University of Groningen **Faculty of Mathematics and Physical Sciences**

Department of Computing Science

Neural Control of Artificial Human Walking

Alexander Ypma

begeleiders: J.A.G. Nijhuis
R.S. Venema

december 1995

RuG

Abstract.

The ultimate goal of much research in biomedical engineering is the construction of an artificial walking system, i.e. a system which enables paraplegics to walk again, using *Functional Electrical Stimulation* (FES) and biofeedback. *Infotronic* at Hengelo (NL) produces a system for measurement of certain body signals that originate from human walking, UltraFlex, which is subsequently used to acquire data from normal human walking.

We try to obtain a solution for the *inverse dynamics* problem in human walking by means of neural networks. Once a network has learned the right instantaneous mapping, it has to be extended to make time–lagged mappings, i.e. to predict future muscle activation on the basis of past activation and movement information. A neural network is expected to predict the human EMG signal better than classical statistical predictors. This advantage will become even auspicious when EMG arising with FES has to be predicted: muscle fatigue and disturbances cause a non–stationary signal the predictor has to adapt to.

A major item in neural system design is the particular *choice of network dimension and data preprocessing* that leads to a satisfactory solution for the problem at hand. We propose a *structured approach* based on the relation between certain signal characteristics and network architecture. A relation is found between a signal's correlation time and its generalization performance with tapped delay–lines of a certain dimension.

This criterion and two benchmark criteria are applied in the prediction of the human EMG. For a synthetic EMG–equivalent, no conclusions can be drawn w.r.t. the suitability of all three design criteria (whether prediction is performed using conventional or neural techniques). For a real segment of EMG, a predictor order somewhere in the middle of correlation time and Final Prediction Error values seems suitable, especially when statistical and movement features are added. Hence, the suitability of the correlation time criterion can be doubted for complexer signals exhibiting nonlinearities (like the EMG).

Ultimately, a neural 400–lag predictor is obtained, that tracks the "amount of muscle activity" conveniently, whereas linear predictors of comparable order show poor tracking performance.

Chapter 1. Introduction

Over the past decades much research effort in Biomedical and Rehabilitation Engineering has been aimed at the rehabilitaion of paraplegia. In particular, the effective application of Functional Electrical Stimulation (FES), is studied, whereby the injured muscles of a disabled person are to regain functional gait or movement. Another area of interest is Biofeedback, i.e. providing patients with sensory (visual, auditory, vestibular, etc.) information to support the rehabilitation process. Lately, there have been attempts to combine these approaches, by utilizing specific information about body configuration, balance and muscle activity (called "proprioceptive information") and visual input ("exteroceptive information") in their attempts to control the process of stimulating a paraplegic's disabled lower extremity musculature. Furthermore, it is used in the design of flexible, adaptable ("EMG–driven") prostheses.

1.1 Current situation

To date, the ultimate goal of artificial human walking has not yet been reached. Some problems that arise with FES (muscle fatigue, non–smoothness of movements, inflexibility, i.e. some fixed set of possible movements is offered to a patient by preprogramming certain stimulation sequences, and tailoring to a particular person requires a time–consuming process of trial–and–error [3]), prevent a broad utilization of this method: as of yet, patients think of a wheelchair as the more reliable, safer, less fatiguing and more familiar alternative.

As additional problems are named [22]: stimulation only provides marginal hip and trunk stability, whereas additional external bracing, to yield sufficient stability, has unnatural appearance, results in slow walking and is therefore not practical in daily life. Moreover, FES–induced unbraced walking takes much energy, because the intact upper extremities are used as power sources (surface electrodes are not selective at the muscle level), and so far has resulted in relatively fast but unnatural and time–invariant gait. Furthermore, problems exist with actuator (stimulator) selectivity, convenience of feedback sensors, lack of artificial sensors for muscle force and immature technology for intelligent subconscious patient interfacing. In short, with or without bracing, FES for artificial walking is not technically reliable and safe, cosmetically unacceptable and cost too much.

Current control systems also appear to be deficient [22]: a. despite the advent of implanted electrodes, stimulation selectivity is not yet at the motor unit level, so recruitment order is nonphysio-

logical, considerably increasing muscle fatigue; b. availability and quality of feedback signals (e.g. joint angles and torques), as necessary for feedback control of lower limb movement, are limited compared to the numerous natural lower limb sensors; c. lack of systematic methods for deriving initial stimulation patterns to achieve coordinated muscle actions, and d. lack of strategies to adapt these patterns to external or internal (muscle fatigue) changing conditions (e.g. reflex strategies are not feasible for paraplegics); e. controller design has to be extended; etc.

True proliferation of FES in artificial walking, providing autonomity, adaptability, health benefits and user–friendliness) will not start until these serious shortcomings have been overcome [22].

1.2 Opportunities and requirements of FES

Complete paraplegia, which can be caused by a spinal cord injury (SCI), results in total paralysis of the lower extremities: the neural pathways between the Central Nervous System (CNS) and the muscle are disrupted. In many cases, however, an intact Peripheral Nervous System (PNS) still allows to artificially stimulate the paralyzed muscles: FES can be used to bypass the disconnected pathways. FES is currently confined to selected (absence of motor neuron and upper extremity injury as well as the absence of lower extremity contractures) and well motivated patients; a daily life user potential of some 10% of all paraplegic SCI subjects has been reported (which equals to some 315 candidates in the Netherlands) [23].

Human standing and locomotion requires a balanced activity of many muscles; this is established in the human motor control system by a continuous bidirectional flow of neural signals between CNS and musculature: *efferent* nerve fibers pass control signals from the CNS to the muscle and *afferents* return information about the resulting contraction. This feedback mechanism allows the CNS to actively control and adapt muscle performance during various tasks under a wide range of conditions (correcting for external disturbances). Hence, the basic requirements for artificial human motor control (as with FES) are actuator and sensor technology (artificial efferent and afferent signals) and a (feedback) control system (artificial CNS and PNS) [22][23].

1.3 Research focus

In this Master's thesis research we are occupied with the design of (a part of) a neurocontroller for a biofeedback–driven system for (FES–induced) artificial walking. More specifically, we are attempting to bridge the gap between research in the fields of FES and biofeedback by mapping exteroceptive and proprioceptive data onto a desired joint trajectory, obtaining an adequate instantaneous stimulation pattern (and ultimately: a desired movement).

A **neural network** is expected to be ideally suited for the task of performing this nonlinear, time–varying i/o–mapping: its adaptivity and extendability, together with its potential in nonlinear system identification and modelling without complex mathematical system descriptions give rise to this assumption. Emphasized is the potential of an artificial neural system to perform a well–defined part of this task (solving the inverse dynamics problem), while utilizing a specific set of feedback signals from the body. We concentrate on the prediction of muscle activation using past activation and movement data. Nowadays, much research is directed towards reliable prediction of complex nonlinear (e.g. physiological) signals: besides classical statistical [42][43] and modern chaotic dynamics [45][46][48] approaches, also neural networks seem very promising for this task [44][47].

1.4 Infotronic Medical Engineering

This research was initiated by chairman J.A.M Maathuis of the company

- Infotronic Medical/ Industrial Engineering
- Hengelo (Ov), The Netherlands.

His ultimate ideal is the construction of a system that enables paraplegics to walk again in a human–friendly manner, i.e. without extensive dressing and/ or moving effort, long endurance, robustness and human–like appearance. The company produces a number of measurement devices, that can eventually be used in such a system. The data we used in this research is obtained with these devices (see Chapter 3).

Chapter 2.
Neural approaches to FES control

First, we will take a look at some previous work done on the subject of FES control.This section contains an overview of relevant literature from the many fields covering the various aspects of the problem, such as trajectory formation and gait pattern generation, inverse kinematics and dynamics, fuzzy control, Finite State Control of FES, etc. Also, an overlook of the many aspects involved in artificial gait synthesis is given. Based on the literature survey,we will draw a schematic representation of the components that in our opinion make up an artificial walking system. We will use this schema when we select a particular subproblem to be solved within the framework of an neural controller for FES–induced walking.

2.1 FES–induced walking: an overview.

We start our survey by relating about contemporary FES–related research as reported in [3].

The author, who is with the Cleveland Veterans Affairs Medical Center, describes the **potential** of contemporary FES–systems: it gives some people limited use of their legs; over a dozen people regularly use FES walking systems, having first electrodes implanted in their bodies and requiring the support of a walker; a person generally needs less than 5 minutes to equip himself to walk and average walking distances are measured in tens of meters. Also a number of **limitations** are stated ("formidable mutidisciplinary problems remain"): applying FES to walking is probably the most complex work being carried out, because of the strains involved in the leg muscles and the urgency of maintaining balance during each walk cycle of 2 seconds duration (on average), while an able–bodied person may expend only one–third of his energy on walking, a disabled person in an FES gait will be using his muscle system to the full and it is reminded that with FES the axons cannot be fired such that the muscles are stimulated in a natural, asynchronous way.

He states that a **long–term goal** for FES researchers is "unobtrusive implantable systems that could automatically adjust the stimulations" and the result would be "a less cumbersome apparatus that would provide near–normal functions to people with paraplegia, most likely with the support of crutches or a cane". As to the **user potential** he remarks that "every year as many as 12,000 people in the United States alone are paralyzed because of a spinal injury" and that "the US population of the wholly or partly immobilized includes some 186,000 people with spinal cord injuries". Furthermore, he gives two **alternative approaches** for the job of conferring mo-

bility on injured people: repair of damaged nerve tissues (a long–term approach) and in the near term "improvements in powered wheelchairs and more access for their users to buildings and public bathrooms".

About FES.

He explains about FES systems: "Basically, a computer coordinates sequences of electrical pulses through a network of electrodes attached to the body. This causes the appropriate muscles to contract and effect movement and control". More specifically, he states that "in contrast, FES helps handicapped people walk by recruiting motor neurons in reverse, energizing those in the fast–twitch muscle fibers first".

With respect to the stimulation control and the job of programming the muscles, a number of remarks are made: "The sequences of pulses vary, of course, depending on the desired movement. Within repeated movements, pulse widths are often varied for a given muscle to recruit different fibers" (in order to minimize fatigue, just like the mechanism present in the human body); "Feedback comes from up to eight channels of force–sensing resistors for measuring foot contact pressure. Other channels are available for accelerometers, for goniometer sensors to measure joint angles, or for other sensors that may also help in the closed–loop control of the stimulation"; "each person using an FES walking system requires different programming, and programming needs alter over time. It is now a rather imprecise art that relies mainly on preprogramming and trial–and–error tactics using general motion analysis models. Nor is there much real–time individual feedback, other than pressure sensors on feet" and finally: "Tuning of stimulation patterns is essential until a stable state is arrived at (...) **When motion and analysis data are combined with ground reaction forces (...) there is enough information to determine both the positions of body segments in three–dimensional space and the joint moments generating the movements**. As a result, changes can be made in stimulation patterns to create a more natural–looking gait".

Closed–loop control of FES.

Clearly, an adaptive, closed–loop approach to stimulation control is favoured by the author. Earlier he stated: "Providing the ability to stand alone will not be possible (...) until real–time feedback control with a greater degree of sophistication – the coveted closed–loop system – is made practical", while in the end he remarks: "An elusive goal is resposive closed–loop systems that adjust in real–time to the changing state of the subject's muscles and joint angles. (...) closed–loop system would be required to give a person more stability and the ability to stand erect with both hands free (...) But closed–loop systems require information from sensors mounted on the skin or from braces that would modulate stimulation in real–time to produce the desired results. (...) this information may be too difficult to achieve for **walking**, a feat that combines **nonlinear properties of muscle** with **time dependency**". This is in accordance with his earlier statement, that researchers still have much to learn about controlling this bio–electric system, with its multiple degrees of freedom and nonlinear, time–dependent variables".

To overcome these difficulties, he thinks of *fuzzy logic control* as a much more promising approach. With fuzzy control, "corrections, based on experience gained by experts with preprogrammed stimulation, are made not in real–time but at each succeeding step. Adjustments are made by measuring the deviation from parameters such as knee angle, with adjustments continuing until the deviation is brought to zero".

Summarizing, the author advocates the use of biofeedback in order to obtain closed–loop controlled FES walking systems, but casts some doubt on the feasibility of real–time adaptive controlled artificial human walking. He thinks of "cycle–to–cycle" fuzzy control as a much more promising approach.

2.2 Control of FES: conceptual framework.

In the attempt to provide paraplegics with the ability to walk, many different aspects of artificial gait synthesis have to be taken into account. Based on [22] and [24], we give an overlook of relevant research issues.

2.2.1 Modeling, control and optimization of FES–induced walking.

According to the MObility REStoration–project (MORES), a number of research aspects have to be taken into account and combined in order to obtain means to restore functional movement (of the lower extremities) in paraplegics:

- development and refinement of *FES techniques* and orthoses
- selective motor unit *stimulation* with endoneural prostheses
- *control* of FES in hybrid systems for the restoration of walking
- evaluation of *walking itself*

In other words, aspects concerning stimulation technology, body stabilization, gait analysis, and stimulation control have to be considered (amongst others). This implies the existence of a biomechanical model of of the (walking) human body. Also, because of underdetermined problems, optimization techniques have to be applied: "The desired movements of the biomechanical system are given in terms of movement parameters, describing the walking pattern. The muscles to be stimulated and the joints to be braced to achieve these movements are not a unique set. The problem is redundant".

Note that *hybrid* systems are considered, in which FES is combined with mechanical components: "Passive braces offer a means of support, alleviating the need to continuously stimulate support muscles" and also: "In each leg, about 46 muscles are used during normal walking. It is in practice impossible to stimulate all of these muscles. When only a few muscles are stimulated to achieve a normal walking pattern, additional stabilization provided by a walker or crutches will often be necessary". Artifical walking using orthotics only is disregarded, because they provide limited mobility at high metabolic energy cost and its benefit is limited to offering support to certain joints only. In the Introduction a number of problems with the sole utilization of FES in artificial walking have already mentioned.

2.2.2 History and characteristics of FES.

In 1951, Liberson designed a portable stimulator to obtain functional movement of the foot. Many applications have been developed ever since; however, these have not been widely accepted due to lack of safety, comfort, cosmesis and other problems. In Rehabilitation Engineering, FES is mostly used to help restore partially the functional movements of paralyzed limbs in humans.

Current FES systems stimulate the peripheral nerve (or its terminal fibre); the muscle itself is not stimulated due to a higher (excitation) threshold value. FES can be applied in three different ways, each having its own distinct (dis)advantages:

- transcutaneous stimulation: a surface electrode technique, which is widely used because of its noninvasive character. Problems that occur are (a) poor muscle selectivity, due to misplacement of the electrodes and the geometry of the conducting medium during stimulation and (b) poor day–to–day repeatability in muscle response;

- percutaneous stimulation: wire electrodes through the skin make contact with the underlying tissue, improving selectivity. This technique brings about high risk of electrode dislocation, infection and electrode breakage, and it is conderably less patient–friendly;
- implanted systems: theoretically, all the above problems can be circumvented, although this has not yet (1991) been validated. This technique is likely to become the future standard in FES systems;

Design of electrodes that provide safe, repeatable, specific and graded activation is a research topic of current interest.

Contemporary clinical FES systems.

Research in FES can be divided into basically six areas, that can be studied separately, although they are interrelated:

- 1. patient interface with the FES system; the user must be able to initiate (perhaps: regulate) the muscle function in the lower extremities to generate locomotion, both by voluntary and reflexive upper body control. The *command control interface* (e.g. a push–button) has to be user–friendly and must not interfere with voluntary movements. *Intention sensors* can be used to detect the possible ways to pursue a certain mode of locomotion
- 2. electrode development; improving electrode selectivity can ultimately overcome the nonphysiological recruitment order of muscle units, that arises with electrical stimulation
- 3. control of stimulation: open–loop; a preprogrammed stimulation process is practiced, so response of stimulated muscles is not used in adaptation of the stimulation parameters. It is assumed that the muscle is sufficiently strong and fatigue–resistant, such that the desired response is always obtained. The inherent time varying properties, however, limit the use of open–loop control
- 4. control of stimulation: closed–loop; the problems just mentioned can be overcome by on–line feedback of movement data (such as force and angular data). The resulting closed–loop controller should be robust, i.e. be stable and offer satisfactory peformance over a wide range of muscle properties
- 5. sensory feedback to the patient; feeding back sensory signals from the lower extremities to an area of the patient which is still sensitive (artificial *proprioceptive* feedback, i.e. providing internal stimuli to signal the relative positions of body parts), will provide the patient with comfortable confidence, which enhances the performance of the FES system. This can be extended by external stimuli, like visual of auditive *exteroceptive* feedback
- 6. hardware of the FES system; issues like controller type (currently, non–portable systems are driven by a computer, whereas portable systems use a programmable processor–driven stimulator) and choice of electrodes and their powering emerge. It has been suggested that a modular design of FES systems will prove to be most efficient

Currently available hybrid systems for FES include the following:

- Floor Reaction Orthosis (FRO), which is a knee–ankle–foot orthosis (KAFO). Stimulation is done on the quadriceps muscles (in order to obtain knee extension at the end of the swing phase and the beginning of the stance phase) and ankle dorsiflexion is caused by using the flexion withdrawal reflex (FWR) in the peroneal nerve. The system is controlled using closed–loop finite state, rule based control; measured signals are force signals from the walking aid and the FRO. Application showed results of varying success

- Para Walker (hip guidance orthosis, HGO), which is a hip–knee–ankle–foot orthosis, HKAFO). Again, the quadriceps are stimulated for walking, standing up and sitting down. Initiation of the swing phase is done by stimulating hip extensors and using the reciprogating action of the orthosis. Hip abductors are stimulated to prevent heel touch during swing. The system is controlled **open–loop**; application showed that locomotion was very restricted, energy consuming, and, **due to the unintelligent control, not very natural**

- LSU Reciprogating Gait Orthosis (LSU–RGO), which is also a HKAFO. Stimulation of the quadriceps is used for standing up and sitting down, while the Glutea muscles are used to stabilize the hip and prevent hip–joint deformation. Propulsion can be achieved by stimulating the hamstrings (to initiate the reciprogating action) or using the FWR. The system is controlled in a **hierarchical fashion**: on a high level, the patient can initiate the desired step and forward velocity, and initiate a step forward by simple push–buttons. On a low level, closed–loop control is done using measurements of angular position (hip, knee, angle) without interference of the patient. Application of the system showed a great improvement in energy consumption compared with FES only experiments

This description is by no means complete, but is included to give the reader an idea of the practical implications and utilization of FES systems.

2.2.3 Biomechanical modeling and optimization.

When we want to control the human body during FES–induced walking (whether this is done open–loop or closed–loop) we need to model both the human body and the activity of (regular) human walking. Ultimately, FES–induced walking should be modeled also.

