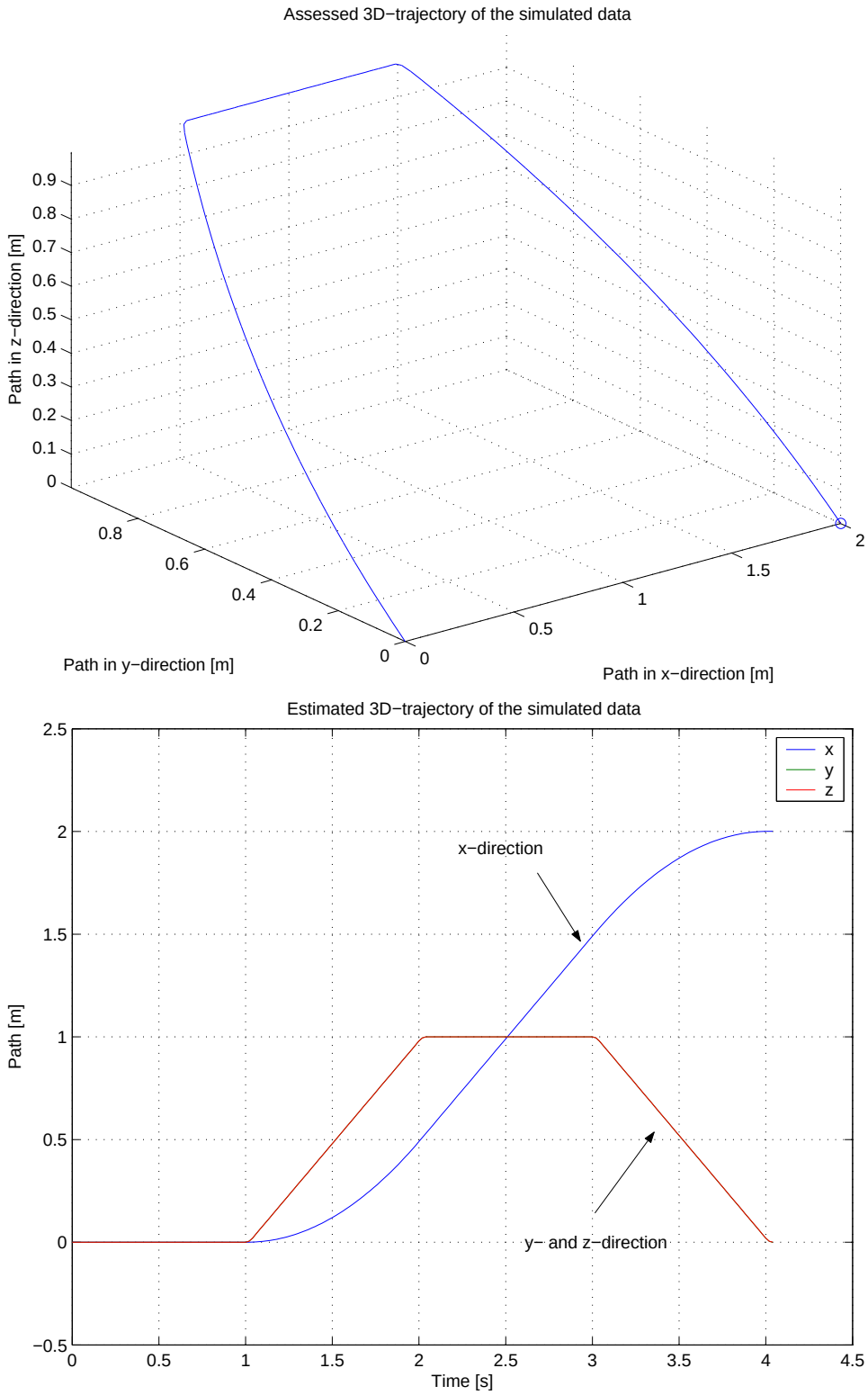


Figure 5.10: Estimated 3D-trajectory of the simulated acceleration

5.4 Comparison to a reference system

One aim of this pre-diploma thesis is the preliminary investigation of the accuracy of the inertial sensor. So it is necessary to compare the estimated trajectory with the trajectory measured by a reference system. For this reference the ultrasonic equipment created by Zebris Medizintechnik GmbH, Germany[2] is used. This equipment consists of a basic system, a transmitter and some markers. The transmitter has got three transmission units, placed at the end of a trident star. Every marker consists of a microphone which receives the ultrasonic waves. The markers are tightened to the subject which is observed. Based on the running time of the waves and on the differences between the running times of the waves from the different receiver units the position of the marker can be calculated. If instead of a single marker a triple marker is used (three markers that are put together to a trident star with defined distances and orientations to each other), also the orientation of the triple marker can be calculated.

For the measurement one triple marker is used. That means that the data record contains nine different values for every sample, three position values (x;y;z) for every one of the three single markers. The data has to be filtered due to failures in the measurement. Normally the failure data is zero, but sometimes there are high peaks. The first filter checks the data if there are too big outliers in it and sets the according data to zero. The second filter checks the data for zeros and interpolates the values.

The calculation of the distance is much easier than the estimation using the Xsens sensor data. The orientation is not regarded, so the mean position data of the three single markers is used. The detection of the starts and stops of the movements is described in section 5.2. For the covered distance of the movement the differences of the start and stop position data in x- and y-direction have to be calculated and summed up vectorial. The height of the movement is the maximum of the z-difference.

$$height_{ref} = max(s_z^I - s_{z,start}^I) \tag{5.26}$$

$$distance_{ref} = \sqrt{(s_{x,stop}^I - s_{x,start}^I)^2 + (s_{y,stop}^I - s_{y_start}^I)^2} \tag{5.27}$$

For direct comparisons it has to be noticed that the Zebris data includes a time delay of 0.08 s which has to be removed.

[2] www.zebris.de

5.5 Experimental setup

The left photo of Figure 5.12 shows the setup for the experiments. The triple marker is fixed to the inertial sensor. With this measurement object pictured in the right photo of Figure 5.12 an arbitrary movement takes place. Here the object is moved 25 times on a three dimensional circular way, drawn in Figure 5.11. Between the movement, the object is placed shortly at the starting and landing point which should be hit as exactly as possible.

The markers receive the ultrasonic waves, and the inertial sensor measures its data. The sampling frequency is 400 Hz. To compare the data, the coordinate systems of the Xsens sensor and the Zebris system should be as similar as possible. For this the sensor and the marker are orientated to each other exactly. One remaining problem is that the points of origin cannot be put together exactly, because there is a displacement of two to three centimeters in z-direction due to the housing of the sensor and the support of the marker. In x- and y-direction the exact placement is also difficult, because the point of origin of the sensor lies in a corner of the housing, and the markers cannot be fixed there.

To obtain gait parameters, the measurement object has to be fixed on the foot.