The human body is very complex, all biological material present exhibits nonlinear and non–isotropic passive and/ or active behaviour. As an exact representation of the human body in mathematical formulas will lead to an extremely complicated model, different simplifications have to be carried out, e.g. restricting walking, which is a 3D–movement, to the sagittal ("forward movement") plane and assuming left–right symmetry. A total walking model includes the following parts:

- I. segmental model of the lower skeletal extremities, consisting of a number of segments (rigid bodies with dimensions and inertial properties that can be deduced from the human body) [21]
- II. anatomical model, incorporating anatomic properties of the body parts represented by the previous model (such as masses, moments of inertia, lengths, etc.)
- III. muscle model, representing a single muscle and (sometimes) incorporating the relations between muscle activation and force development
- IV. muscle attachment model
- V. models for a specific joint, incl. muscles, ligaments and special characteristics

Modeling FES–induced walking implies some differences with modeling normal walking. We have to take into account

- external muscle stimulation (because of low muscle saturation, this is often done in an on/off–mode)
- fewer muscles to model (with the possibility to choose the muscle set)
- use of a walking aid (often), leading to system undetermination
- high rate of fatigue
- low ultimate strength of stimulated muscles

Also, considering that one of the problems with FES is, the selection of muscles to stimulate for an optimal result, it is obvious that the dynamical characteristics of musculotendon systems, i.e. force–length and force–velocity properties of the muscle, should now be included in the total system. These characteristics can be modeled by one basic equation:

$$\frac{dF^T}{dt} = f\left[F^T,\ l^{MT},\ v^{MT},\ a(t)\right] \tag{1}$$

where

- F^T is the instantaneous tendon force
- l^{MT} is the instantaneous musculotendon actuator length
- v^{MT} is the instantaneous musculotendon actuator velocity
- $a(t)$ is the muscle activation

Notice that prediction of muscle activation can help us in predicting future exerted muscle force. Based on current muscle activations and kinetic and anatomical information, muscle fatigue can be detected and compensated. An alternative method to this way of forecasting muscle fatigue is by including recruitment characteristics of motor units in the control variables of each muscle (group).

Analysis and synthesis of FES–induced walking using biomechanical models has not been studied extensively; investigations are usually directed to the *control* of FES systems. In other words, a *black box approach* is usually adopted, probably due to the complexity of analysis and synthesis of FES–induced walking. This resembles our approach ...

Optimization in (FES–induced) walking.

A number of problems in artificial gait synthesis exist, that have to be solved using optimization techniques:

- during double leg support (in human walking), the number of unknowns in the system (joint forces or rotations) exceeds the number of equations of motion, so the system is underdetermined; attempts have been made to solve it by minimizing joint–torque
- a number of kinematic contraints have to be fulfilled at the same time [21]
- *force sharing problem*: the muscle redundancy of the biomechanical system ("a certain movement can be caused by a number of joint torques or muscle activations") can be tackled by optimizing an object function, such as minimization of (weighted) muscle force ("total muscular effort–minimization"), minimization of the differences between

the computed and observed displacements using a direct dynamics model or maximization of endurance time ("fatigue–minimization"); taking all available muscles into account (some 46) leads to an extensive optimization process, so the additional problem arises which muscles to include (and, within the context of our problem also: stimulate)

- an optimal activation pattern is to be determined for FES–induced walking

The force sharing problem can be seen [40] as indeterminacy (flexibility) at the motor level. Walking involves the integrated movement of muscles acting across many joints. It is quite possible to achieve the same movement (as measured kinematically) from a score of different combinations of muscle patterns: many muscles involved in walking are *synergists* or *antagonists*.

Optimization methods include linear programming and dynamic methods like *optimal control*, which is characterized by 7. a state equation, 8. initial conditions for the state variables, 9. a set of allowable controls and 10. an objective function that has to be optimized. Furthermore, optimization of an (unconstrained) function can be done with direct and gradient search techniques (like e.g. simulated annealing, that can be implemented using neural networks).

2.2.4 FES control schemes.

With stimulation and biofeedback technology available, the link between them is established by the *system controller*. It is, however, not a priori evident that feedback signals are being used; also, the level that control is being performed at (directly stimulating all the muscles, or adopting a hierarchical strategy) can differ. Currently [24], we can divide FES control approaches in two ways:

- high–level vs. low–level control, where a hierarchical approach is followed; lower level subsystem controllers generate stimulation sequences to achieve a desired movement (e.g. in terms of biomechanical output) and a high level supervisory controller coordinates the actions at the lower level [24]. Sometimes, the *conscious level* is also considered: FES system interfacing *to* (computer display, sound, light) and *from* ((foot–) switches, joysticks, voice actuation, intact upper body EMG, insole or crutch force sensors, goniometers or accelerometers) the patient. The ultimate goal is, of course, to enable *subconscious* interfacing; this could be achieved by using interfaces that do not require conscious effort of the patient (avoid manual switches and voice actuators; provide the (intact body parts of the) patient with proprioceptive sensory feedback) in combination with an *intention detection scheme*, implemented in the high–level controller [22]
- open–loop vs. closed–loop control; current clinical hybrid systems for locomotion rely to a large extent on supervisory control without kinematical feedback, i.e. open–loop control: muscles are driven into saturation in order to obtain a (relatively) reliable response. In closed–loop control, position and motion data (perhaps also commands) from the patient are used to adapt the stimulation, in order to account for disturbances and fatigue [24]

We will make further comments on these approaches below.

2.2.4.1 System characteristics.

The lower extremities exhibit highly nonlinear characteristics [22]: the passive limbs show a dominating inertia component together with nonlinear effects of gravity, damping and stiffness; FES–induced muscle force depends in a nonlinear way on fiber and recruitment characteristics and actual muscle length and shortening velocity; electrically stimulated muscle exhibits a dead band (a threshold below which stimulation input does not evoke muscle contraction) and a significant **delay** between stimulation onset and elicited muscle contraction; FES–induced joint torques are small compared to the inertial properties of the lower extremity system, so lower limb movements elicited with FES are mainly determined by the inertial constraints. Note that the last characteristic also implies that **the feasibilities for the compensation of significant external disturbances within a step cycle are limited**.

The characteristics of the system have implications for the control of it. Lower extremity movements show a repetitive invariant character (walking is a cyclic activity; dominating inertial characteristics support the invariant character), so *cyclic parametrization* (identification of these invariants) can be helpful in the design of an FES control system, e.g. to find a *finite state* representation, which is essential for the coordination of task specific muscle actions. Invariant characteristics can also be observed in other lower extremity tasks, such as standing up and sitting down.

The *conventional* way to perform closed–loop control (by *trajectory following*) in such problems (e.g. in Robotics) can be of use in controlling hip and trunk stability [22], and also seems suitable in the control of upper extremity neuroprostheses.

2.2.4.2 Low–level control.

In this mode, a single muscle (group) is actuated (stimulated) to control a single degree of freedom. The electrically stimulated muscle is a highly nonlinear, time varying system with low saturation. Generated force depends on a number of factors:

- number of recruited motor units
- dead band of the stimulated muscle (time–varying, due to fatigue)
- saturation of the stimulated muscle (time–varying)
- reverse recruitment order of the muscle fibres

The number of motor units that is recruited can be adjusted by modulating either pulse width or amplitude. Unfortunately, dead band and saturation seriously worsen controllability of the system. In addition, most FES systems recruit the motor units in a nonphysiological order, i.e. big fast twitch units are excited before the small slow twitch fibers.

In both open–loop and closed–loop control a model is necessary to either calculate the (preprogrammed) stimulation sequences or tune the closed–loop controller. This requires *identification* of the biomechanical system at the level of interest, e.g. identify the muscle–actuator dynamics. More specifically, this means identification of the relation between e.g. contraction velocity, motor unit recruitment, and muscle activity; recruitment order, fiber composition and the force–velocity relationship; muscle stimulation and response, i.e. pulse width and produced torque; muscle stimulation and resulting flexion withdrawal reflex (FWR), which is often used to produce the swing phase of gait and has to be characterized in order to be able to control it. FES offers the possibility to stimulate the muscle (open–loop) to identify the muscle dynamics, which is particularly efficient for muscles in paraplegics that cannot be controlled voluntarily. This method is the alternative of **EMG measurements during voluntary muscle contractions** of normal patients.

Open–loop control at this level amounts to the application of preprogrammed stimulation sequences (calculated from identified models or obtained by trial–and–error); it is assumed that

the response of the stimulated muscle is always sufficient to generate the desired torque. From studies on open–loop control some interesting facts can be concluded: high frequency block stimulation reverses the recruitment to normal; efferent stimulation has the advantage of reducing latency and variability and avoiding habituation and thus improving the feasibility to control the response; and trial–and–error determination of stimulation sequences is not very useful (error prone because of day to day changes in muscle fatigue; hence, the procedure has to be repeated every time prior to an application). It is reported [22] that current (1991) on–line closed–loop control schemes **do not seem to function much better** due to unknown muscle properties, incorrect adaptation due to time delays and dead bands, inadequate sensor signals, etc. The feedback approach, that theoretically should exhibit distinct advantages, has to be **investigated extensively** before it becomes de facto preferable to preprogrammed control.

In low–level *closed–loop control* stimulation patterns are modified by using biofeedback signals. Modulation of force can be obtained by either adjusting pulse frequency (adapt *temporal summation*) or pulse area (also called *interpulse interval stimulation*: modulation of recruitment). Alternatively, burst time (or burst duration, time during which is stimulated) can be adjusted. Both modulation techniques offer a means to compensate for fatigue. PID–controllers have been used for closed–loop control using recruitment or burst time modulation, and to control the cyclical movement of the lower leg with electrical stimulation of the quadriceps muscles (optimizing knee torque, using reference angles for every cycle, and using maximum (physically allowable) amplitude in an on/off manner) [16]. More sophisticated approaches use adaptive recruitment modulation, in which the control algorithm is updated based on input and output measurements (*model reference adaptive control*), or practice nonlinear control (optimal control, **neural network** control [16]a , etc.)

Feedback signals used with closed–loop controlling FES include: positional data (e.g. knee angle [16]) and (isometric) torque signals to be compared with a reference value, EMG, force distribution under the foot, muscle tone and discrete events in a cyclical movement, such as maximum angle [16], and discrete events describing **part** of the walking cycle (measured by positional data and/or foot switches). The discrete events play a more important role in the high–level control system [24].

2.2.4.3 High–level control.

We already stated that the multi–goal and multi–variable control problem under consideration is often decomposed into sublevels, e.g. in a hierarchical way. At the top level, (sub)conscious patient intention to initiate or pursue a certain (locomotive) movement is registered, and low–level control is synchronized, activated and supervised (controlled) according to these wishes. High–level *system identification* consists of the identification of all low–level (sub)systems and their interaction, and the design of a special protocol to identify multi–articular driven biomechanical systems (e.g. the FWR recruits muscles that activate several links of the lower extremities); for example, the following strategy can be used: system modeling, structure and parameter identification, model adaptation. The identification of lower extremities biomechanical properties can be divided into several parts, described already in Section 2.2.

Open–loop control at the top–level consists of (synchronized) preprogrammed stimulation sequences for the different low–level control units. Initialization of a stimulation sequence is mostly done by a patient pushing a button (hence having no influence on the stimulation timing). Main results obtained with open–loop high–level control of FES include: open–loop control of the freely–swinging paralyzed leg, using an iterative trial–and–error procedure to cause the knee angle to follow a desired trajectory (where high– and low–level control are not explicitly distinguishable), standing of spinal cord injured, and functional walking in paralyzed patients by means of intramuscular stimulation. Also, a framework for a certain FES system is reported, containing: (commands) – command processing – (control parameters) – movement planning –

(movement parameters) – coordination and regulation – (stimulus parameters) – stimulus generation – (electrical stimulation); where data is written inside parentheses, separated by processing units.

High–level closed–loop control systems.

High–level *closed–loop control* implies the existence of feedback signals, such as gonio data, force data, etc. A number of existing high–level control systems is now described.

Initially designed to function with the hybrid system based on the FRO (see Section 2.2), a control system based on *finite state rule–based hierarchical control* is reported. Standing up, standing and walking using the system are described by a finite number of states, with corresponding state transitions (designed to enable paraplegics to take occasional rest breaks to recover from (physical– ; FES–induced muscle–) fatigue. Transfer from one state to another can be induced by the patient by means of a user interface, perhaps facilitated by EMG measurements, and foot pressure or crutch load measerements.

In another approach, the use of EMG as a feedback (or feedforward: *predictive control*) signal to control FES in a hybrid system is investigated. The system employs above–lesion surface EMG to activate standing and walking in a patient–responsive manner, and below–lesion EMG for the regulation of stimuli levels in the face of fatigue. **Patient intention is extracted on–line via identified time series parameters of EMG–signals from voluntary contractions of upper limb muscles**. It turned out that below–lesion EMG control of FES levels had a better performance in terms of standing duration than above–lesion EMG control. It is also believed that the strategy can be used with all FES techniques present; however, with feedback of below–lesion EMG, stimulus artefacts will make distinction of the actual EMG difficult.

Hybrid Assistive System (HAS) and related research.

Popovic [4] describes a skill–based expert system for the control of motion in biomedical robots. This is an extension of the Hybrid Assistive System (HAS) controller, which is nonnumerical and based on *artifical reflexes*: externally powered and controlled joints are activated by artificial proprioceptive and exteroceptive sensors. Input to the joint controller is also used for recognition of the sensor patterns (by nonnumerical, logic expressions), which are directly matched to one of the actuator states *loose*, *locked*, *flexion* and *extension* (see also [23]). The logic rules ("pattern matching operator") are derived by knowledge capturing using gait records, called *gait invariants*, i.e. sequences of singular events with a specific locomotion cycle. The resulting rule base contains three kinds of rules: regular, hazard, and mode rules, so a certain mode can be triggered (e.g. a regular mode, such as normal walking) while emergency phases will also be detected. Transition between phases in gait is governed by the rule base also. **The heuristic approach is applied since there do not exist general algorithms for the problem at hand**.

Implementation of the above described system was done for use with an active above–knee prosthesis (AKP) and hybrid orthosis for spinal cord injured paralyzed humans (HAS). Experimentation, limited to quasi–static movements of the patient with the help of walking aids, proved the feasibility and efficiency of the approach and it is stated to have great potential for improvements. It is also reported that "a more complex system for rehabilitation of lower limb amputees" is developed, offering: walking on level ground, walking up and down the ramp and stairs, automatic speed adaptation, standing up and sitting down, walking backward and turning around". Furthermore, the approach, derived by observing heuristics of human motor acts and neurophysiological organization of movements, is stated to be a relatively simple control solution, since the "multivariable nonlinear nature of motor control in man and animal can not be easily mastered by numerical methods" and "the large diversity of sensory–driven functional motions encountered in nature represents a serious challenge to control theory". The author also thinks that "the learning process by which a skill becomes controlled by reflex and automatic mechanisms

is also not understood", but that "by multidisciplinary efforts of life and engineering sciences it should be possible in the future to arrive at a much better understanding of heuristics of motor skills" [4][24].

We want to mention briefly the work done by Popovic et al. [4]a b c. In the attempt to determine gait invariants (identification of invariant features) for closed–loop control of FES, adaptive logic networks (ALNs, artificial neural networks with nodes realizing AND and OR functions, thus resembling a rule–based system) are used for the automatic recognition of sensory nerve recordings. In fact, a mapping is established between afferent neural signals (like *electroneurograms* (ENGs) from *superficial peroneal* (SP) and *tibial* (TI) nerves) and muscle activation of the same or the opposite limb (EMGs from *medial gastrocnemius* and *tibialis anterior* muscles). It is found that the processed ENGs are much more reproducible in amplitude from step to step than the corresponding EMG signals (see also Chapter 3). Furthermore, the ALN could reliably match the exact timing of EMG onset and average duration (EMG was represented binary in the ALN). The feasibility of multielectrode recordings from sensory nerves for rule–based control was demonstrated with a chronic cat model: long term recording and stimulation was effective (recordings from TI and SP were sufficient to trigger the appropriate periods of stimulation reliably to ankle flexor and extensor muscles). The reason that the ANN–approach is adopted (in "the nonparametric identification of the system consisting of two peripheral nerves and two muscles in the freely moving cat") is the lack of a good mathematical model for the transfer function between sensory nerve signals and myoelectric activity of ankle flexors and extensors. According to the authors, ALNs offer good performance with fast supervised training and execution and easy hardware implementation (as opposed to conventional backpropagation ANNs).

Continuing with our description of existing high–level control systems, another approach to closed–loop FES control is to match the i/o–relation of a joint to a desired system (usually second order linear); stimulation parameters are adjusted according to deviations between actual and desired outcomes, in other words practice *model reference adaptive control* (MRAC). Problems that occur with this approach are inaccurate modelling of the (time–varying) muscle output saturation and low muscle saturation (demanding on/off control). Opinions concerning the feasibility of MRAC to control FES differ.

At the University of Twente, the principle of finite state control is applied to the knee joint during standing [23]. The resulting controller showed robustness, allowing reduction of average muscle force (compared to open–loop simulation) and provide continuous dynamic activation of the knee extensor muscle. Repeated calibration and reduction of effective stimulation time (because of intermittent stimulation) were reported additionally.

2.2.5 Artificial walking system.

On the basis of the previous material and the surveyed literature reported below, we propose a schematic framework for a future system for artificial human walking in Figure 1. The reported research can be positioned according to this schema.

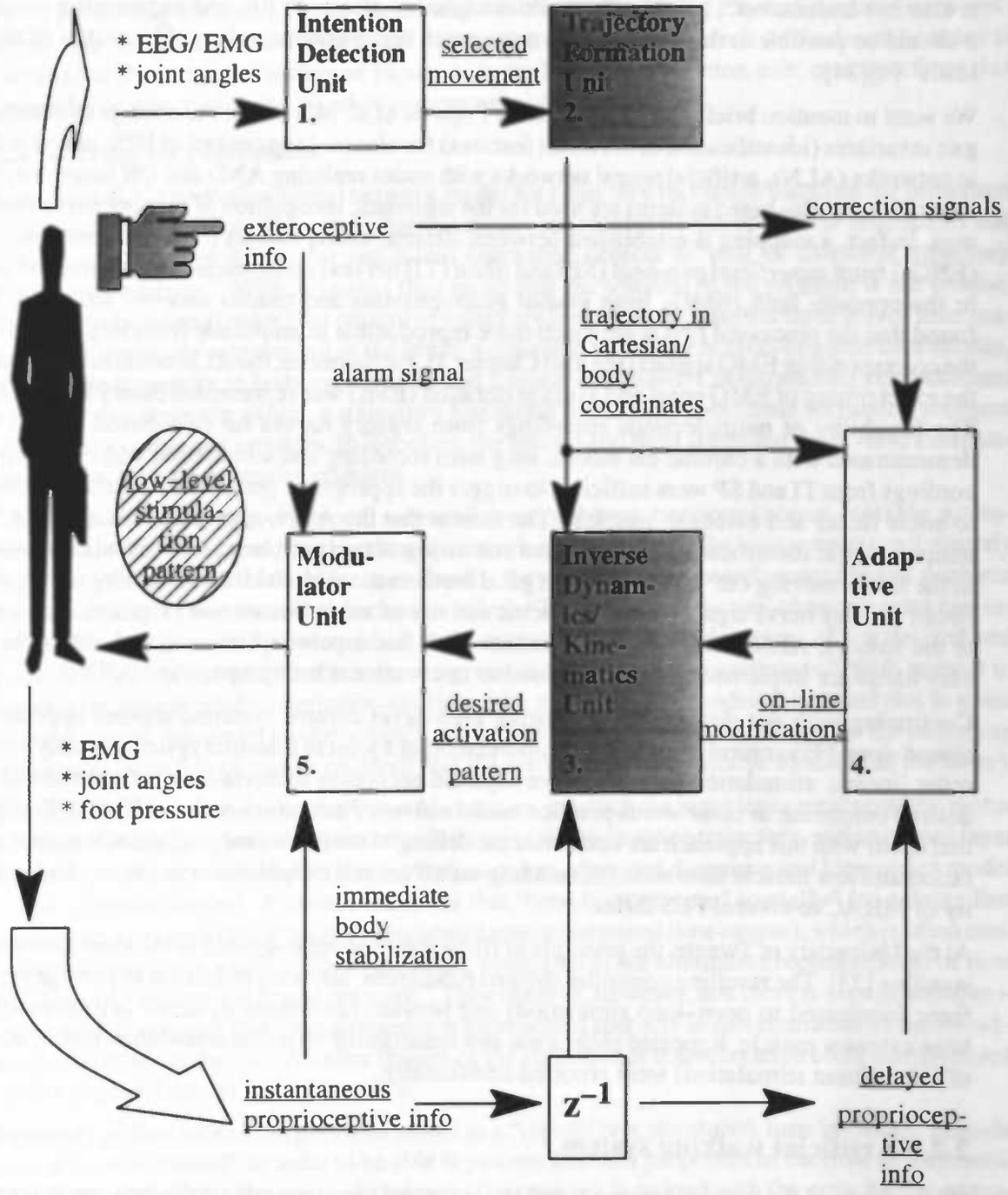

Figure 1. *Components of an artificial walking system*

2.3 Literature review.

We now describe some investigations into different (areas concerning) parts of our anticipated walking system. Often we turn to biomedical engineering literature on this subject, where an increasing number of investigations is directed to neural control of FES.

It is also worth noting, that a good deal of research in the area of *Robotics* is of interest to the problem of artificial human walking. Material on autonomous vehicle control, generation of gait patterns, control of artificial limb movements, sensorimotor control, etc. offers many ideas and methods that are applicable to the problem at hand (although they are often designed to operate on a simplified, well–understood model of the human or vertebrate system).

2.3.1 Planning and intending a movement.

In the human motor control system, a movement is initiated and planned in the central nervous system (CNS); there, it gets translated into motor commands. In fact [39], locomotor programming occurs in supraspinal centers and involves the conversion of an *idea* into the pattern of muscle activity that is necessary for walking. The resulting neural output may be thought of as a central locomotor command being transmitted to the brainstem and spinal cord. According to [22] the pattern generator for human locomotion resides in the spinal cord; firing the motor units in the associated leg muscles occurs at various recruitment levels and rates in a preprogrammed manner. Afferent feedback signals are used as indicative data by the CNS, to trigger certain locomotion phases and adapt the motor program for ensuing steps (e.g in muscle fatigue). Triggering reflexes (e.g. with sudden disturbances) by the CNS using afferents is limited due to the delayed arrival of afferents and slow transient response of associated muscles.

In the peripheral nervous system (PNS), the motor commands are modulated via sensory feedback to achieve actual muscle activity and functional movement [12]. Also in [40], the CNS must integrate desired efferent commands with peripheral feedback and vestibular and visual inputs to generate the correct patterns of *moment of force* at each joint. Referring to Figure 1.we could think of the first four components as replacing the CNS and the fifth component as an artificial PNS (or, alternately, the first four components represent the high–level controller, whereas the modulator implements the low–level control).

2.3.1.1 Approaches to stimulation pattern generation.

We remark that units 2., 3. and 5. together in fact constitute a stimulation pattern generator. Much work on neural networks addresses this more general problem, i.e. how to generate on–line an adequate stimulation (or, in Robotics, actuation) pattern, given a specific movement (e.g. grasping or walking). These systems often contain a generator of the desired trajectory (e.g. on the basis of *optimal control theory* or some fatigue–minimizing or smoothness criterion) and a neural network which maps it onto a desired (joint) torque, (muscle) activation or (muscle) stimulation pattern.

Another approach could be to use a library of low–level movements (body coordinate trajectories, preprogrammed stimulation patterns), combine these with a fuzzy system to form new movements and (in the trajectory case) derive the new stimulation pattern from the existing.

Yet another is the use of a preprogrammed stimulation pattern, that will do fairly well on the average and adapting it to the patient and the environment (real–time or cycle–to–cycle), while satisfying a number of (physical and practical) constraints. With this approach, the desired trajectory is often expressed as a target pattern of joint angles or other objectives as a function of time or walking cycle.

Intention detection.

The concept of *intention detection* is brought into play because, when mimicking the CNS, we would like to be able to acquire the user's intended movements. Nowadays, this is usually done by letting the user push a button or perform some movement by his (intact) upper extremities. The most natural way would, of course, be to extract the intention automatically out of body configuration and muscle activity patterns (ground reaction forces, goniometry, EMG) or, eventual-

ly, by processing the user's motor commands directly (by analyzing ElectroEncefaloGram–patterns). We remark that the vast amount of research into *EMG–analysis and –decomposition* is included in the proposed system of Figure 1. in the Intention Detection unit; concepts from this research can also be helpful to other components of the system.

2.3.1.2 Trajectory formation.

Kawato et al. [2] propose a neural network model for trajectory formation based on the minimum torque–change criterion. They state that this model can resolve ill–posed inverse kinematics and inverse dynamics problems for redundant controlled objects as well as ill–posed trajectory formation problems". With *ill–posed* they mean that the solution to the problem is not unique. Their "computational model for control of voluntary movements" assumes three sets of information to be internally represented in the brain: a desirable trajectory in task–oriented coordinates, a desirable trajectory in body coordinates and a motor command. Corresponding computations which derive them are called: a trajectory formation problem, a coordinate formation problem and a motor command problem. In Robotics, the second and third are called the inverse kinematics and the inverse dynamics problem.

The proposed neural network model solves all three problems mentioned above. Notably, a computational scheme adopted in this neural network model obtains the motor command directly from the goal of movement. More specifically, they solve the ill–posed motor control problem by introducing a smoothness performance index: "The minimum torque–change model can resolve all three ill–posed problems (...) at the same time when the locations of the desired end point, desired via points and obstacles are given in task–oriented coordinates." With respect to another frequently used smoothness–criterion (the minimum jerk model) it is stated that in some experiment the minimum torque–change model predicted the real data better than the minimum jerk model (which is of course no proof that the former is a more suitable model than the latter).

The network has a five–layer feedforward structure, in which the input layer represents the motor command (or, in fact, torque), an intermediate layer the body space trajectory, and an output layer the task space trajectory. Hence, the network acquires a forward dynamics and kinematics model of the controlled object. A salient feature is that "time is represented spatially" (so a delay line is required to transmit the spatially represented motor command time course), which is often considered biologically implausible in the brain, compared to the alternative representation of time (as "time of a dynamical neural network". They observe, however, that there is experimental evidence which suggests that time difference is represented spatially in determination of the sound–source azimuth by the owl. Another feature of the algorithm is that relaxation computation based on the acquired model takes place in it.

Proceeding, they extend the previous model to a "repetitively structured, time invariant, cascade neural network model" in order to be able to possess intrinsic properties of the flow of a dynamical system, describing the controlled object. The network is trained with the error backpropagation algorithm, using the realized trajectory of the controlled object (which is provided with the same input torque as the neural network model) as a teaching signal to acquire the forward dynamics model between the first and the third layers of the network unit. In the testing, or pattern generating, phase, the central commands which specify the desired end–point, the desired via–point and the locations of obstacles are given to the 4th layer from the higher motor center, which receives the necessary information for the trajectory specification. Stated otherwise: "In the learning phase, the output line line is inhibited. In the pattern generating phase, the input signal is inhibited"

The system is designed to relax to a stable equilibrium point of the network dynamics and hence output the torque which realizes the minimum torque–change trajectory. This method bares a strong resemblance to a well–known numerical method in optimal control theory, but differs in

that it does not utilize any co–state (in this case: the derivative of the torque). The most marked advantage of the ANN is that contraints can be imposed on the motor command by directly constraining states of the neurons which represent the motor command. With respect to the redundancy–issue they state that "if the controlled object has redundancy at the inverse dynamics level, we only need to increase the number of neurons representing motor commands". The ill–posed inverse kinematics problem is also solved, although in a less trivial way.

Simulations are done on a two–link arm model, which implies that: "Because the manipulator has two degrees of freedom within the plane, there is no redundancy at the kinematics level. Because the manipulator has only one actuator at each joint, there is no redundancy at the dynamics level". Results from learning the forward dynamics and kinematics model and generating the trajectory between two points and a via–point trajectory, respectively, show that the estimated trajectories by the network were almost identical to the teaching trajectories after learning, that there are consistent errors between the minimum torque–change trajectory, the estimated trajectory and the realized trajectory and that the estimated and realized trajectories were in fairly close agreement with the trajectory calculated by the Newton–like method.

Discussing their work, the authors think that "the cascade network can compute the trajectory in real time" despite the fact that it requires relaxation time (compared to the recurrent network model of Jordan, in which instantaneous generation of a trajectory takes place). One major advantage of their model is especially important if the network is used for trajectory formation of an articulator: "once it learns the forward model of the controlled object, it can generate any trajectory regardless of locations of the end point, intermediate points and obstacles to be avoided".

We think of this research as important and promising, although some doubts concerning the applicability to our problem remain: a paraplegic cannot provide a realized trajectory of a controlled object, the method does not apply to cyclical movements, no mention is made of closed–loop control by using feedback, i.e. muscle fatigue and body stabilization is not accounted for, etc.

2.3.1.3 Artificial locomotion.

The research in [6] deals with the problem of calculating a set of control torques that are used to generate a locomotive gait for a bipedal robot. In developing a formal control model of biped walking, the authors report some facts: "Locomotion is accomplished by alternately placing one foot on the terrain so that the supporting leg can propel the biped forwards while the other leg swings into place. From obsevations of normal human walking, the gait cycle can be subdivided into four unique phases:

- (i) right leg support phase
- (ii) right–to–left support phase
- (iii) left leg support phase
- (iv) left–to–right leg support phase

During right (left) leg support, the right (left) leg is in contact with the ground while the left (right) leg is in the free swing phase. In the support exchange phase, support is shifted from the right leg to the left leg, or vice versa. These phases are referred to as the *double support phases*. From the observations of normal human locomotion, it is clear that the ankles contribute very little to the overall locomotion cycle of the biped, in terms of the controlling torques. The dimensions of the feet are comparatively small when considering the large leg displacements produced during normal walking. The ankles would have no significant effect on the control of the biped over the complete locomotion cycle; the hip actuators can be effectively utilized to generate a required locomotion cycle".

Their *neural network torque generator* consists of two leg support NNs (one for each leg) and a leg support exchanger. The NNs (two fully–connected multilayered Perceptrons) were trained with the error backpropagation algorithm, using training data from three different linear control models. The inputs to the network were the state variables of the dynamic system (body angles), and the outputs were the required hip actuator values; thus the inverse dynamics problem is considered. During training, the body attitude of the biped was varied so that information on stabilizing body posture was included; the hip actuators were responsible for body stabilization during locomotion. After 50 hours of training, the biped was able to maintain the gait cycle for a considerably longer period of time when compared to the linear controller. At some point in the simulation, when the biped has almost fallen backwards, the neural network was able to generate an appropriate torque value at the hips to force the biped to continue its gait.

The authors suspect many implications of their results: as it is possible to determine the torque requirements of a human subject, "a neural network may be trained to represent these (...) for certain unique values of the system state". Eventually, it is remarked that "only position and velocity feedback were used to determine the proper step to take during walking. If one were to study a human subject attempting to walk, it would accept a variety of sensory feedback, which may include visual, balance, position, velocity and tactile inputs, analyze this information in real time and effectively plan a series of necessary steps to complete its gait cycle. It is clear that more of the feedback senses must be utilized in order to determine a sufficiently successful gait cycle".

Clearly, a plea for real–time closed–loop control of artificial walking, stemming from an attempt at gait synthesis in a realistic, human–like setting, is being made.

2.3.2 Relating spatial coordinates to muscle activation.

Then we turn to the heart or our system: the unit that maps a desired trajectory onto a suitable sequence of muscle activities, i.e. solves the *inverse kinematics* and *inverse dynamics* problems. We already noted that systems addressing this issue often contain a gait pattern generator or have at their disposal a desired trajectory in Cartesian or body–coordinates.

2.3.2.1 Predicting body coordinates from myographic data.

In [11] the authors are trying to model the relationship between muscle activity and lower–limb dynamics during human gait by means of an artificial neural network. They describe a cause–and–effect sequence of events that takes place for human walking to occur:

- VI. registration and activation of the gait command in the CNS
- VII. transmission of the gait signals to the PNS
- VIII. contraction of muscles that develop tension, with concomitant generation of electromyographic (EMG) signals
- IX. generation of forces at, and moments across, synovial joints
- X. regulation of the joint forces and moments by the rigid skeletal segments based on their anthropometry
- XI. movement of the segments in a manner that is recognized as functional gait
- XII. generation of ground reaction forces

Furthermore, they claim: "The relationship between steps VIII. and IX. has been the focus of considerable research in biomechanics (...) The basic problem, facing both the human walker and biomechanics researchers, is one of *distribution* (also referred to sometimes as *redundancy*): how are the large number of muscle forces distributed among relatively few joint moments?" Of the three strategies that have been followed in attempting to elucidate this question, nl.

- a. studying the relationship between EMG and muscle force
- b. estimating individual muscle forces by means of mathematical optimization theory
- c. trying to correlate EMG and joint moments

the *third* is believed to have the greatest potential, especially when approached with an artificial neural network algorithm.

Data for training the network were taken at twenty evenly spaced time intervals and were selected from the gait cycele of normal subjects walking at natural cadence. Furthermore, it originated from 16 muscles and three joint moments and angles (hip, knee and ankle) and were adapted from Winter [40], based on more than 10 normal subjects. Whereas the EMG patterns had large standard deviations, the joint angle and joint moment data tended to be much less variable, as would be expected from the literature.

Next, they propose a neural network with 16 inputs, one hidden layer with 32 units and three outputs, which was trained with the backpropagation algorithm. Results showed that the neural network can be used very succesfully to map an input vector of 16 muscles onto an output vector of three joint angles. The same was true for the second network, mapping EMG onto joint moments. But, since the succesful training of an artificial neural network does not necessarily constitute validation of the model, both nets were subjected to perturbations. These slight perturbations led to similar output predictions, which suggests that both nets are quite robust. Dramatic reduction to single muscle inputs (30% in *soleus* activity, 100% in *rectus femoris*) led, in the case of Net 2, to reasonable simulated predictions.

Some assumptions and limitations involved in the work are stated: "We have assumed that a particular combination of muscle activities and external loads will yield only *one* possible set of resultant joint moments. Mapping in the opposite direction is **not** possible: a particular moment or joint angle pattern could be generated by an *infinite* combination of muscle activity patterns", hence emphasizing the problems with an inverse dynamics approach to the problem of artificial human walking. Also, they "did not model the physical arrangement of muscles, force–velocity relationships, or sensory feedback information, although we have previously used the neural network approach for this purpose". Finally, the neural network model "did not include *temporal relationships* (i.e. parameters at time $t+1$ are a function of what happened at time t) although the learning algorithm did present the training patterns in sequential order (...) time does not appear explicitly".

Then, the question of choosing between mathematical (biomechanical) modelling or computer simulating a system is commented on: "The major shortcoming of these two [mathematical] models was that the authors did not set them up to explore the prediction of behavior (...) In contrast, we have emphasized this feature in the model presented here (...) The speculative nature of computer simulation is both its appeal and weakness, but rigorous validation still tends to be an extremely difficult task (...) Relatively few workers in the field of human gait biomechanics have attempted simulations of pathologic conditions (...) None, to our knowledge, have either validated or made a strong case in favor of accepting their simulations". They proceed with a description of the results from some surgical experiments, which "appear to support the verisimilitude of our prediction".

Finally, it is concluded that the joint moments simulated by Net 2 could be used as the input to the forward dynamics problem to solve for limb trajectories. Also, they think that further work should be done with the model so that it is trained on a larger data set of both normal and pathological subjects.

This research gives some very interesting insight into the possibilities and limitations of (neural network) modeling of gait dynamics. We note the serious limitation of an inverse dynamics ap-

proach: because of the redundancy of the human motor control system, mapping of desired joint moments onto necessary muscle activities is not possible, unless optimization criteria are used, or certain choices are made. Prediction of muscle activation in a multiple muscle system is hence much more complicated than in a system constrained to one single muscle. Also, the necessity of incorporating temporal features other than the presentation of training patterns in a sequential fashion in order to catch gait parameter characteristics (which are, in fact, *time series*) is established. The feasibility of neural networks direct dynamics mapping with clinical applicability is made

2.3.2.2 Control of cyclical movements.

An adaptive FES control system, which incorporates automatic adjustment to changing temporal characteristics is proposed in [16]a In particular, the paper deals with a neural system for control of the swinging lower leg. Cyclic movements of paralyzed limbs are generated by stimulating the quadriceps: "The torque that can be generated by the quadriceps depends in a nonlinear fashion on the angular position and velocity of the lower leg. Together with the nonlinear passive elasticities and gravitational force, this makes the relation between the movements of the lower leg and the stimulus of the quadriceps highly nonlinear." There are, however, some constraints in stimulating the quadriceps: the lower leg has to move cyclical and has to reach a maximum reference angle after each swing, while disturbances may be present; also, muscle fatigue must be minimized. Also, "the system should adapt to changing system parameters, in particular to the maximum achievable quadriceps torque, which decays due to fatigue".

For the simulations, a model that describes the relation between stimulus and angular position and velocity of the lower leg is extracted from the literature. Notably, three state variables (activation of the quadriceps, angle and angular velocity) are present in this model. The stimulation constraints are represented by a *cost function J*, which consists of two terms: the quadratic difference between actual and *maximum reference angle*, plus a term that is proportional to the time integral of the quadratic stimulus (fatigue is assumed to be related to this integral).

A 2–50–1 ANN (multilayered Perceptron) was trained by supervised learning (SL) on the basis of (*state, optimal stimulus*) – pairs, and by the *backpropagation–through–time* algorithm (BTT). When the network was trained with BTT after having been initialized with SL training, the best approximation of the optimal end–angle together with a minimum stimulus–magnitude was obtained. Performance of the neural controller in a number of situations (with or without disturbances, decaying maximum torque) demonstrated to be quite good (the reference angle was obtained within a few degrees, while, without adaptation, *average end angles* decreased significantly). It is concluded that, for testing the controller experimentally on a human leg, faster adaptation is desired.Weight adaptation will be evaluated on transputers.

In fact, this work embodies the cycle–to–cycle adaptation of stimulation advocated by Kobetic [3], though not in a fuzzy, but in a neural fashion. A main issue with on–line adaptation is, evidently, the speed and accuracy of the adaptation algorithm, even for cycle–to–cycle control, using precomputed (optimal) stimulations.

2.3.2.3 Fuzzy generation of motor commands.

A *fuzzy* approach to the inverse dynamics problem is given in [12]. In the introduction, some facts from physiology are memorized: "the control is not dictated solely by the nervous system in the brain; the muscle system also forms a local control system that adapts to different movements, loads and environments, in addition to accepting commands from the brain" and "voluntary movements are free motions dictated by the central nervous system via motor commands under different demands, such as smoothness and time constraint". The authors continue, stating that "the proposed control scheme will generate proper control signals to bring the limbs to various desired positions under different velocities and loads". In fact, the fuzzy system acts as a kind

of artificial CNS, whereas a local controller, representing the muscle system, is developed to modulate the motor commands via sensory feeedback. It is remarked, that a fuzzy system is chosen for the CNS rather than a neural network, because fuzzy logic is more effective in representing biological control strategies.

The control strategy is based on *equilibrium point control* (EPC), which has been hypothesized to be the strategy used by the neuromuscular system. The proposed scheme is said to be generic to the number of joints. With EPC, movement is treated as a transition between postures. "The CNS needs only select new levels for the central commands sent to the motor servo controlling the body's musculature. The subsequent result, mediated by autogenetic reflexes and mechanical properties of the muscle, should be a smooth transition from one posture state to another. Slow movements can be produced by progressive shifts in reflex thresholds. Movements can be speeded up by producing an initial threshold shift that is larger than necessary, followed by a return to an appropriate static level".