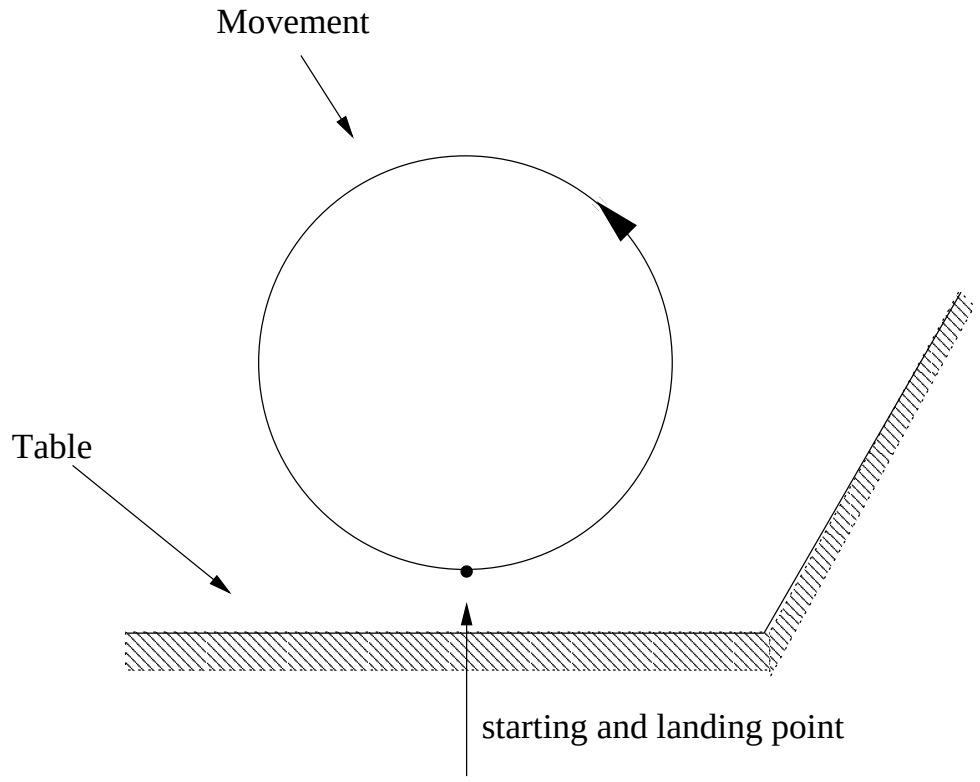


Figure 5.11: Drawing of the movements

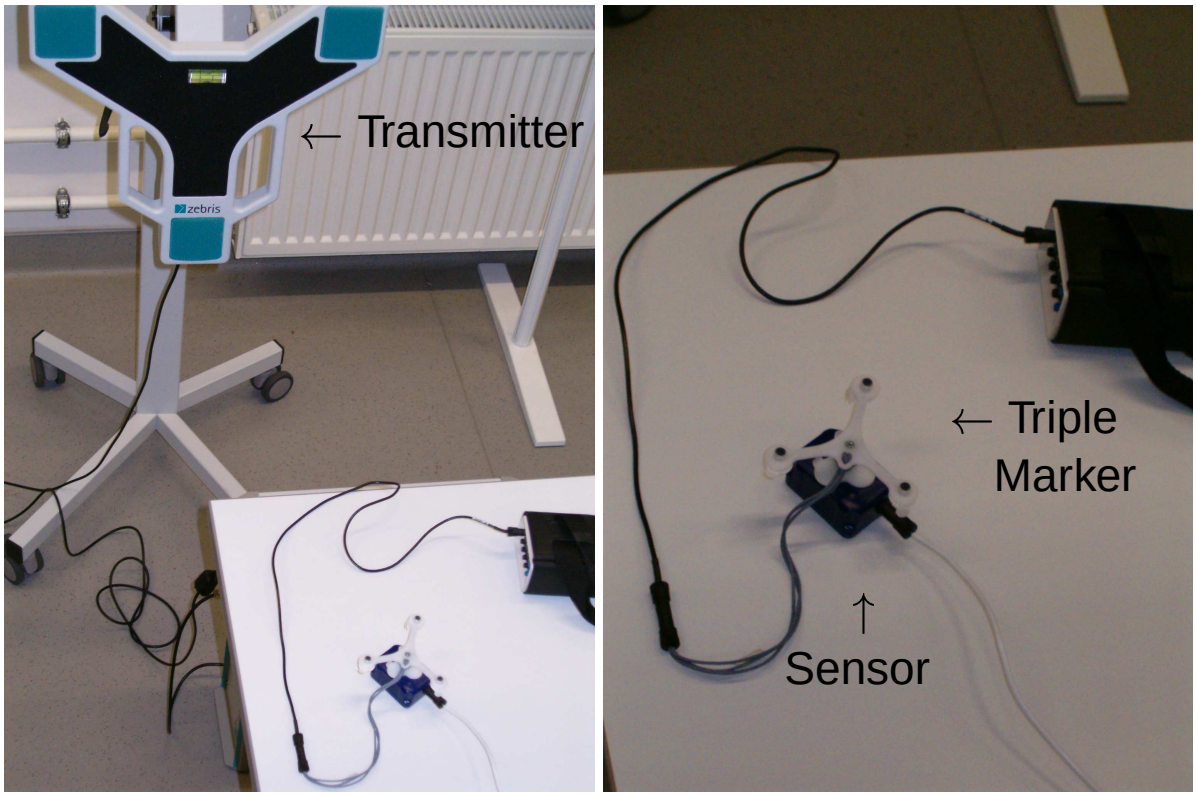


Figure 5.12: Experimental setup and measurement object: Xsens sensor and Zebris marker

5.6 Results

The estimated 3D-trajectory of the data record is given in Figure 5.13. Here the paths in every direction are plotted against the time. The enlargement in the second 2D-subplot and the 3D-plot in Figure 5.14 represent the 19^{th} movement.

At the beginning of the movement the path data are set to zero. At the end of the movement the orientation in x- and y-direction may be changed, but depending on the exactness of the landing point with respect to the starting point the paths in all three directions should come to zero. The Zebris data fulfills this requirement, but it is noticeable that there are still some inaccuracies in the Xsens estimation.