Of interest is also, that "in light of both physiological and engineering considerations, the number of pulses in the motor command should be kept fairly small". The authors propose to use three–pulse motor commands as the control signals, based on the investigation of rapid voluntary wrist movements, which normally demonstrate motor commands with a three–pulse pattern for a light load: a first agonist burst, followed by an antagonist burst, followed by a second agonist burst. Correspondingly, three major alternating phases of net force are observed: first in the direction of the movement, then opposed to the movement, and finally in the direction of the movement again (for a heavy load, however, the second agonist burst was not very apparent).In fact, a large burst generates a large initial speed, then a small burst brakes the movement, and finally a step brings the limb into the equilibrium posture assigned by the nervous system. Later on, it is said that when the control signal is specified in this simple form, the learning space is tremendously reduced for dealing with diverse variations exhibiting in different movements". In other words, an attempt is made at tackling the problem of *redundancy.*

The fuzzy system is used for the generation of proper three–pulse motor commands according to different distances, velocities and loads. Moreover, the authors attribure the intelligence of the proposed scheme to the fuzzy system and the three–pulse motor command pattern, and use the local controller to deal with sensory feedback only. In fact, the fuzzy system maps load, distance and velocity information to a motor command, which is modulated by the local controller and ultimately generates movement by activating a forward model of limb dynamics. First, single–joint movements are considered, proceeding they also include multi–joint movements.

The "proper (reference) motor command patterns" are derived by a learning process: a simulation system is developed (based on second–order modeling of a single–joint) to derive the force profile from desired movement positions, velocities and accelerations. Then, a fuzzy mapping between inputs and motor commands (or, actually, parameters characterizing motor commands) is made by evaluating simulated force profiles with desired profiles. A notable point is, that although three inputs, distance, velocity and load are called for in this fuzzy system, only velocity and load are sent into the fuzzifier. This is because the input distance will be directly proportional to the final step of the three–pulse motor command under the equilibrium point control hypothesis.

When the extension from single–joint to multi–joint movements is being pursued, the problem of *dynamic couplings between joints* emerges: "In Robotics interaction terms are generally compensated for through the use of feedback. For biological arms, however, feedback compensation through neuronal loops does not seem to be a viable option, because of feedback delay (...) In our design, dynamic interaction is tackled via an open–loop control using the fuzzy relation and three–motor command pattern". Also, loads, velocities and distances that are stated in Cartesian space now have to be related to individual joints: "In addition, as mentioned above, the distance,

velocity and load inputs are given in Cartesian space. To resolve these problems, we need to provide a proper mapping between Cartesian and joint space". In this design, apparently, both inverse kinematics and inverse dynamics problems are considered.

From simulations, it is observed that the relationship between the effective inertia and dynamic coupling is well captured by the rule–based system for load mapping in both cases; also, reasonable profiles were observed when the movements were within a short range under both whipping and reaching actions. Furthermore, the authors claim that "simulation results give evidence that the planning of the sensory motor system may be in joint space; they do not attempt to tackle this question, however. In future work, practical applications of industrial robot manipulators will be dealt with, including effects of gravity.

This "novel scheme for controlling multiple–limb movements" could be of interest for the problem of artificial human walking, although straightforward application to the real–world problem at hand is likely to result in many artefacts: extensive tuning of the fuzzy rules on the basis of desired force profiles has to be performed, while evaluation implies the use of a (complex, multi–joint) limb dynamics model. Once tuned, it can provide accurate and efficient open–loop control of a specific limb movement. In human walking, detailed mathematical models may satisfy for an initial situation, but do not account for varying system parameters. Fuzzy rules can get "out of date" this way; immediate adaptations are also not accounted for. Also, desired force–stimulation profiles may differ between individuals and even for a specific person; modeling of stimulation–force profiles may be significantly harder with FES than with Robotics applications, because of the nonlinearities and nonstationarities present.

2.3.2.4 An autonomous walking robot.

Another Robotics work that could bear some relevance to our problem is [10]. This article presents preliminary real–time results of a study of the application of on–line neural network learning to the problem of biped walking with dynamic balance. It is claimed that the experimental biped (a ten axis biped robot with foor force sensing) has learned the closed–chain kinematics necessary to shift body weight from side–to–side while maintaining good foot contact, has learned the quasi–static balance required to avoid falling forward or backward while shifting body weight from side–to–side at different speeds, and has learned the dynamic balance required in order to lift a foot off of the floor for a desired length of time, during which the foot can be moved to a new location relative to the body. Using these skills, the biped is able to march in place and take short steps without falling (too often). Interestingly, in the robot, feedback was provided by ten joint position sensors (providing artificial *proprioception*) and four force sensors in the sole of each foot, which resemble some of our own biofeedback signals. The author states that the goal of the research reported in this paper was to develop low level on–line learning control strategies which would enable the biped to balance during changes in standing posture, and to link short steps without falling. This seems like an interesting, realistic and integral approach to artificial biped locomotion.

The author proposes an integral biped control system, which consists of four units: a Fixed Gait Generator, which produces a periodic 4–phase gait for each foot, using heuristics and a simple planar model of the leg kinematics (supervised by a keyboard command input level), and three CMAC neural networks ("basically an associative memory technique") to provide right–left and front–back balance plus *closed–chain kinematics* (i.e. it learned kinematically consistent robot postures : it predicted appropriate positions for the frontal and lateral ankle rotations in order to keep the biped feet parallel to the floor with spatially balanced force on the sole of each foot).

Ultimately, the system maps measured (actual) foot forces and joint positions/ angles onto a pattern of desired joint positions. Actuation of the robot to achieve this posture (or: body configuration) is always well–defined (compare this e.g. with the natural (human) situation with FES,

where effects of stimulation are often assumed to be well–defined and predictable, but likely to get more doubtful as stimulation of a certain muscle (group) continues). Furthermore, the model totally ignored the coupling between the leg segment orientations in the frontal and lateral planes, and the coupling between the two legs via their common connection to the body. We could say that the system solves the inverse kinematics problem without requiring a detailed mathematical model.

Of course, in order to achieve actual propulsion, a (preprogrammed) gait pattern generator is included. Although fixed for one particular movement (instead of offering a whole range of desired movements), it seems relevant to our research: in earlier work on on–line gait modification the author combined standard supervised learning and temporal difference learning for achieving sudden body translational accelerations and decelerations, and for recovering from unexpected disturbance forces, so a considerably realistic strategy is proposed.

After 90 minutes of training (compare the 50 hours of training reported in [6], a similar kind of work) a reliably walking robot is observed, despite the fact that "the biped was fundamentally unstable and thus unsuitable for control using purely reactive, error driven control". Here, the obtained performance was achieved with "accurate and smooth feedforward control, which was provided by the neural network in lieu of detailed system modeling". The CMACs are trained using supervised learning and *temporal difference learning*, resulting in a controller that learned the correct time sequences of body postures.

Some important facts that apply to artificial human walking can be learned from this research: control of biped walking has to be done in a closed–loop fashion. This, however, does not imply on–line gait pattern generation (fundamental unstability during walking prevents error–driven control), but had better be done using on–line modification of preprogrammed gait patterns. Another feature of walking is, that it comprises temporal sequences of body postures, so controlling walking with neural networks implies that temporal patterns have to be represented and learned by the network.

Lack of a mathematical model within the system seems to enlarge the applicability of the work to FES–control; ignorance of dynamic couplings and disregarding recruitment characteristics in performing joint actuation, for instance, create yet a significant gap between natural biped walking and the above research.

2.3.2.5 Neural open–loop control of cyclic movements.

The work in [1] was supported by the US government (more specifically: by the Rehabilitation Research & Development Service of the Department of Veterans Affairs). The researchers are with the famous Cleveland Veterans Affairs Medical Center and Case Western Reserve University in Cleveland, Ohio, where much important work on FES is being done. They propose an adaptive neural network control system for the purpose of controlling cyclic movements of nonlinear dynamic systems with input time delays (like artifical human walking). It is designed to address two major problems with FNS (Functional Neuromuscular Stimulation, equivalent to FES [23]) systems:

- the stimulation patterns must be customized for each individual

Currently, a time–consuming trial–and–error procedure is used to specify stimulation patterns for each muscle of each individual. The control system addresses the first problem by providing **automated customization** of the stimulation patterns.

- musculoskeletal system properties may change while the system is functioning

Currently, stimulation parameters are fixed. Thus, performance will degrade as system properties change, e.g. due to muscle fatigue. This control system addresses the second problem by providing **on–line adaptation** of the stimulation patterns.

The authors remark that ANNs are limited in their ability to generate temporal patterns of activity and explain that the neural network control system presented here utilizes the adaptive properties of ANN techniques and the pattern generating capabilities of more detailed models. Their system consists of a **pattern generator** (PG), which uses a set of neurons in a physiologically–motivated architecture and consists of a set of coupled unit pattern generators, one pair for each joint, and a **pattern shaper** (PS), an adaptive filter of the PG signals, which directly addresses the problems of *customizing the control system* and *adapting the control system*.

The system has been evaluated in computer simulation on a musculoskeletal model which consisted of two muscles acting on a swinging pendulum. Furthermore, "The model of electrically stimulated muscle takes stimulation as its input and generates torque at its output. The model includes nonlinear recruitment (i.e. relationship between stimulation and activation), (...) second–order linear dynamics with an input time delay (...), and force–length and force–velocity multiplicative factors (...)." In other words, stimulation of muscles is realistically modeled within the simulation. Then, a "one–segment planar swinging pendulum with linear stiffness and damping was used to model the skeletal segment. The single–segment was acted upon by two electrically stimulated muscles".

Automated customization was achieved with the following procedure: the PS output weights were initialized to zero, a proportional derivative (PD) feedback controller was used to supplement the action of the FF controller, and the adaptation algorithm was enabled. The PS weights were adapted, i.e. the FF controller was customized, in a simulation run consisting of 20 cycles of movement (20 sec). This was performed in a series of simulation runs on a population of 71 different musculoskeletal systems. Evaluation showed that for all of the systems, the rms tracking error for the customized FF controller was less then 5% of the peak–to–peak amplitude of the signal. The on–line adaptation procedure was evaluated in a simulation run in which muscle fatigue was simulated by a decay in muscle gain. It turned out that "as muscle gain gradually decreases to half of its original value, the adaptive FF maintains good tracking performance".

In future work, the authors will include feedback into the pattern generator and expansion to coordinate muscular activity in a multi–joint system with biarticular muscles.

We think of this approach as promising, because it deals with the problem of controlling FNS– (or FES–) induced human walking in an integral (from pattern generation to actual stimulation) an complete (incorporating real–world problems, such as muscle fatigue and gait customization) manner. Nevertheless, in our opinion there are still some limitations present: first, a PD controller acts as the initial FF controller, which necessitates the presence of a detailed biomechanical model of the human musculoskeletal system. Second, observe that the FF controller is adjusted to a particular person by feeding back the error between the actual and the desired joint angles. In this study, a desired angular movement of the segment was formulated, hence defining the desired joint angle pattern. Customization is then regarded as the modification of the initial FF controller when mediation of the desired stimulation pattern through the particular musculoskeletal system takes place. Will it be possible to find a desired joint angle pattern for human walking that is applicable to a broad range of people? And if so, can the neural controller be accustomed to a multitude of training patterns, e.g. to climb a hill, to walk slower and faster, or be trained to correct for various disturbances (keeping balance, swinging of the upper extremities, changing environment, etc.) Also, we expect that things will get far more complicated when using a more realistic multi–joint system (compare e.g. [12]), because of the redundancy that is present in the human motor system, and dynamic couplings between joints. A lot of work lies still ahead.

2.3.2.6 Neural stimulus pattern generation.

We will now take a look at a paper that addresses similar problems as the former [5]. A number of problems with FES are described: "Paralyzed arm functions resulting from upper motor neuron lesions can be restored by electrical activation of the peripheral nervous system (...) However, the control of this motor function is complicated by the highly nonlinear, length– and velocity–dependent, time varying behaviors of electrically stimulated muscles (...) In [closed–loop control] systems, sensor–derived information is essential for correction of performance errors and maintenance of stability. More complex motor tasks, such as multifingered grasping, standing, and walking, are still controlled with open–loop paradigms, owing to the limited sensor availability and the complexity in implementing closed–loop algorithms".

About open–loop control is remarked: "Although open–loop control is capable of providing tracking of a desired trajectory (...), it is often perceived as insufficient because it cannot compensate for changes in the system, such as muscle fatigue. In addition, the procedure of programming muscle stimulation patterns remains mostly a trial and error process which is not efficient in practice (...) [when] movements of different amplitudes and speeds must be controlled, the use of trial and error programming is limited since the stimulation pattern obtained for a particular movement may not be generalizable to other movements". Some potential advantages of open–loop control are also mentioned: "First, (...) The use of a neurobiomechanical model of the motor system to determine optimal muscle stimulation patterns is mathematically possible (...) These studies provide an empirical and theoretical basis for the design of an artificial motor program which can generate a generic muscle stimulus pattern for movement control (...) Second, movements elicited by this artificial motor program capture the features exhibited in voluntary movements in humans, such as normal looking appearance and minimized muscle stimulation (...) Thirdly, (...) artificial neural networks can be used to learn and store the optimal muscle stimulus patterns and to perform on–line control for the desired movements. Finally, a more effective strategy for movement control may be the combination of the open–loop pattern generator with a closed–loop controller (...), or an on–line adaptive algorithm which corrects for time varying properties such as muscle fatigue"

The goal of the research is to explore the use of an ANN as an open–loop muscle stimulation pattern generator for control of discrete limb movements; the authors think of neural networks as promising as pattern generators because of their enormous capacity to learn and to memorize complex nonlinear i/o–information (compare this to the statement made w.r.t neural pattern generation in the previous work). With respect to the network structure they note that "in movement control, however, a spatiotemporal control signal must be computed" and they announce the investigation of a standard feedforward, a recurrent, and a feedforward network with recurrent feedback and time delays in the inputs as a "stimulus pattern generator for a pair of flexor and extensor muscles to produce a range of point–to–point movements". All three networks are trained with the error backpropagation algorithm (generalized delta rule). Target data were obtained from an optimal control model for single–joint point–to–point movements.

The goal of movement control was to minimize an *effort functional of Hasan*, a cost function consisting of the product of the time derivative of the equilibrium trajectory and the joint stiffness integrated over a certain time period T. The obtained optimal movements exhibit the important features also seen in voluntary movements made by normal human subjects, i.e.

- they are normal looking, having a bell–shaped velocity profile
- muscle activations show a triphasic burst pattern
- there is little or no cocontraction in the antagonist muscles
- their velocity profiles are scaled with each other in time and size

where the last feature is a unique characteristic of optimal movements. Simulations of open–loop controlled joint movements were done with a three–factor nonlinear muscle model, employing muscle activation dynamics, linearized torque–angle characteristic and linearized torque–velocity relation.

Results showed that recurrent feedback significantly speeds up the learning process, that the pure feedforward networks were not efficient for learning spatiotemporal signals and that time delays appeared to enhance mainly the neural network's ability to generalize the learned control patterns to new movements. This generalization was, however, limited only to the movements that had scaled velocity profiles. In the study it is also suggested that "the extent to which the ANN controller can extrapolate the learned control within the class of movements is speed–limited, since the trained ANN does not seem to predict muscle stimulation well for a considerably slower movement" and "the ANN–produced movement may also deviate from the optimal ones, particularly at the termination of the movement".

In the Discussion it is remarked that neural networks are often used in system modeling and adaptive control, in which neural networks learn either the inverse model or the forward model of the system. The authors address the issue of *how a neural network can be taught to perform a specific motor task*, a practical issue in FES control, by stating that direct copying of an existing controller is "useful for FES applications because we wish to copy the strategies employed by the central nervous system for motor control". Furthermore, the neural network can be trained off–line and more effectively adapted on–line during real–time control: "For training the neural network, information obtained during motor performance by normal human subjects may be employed (...) a better alternative is to create an artificial motor program that can replicate the features of voluntary movements made by normal human subjects. This idea has led to an investigation of a minimal effort control strategy for the formation of movement trajectories and muscle controls".

The paper is concluded with the proposal of two countermeasures that can be taken to improve the performance of the open–loop ANN controller. First, the authors think the ANN may be combined with an on–line adaptation algorithm, which performs fine adjustment of the controller's internal weights to compensate for the difference between training and actual movements. Second, a feedback controller can be added to the ANN so that the performance errors are corrected through the feedback controller that directly modulates the muscle stimulation. Finally, it is stated that in either case "the open–loop neural network controller provides the basic pattern of muscle stimulation, while the on–line adaptation algorithm or the feedback controller assists in refining muscle stimulation so as to improve the overall performance". This approach to movement control may be applicable to more complex tasks that are restored by FES. In the authors' opinion, activities such as human standing and walking should be thus controlled in an open–loop manner using on–line modifications based on biofeedback,

2.3.2.7 Predicting muscle activation from surface EMG.

The last piece of work we include in this review deals with the relation between muscle activity and EMG patterns. These may seem equivalent at first sight, but in fact they are not. Referring to the "cause–and–effect sequence of events that takes place for human walking to occur" [11], it is clear that generation of an EMG signal is a side–effect of a contracting muscle. The EMG consists of a summation of separate *motor unit action potentials,* each indicating a particular motor unit activity. With surface EMG, the ensemble myographic activity in the vicinity of the skin electrodes are registered, hence a nonlinear relationship between separate muscle activities and the registered EMG will be present. More about the EMG signal will be said in the next chapter of this thesis.

When the complex relationship between EMG, the only noninvasive indicator of muscle activation intensity, and muscle force can be determined, it will provide the community with accurate

noninvasive access to (estimations of) human muscle forces, providing engineers with valuable design parameters to be used in (EMG–driven) prosthetic limb and joint design. This relationship is investigated by braking it down into three parts:

- 1. observable EMG and muscle activations
- 2. individual muscle activations and forces
- 3. individual muscle forces and net joint torque

The second and third segments are modeled biomechanically (including one DOF, flexion/ extension of the biceps, brachialis, brachioradialis and triceps muscles), while the first relationship is modeled with neural networks.

The system was provided with distinct images of muscle activation patterns from a ballistic movement (a single maximally rapid flexion with an abrupt stop). The surface EMG data were sampled at 1000 Hz, zero–mean averaged, full–wave rectified and low–pass filtered. Each NN training set consisted of processed EMG, net joint torque, and angular position and velocity. Feedforward networks using the backpropagation algorithm and batch error updating were utilized in the mapping procedure.

The hybrid (neural–musculoskeletal model) system was trained in several configurations and tested over a variety of situations. During open–loop learning, accurate torque profiles were obtained, but the activation profiles of the included muscles were not physiologically feasible (the flexors did not act synchronously), although the solution was valid. The closed–loop trial (where measured kinematic data, i.e. angular position and velocity, were provided only initially, and predicted kinematics used for this purpose later on) showed signifcant destabilizing tracking errors for joint positions; torque error decreased significantly. This made sense, since the network used joint torque as the sole teacher and basis for the error function, so overall torque error was minimized without regard to joint kinematics.

In order to force the network to a physiologically realistic solution, an arbitrary loss function was added to the existing torque–error loss function:

$$J = \frac{1}{2}\left[\left(\Gamma - \Gamma_d\right)^2 + \phi(a)\right] \tag{2}$$

where

- Γ is the realized system torque
- Γ_d is the desired system torque

and the activation loss function is represented by

$$\phi(a) = k\left[\left(\phi_1 a_{bic}\right)^2 + \left(\phi_2 a_{bra}\right)^2 + \left(\phi_3 a_{brd}\right)^2 + \left(\phi_4 a_{tri}\right)^2\right] \tag{3}$$

with

- k a variable learning constant
- $\phi_i,\ i = 1, 2, 3, 4,$ weighting functions

This particular activation loss function was chosen over other functions (minimum jerk, minimum effort, minimum torque–change) because **muscle activation is independent of other muscle parameters, such as force and length**, and easily accessible. The underlying reason for the utilization of an activation loss function is that *balanced muscle activations* (or evenly distributed force) result when overall muscle activation is reduced, while also maintaining accurate

torque tracking (the human body tends to perform activities in a metabolically efficient manner). Results using this (extended) loss function showed a more balanced distribution, while all muscles were brought to full activation simultaneously and synergistically, which is physiologically plausible.

Generalization of a trained network to unseen flexion EMG data resulted in erroneous timing of the primary activation peak and temporal displacement of the torque waveforms, so considerable tracking errors were obsereved. Significantly better results were obtained when generalization capabilities of multiple–trial trained networks were investigated; moreover, a classic triphasic (agonist–antagonist–agonist) activation pattern was observed, hence the NN was detecting underlying features common to typical ballistic EMG signals.

This work demonstrates once again the capacities of neural networks in performing difficult redundant mappings within parts of our problem. These mappings can be seen to follow certain physiological principles, while also good generalization can be provided.

Chapter 3.
Data analysis and measurement

We already stressed the vital role of body coordinates in describing the walking cycle. This description seems necessary when keeping track of the walking movement in case of an artificial walking system. These data, however, would not suffice for such an application: we would also like to gain insight in the internal and external forces acting on the stimulated subject, e.g. to ensure balance and to be able to foresee the effect of a certain stimulation on a certain moment or the pattern of future muscle activities and forces. These forces can partly be measured, and partly derived from the measured forces and the body configuration [39]. Furthermore, the human motor control system is highly redundant in that many muscle activities could be underlying a certain joint movement. For a complete description of the (mechanical) processes going on in the human body during walking, we need information about the particular muscle activities at a particular moment.

3.1 Gait analysis.

In interpreting the signals mentioned above we are actually performing a kind of *gait analysis*. Vaughan et al. [39] look at gait analysis as the "search for the homunculus", a little puppeteer, which acts as a supreme commander of our locomotor program. We already stated their "sequence of gait–related processes" in the previous chapter, the process that starts as a nerve impulse in the CNS and ends with the generation of ground reaction forces [11]. This top–down approach may be useful in understanding a pathologhy: "The higher the lesion, the more profound the impact on all the components lower down in the movement chain."

The search for the homunculus can be stated in terms of the measurement of causes and effects of the walking pattern: "The direct measurement of the forces and moments transmitted by human joints, the tension in muscle groups, and the activation of the peripheral and central nervous systems is fraught with methodological problems. That is why we in gait analysis have adopted the indirect or inverse approach."

Another aim in gait analysis is to parametrize walking in such a way that normal and pathological walking can be discriminated, and the kind of pathology can be extracted.

The following parameters are used for this purpose [25]:

- time– and distance variables (e.g. steptime, stride length, cycle duration, stance phase, swing phase, double support, cadence)
- kinematic variables, i.e. (angular) displacements and velocities
- kinetic variables, i.e. forces, moments and energies

Ultimately, anthropometric (body segment parameters), kinematic and kinetic data must be integrated to derive (via the *equations of motion*) joint forces and moments. Combining these with muscle activation, muscle tensions can be extracted [39], so the little homunculus pulling the strings can be tracked (in principle, that is).

3.2 Description of available data.

A more detailed description of the measured signals seems now in order. We look for characteristics that contain useful information about (current and future) state of the walking movement or the signals themselves and their interrelationships. At a later stage in the research, these could be used as *features* for a neural network.

3.2.1 Computer DynoGraphy (CDG): ground reaction forces.

The CDG–system can be used to measure the vertical ground reaction force. Originating from Newton's third law of motion, during walking, a resultant force from the ground acting on a subject's foot is generated, which is alternatively acting backward and forward (i.e. negative and positive in the sagittal plane). Its vertical component, i.e. the component orthogonal to the transversal plane, is the force under consideration.

It exhibits [25] a characteristic shape: two peaks occur at heel–strike (F1) and toe–off (F3), while a drop occurs at midstance (F2) (Figure 2.). Many (pathologic) features can be extracted from the shape of the vertical ground reaction force.When ground reaction forces are measured with multiple foor–sensors (like with our measurement system UltraFlex, i.e. 8 sensors per foot), it is common practice to display the aggregate force obtained from all sensors. A typical ground reaction force (from the right foot) during normal walking is shown in Figure 2..

Note that the sample frequency (which is low) causes is that the peak at heel–strike is absent in this figure.

The ground reaction force can be used in gait analysis fairly well: it is well reproducible, and there is a immediate relationship between forces and pain or stress. Also, step parameters can be very accurately timed from its shape: unequal zero, it indicates the stance phase, otherwise the swing phase of gait. There is, however, some dependence on: walking speed, age, gender, body height, weight, etc. Often, ground reaction forces are normalized to body weight [25].

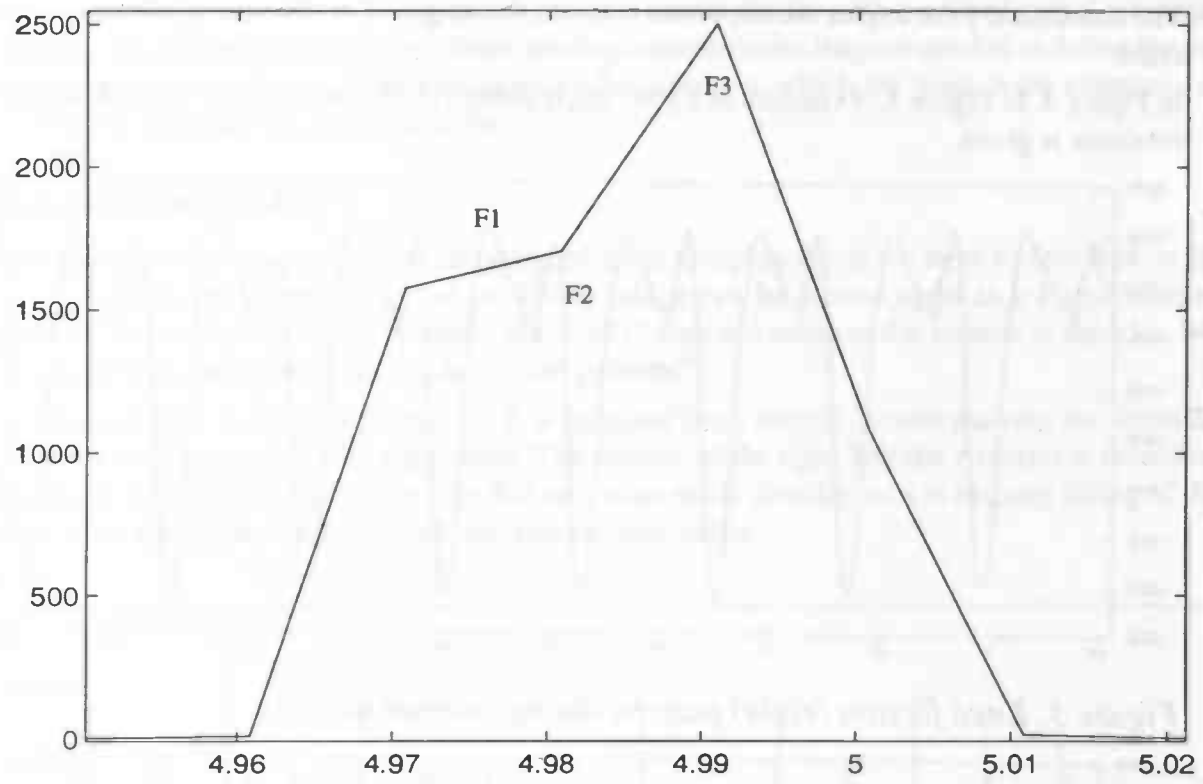

Figure 2. Ground reaction force during normal walking

Koopman [21] remarks in his thesis on the subject of kinetics: "An alternative description of the walking movement is provided with the internal and external forces and moments of force acting on the body segments. Of these forces only the ground reaction forces can be measured directly. (...) The joint moments of force can only be determined by analyzing and modeling the registered movement. (...) The joint moments of force are then computed from the measured ground reaction forces and the joint rotations."

3.2.2 Goniometry.

In human walking, joint dynamics are the causes and kinematics (displacement, velocity, acceleration) the effects. Koopman [21] states: "joint rotations are measured at each point of time of the walking cycle. (...) the kinematics of the walking movement could also be described with the displacements of the joints as time functions or with a combination of displacements and rotations. The choice for one of these possibilities mostly depends on the measuring system that is available."

Furthermore, in [39] 3–D angular orientation of the body segments is expressed in two different ways: with *anatomical joint angles* and with *segment Euler angles*. In the first approach, joint angles are defined as a rotation of the distal segment relative to the proximal segment (each joint has a reference frame in the proximal and distal segments; e.g. for the hip joint, these are the pelvis and the thigh). The rotations may be defined, in general, as

- flexion and extension (plus dorsiflexion and plantar flexion) take place about the mediolateral axis of the proximal segment
- internal and external rotation take place about the longitudinal axis of the distal segment
- abduction and adduction take place about a floating axis that is at right angles to both the flexion/ extension and internal/ external axes

The anatomical joint angles are important because the ranges of motion are of interest to clinicians.

In Figure 3. to Figure 5., examples of a joint angle pattern during walking for each of the above rotations is given.

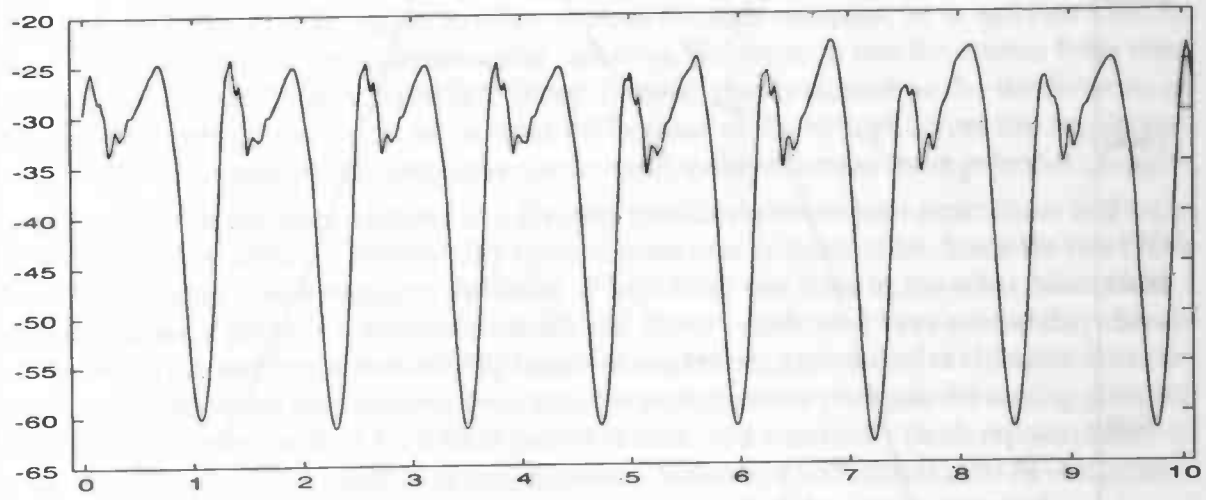

Figure 3. Knee flexion (right) pattern during normal walking

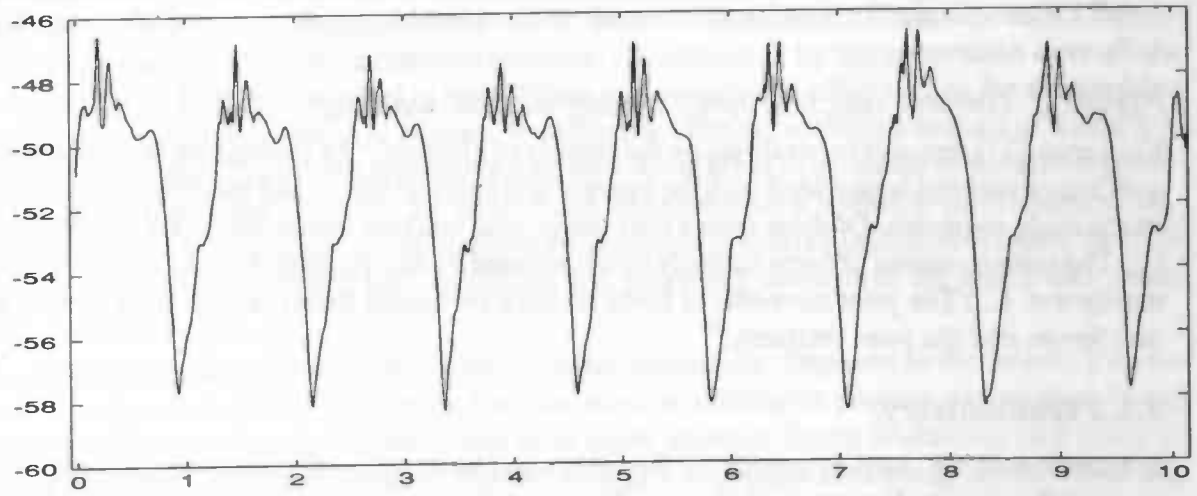

Figure 4. Knee rotation (right) pattern during normal walking

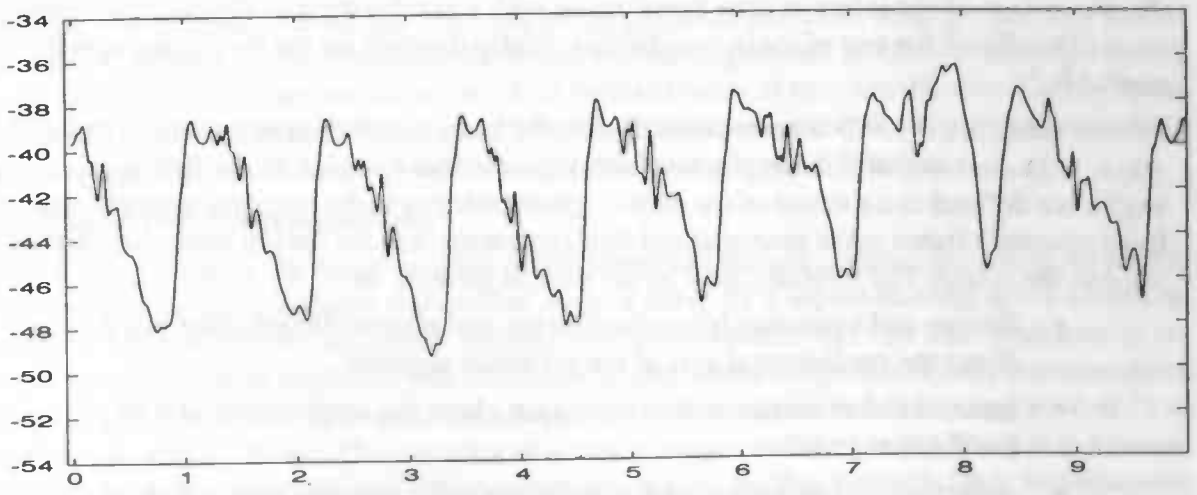

Figure 5. Hip abduction (right) pattern during normal walking

The second approach defines how one segment is oriented relative to the fixed global reference frame – the segment Euler angles. These are important because they are needed to define the angular velocities and angular accelerations of the segments, which are used in the equations of motion to calculate the joint moments.

3.2.3 Surface EMG.

Perhaps the most interesting and certainly the most complex signal we have to deal with is the human electromyographical signal, or EMG, which can be looked upon as a by–product of muscle contraction. In [39] it is stated, that EMG "does not measure the tension in muscles, but it can give us insight into muscle activation patterns".

A typical EMG is shown in Figure 6.. It is acquired from normal human walking for about 20 seconds, using the UltraFlex equipment. The activity of the right Tibialis Anterior is recorded, a muscle in the lower part of the foot that acts as an ankle dorsiflexor, i.e. causing lifting of the foot from the ground after toe–off and before heel–strike.

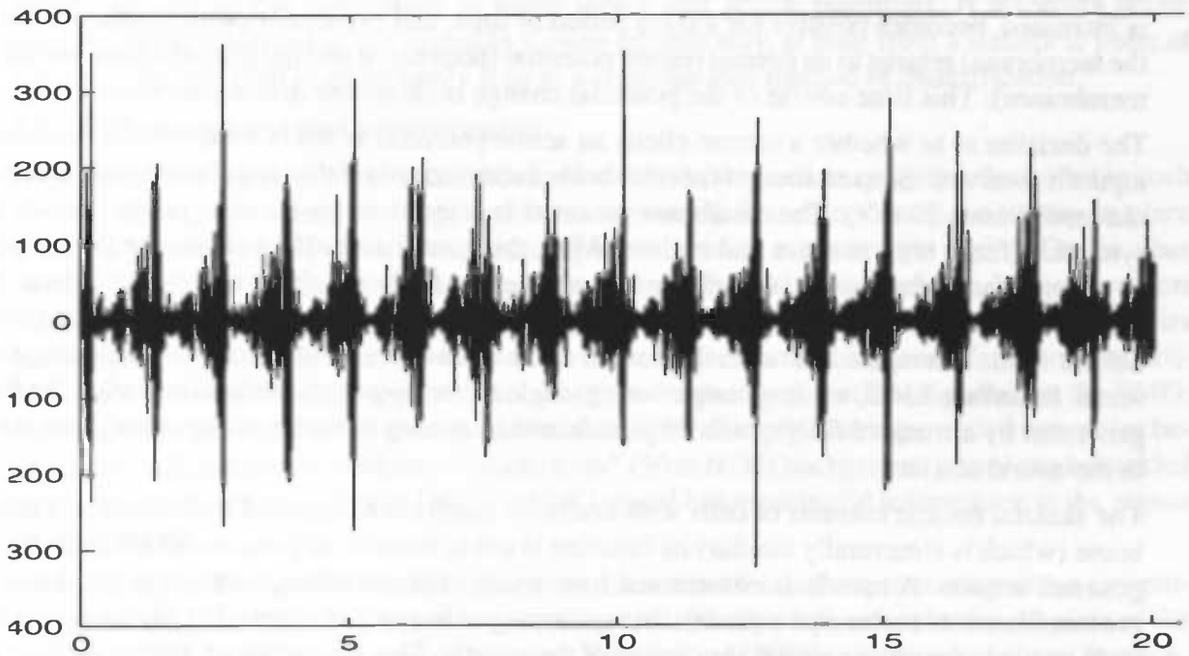

Figure 6. Raw EMG from right Tibialis Anterior

When this muscle is disturbed, a phenomenon known as dropfoot occurs, in which the foot is dragged along the ground during walking.

3.2.3.1 Physiological origin.

In order to understand the mechanism that underlies the human EMG, we repeat some basic notions from physiology [30].

Many functions of neural and muscular cells are chemical in nature. These functions produce changes in the electrical field that can be monitored by electrodes. Neural or muscular cells, however, do not function alone but in large groups, so muscle activity that is measured by skin electrodes represents in fact the electric field in the vicinity of the electrodes that propagates in the **volume conductor** (consisting of the various tissues of the body), produced by the accumulated effects of all active cells in this vicinity.