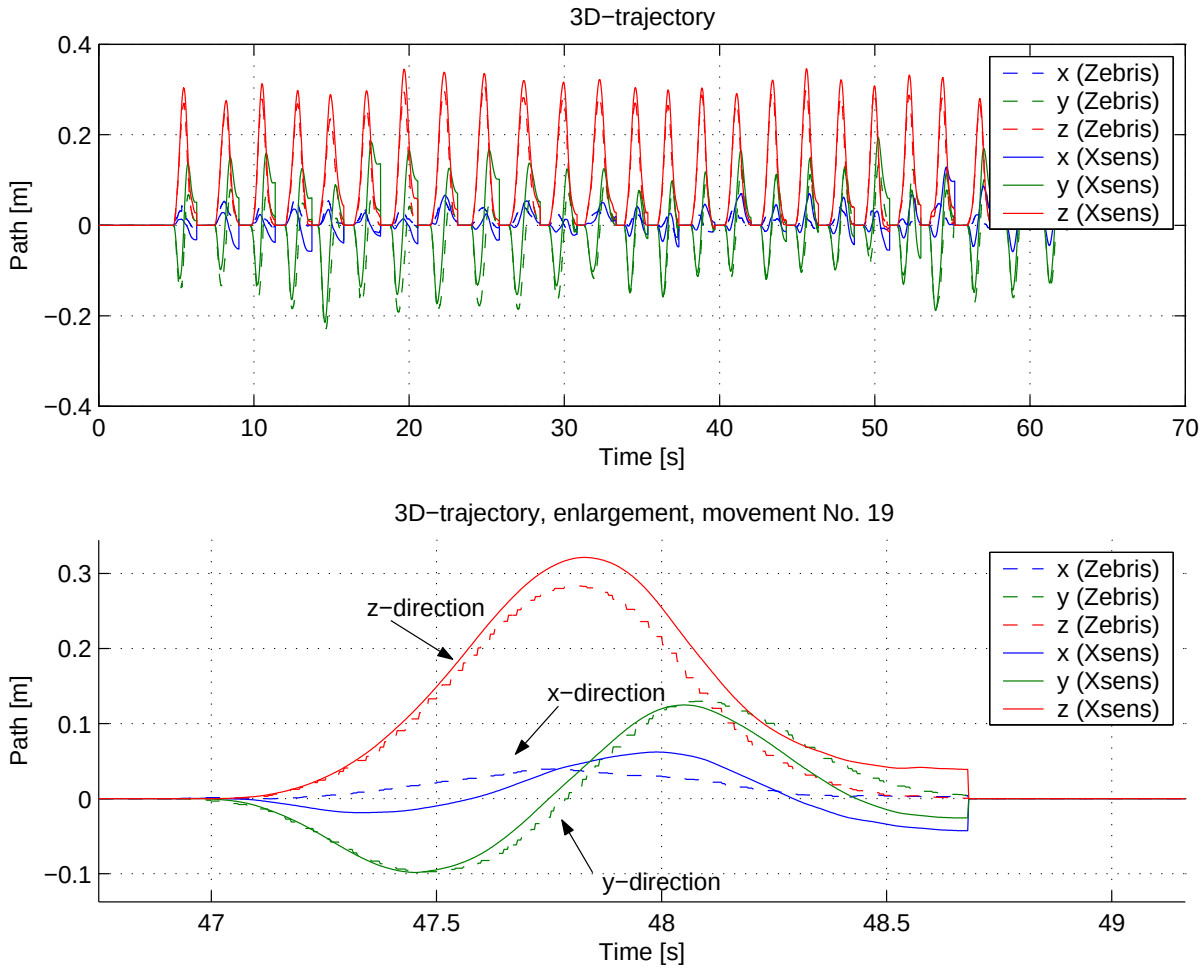


Figure 5.13: Estimated and calculated 3D-trajectory

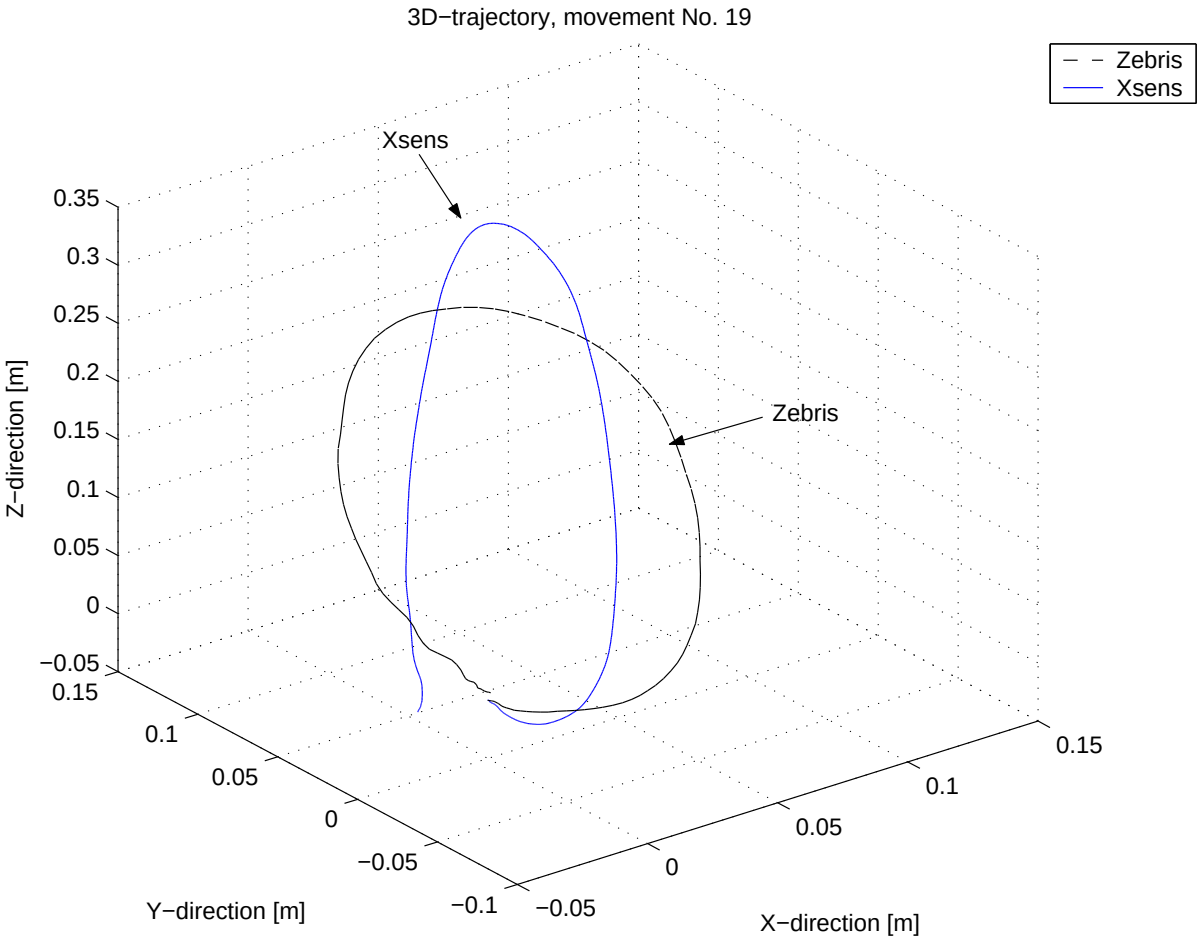


Figure 5.14: Estimated and calculated 3D-trajectory, movement No. 19

In Figure 5.15 the differences in the path between the Xsens sensor estimation and the Zebris reference calculation are plotted. The mean differences are calculated according to the following equation:

$$\vec{mean}_{diff} = \frac{1}{n} \sum_{k=1}^{n} \left(\vec{s}^{I}_{ref}(k) - \vec{s}^{I}(k) \right) \tag{5.28}$$

n - number of samples

For the standard deviation the next equation is used:

$$\vec{std}_{diff} = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} \left(\left(\vec{s}^{I}_{ref}(k) - \vec{s}^{I}(k) \right) - \vec{mean}_{diff} \right)^2} \tag{5.29}$$

The mean differences are -0.003 m in x-direction, -0.021 m in y-direction and 0.014 m in z-direction. The standard deviations are 0.025 m, 0.035 m and 0.017 m.

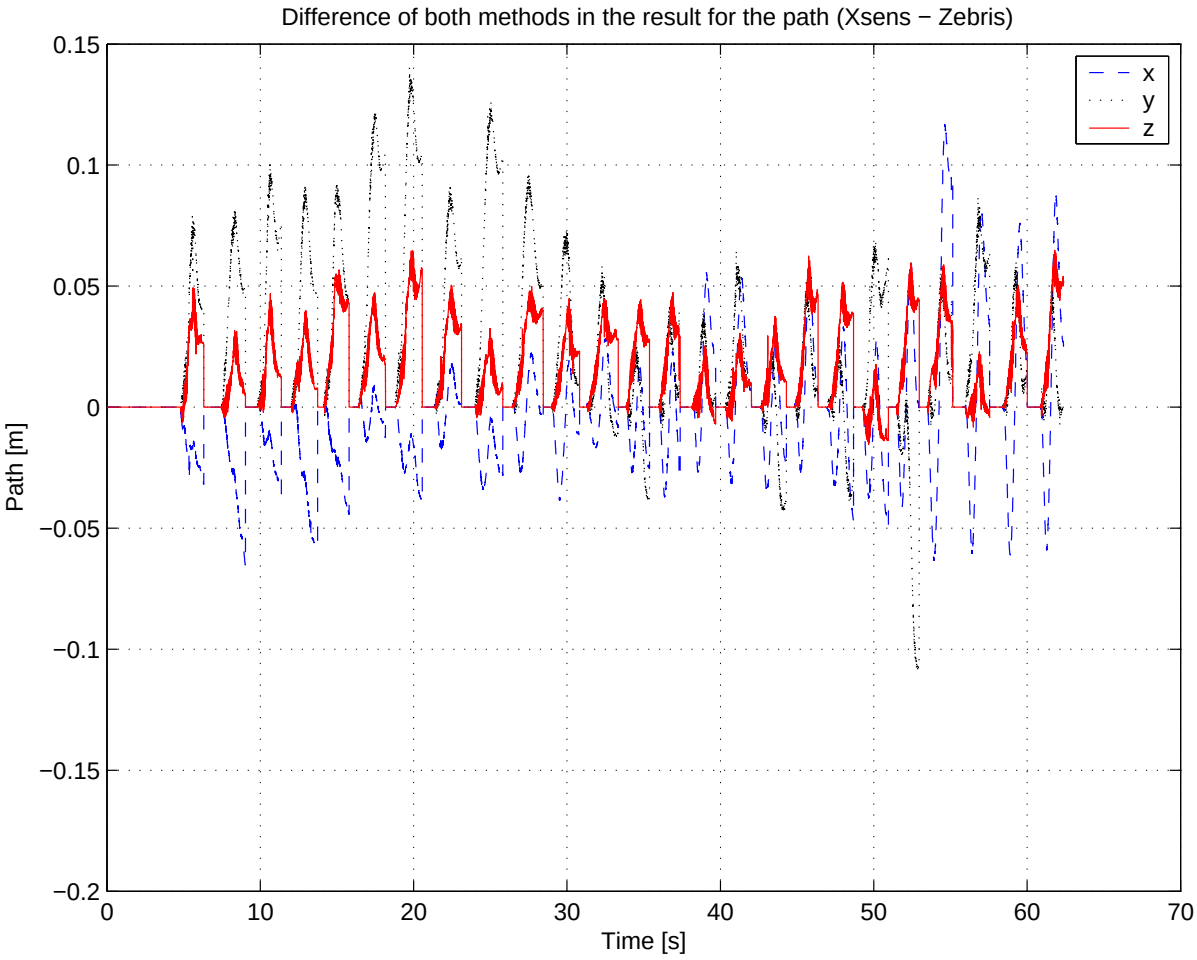


Figure 5.15: Difference between estimated and calculated 3D-trajectory

Figure 5.16 shows the estimated bias of every movement, which is very small. In x-direction the mean bias is around 0.042 $\frac{m}{s^2}$, in y-direction 0.2 $\frac{m}{s^2}$ and in z-direction -0.062 $\frac{m}{s^2}$. The standard deviations are 0.045 $\frac{m}{s^2}$, 0.066 $\frac{m}{s^2}$ and 0.018 $\frac{m}{s^2}$.

In Figure 5.17 the estimated horizontal movement distances in comparison to the reference distances are plotted. Here it is observable that the estimation is not so accurate. The used formulae are

$$mean_{diff_{distance}} = \frac{1}{m}\sum_{i=1}^{m}\left(distance_{ref}(i) - distance(i)\right) \tag{5.30}$$

m - number of movements

$$std_{diff_{distance}} = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}\left(\left(distance_{ref}(i) - distance(i)\right) - mean_{diff_{distance}}\right)^2} \tag{5.31}$$

The mean of the differences is 0.053 m, and the standard deviation is 0.033 m.

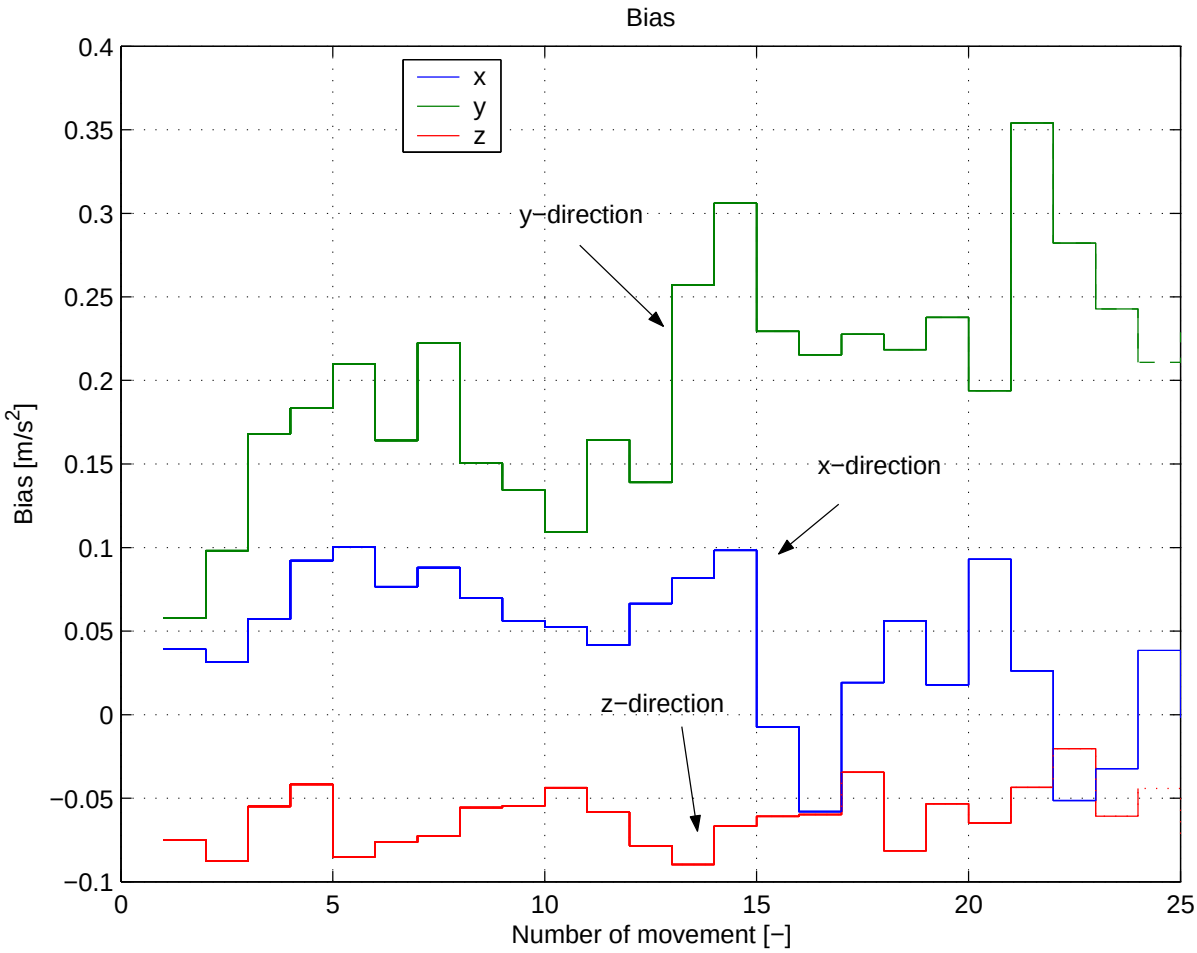


Figure 5.16: Estimated bias

Figure 5.18 shows the estimated height in comparison to the reference height. Here the estimation works better. The formulae are:

$$mean_{diff_{height}} = \frac{1}{m} \sum_{i=1}^{m} \left(height_{ref}(i) - height(i) \right) \tag{5.32}$$

$$std_{diff_{height}} = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} \left(\left(height_{ref}(i) - distance(i) \right) - mean_{diff_{height}} \right)^2} \tag{5.33}$$

The absolute mean of the differences is 0.034 m, and the standard deviation is 0.012 m.

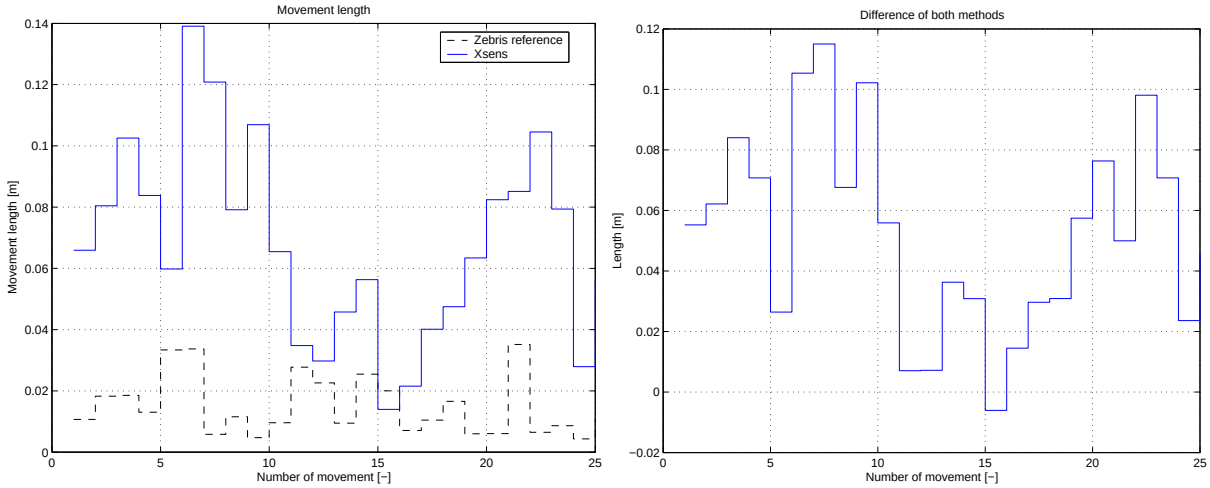


Figure 5.17: Comparison of the estimated and the calculated movement length

The estimation of the movement height is better than the estimation of the distance. One difference between both is that the measured movement distance is nearly zero, and the height is bigger. One possible reason for this effect is the uncertainty in the movement detection. Inaccuracies in this part of the estimation lead to large errors in the results.

The error in the estimation seems to be mainly systematically, not so much random. The results of the Xsens estimation are bigger than the results of the Zebris reference measurement. Also the bias is relatively equal. The bias in y-direction is the highest one, the bias in z-direction is always negative and the bias in x-direction lies in the same order than the bias in z-direction, but it is mainly positive.

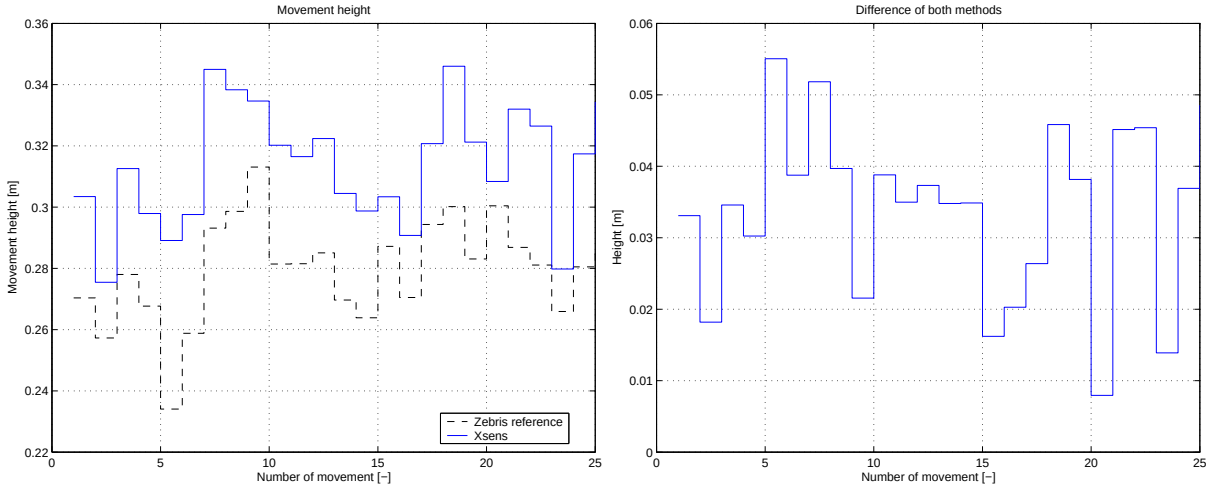


Figure 5.18: Comparison of the estimated and the calculated movement height

This leads to the issue of the sensor calibration. The sensor was calibrated by the help of the Xsens calibration equipment and algorithm. The new axes were compared to the old axes, and the old axes were better. So it was decided to use the preset calibration values. Later N.-O. Negård of the Max-Planck-Institut Magdeburg developed an own calibration algorithm. After applying this algorithm the errors looked more random, but the amount of the errors were a little bit higher.

Chapter 6

Conclusions and outlook

In the first part of this pre-diploma thesis a new physiological stimulation pattern was developed and implemented as a finite automaton in Matlab / Simulink. The stimulation pattern uses three channels per leg to stimulate three different muscle groups and it subdivides the gait cycle into four states. The automaton is designed very flexibly and can be used for a wide range of applications. It can be used for patients walking with or without a treadmill. Also the automaton can be used for hemiplegic and paraplegic patients. It can be enlarged easily for more channels or more states and is not determined to be used only for gait stimulation. It can easily be adapted for application on another limb, e.g. the arm.

A remaining problem is the detection of the gait phase. In the actual implementation there are foot sensors used, and the problem is their short life-time and how to define the switching threshold. It should be investigated if the gait phase can be detected by the interpretation of the acceleration data which is measured with an inertial sensor.