The main tasks of the basic processing unit in the neurophysiological system, the *neuron*, are information processing, transfer and acquisition. A variety of mechanisms exist to transduce

many kinds of stimuli into electrical and chemical signals, whereas the CNS deals wth the task of information processing and control. The important parts of the neuron are the cell body (*soma*), the *dendrites* and the *axon*. The cell body is surrounded by an excitable membrane, which is extended in various places to generate root–like structures (called dendrites) that are used for interconnections with other nerve cells. The axon serves as the output of the nerve unit; the tips of the axon serve as inputs to other neurons through synapses, or to activate muscles through special synapses: the neuromuscular junctions. Information into the neuron from other neurons is introduced through a junction called *synapse,* that is located on the dendrites or on the soma. The synapse can cause an increase or decrease of the voltage across the membrane; the cell function is based on the integrative (in time and space) effects of these potential changes.

The cell membrane can be considered as a dividing medium between the extracellular and intracellular fluids, with different permeability to the various ions in the solution. Since the two fluids have different ionic concentrations, diffusion of ions from one fluid to the other occurs and a resulting voltage is generated across the membrane. Some membranes have *excitability* characteristics: when the membrane is excited by means of **electrical** , mechanical or chemical stimulus the permeabilities of the membrane to ionic transfer undergo some changes: the resting potential is increased, becomes positive for a short period of time, and eventually (with repolarization of the membrane) returns to its normal resting potential (both nerve and muscle cells have excitable membranes). This time course of the potential change is called the *action potential.*

The decision as to whether a neuron elicits an action potential or not is a function of "outside" inputs it receives: the excitation of the membrane is caused only if the stimulus exceeds a threshold level (about 20 mV). The membrane potential is determined by the integration over all the synaptic effects, both in space and in time. When this cumulative effect causes the threshold to be crossed, an action potential is elicited. It propagates further without attenuation, since it is locally regenerated. The information carried by the neuron is not in the shape of the action potential but in the interspike intervals: the neuron can be considered a stimulus–to–frequency converter. In surface EMG, we are not monitoring single action potentials (spikes) but rather the field generated by a trunk of fibers; both amplitude and frequency contents of the signal then relate to the neural activity.

The skeletal muscle consists of cells with excitable membrane. Opposed to the neuron's membrane (which is structurally similar) its function is not to transfer or process information, but to generate tension. A muscle is constructed from many seperate fibers, containing two kinds of protein filaments (*actin* and myosin), that are arranged in parallel interlacing layers which can slide one into the other causing shortening of the muscle. The generation of motion or force by the muscle is activated when the fiber membrane is excited. An action potential then propagates along the surface membraneof the fiber, triggering chemical reactions that, in turn, cause fiber contraction. When a muscle contracts, the action potentials generate an electric field that can be monitored by means of surface electrodes. This field is a result of the contribution of many fibers at different times and with different rates. The EMG–signal monitored this way will thus be a random signal with statistical properties that depend on the muscle function.

Dipole model of action potentials.

Another way to clarify the meaning of the action potential (the basic building block of the EMG) is by modeling a nerve impulse transmitted down a nerve by a dipole moving down a volume conductor [39]. When the difference between two points (electrodes) on this conductor is recorded by a differential amplifier, this causes the generation of a triphasic signal, that resembles the acual action potential.The nerve axon is part of a motor nerve, and innervates several (3 to 2,000) single muscle fibers. The complex of a motor neuron and the muscle fibers that it innervates is referred to as a *motor unit.* Once an action potential reaches a muscle fiber, it propagates proximally and distally, and is called *single fiber action potential* (SFAP). A *motor unit action*

potential (MUAP) is a spatiotemporal summation of SFAPs for an entire motor unit. Of course, a muscle consists of many motor units, which contract asynchronously (in non–fatigued muscles of healthy subjects) [29]. It turns out that an EMG signal is the algebraic summation of many repetitive sequences of MUAPs from all active motor units in the vicinity of the recording electrodes.

Surface EMG.

With *surface EMG*, which is practiced with the UltraFlex–system, a pair of electrodes placed over a certain muscle will almost certainly pick up the signal from nearby muscle, so attention has to be paid when a high degree of correlation between certain muscles is found to exist. It may be a result of the data capture procedure rather than a true manifestation of the underlying neuromuscular control system [39]. This "non–specificity" of surface EMG (compared to needle EMG, an invasive method for acquiring EMG) has also some advantages, according to Hermens et al.[13]:"Patterns obtained by means of needles are almost always only related to single motor units and the relative position of the leadoff area causes a large variability with respect to the registered potentials. (...) With surface EMG, (...) The signal is dominated by an interferenced pattern caused by the summation of many motor unit action potentials. If disorders occur, changes can be detected in this signal in a more reliable way, at least from a statistical point of view (...) surface EMG can certainly lead to a reproducable method of quantification."

3.2.3.2 Measurement and preprocessing.

Acquisition of the signal is not a trivial task. Not only do we need to apply the electrodes in such a way that the right muscle (group) activity is measured. The electrodes have to stay in place when the subject is moving, otherwise causing lateral motion or electrolyte drying. On the other hand, the attachment has to be loose enough to ensure a freely moving subject. Motion artifacts can be overcome also by choosing the right electrodes: electrodes with different characteristics (half–cell potential, *double layer*) show different robustness to disturbances. Furthermore, artifacts due to movement often appear as noise of low–frequency ("spurious signals") in the EMG: most gait signals repeat about once per second, i.e. 1 Hz. Simply filtering out all frequencies below 20 Hz will reduce this problem. "Mains hum" (50 or 60 Hz background noise) can be tackled the same way [39] (according to [30], the EMG signal has meaningful information in the region of 50 to 60 Hz).

When the signal has been measured, it has to be discretized for further processing with a computer system. Vaughan et al. [39] state that according to standards prepared under the auspices of the International Society of Electrophysiological Kinesiology (ISEK), the range of signal frequencies for surface EMGs is from 1 to 3, 000 Hz. With the Shannon theorem in mind, this would imply that sampling of the signal has to be done at 6,000 Hz or more. Since most of the power of the signal is in the range of 50 to 150 Hz, a sampling frequency of 500 Hz suffices.

Winter [40] states that it is difficult to interpret the amplitude and shape of the raw EMG, and it is impossible to quantify their stride–to–stride variability. Hence, to provide the assessor with a "pattern" that can be justified on some biophysical basis, the signal has to be processed. The selected smoothing technique should bear some relationship to the biomechanics of the movement and muscle or joint moment of force, in order to make intelligent correlations with the observed movement. He also notes that a processing scheme must have an appropriate *delay*: the muscle tension waveform will have a significant phase lag behind the associated EMG profile, since there exists an electromechanical delay between the MUAP and the resultant twitch waveform. ISEK has recommended some methods for processing the raw EMG signal. These include [39] :

- full–wave rectification, in which the absolute value of the signal is taken

- a linear envelope detector, which consists of full–wave rectification followed by a low–pass filter; in [40] is stated that the linear envelope of the EMG is seen to closely follow the rises and falls of muscle tension
- integration of the linear envelope ("area under the curve") over the period of interest
- a simple binary threshold detector, in which the muscle is designated to be either *off* or *on*

The processing of the raw EMG can be put further. Kleissen [14] reports that "a muscle's EMG profile may be observed by a number of gait cycles, with the beginning of each new gait cycle defined by the moment of heelstrike" and describes the following procedure: first, the signal is led through a linear envelope detector. (The resulting output signal represents the time course of the estimated intensity of the EMG signal. This "intensity" of the EMG (sometimes also called the "amount of activity") indicates the muscle's myoelectric activity. Also, the accuracy of the estimate in this first step is governed by the bandwidths of the raw EMG signal and the low–pass filter used and is limited).

In the second step, the time scale of the recorded intensity signal in each individual gait cycle is normalized to a standard cycle length. The resulting intensity signal frames, which correspond to the individual cycles, are ensemble–averaged to form the final EMG signal intensity profile. This ensemble–averaging process improves the accuracy of the intensity information, because random fluctuations tend to cancel each other. Furthermore, it is stated that the particular choice of the low–pass filter in a linear envelope detector influences parameters like cycle–to–cycle variability, time–lag w.r.t. the raw EMG and inter–subject variability. Caution must be payed when comparing EMG profiles recorded with different detectors.

Kleissen et al. [15] give some relevant information about the recording of EMG: "The intensity of the recorded EMG reflects the level of muscle activation. This intensity varies slowly compared with the fluctuactions which are found in the EMG, so that a signal representing the EMG intensity has a considerable lower bandwidth than the raw EMG itself. Recording this intensity signal (...) implies substantial data reduction. (...) this signal is derived (...) by estimating its standard deviation over a time interval short enough to be able to observe changes in the EMG activity related to movements, but at the same time as long as possible to obtain the best possible estimate over that interval; clearly a compromise." The way this parameter is estimated, is reported below.

3.2.3.3 About the EMG.

Finally, we end up with a digital signal, a time series which is subject to the vast amount of theory from signal analysis, modeling and estimation. We want to characterize the signal on the basis of this theory and extract a number of useful parameters from it.

We already mentioned the deterministic nature of EMG: conceptually, it consists of an algebraic summation of individual MUAPs, each caused by a propagated motor nerve action potential (ultimately originating from a motor command or even: motor program). Indeed, after suitable preprocessing (smoothing, coherent averaging, etc.), individual MUAPs can be observed in the EMG [29].

Because the data originates from a cyclical movement, the measured signals will exhibit a certain periodicity. For the force and gonio–signals this will be evident; however, the EMG shows a more complex behaviour: in a qualitative way (muscle activity on–off), it can be observed to be repetitive, when looking more closely (quantitatively), a random character seems to dictate the behavior of the signal; the EMG itself can be classified as a random signal, despite its underlying

determinism. There are many factors that determine the exact form of a registered MUAP (muscle fiber characteristics, geometry within the motor unit, filtering by intermediate tissue and skin–electrode transition), particular electrode–configuration, characteristics of electrodes and measurement equipment) [29]; also, the muscle units together do not produce pulse–trains of constant frequency (although the individual MUAP–trains do [29]), so muscle activity doesn't show a constant pattern.This is illustrated by Hermens et al, [28] (pp. 231 – 234). According to them, different characteristics of the EMG spectrum can be attributed to deviations in the MUAP–forms or in the stochastic distributions of the MUAP–trains.

Simulations with the reported *stochastic model for simulation of surface EMG patterns* indicated that "the lower–frequency part of the power spectrum corresponds mostly with the MUAP firing characteristics, while the higher–frequency part is mainly determined by the MUAP–forms". Hermens [20] remarked on this subject: "the power spectrum of the surface EMG signal consists of three components: the auto and cross power spectra of the firing processes and the power spectra of the MUAP's. (...) Simulation clearly predicts that the median frequency is especially sensitive to varations in MUAP shape, the MUAP peak–peak time and synchronization. The influence of the firing process parameters was predicted to be marginal" Also, a study is described in which surface EMG recordings are used to evaluate the effect of a specific treatment. This is done using the *RMS value* of the EMG.

3.2.3.4 Characteristics.

Hermens et al. [13] provide a number of parameters that can be extracted from the surface EMG (which they want to use as a "tool for quantification"). First, it is remarked that the total power of the signal is given by the RMS value of the signal. Instead of this parameter, that is indicative of muscle activation, the integrated EMG is often used.

Then, they state that the amplitude distribution function (estimated by the histogram of measured samples) is, generally, a Gaussian with zero mean. An explanation for this statement can be found in [30]: "Since the sum of independent random variables tends to be Gaussian distributed, we would expect, for example, the EEG (but also the EMG, red.) to be almost Gaussian distributed. This is because the EEG can be, in some cases, considered the sum of fields generated by the activity of many neurons (motor units, red.) operating independently of one another."

It can be shown also that there is a linear relation between the integrated EMG (here defined as the mean rectified value ot the EMG) and the standard deviation of the EMG, which in turn gives rise to a linear relation between this standard deviation and the exerted isometric (i.e. with constant muscle–length) force. Winter [40] speaks more generally about linear relationships between muscle tension and various measures of EMG amplitude (linear envelope, integrator resets per unit time, average full–wave rectified) during isometric contractions. He reports also that some studies have shown varying degrees of nonlinearity in some or all of the muscles tested; this occurs, however, only at high tension levels, well above that seen in most muscles during normal walking.

A number of parameters is presented in [13], of which we will name a few:

- V_u, mean peak–to–peak amplitude, which is dependent on the noise level
- N_t, number of peaks per second, also dependent on the noise level
- F_f, the frequency of the first peak or bending point in the spectrum (2<Ff<30 Hz)
- F_{max}, the frequency at which the maximum of the spectrum occurs
- F_{med}, the frequency that divides the spectrum into two parts of equal power

They also relate statistical and frequency parameters to physiological mechanisms: "The increase of standard deviation together with the shift of this low frequency peak [F_f, related to the mean firing frequency of the active motor units] during an increase of force, can provide information about some aspects of the recruitment mechanism. For instance, if at an increase of force the low frequency peak does not show any shift, it is reasonable to assume that the increase of force is mainly caused by an increase in the number of active motor units." About the parameter F_{med} is said, that it shifts to lower frequencies with increasing fatigue. "In order to avoid effects of fatigue the subjects have to contract for only two seconds at relative low forces." Furthermore, they state that with certain neuromuscular disorders deviations of the Gaussian distribution are found. These deviations can be described by means of the third and fourth moments, or *skewness* and *kurtosis* of the distribution. With a normal subject, the amplitude histogram will be almost Gaussian, so the skewness will be zero and the kurtosis will be three. It is noted, that the general shape of the power density function depends on the muscle investigated. The spectrum of the Tibialis Anterior, deviates e.g. significantly from the spectrum of the Biceps (centre frequency is app. 80 Hz).

An interesting remark is made w.r.t. the center frequency: it is stated that it is a good reproducable parameter and that it will decrease when synchronization of the motor unit firing occurs (as is the case with FES!).

We already mentioned the importance of the standard deviation as a parameter to describe the EMG. Kleissen et al. [15] give the following explanation:

When n(t) is a stationary, ergodic N(0,1) distributed Gaussian process and f(t) describes the intensity of the EMG signal (...), an expression for the EMG signal e(t) is found:

$$e(t) = n(t)f(t) \tag{4}$$

. It is clear that e(t) is not stationary any more. However, when f(t) is approximately constant over a short time interval, e(t) can be considered stationary and ergodic again over this interval, and normally distributed. For a normally distributed random variable e(t), it is known that

$$E\{|e(t)|\} = \sqrt{\left(\frac{2}{\pi}\right)E\{e^2(t)\}} = \sqrt{\frac{2}{\pi}}\sigma_e(t) \tag{5}$$

so that an estimate for the standard deviation can be determined as a time average of the rectified EMG signal over a short interval. This time average is approximated by a convolution of the rectified EMG signal with a suitable low–pass impulse response (exhibiting linear phase characteristics to avoid waveform distortion, strong high–frequency roll–off, and prevention of aliasing).

We conclude this discussion of EMG characteristics with the remark [30] that muscle fatigue can be characterized and, to some extent, predicted by processing the EMG spectrum (or: its power spectral density).

3.3 Relating the data.

Once we have obtained the data in a MatLab–ready form, another difficult part emerges: bestowing the data with a meaning. The EMG from a particular muscle may already exhibit considerable complexity, understanding the simultaneous activity from all muscles involved in gait yet is another difficult task. In effect, we now have to relate muscle activities to their antagonists and synergists activities, and also to the movement they are indicative of. This is again a matter of gait analysis, so we take anther look at [39] and [15].

3.3.1 Relating movement and EMG.

The question that arises may be stated as: "Are there some underlying trends that in combination can explain all the variations in a set of EMG traces?" This question is important, because if it's found that there are fewer factors than there are muscles, the neural system may not need one program for each EMG signal. Thus, by decomposing the original data, we can ascertain how many independent patterns are needed to reconstruct all of the EMG patterns. In fact, we are aiming at discovering the control processes involved in regulating muscular activity during gait.

A way to relate muscle activities to each other is by means of statistical techniques like *factor analysis*. Applying this, it turns out, that EMG patterns of 16 leg muscles can be described by 4 factors, nl. a heel strike factor, a propulsion factor, the loading response factor and the biphasic factor. Muscles that are representative of each of the four factors are: Tibialis Anterior, Lateral Gastrocnemius, Gluteus Medius and Erector Spinae, respectively.

A simple way to study the relationships between different muscles, is by plotting the activity levels of one muscle as a function of the other. Antagonists, like the Triceps Surea and the Tibialis Anterior, will exhibit an L–shape (when one is active, the other is quiescent and vice versa), while synergists, like the Rectus Femoris and the Gluteus Medius, will resemble a straight line passing through the origin and the top right–hand corner of the graph. When there is a loop in the graph, it can be deduced that there is some phase lag between the two individual EMG traces.

Kleissen et al. [15] have the opinion that kinesiological analysis has to involve both movement of upper and lower extremities and the related muscle activity: "EMG activity is a measure of muscle activity, but it has hardly any significance under dynamic conditions unless the simultaneous movement is recorded as well. Functional effect of muscle activity strongly depends on joint position, so that qualitative interpretation of the EMG patterns without reference to movement patterns is disputable." The way the movement and EMG data are related is by sectioning the time series into segments belonging to a walking cycle (indicated by heel–on = 0/100 % and toe–off = 55 % –detectors), normalizing to a standard cycle length and averaging the segments (16, in this case) in order to obtain normal patterns.

3.3.2 Relating angles and forces.

A statistical study of movement data (joint angles and ground reaction forces) is reported by Loslever et al. [8] First, the six signals (hip, knee and angle excursions in the sagittal plane; the three components of the ground reaction force) of each step are described through sliding averages computed for successive time windows. The internal range within a window is cut into fuzzy modalities. Then the six step–by–window tables are analyzed separately through *principal component analysis* to reduce the excessive quantity of data. The most informative windows are then analyzed simultaneously. The resulting table is investigated using *multiple correspondence analysis*. The outputs are gait patterns combining both time and space aspects.

It is remarked that gait patterns vary more or less among normal subjects and even among patients with a similar diagnosis. The main difficulties encountered in performing gait analysis are data characterization and data comparison. In the data characterization stage, a Fourier analysis is performed. From the literature it is mentioned that the number of essential harmonics to reconstruct the ground reaction force runs from 3 to 20 for the vertical component. It is reported also that seven harmonics are considered to be sufficient to describe the joint angles.

When comparing individuals, the ground reaction forces need to be normalized to the body mass. The hypothesis that data within a time window can be described with mean and standard deviation (thus be Gaussian distributed) is verified for 25 windows of each of the six curves, and confirmed. The influence of such a normal model is that the gait variations over the population are

due to the influence of numerous parameters, varying independently and with additional effects. Winter [40] found that the variability decreased from proximal to distal. He concluded that the lower variability at the ankle indicates that the motor tasks of the ankle muscles are more fixed over the stride than with the two other joints. The independence of the ankle excursion and the two upper joints is confirmed by the research here reported.

The following outcomes of the factor analysis are reported:

- the MCA showed that the most discriminant windows and signals are those of the hip and knee joint angles (the hip joint in the gait mechanical system has a major role in yielding large differences in motor patterns, which have to control the mass of the remaining body while allowing global head movement)
- for the ankle joint different curve patterns can also be drawn
- two opposed behaviours are drawn at the end of the propulsion phase, which represent two different motor configurations

It is concluded that the patterns show that the walking mechanism can be reached using very different motor actions. Also, the authors state that the hypothesis of the research ("gait patterns do exist", despite inter– and intra–individual variations) has been confirmed.