In the second part of this pre-diploma thesis an algorithm was given that estimates gait parameters like step length and step height and it was tested on simulated data and on real test data. The results for the real data are not as accurate as desired, so further research concerning the accuracy must take place before the inertial sensor can be used practically. Furthermore, the algorithm should be tested for real gait data, because the influence of little inaccuracies (e.g. in the stride detection) is not so high for longer covered distances.

There are also other problems which remain to be solved. One of them is the influence of the treadmill on the estimation. It has to be investigated whether the treadmill has got any influence on the acceleration measurement or not. If the influence has to be calculated, the step time is multiplied by the velocity of the treadmill. The obtained path is added to the estimated path. In a non-academic experiment in which a test subject was walking on a treadmill, the results without regarding the treadmill looked much better than the results with regarding it. It is also desirable to convert the offline estimation of the gait parameters to an online algorithm.

The final aim of the project is a closed loop control system. The movements of the healthy limb are measured, and the electromyography (EMG) and the voluntary muscle control of the paretic limb are measured. According to this measurements and estimations the finite automaton is controlled, so that the movements of the paretic limb are adapted to the movements of the healthy leg by using Functional Electrical Stimulation.

List of Figures

List of Tables

Glossary

antagonists ..... the flexor and the extensor that move the same limb; they should be balanced

contralateral ..... related to the other side (opposite: ipsilateral)

drop foot ..... popular after a stroke: the affected person is not able to lift his resp. her toe during gait

ipsilateral ..... related to the same side (opposite: contralateral)

offline gait analysis ..... the desired calculations take place after the end of the gait session

online gait analysis ..... the desired calculations take place during the session (e.g. after every single step); requires real-time-ability

perpendicular ..... normal

reflex ..... a consistent rapid automatic response to a stimulus, controlled in the spinal cord [13]

spasticity ..... enhanced tonicity of the sceletal muscles; normally the flexor muscles are affected and resist against the antagonistic extensor muscles [13]

step ..... interval between the back-to-back initial contacts of both feet

stimulation pattern ..... a pattern that determines which muscle is stimulated at which time

stride ..... interval between two back-to-back initial contacs of the same foot (also called double step)

stroke ..... a celebral infarct, in 80% of the cases due to ischemic apoplexia (acute occlusion of a cerebric artery), in 20% due to cerebral haemorrhages [13]

tibia ..... shinbone

transition ..... condition for the change to a following state

Bibliography

[1] U. Bogataj, N. Gros, M. Kljajić, and R. Aćimović-Janežič. Enhanced Rehabilitation of Gait after Stroke: A Case Report of a Therapeutic Approach using Multichannel Functional Electrical Stimulation. *IEEE Trans. Rehab. Eng.*, 5(2):221–232, 1997.

[2] T. Fuhr and G. Schmidt. Neue Ansätze zur Steuerung und Regelung einer kooperativen Gang-Neuroprothese. *at - Automatisierungstechnik*, 50(7), Jul. 2002.

[3] W.T. Liberson, H.J. Holmquest, D. Scott, and M. Dow. Functional Electrotherapy: Stimulation of the Peroneal Nerve Syncroized with the Swing Phase of the Gait of Hemiplegic Patients. *Archives of Physical Medicine and Rehabilitation*, 21:101–105, 1961.

[4] H.J. Luinge. *Inertial Sensing of Human Movement*. PhD thesis, University of Twente, 2002.

[5] H.J. Luinge, P.H. Veltink, and C.T.M. Baten. Estimating orientation with gyroscopes and accelerometers . *Technology and Health Care*, 7:455–459, 1999.

[6] G.M. Lyons, T. Sinkjaer, J.H. Burridge, and D.J. Wilcox. A Review of Portable FES-Based Neural Orthoses for the Correction of Drop Foot . *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(4):260–279, 2002.

[7] N.-O. Negård. Kalman-filter for orientation. Technical report, Max-Planck-Institut für Dynamik komplexer technischer Systeme Magdeburg, 2004.

[8] I.P.I. Pappas, T. Keller, S. Mangold, M.R. Popovic, V. Dietz, and M. Morari. A Reliable Gyroscope-Based Gait Phase Detection Sensor Embedded in a Shoe Insole. *IEEE Sensors Journal*, 4(2):268–274, April 2004.

[9] I.P.I. Pappas, T. Keller, and M.R. Popovic. Validation of a new Gait Phase Detection System. *Gait and Posture*, 13(3):301–302, May 2001.

[10] I.P.I. Pappas, M.R. Popovic, T. Keller, V. Dietz, and M. Morari. A Reliable Gait Phase Detection System. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 9(2):113–125, June 2001.

[11] J. Perry. *Gait Analysis: Normal and Pathological Function*. Slack Incorporated, Thorofare, New Jersey, 1992.

[12] C. Schlossbauer. Entwicklung und Testung eines inertialen Sensorsystems zur Messung von Translation und Rotation bei Körperbewegungen des Menschen . Master's thesis, Ludwig-Maximilian-Universitaet München / Technische Universitaet Mn̈chen, 1996.

[13] G. Thews, E. Mutschler, and P. Vaupel. *Anatomie, Physiologie, Pathophysiologie des Menschen*. Wissenschaftliche Verlagsgesellschaft mbH Stuttgart, 5 edition, 1999.

[14] Krauth + Timmermann. TransStim© , 2003. Stimulation Program .

[15] P. H. Veltink, P. Slycke, J. Hemssems, R. Buschman, G. Bultstra, and H. Hermens. Three Dimensional Inertial Sensing of Foot Movements for Automatic Tuning of a Two-channel Implantable Drop-foot Stimulator. *Med. Eng. Phys.*, 25(1):21–28, Jan 2003.

[16] P.H. Veltink, H.J. Luinge, B.J. Kooi, C.T.M. Baten, P. Slycke, W. Olthuis, and P. Bergveld, editors. *The artificial vestibular system - design of a tri-axial inertial sensor system and its application in the study of human movement* . International Society for Postural and Gait Research, 2001.

[17] WHO. World health report 2002. www.who.int/whr/2002/en, 2001.

[18] Xsens Technologies B.V., Enschede, The Netherlands. *Technical Documentation MT9 and MT6*, 2003.