3.4 The UltraFlex system.

The measurement system we used in our research, the UltraFlex system for gait analysis from Infotronic Medical Engineering, is capable of acquiring all the specified signals. It can be used to measure:

- (vertical) ground reaction forces (*computer dynography*, or *CDG*)
- joint angles (goniometry, or *Gonio*)
- surface EMG (*EMG*)

3.4.1 UltraFlex system.

The *UltraFlex portable datalogger* consists of a microprocessor unit (68020, opt. 68882 co–processor, 16/ 50 MHz), optical link of 10 Mb/s (for real–time applications), SCSI/ ISA–interface, removable memorycard (up to 64 MB), expansion unit (16 channel differential inputs with programmable gain and filters, to be used with measurement devices), power supply and optional additional expansions. It is used in conjunction with several measurement devices, such as CDG, Gonio and EMG. Measurements are done without immediate connection to a host (personal) computer; a total of 4 measurements of 20 sec can be stored in the measuring unit's memory before connecting the unit to the base station for downloading the data [26][39].

3.4.2 CDG–system.

The CDG [39] consists of the following components: a pair of pressure insoles, each having eight capacitive transducers; a battery–powered measuring unit which is worn either around the subject's neck or waist and records 16 channels at ca. 100 Hz; and an analysis program with pull–down menus and windows. The CDG transducers have a range of 0.004 to 1.11 MPa, and the measuring unit has a resolution of 0.004 MPa (i.e. 8 bits, or 256 levels).

The analysis software includes presentation of: force graphics (forces/ sensor vs. time, summed forces for one foot vs. time, mean and standard deviation per foot for multiple steps, etc.), gaitline (point of application of the resultant force per measurement–cycle per foot), histogram (on–line

simulation of force distribution over all sensors), cyclogram (displacement of the point of application, resultant vertical ground reaction force for both feet, projected onto a virtual horizontal plane), steptime and symmetry parameters. The cumulated forces per foot indicate the total (vertical) ground reaction force per foot. Strictly speaking, this doesn't equal the force of the body on the ground (like Newton's third law states), since the surface occupied by the sensors doesn't equal the surface of the foot causing the body transfer [25].

3.4.3 Measurement of joint angles.

The Gonio system [26] is used for dynamic measurement of several lower extremity joint angles (max. 16 channels) during walking. Two axis goniometers enable 2D angle measurement, so combined joint motions are allowed. Data is sampled at ca. 100 Hz. Accompanying software displays angle/ time–, and angle/ angle– (e.g. detect the goniometric ratio) plots.

3.4.4 Surface EMG.

Finally, the EMG measurement system [26] measures raw surface EMG from max. 16 different muscles offering noise reduction and accurate 12–bits data. The EMG is sampled at 1000 Hz; anaysis software features display of EMG amplitude, envelope, histogram and power spectrum.

Chapter 4.
Problem identification

Now that we have gained some insight into the data and its meaning, we are ready to postulate the actual subject of our research. This implies that we choose a part of the total problem ("designing a neurocontroller for an artificial walking system") for further investigation.

4.1 Research focus.

In choosing a suitable sub–problem that can be tackled with a neural network, we have to keep in mind that research into *muscle stimulation and activation* can only be done with the necessary "material" (FES–equipment, biofeedback–unit), people (knowledge about stimulation patterns, clinicians, patients) or simulation models present. Presently, these resources are not generally available at our institute. Therefore, we will focus on the *inverse dynamics* problem, trying to relate movement to its underlying muscle activation pattern. With the measurement–system available, this means that we are looking for

"a mapping between joint angles and ground reaction forces on the one hand and EMG patterns on the other"

Since walking is a cyclic activity, we expect the body signals concomitant with walking to be also repetitive. Therefore, an attempt will be made to relate the EMG of a muscle to its own past. In fact, we will be performing *time series prediction* of a physiological signal, while additional information can be obtained from characteristic features of the signal and related movement data. A neural network seems to be the right tool in performing this prediction, since it is flexible, extendable and it can approximate any nonlinear mapping.

4.1.1 System identification.

The problem can also be stated another way: it is concerned with the *identification* of the virtual system underlying the inverse dynamics–behaviour, modeled as a nonlinear plant (Figure 9.). According to Narendra & Parthasarathy[18], system identification and characterization are fundamental problems in systems theory. In system *characterization*, i.e. finding a mathematical representation of a system, a model of a system is expressed as an operator P from an input space $\mathfrak{U}$ into an output space $\mathfrak{Y}$, and the objective is to characterize the class $\mathcal{P}$ to which P belongs. In system *identification* the problem is to determine a class $\hat{\mathcal{P}} \subset \mathcal{P}$ and an element $\hat{P} \subset \hat{\mathcal{P}}$

so that $\hat{P}$ approximates P in some desired sense, given a class $\mathcal{P}$ and an operator $P \subset \mathcal{P}$. With static systems, the spaces $\mathcal{U}$ and $\mathcal{Y}$ are are subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. A typical example of identification of static systems is *pattern recognition*, where compact sets $U_i \subset \mathbb{R}^n$ are mapped into elements ($y_i \in \mathbb{R}^m$, i = 1, 2, ...) in the output space by a decision function P. With dynamical systems, as in our case, spaces $\mathcal{U}$ and $\mathcal{Y}$ are generally assumed to be bounded Lebesgue integrable functions on the interval $[0, T]$ or $[0, \infty]$. In both cases, *the operator P is defined implicitly by the specified i/o–pairs*. The particular choice of the class of identification models $\hat{\mathcal{P}}$ and the specific method used to determine $\hat{P}$ depends on a variety of factors; in practice, many decisions naturally depend on the *prior information* that is available concerning the plant that is to be identified.

The objective of identification is to construct a suitable identification model, which, when subjected to the same input $u(k)$ as the plant, produces an output $\hat{y}_p(k)$ that approximates $y_p(k)$ in the sense

$$\| \hat{y} - y \| = \| \hat{P}(u) - P(u) \| \leq \epsilon, \tag{6}$$

where

- $u \in \mathcal{U}$
- ϵ a specified positive constant
- $\| \cdot \|$ a suitably defined norm on the output space $\mathcal{Y}$

This is shown graphically in Figure 7.:

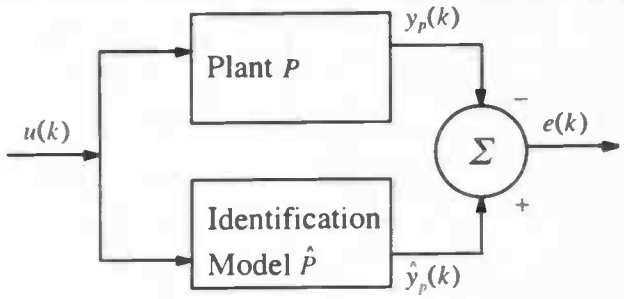

Figure 7. System identification

4.1.2 Linear and nonlinear systems.

The behaviour of linear time–invariant discrete–time systems can be expressed by

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) \quad\quad = Cx(k) \end{cases} \tag{7}$$

where A, B and C are matrices with dimensions that match the dimensions of u, x and y, the input, state and output vector, respectively.

The theory of linear time–invariant systems, when the above matrices are known, is very well developed and concepts such as controllability, stability, and observability of these systems have been studied extensively. For the more general case, when we consider nonlinear systems, much less methodology is present. In spite of many attempts, constructive procedures do not exist. The behaviour of a discrete–time nonlinear system can be described by

$$\begin{cases} x(k+1) = \Phi[x(k), u(k)] \\ y(k) \quad\quad = \Psi[x(k)] \end{cases} \tag{8}$$

where the u, x and y vectors are discrete time sequences. Since the problems involving nonlinear equations of the form (8), where the functions Φ and Ψ are known, result in nonlinear algebraic equations (for the solution of which powerful methods do not exist), the choice of identification and controller models for nonlinear plants is a formidable challenge. Consideral progress in nonlinear control theory will be needed to obtain rigorous solutions to the identification and control problems.

The choice of structures for identifiers and controllers in the nonlinear case is motivated by the following linear model–reasoning: when, for instance, a SISO–system is represented by (8), it will be assumed that the state of the system can be reconstructed from n measurements of input and output. In other words,

$$\begin{cases} y_p(k) = \Psi[x(k)] \\ y_p(k+1) = \Psi[\Phi[x(k), u(k)] \\ \cdots \\ y_p(k+n-1) = \Psi\big[\Phi[\cdots \Phi[\Phi[x(k), u(k)], u(k+1)], \cdots, u(k+n-2)]\big] \end{cases} \tag{9}$$

yield n nonlinear equations in n unknowns $x(k)$, if $u(k), \cdots, u(k+n-2), y_p(k), \cdots, y_p(k+n-1)$ are specified. Furthermore, it is assumed that for any set of values of $u(k)$ in a compact region in $\mathcal{U}$ a unique solution to the above problem exists. This permits identification procedures for nonlinear systems to be proposed along lines similar to those in the linear case. In fact, it motivates the choice of structures for identifiers and controllers in the nonlinear case; we will incorporate *neural networks* in these structures.

We can express the i/o–mapping of a multilayer feedforward network (with one hidden layer) by

$$y = N[u] = \Gamma\big[W^2\Gamma\big[W^1u\big]\big] = N_2N_1[u] \tag{10}$$

where

$$N_i[u] = \Gamma\big[W^iu\big] \tag{11}$$

represents a layer of the network, and

- W^i a matrix of weights w^i_{jk}
- Γ a diagonal nonlinear operator with identical sigmoidal elements γ

Schematically, this is shown in Figure 8..

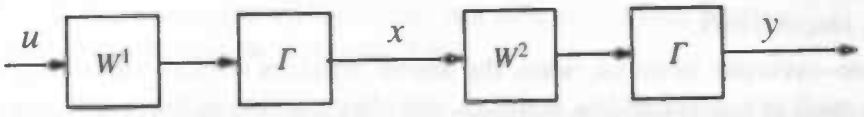

Figure 8. Block diagram of 3–layer feedforward network

Neural networks can thus be included into the above model for system identification. We could display this in the following way:

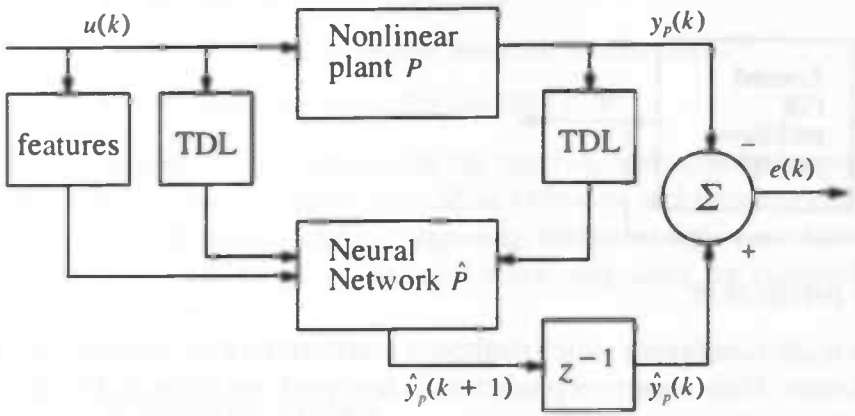

Figure 9. Identification of nonlinear plants using neural networks

Note that we also included feature data into the model of Figure 9.. By *features* we mean inputs containing information about the input signal, that can aid the network in thelearning process. In our problem, we can e.g. think of joint angles and ground reaction forces as features. The denotation **TDL** refers to a *tapped delayline*, i.e. a device that stores past samples of the input signal $u(k)$. In Figure 10., it is shown in combination with a linear neuron.

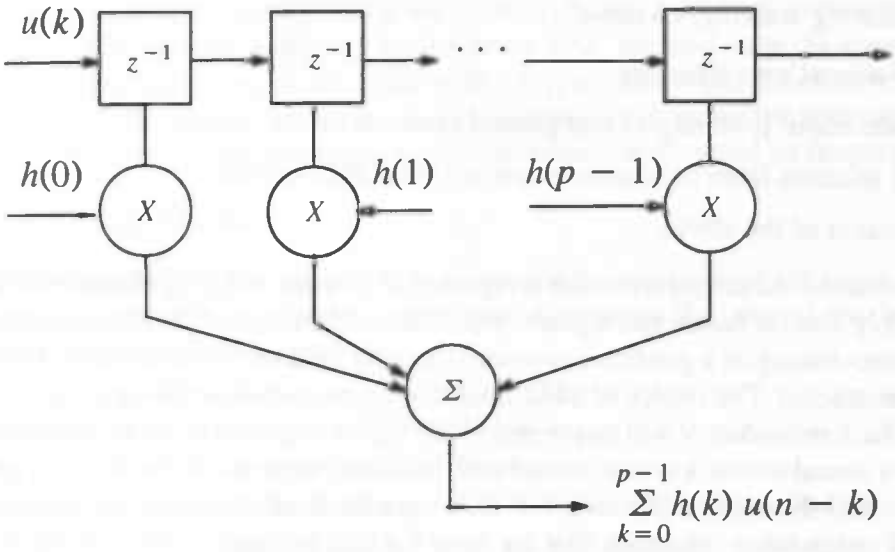

Figure 10. Tapped delayline with a linear neuron

4.1.3 Predicting further ahead.

Furthermore, note that in this model the estimate of the plant output is based on the immediate past of the realized input and output of the plant, so it represents *1–lag prediction*. In our anticipated neural predictor we will initially adopt this same strategy; for more practical (evaluation) purposes, a *recursive 1–lag predictor* [33] will ultimately be contructed, i.e.

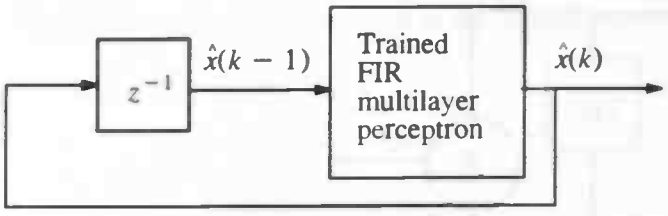

Figure 11. Recursive prediction

In [46], some remarks are made concerning which method is more preferrable, direct *n*–lag prediction or recursive prediction. Since recursive prediction is favoured, we chose to practice recursive prediction in this research.

Choice of particular EMG and associated biofeedback–signals.

We will focus on the EMG from the right Tibialis Anterior muscle, which acts as an ankle dorsiflexor at the beginning and end of the walking cycle. Movement data that are available as features for a neural predictor, include joints angles from hip and knee and ground reaction forces. In the experiments, we will look for the most suitable signals to include in the predictor.

4.2 Structured design of neural systems for temporal processing.

Restating the previously mentioned research goal, we have to predict the human EMG–signal (from the Tibialis Anterior muscle) some timesteps ahead. We would like to do this on the basis of all data available during walking, i.e using

- the EMG–signal's own history
- information about joint angles and ground reaction forces
- statistical features from (a time–window of) the EMG–signal
- a combination of the above

Utilization of additional movement information is reported to be essential for gait analysis (Section 3.3), so presumably also for future gait signals inference. Addition of additional (movement) features for further fine–tuning of a predictor can conveniently be done when a neural network based predictor is constructed. The choice of additional movement data describing the gait cycle is a nontrivial task. Much redundancy will be present in the inputs supplied to the neural network (Section 3.3). When a neural network is confronted with too many degrees of freedom, its generalization capabilities will deteriorate (Section 6.1.1), so practical utilization of the system requires a reduction of redundancy. Methods that are used for this purpose in the context of gait analysis include *Factor Analysis* and *Principal Components Analysis* (Section 3.3).

Note that the sampled EMG–signal is a *time series* (see Section 5.2). The second theme of this research could thus be formulated as:

"how do we capture the temporal properties of a stochastic, nonstationary signal such as the human EMG with a neural network?"

Only when we are able to catch these properties well, a correct predictor can be expected to emerge. For a start, we will investigate the one–lag prediction capacities of the neural network. Later on, we may verify the results by recursively predicting the whole time series. There are in fact three ways in which we can express temporal information with a neural network:

- by preprocessing the inputs
- by choosing a suitable network architecture
- by choosing a suitable learning rule

We will mainly be concerned with the first two methods; the learning rule we choose is the standard error backpropagation–rule. More elaborate architectures enable more sophisticated learning rules, such as *time–delayed learning*, *temporal difference learning* and *real–time recurrent learning*. We will focus on temporal processing using the standard multilayered Perceptron architecture instead.

4.2.1 Structured design.

A major issue in our quest for the best neural EMG–predictor is to look for it in a *structured* manner, i.e. to take decisions about network architectures, learning rule and preprocessing in a sensible way. An important decision that has to be made in neural system design concerns the dimensions of the network, i.e. the number of input, hidden and output units. Many techniques for optimization of network dimensions exist, such as *network pruning* and *growing* [33][47]. When dealing with time–delayed neural networks (TDNN's, Section 6.1.2) benefit could be taken from the fact that time is represented spatially, so networks dimensions correspond to data characteristics.

4.2.2 Data characteristics and network dimensions.

In a structured design approach we want to choose a network architecture. A possible criterion is the characteristic exhibited by the input data. We now state the assumption that, in temporal processing using neural networks with a tapped delay–line (Section 4.1.2), the number of time delays in the tapped delay–line that is necessary to capture the temporal properties of the input data depends on the input data autocorrelation function and its frequency contents.

4.2.2.1 Hypothesis.

More specifically, the maximum number of lags during which the signal is reasonably autocorrelated is expected to define an upper bound, whereas the ratio of maximum to minimum signal frequencies defines a lower:

$$\frac{f_{\max}}{f_{\min}} \leq ND \leq k_{\max} \tag{12}$$

where ND denotes the number of data samples in the delay–line and

$$k_{\max} = \min\left(k : k \in \mathrm{N} : r_{11}(k) \leq \epsilon\right) \tag{13}$$

with ϵ a suitable constant from R and the sample autocorrelation function (ACF) of the input–signal x of length N defined by [31]

$$r_{11}(j) = \frac{1}{N} \sum_{n=0}^{N-1} x_1(n)\, x_1(n+j) \tag{14}$$

When ϵ is taken zero, $k_{\max}$ is also denoted [32] as ***correlation time*** (which is dimensionless), i.e. the maximum lag of the ACF $r_{xx}(j) = E\{x(n)\, x(n+j)\}$ for which the ACF is essentially nonzero.

The lower bound can be motivated the following way: the maximum signal frequency $f_{\max}$ determines the sample frequency f_s. Hence, it also determines the number of samples in a period of

the lowest frequency f_{min}. The tapped delay–line has to contain enough samples to be able to reconstruct the longest period in the signal, hence the ratio $\frac{f_{max}}{f_{min}}$ emerges.

Chapter 5. Conventional methods for time series prediction

5.1 Temporal processing: linear and nonlinear methods.

We now turn to a central theme of our research: *how to capture temporal features of a (temporal) data vector in order to make reliable future predictions*. Many methods for this purpose rely on statistical theory, which usually amounts to estimation of features and parameters from the data using linear relations and least–squares minimization techniques only. These methods are usually computationally feasible and, for simple cases, reasonably accurate. When dealing with more complex data, e.g. chaotic data, they are not adequate any more. Nonlinear dependencies can be tackled by employing the familiar idea of linear approximation of a nonlinear (differentiable) function in a small area (*local linear prediction*) or by modeling the nonlinear temporal function directly, e.g. using interpolation or least–squares methods, or *by using neural networks*. Also, because of similarities in the chaotic and neural approach to temporal processing, we could benefit from insights gained by analyzing the data according to chaotic dynamics methods.