Appendix A

Programs

A.1 Filtering the ultrasonic position data

A.1.1 Detection of failures

```
%filterzebris.m
%detects failures in the position data measured by the ultrasonic Zebris system
%If the differences between a value and its both neighbour values is
%higher than a threshold, the value is set to zero.
%Respectively, a matrix of zeros is generated, and if at least one
%of the differences is smaller than the threshold, the value is put
%into the matrix of the zeros at its original indices. At the place
%of the failures the zeros remain. After that the data is filtered by
%FilterZeros.m which detecs zeros and interpolates them.

function signalout = filterzebris(signalin)

threshold = 0.05;
maxindex    = max(size(signalin));
signaltemp = zeros(size(signalin));

%The first and the last values are inherited from the original signal.
signaltemp(1,:)=signalin(1,:);
signaltemp(maxindex,:)=signalin(maxindex,:);

for i=2:maxindex-1,
  for j=1:9       %there are nine channels to filter (three coordinates for
                  %every one of the three markers)
    if (signalin(i,j) - signalin(i-1,j) < threshold) & (signalin(i+1,j) - ...
                signalin(i,j) < threshold)
      signaltemp(i,j) = signalin(i,j);
    end
  end
  signalout = signaltemp;
end
```

A.1.2 Interpolation of failures

This routine was programmed by Jaques de Gersigny (Max-Planck-Institut für Dynamik komplexer technischer Systeme, Magdeburg).

```
function signalout = filterZeros(signalin)
%  signalout = filterZeros(signalin)
%  Author: Jacques de Gersigny
%  Date: 22 March 2004
% AIM: filter signals having zeros values

signalsize = size(signalin);
signalin = signalin(:);


maxindex    = max(size(signalin));
signaltemp = zeros(size(signalin));

for i=3:maxindex-6,
        %angledif1(i) = signalin(i+1)-signalin(i);
        if (signalin(i) == 0)
                j=i;
                %while (signalin(j) == 0)
                %       j = j+1;
                %end

                % verify that there is not two occlusions on after the other
                while ((signalin(j-1)==0)|(signalin(j)==0)| ...
                       (signalin(j+1)==0) | (signalin(j+2)==0) | ...
                       (signalin(j+3)==0) | (signalin(j+4)==0) | ...
                       (signalin(j+5)==0)| (signalin(j+6)==0))
                        j = j+1;
                end

                for k=i-1:j+1,
                        %interpolation between first point and last point (included)
                        signaltemp(k)= signaltemp(i-2)+((signalin(j+2) ...
                                        -signaltemp(i-2))*(k-i+2)/(j-i+4)) ;
                end
        else signaltemp(i)= signalin(i);
        end
end

% reshape the signal to its original architecture
signalout = reshape(signaltemp,signalsize(1),signalsize(2));

%subplot(2,1,1);
%plot(reshape(signalin,signalsize(1),signalsize(2))); % signal before
%subplot(2,1,2);
%plot(signalout);  % signal after
```

A.2 Detection of steps

```
%detect_step_xsens.m
%detects the begin and the end of the step with the Xsens sensor

%For every value the mean and the standard deviation of the value and its
%neighbors are calculated. If both are small enough, the value is assumed
%not to be part of a step. By filtering the data again, this time with a
%median filter which removes outliers, little failures are corrected. By
%the help of an edge detection, the start and stop values of the steps
%are extracted.

function [start,stop]=detect_step_xsens(acceleration)

acc=sum(acceleration')-9.81; %total acceleration without gravitation

threshold_mean=0.5;  %allowed maximum difference of the mean to zero
threshold_std=0.3;   %allowed maximum difference of the standard
                     %deviation to zero
restingsamples=50;   %size of the mean filter (how many neighbor
                     %values are used (in one direction))
medfiltsize=50;      %size of the median filter

N=max(size(acc));    %Number of samples
starts=zeros(1,length(acc));
start=[];
stops=zeros(1,length(acc));
stop=[];
nostep=zeros(1,length(acc));

for k=restingsamples+10:N-restingsamples-10
  nostep(k)=1;
  mean_for=mean(acc(:,k:k+restingsamples));   %forward mean
  mean_back=mean(acc(:,k-restingsamples:k));  %backward mean
  std_for=std(acc(:,k:k+restingsamples));     %forward standard derivation
  std_back=std(acc(:,k-restingsamples:k));    %backward standard derivation
  if(abs(mean_for)<threshold_mean & std_for<threshold_std) | ...
        (abs(mean_back)<threshold_mean & std_back<threshold_std)
  nostep(k)=0;%If the conditions concerning the mean and the standard
              %deviation are fulfilled, this point k should indeed lie
              %between the steps.
  end
end

nostepfilt=medfilt1(nostep,medfiltsize);%If the conditions before didn't
                                        %work perfectly well, the median
                                        %filter removes outliers.

%The data 'nostepfilt' should look like pulses which are 1 exactly during
%the steps and 0 between the steps. With an edge detection the starts and
%stops are detected. Because of the fact that the mean filtering blurs
%the exact edges, the starts are shifted forward a little and the stops are
%shifted backwards a little. One remaining problem is that the increases
%and the decreases of the pulses often takes two samples. To avoid this
%error, it is checked wether the value before is also a start resp. stop
%value or not before the value itself is declared as start or stop.


for k=restingsamples+1:length(nostepfilt)-restingsamples
diff(k)=nostepfilt(k)-nostepfilt(k-1);
  if diff(k)>0 & starts(k-restingsamples-1)==0
    starts(k-restingsamples)=1;
    start=[start k];
  elseif diff(k)<0 & stops(k+restingsamples-1)==0
    stops(k+restingsamples)=-1;
    stop=[stop k];
  end
end
```

A.3 Estimation of gait parameters

A.3.1 Calling the different functions

```
%steplength_function_call.m
%loads the data, calls the functions that assess trajectories and plots them

clear all;
close all;

load sl_data/preparated_tabledata_31;

%Lowpassfiltering of the acceleration data
[B,A]=ellip(5,0.5,20,[100/200]);
freqz(B,A,1000,1/(1/400));
acceleration_xsens=filtfilt(B,A,fCalibrated(:,1:3));

%Correction of the Zebris time delay, removing the mean of the Zebris
%data for the step detection
time_del=0.08/Ts;
xyz=mean(ankleTriPosFil(1:1800,:));
for k=1:length(time)-time_del
  ankleTriPosFilwithoutdelay(k,:)=ankleTriPosFil(k+time_del,:)-xyz;
end

v_tm=0;  %In this experiment there is no treadmill

%Detection of the start and stop values by the use of the inertial sensor
%(from Xsens)
[start,stop]=detect_step_xsens(acceleration_xsens);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Zebris reference data                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Movement parameters with ultrasonic measurement (Zebris)
[step_length_zebrisref,s_i_zebrisref,step_height_zebrisref]= ...
    steplength_zebris(ankleTriPosFilwithoutdelay',Ts,v_tm,start,stop);

figure(3)
stairs(step_length_zebrisref,'k--')
grid on;hold on;
title('step length');
xlabel('Number of steps');
ylabel('step length [m]');

figure(9)
stairs(step_height_zebrisref,'k--')
grid on;hold on;
title('step height')
xlabel('Number of steps');
ylabel('step height [m]');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             Xsens data                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Movement parameters with inertial measurement
[step_length_xsensrotacc,s_i_xsensrotacc,step_height_xsens,bias_s]= ...
    steplength(acceleration_xsens,orientation_xsens.signals.values, ...
               start,stop,Ts,v_tm,2);

figure(2)
title('Bias')
legend('x','y','z',0)
xlabel('Number of movement [-]');
ylabel('Bias [m/{s^2}]');
```

```
%The intrinsic plot takes place inside the function.

figure(3)
stairs(step_length_xsensrotacc)
legend('Zebris reference','Xsens',0)

figure(9)
stairs(step_height_xsens)
legend('Zebris reference','Xsens',0)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Simulated data                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%acceleration_sim=sig6(2:4,:)'; %Simulated 'acceleration' data
%orientation_sim=test_rotmatrix.signals.values; %rotation matrices taken
                                                %from an arbitrary data record

%start_sim=round(1.02/Ts); % 51
%stop_sim=round(4.06/Ts);  % 202
%[step_length_sim,s_i_sim,step_height_sim,bias_sim]= ...
%    steplength(acceleration_sim,orientation_sim,start_sim,stop_sim,Ts,0,6)

%figure(6)
%title('Bias simulated data')
%The intrinsic plot takes place inside the function.
```

A.3.2 Calculation with the ultrasonic reference data

```
%steplength_zebris.m
%Calculates the steplength with position data of the ultrasonic Zebris
%system.
%The algorithm uses the differences of the location between the start and
%the stop value. If a treadmill is running, its velocity is multiplyed
%by the time passing between the foot's leaving the treadmill and hitting
%it again. This term is added to the absolute difference of the position.

function [step_length,locus,step_height]= ...
      steplength_zebris(coordinates,Ts,v_tm,start,stop)

%Resorting the position data and using the means of the three markers.
x_coordinates=[coordinates(1,:);coordinates(4,:);coordinates(7,:)];
y_coordinates=[coordinates(2,:);coordinates(5,:);coordinates(8,:)];
z_coordinates=[coordinates(3,:);coordinates(6,:);coordinates(9,:)];

for k=1:length(coordinates)
  locus_x(k)=mean(x_coordinates(:,k));
  locus_y(k)=mean(y_coordinates(:,k));
  locus_z(k)=mean(z_coordinates(:,k));
  locus(:,k)=[0;0;0];
end

%Leaving and hitting the treadmill. They are assumed to be the same as
%start and stop of the gait, but this can be changed here.
start_tm=start;
stop_tm=stop;

i=length(start);
number_of_steps=i

for n=1:i %for every step

  %Translating the Zebris data so that the start values are zero
  for k=start(n):stop(n)
    locus(:,k)=[locus_x(k)-locus_x(start(n)); ...
                locus_y(k)-locus_y(start(n));locus_z(k)-locus_z(start(n))];
  end
```

```
  %Change in the absolute position
   slength(n,1)=locus_x(stop(n))-locus_x(start(n));
   slength(n,2)=locus_y(stop(n))-locus_y(start(n));
   slength(n,3)=locus_z(stop(n))-locus_z(start(n));

  %Change in the position due to the treadmill
   treadmill(n)=(stop_tm(n)-start_tm(n))*Ts*v_tm;
   step_length_without_tm(n)=norm(slength(n,1:3));
   step_length_with_tm(n)=step_length_without_tm(n)+treadmill(n);
   step_height(n)=max(locus_z(start(n):stop(n)));
end

step_length=step_length_with_tm;
locus=locus';
```

A.3.3 Estimation with the inertial data

```
%steplength.m
%Estimates the 3D-trajectory passed by an inertial sensor which measures
%acceleration and orientation data.
%Calculates also the movement parameters length and height.

function [step_length,s_i,step_height,bias_s]= ...
      steplength(acceleration,orientation,start,stop,Ts,v_tm,verbose)

a_sgb=acceleration;
rotmatrix=orientation;
g_i=[0;0;9.81];         %Gravitation

start_tm=start;         %leaving the treadmill and hitting ist again
stop_tm=stop;           %here it is assumed to be the same as start and
                        %stop of the movements

i=length(start);
number_of_steps=i
for n=1:i               %for every movement

  %forward motion due to the treadmill
   treadmill(n)=(stop_tm(n)-start_tm(n))*Ts*v_tm;

  %Creating the rotation matrix, removing the gravitation (after converting
  %it to sensor coordinates) from the acceleration data (in sensor
  %coordinates) and converting this data to inertial coordinates
   for k=start(n)-2:stop(n)
     R(1:3,1:3,k) =[rotmatrix(k,1) rotmatrix(k,2) rotmatrix(k,3);
                    rotmatrix(k,4) rotmatrix(k,5) rotmatrix(k,6);
                    rotmatrix(k,7) rotmatrix(k,8) rotmatrix(k,9)];
     RT(1:3,1:3,k)=inv(R(1:3,1:3,k));
     g_s(1:3,k)=RT(1:3,1:3,k)*g_i(1:3);
     a_sb(1:3,k)=a_sgb(k,1:3)'-g_s(1:3,k);
     a_ib(1:3,k)=R(1:3,1:3,k)*a_sb(1:3,k);
    %Acceleration in inertial coordinates with bias, but without gravitation
   end

  %Calculating the bias in sensor coordinates

  %Integration of the acceleration data with bias and without gravitation
  %in inertial coordinates
   for k=start(n):stop(n)
     v_ib_help(1:3,k)=0.5*Ts*(a_ib(1:3,k-1)+a_ib(1:3,k));
     v_ib(k,1)=sum(v_ib_help(1,start(n):k));
     v_ib(k,2)=sum(v_ib_help(2,start(n):k));
     v_ib(k,3)=sum(v_ib_help(3,start(n):k));
   end;

   sR(:,:,n)=sum(R(:,:,start(n):stop(n)),3);
   bias_s(1:3,n)=inv(sR(:,:,n))*v_ib(stop(n),1:3)'/Ts;
  %Bias of the acceleration data in sensor coordinates
```

```
  figure(verbose)
  title('Bias_in_sensor_coordinates')
  stairs(bias_s');
  grid on;hold on;
  drawnow;
  %legend('','x','y','z',0);

  for k=start(n):stop(n)
    %Removing the bias from the acceleration data with gravitation in
    %sensor coordinates
     a_sg(1:3,k)=a_sgb(k,1:3)'-bias_s(1:3,n);

    %Converting the acceleration data without bias and with gravitation
    %into inertial coordinates
     a_ig(1:3,k)=R(1:3,1:3,k)*a_sg(1:3,k);

    %Removing the gravitation in inertial coordinates
     a_i(1:3,k)=a_ig(1:3,k)-g_i(1:3);
  end;

  %Integration of the acceleration data without bias and without
  %gravitation in inertial coordinates
  for k=start(n)-1:stop(n)
     v_i_help(1:3,k)=0.5*Ts*(a_i(1:3,k-1)+a_i(1:3,k));
     v_i(k,1)=sum(v_i_help(1,start(n):k));
     v_i(k,2)=sum(v_i_help(2,start(n):k));
     v_i(k,3)=sum(v_i_help(3,start(n):k));
  end;

  %Integration of the velocity data without bias and without
  %gravitation in inertial coordinates
  for k=start(n):stop(n)
     s_i_help(k,1:3)=0.5*Ts*(v_i(k-1,1:3)+v_i(k,1:3));
     s_i(k,1)=sum(s_i_help(start(n):k,1));
     s_i(k,2)=sum(s_i_help(start(n):k,2));
     s_i(k,3)=sum(s_i_help(start(n):k,3));
  end;

  step_length_without_tm(n)=norm(s_i(stop(n),1:3));
  step_length_with_tm(n)=step_length_without_tm(n)+treadmill(n);

  %It is better to clear the variables at the end of the loop ...
  clear treadmill g_s a_sb a_ib v_ib_help v_ib sR a_sg a_ig a_i
  clear v_i_help v_i s_i_help

  step_height(n)=max(s_i(start(n):stop(n),3));

end

%Here the interesting question is asked which value must be taken. In the
%results of non-academic experiments it comes apparent that the results
%without the treadmill are better. But this remains to be investigated.
%Anyway, in the preliminary investigations there is no treadmill.

step_length=step_length_without_tm;
%step_length=step_length_with_tm;
```