5.2 Some properties of time series.

We have to deal with data that can be looked upon as data records, varying with time in a certain way. We could also call them *time series* data. A few definitions seem in order [48]:

Time series (records of observations made sequentially over time) *modeling* aims at uncovering the dynamical law governing the generation of the particular time series. When the underlying theory is non–existent or far from being complete (and we are presented with not much more than the data themselves), the following paradigm may be adopted:

- recognize important features of the observed data
- construct an *empirical time series model*, incorporating as much available background as possible; an empirical time series model represents a hypothesis concerning the probability transitions over time, i.e. the dynamics of the system underlying the time series
- check that the constructed model is capable of capturing the features mentioned above and look for further improvement if necessary

Note that the above paradigm fits quite well into the paradigm of nonlinear system identification; neural networks can thus be expected to play a significant role in the second stage of the above procedure, specifying a fairly wide class of models (let's call them C) within which some optimal search technique (e.g. gradient descent, by employing the backpropagation algorithm) may be employed. The empirical model can never replace the underlying theory, but it can assist the development of the latter; at the same time, advancement in the latter can aid in the constuction of a more satisfactory empirical model. It is clear that time series modelling should not be divorced from the underlying scientific discipline that the final product (a neural or statistical empirical model) is supposed to serve.

Let X_t denote a real–valued random variable representing the observation made at time t. We assume that observations are made at regular time intervals, of one time unit duration. We formally describe a time series:

Definition 1: A *discrete–time time series* $\{X_t\}$ is a family of real–valued random variables indexed by $t \in Z$, where Z denotes the set of integers.

We will mainly look here at those statistical features with time–invariant properties:

Definition 2: The time series $\{X_t\}$ is said to be *(strictly) stationary* if, for any $t_1, t_2, \ldots, t_n \in Z$, any $k \in Z$, and $n = 1, 2, \ldots$,

$$F_{X_{t_1}, X_{t_2}, \ldots, X_{t_n}}(x_1, \ldots, x_n) = F_{X_{t_1+k}, X_{t_2+k}, \ldots, X_{t_n+k}}(x_1, \ldots, x_n) \tag{15}$$

where F denotes the distribution function of the set of random variables which appear as suffices, i.e. the time series is stationary if its statistical properties don't change in time.

Definition 3: The time series $\{X_t\}$ is said to be *weakly stationary* or *wide–sense stationary (WSS)* if, for any $t_1, t_2,\ k \in Z$,

$$E(X_{t_1}) = E(X_{t_1} + k); \quad cov(X_{t_1}, X_{t_2}) = cov(X_{t_1} + k, X_{t_2} + k) \tag{16}$$

where $E(X)$ denotes the statistical expectation X, and the covariances *cov* are assumed to exist.

Consider a stationary time series $\{X_t\}$ with finite variances. The *autocovariance function* of $\{X_t\}$ at lag $(t_2 - t_1)$ is denoted by $\gamma_{t_2 - t_1}$, and has the following properties:

(1) $\gamma_0 = var(X_t)$

(2) $|\gamma_\tau| \leq \gamma_0, \forall \tau \in Z$

(3) $\gamma_{-\tau} = \gamma_\tau, \forall \tau \in Z$

(4) $\sum_{r=1}^{n} \sum_{s=1}^{n} \gamma_{t_r - t_s} z_r z_s \geq 0,$

$\forall t_1, t_2, \ldots, t_n \in Z, \forall n \in \mathbb{N}$, and $\forall z_1, z_2, \ldots, z_n \in \mathbb{R}$

The ratio $\gamma_\tau / \gamma_0, \tau \in Z$ is called the *autocorrelation function* of $\{X_t\}$ at lag τ, and is denoted by ρ_τ. Properties (1), (2) and (3) still hold if the γ_i are replaced by the ρ_i, with corresponding subscripts i. **We may interprete ρ_τ as a measure of *linear association* between X_t and $X_{t \pm \tau}$.**

5.2.1 Linear Gaussian models.

The class of *autoregressive–moving average* models, of order k and l respectively, is denoted by $ARMA(k, l)$ and consists of models of the form

$$X_t = a_0 + \sum_{j=1}^{k} a_j X_{t-j} + \sum_{j=0}^{l} b_j \epsilon_{t-j} \tag{17}$$

where

- $a_j, b_j \in \mathbb{R}$ are nonzero constants
- ϵ_t are zero–mean uncorrelated random variables with a common variance σ_ϵ^2 ($< \infty$), called *white noise*
- b_0 may be set to *unity* without loss of generality

When the sequence of independent identically distributed random variables $\{\epsilon_t\}$ (an *i.i.d.* sequence) each have distribution $\mathcal{N}(0, \sigma_\epsilon^2)$, $\{\epsilon_t\}$ is referred to as *Gaussian white noise*.

An ARMA–model has the significant property that it consists of only a finite number of parameters. From the point of view of model building we may conclude that **if and only if** *the autocorrelations are considered an important feature, the class of ARMA–models constitutes a useful choice of the class of linear models C.*

5.2.2 Virtues and limitations of ARMA–models.

We now look at stationary Gaussian ARMA–models more closely, and state some strengths and weaknesses in modelling time series, keeping in mind our particular physiologic time series [48].

Virtues of ARMA–models.

Some significant **achievements** of ARMA–models are:

(1) mathematically, linear difference equations are the simplest type of difference equations and a complete theory is available; probabilistically, the theory of statistical inference is the most developed for linear Gaussian models. The class of stationary Gaussian ARMA–models has an elegant and fundamental geometric characterization in terms of the concepts of a *predictor space* and a *Markovian representation.*

(2) the computation time for ARMA–models is well within the reach of most practitioners

(3) linear Gaussian models have already been reasonably succesful as a practical tool in analysis, forecasting and control: *they represent the objective world to a good first approximation ...*

Drawbacks of ARMA–models.

Some **limitations** in using ARMA–models are:

(1) linear difference equations do not permit stable periodic solutions independent of initial values, as sometimes occurs in physical phenomena

(2) stationary Gaussian ARMA–models are not ideally suited for data exhibiting strong asymmetry

(3) they are also not well suited for data exhibiting sudden bursts of very large amplitude at irregular time epochs

(4) since the autocovariances $\gamma_j (j \in Z)$ are only one aspect of the joint distributions of $(X_t, X_{t-j}), (j \in Z)$, other aspects may contain vital information missed by the γ_j. For example, the regression functions at lag j, i.e. $E(X_t \mid X_{t-j}), (j \in Z)$, are all linear because of the joint normality. With *strongly cyclical data* (like our human EMG), the autocorrelation function is cyclical also. At those lags for which the autocorrelation function is quite large in modulus, the corresponding regression functions may be sufficiently well approximated by linear functions.

At those lags for which the autocorrelation function is quite small in magnitude, a linear approximation for the corresponding regression functions is not always unquestionable

In the last case, we could imagine that a strong cyclicity of the data may be linked with a strong *association* between X_t and X_{t-j}, $j = \pm 1, \pm 2, \ldots, \pm L$, for some finite integer L. This association will not necessarily be measured by the autocorrelation function, measuring only *linear association*. In this case, a nonlinear approximation for the regression functions may well be more appropriate for those lags with small autocorrelations.

5.3 Linear prediction.

In the previous section some virtues and limitations of ARMA–models for linear prediction of time series were given. We now look at this method in a more detailed manner.

5.3.1 Stationarity.

Before a time series can be predicted using conventional ARMA predictors, it has to be stationary. A simple method for checking this, is by splitting the time series into a number of subsegments, and looking at the statistical properties of each segment. When these do not change (very much), we can presume that the data is indeed stationary. Otherwise, preprocessing (e.g. trend removal) has to be performed to the data before prediction takes place.

5.3.2 The Durbin–Levinson algorithm.

A convenient way to perform linear prediction, is by applying the Durbin–Levinson algorithm for computation of the predictor coefficients. It is a recursive algorithm, in that predictor coefficients are computed iteratively, which is computationally efficient. The algorithm can be found in [42]. We applied it for the computation of *statistical features* from the time series to be predicted, i.e. the (row) entry in the matrix of partial aurocorrelation coefficients having the same number of elements as the "time window" used for prediction, was used to calculate the linear prediction, using these samples, thus "showing the way" to the neural network for EMG–prediction.

5.3.3 Predictor order selection.

Of course, the number of previous samples that is incorporated in the prediction of the next sample (at arbitrary lag), is a main decision within time series prediction. In neural prediction, it is also intertwined with network dimensions, hence a neural architecture could be based on knowledge about signal interdependencies (the main idea of this thesis). There exist two important ways to determine the order of a linear predictor: the first is based on visual inspection of the magnitude of a signal's partial autocorrelations, while the second introduces a penalty for very large predictor orders (enhancing predictor generalization). The important Akaike Information Criterion (AIC) [42] is not used here.

5.3.3.1 Preliminary criterion.

In [42], a method is described for the determination of the linear predictor order. Details can be found in this reference; in short, the partial autocorrelation function of the input data is computed, while also a boundary can be determined that has the following property: if an autogressive model is appropriate for the data, there should be a finite lag beyond which the sample partial autocorrelations $\hat{\phi}_{mm}$, are compatible with the distribution $N\left(0, \frac{1}{n}\right)$. This suggests using as a preliminary estimator of p the smallest value or r such that $|\hat{\phi}_{mm}| < 1.96n^{-\frac{1}{2}}$, $m > r$.

The criterion is shown for our prediction data in Figure 12..

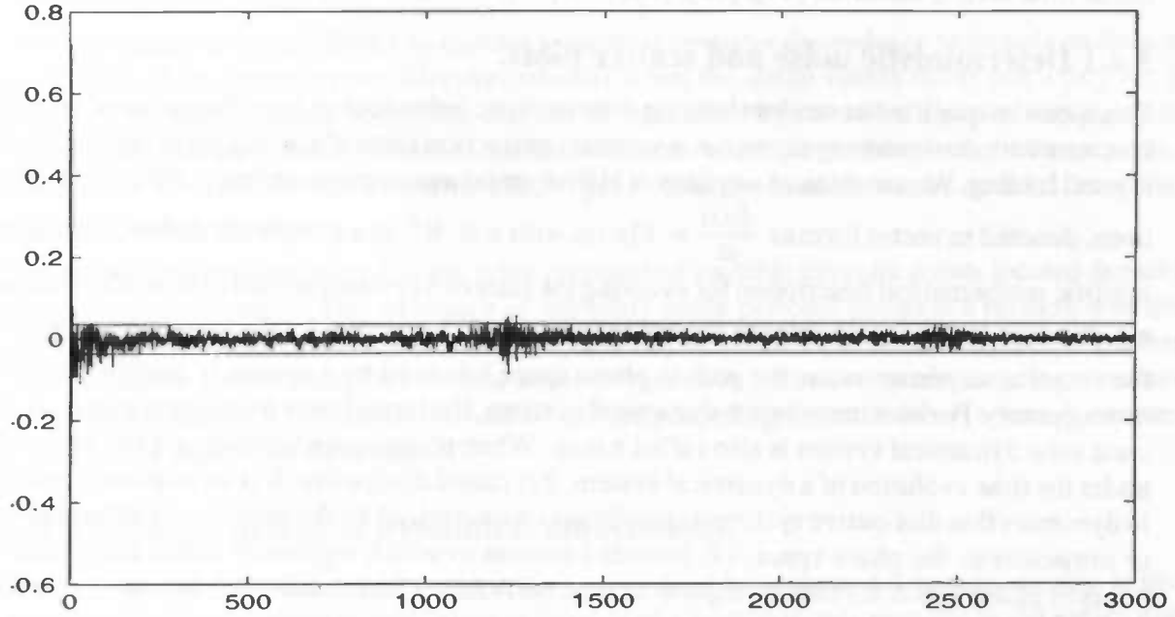

Figure 12. EMG segment partial autocorrelations

We used the preliminary criterion to determine a benchmark order to the previously stated correlation time criterion, and the next

5.3.3.2 Akaike Final Prediction Error Criterion (AC)

In choosing an optimal linear predictor, the problem of overfitting of a time series occurs, i.e. prediction errors can be decreased to small values, but at the expense of bad generalization performance, i.e. prediction errors show a large variance when previously unseen data is presented to the predictor. A cost for increasing the number of parameters in the fitted model is imposed, leading to the Akaike Final Prediction Error

$$FPE = \sigma^2 \frac{n + p}{n - p} \tag{18}$$

where n denotes the length of the time series and p the order of the AR–process that is used to fit the data. We will denote this criterion as the Akaike Final Prediction Error Criterion (AC).

5.4 Chaotic dynamics and time series analysis.

Consider the time series generated with the Logistic map with $\mu = 4$,

$$x(n + 1) \mapsto 4x(n)(1 - x(n)) \tag{19}$$

It can be shown that this series has a mean value of $\bar{x} = \frac{1}{2}$ and autocorrelations $\rho_i = 0, \quad i > 0$. When we construct an optimal linear predictor for this series, predictions will result that are equivalent to the predictions that are made with a predictor constructed with white noise of mean value $\overline{wn} = \frac{1}{2}$. The Logistic map exhibits the properties of *stretching* and *folding*, operations that give rise to chaotic dynamical behaviour5: stretching leads to exponential divergence of nearby trajectories, while folding keeps the orbit bounded. It is clear from this example that more subtle methods have to be adopted in the prediction of chaotic data: the stretching and folding operations introduce nonlinear relationships between data–samples, that cannot be extracted with con-

ventional linear methods. We will review some methods for quantification and prediction of non-linear time series, based on [44], [45], [46] and [50].

5.4.1 Deterministic noise and scatter plots.

Chaos can be qualified as random looking deterministic behaviour of low–dimensional systems. It occurs with dissipative systems, i.e. a volume contraction takes place that gives rise to stretching and folding. We can think of a system of N first–order autonomous ordinary differential equations, denoted in vector form as $\frac{dx(t)}{dt} = F[x(t)]$, with $x \in \mathbb{R}^N$, as a *dynamical system*, i.e. a deterministic mathematical description for evolving the state of a system forward in time. Considering e.g. the case N=3, we can refer to the space $\left(x^{(1)}, x^{(2)}, x^{(3)}\right)$, spanned by the three components of the vector x, as *phase space;* the path in phase space followed by a system is qualified as *orbit* or *trasjectory.* For continuous time dynamical systems, the term *flow* is more appropriate. A discrete time dynamical system is also called a *map*. When phase space volumes are not preserved under the time evolution of a dynamical system, it is called *dissipative*. It is an important concept in dynamics that dissipative systems typically are characterized by the presence of attracting sets or *attractors* in the phase space, i.e. bounded subsets to which regions of initial conditions of nonzero phase space volume asymptote as time increases. *Conservative* dynamical systems do not have attractors.

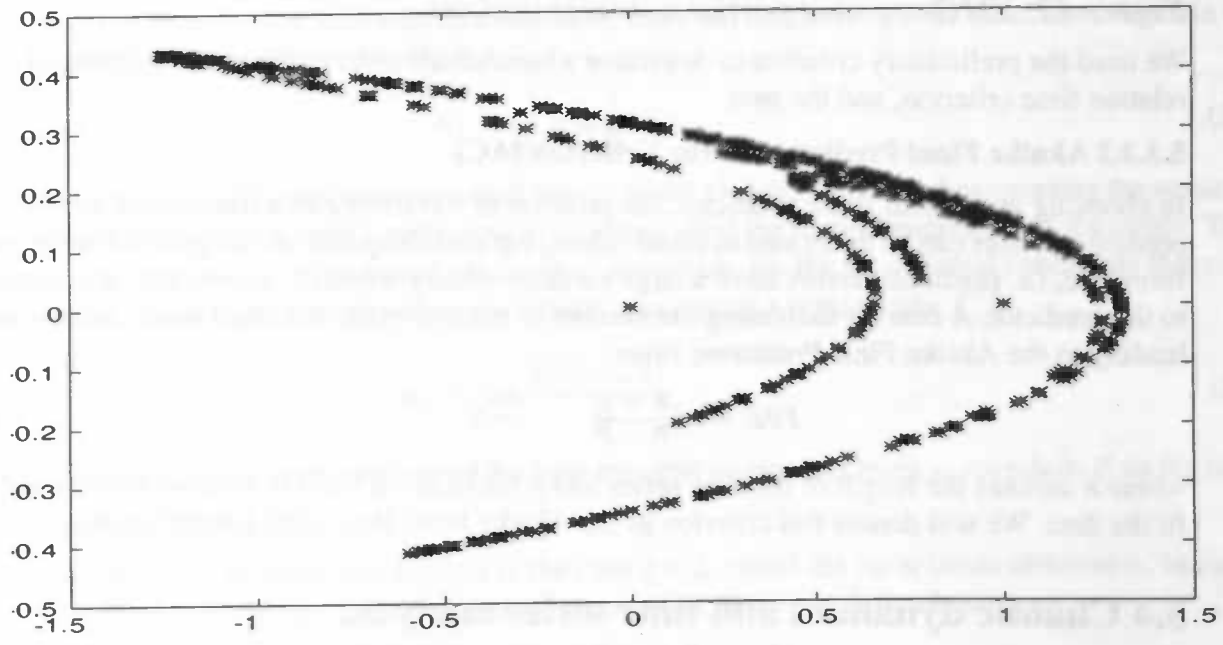

Figure 13. Hénon–attractor

It is a characteristic of chaotic dynamics that the resulting attractors often have a very intricate geometrical structure. According to the standard definition of dimension, these attractors commonly have a value that is not an integer, i.e. have *fractal* dimension. Attractors with fractal dimension are called *strange attractors*. An example of a strange attractor is the attractor obtained for the two–dimensional Hénon–map,

$$\begin{cases} x_{n+1}^{(1)} = 1 - a\left(x_n^{(1)}\right)^2 + x_n^{(2)} \\ x_{n+1}^{(2)} = b x_n^{(1)} \end{cases} \tag{20}$$

for $(a, b) = (1.4, 0.3)$. Plotting a sequence of points that are obtained by iterating (20), is essentially plotting a picture of the attractor (Figure 13.). Zooming into this plot consecutively, reveals

similar structure at all zoom–levels, indicating a dense, complex structure of box–counting dimension (Section 5.4.3) $D_0 \simeq 1.26$.

A major phenomenon exhibited by chaotic systems is *sensitive dependence to initial conditions*: evolutions of the systems can differ dramatically when the initial values differ just a tiny bit. It is caused by the stretching operation, and can be quantified by so–called *Lyapunov exponents*. Trajectories that have nearly identical initial conditions will separate from one another at an exponentially fast rate. In fact, a familiar kind of predictability–measure can be derived from this, naturally called *entropy*.

The third item characteristic of chaos is the presence of unstable periodic points located densely throughout the attractor. This existence of infinitely many periodic points is a reflection of the complexity of chaotic dynamics. It can be displayed by so–called *bifurcation diagrams*. Another way to display periodicities present in a series of evolution points, is by constructing a *scatter diagram*, i.e. a diagram in which $x(t)$ and $x(t–m)$ are plotted agains each other, where m denotes a periodicity present in the series.

5.4.2 Reconstruction of dynamics: embedology.

The insight that data with random appearance can indeed be generated deterministically, is important, because the prediction of future behaviour of a system is only possible to the extent that the system is deterministic (information from the system's past unambiguously determines its future states). For a purely stochastic system, predictions are by definition statistical in nature, and limited to mean values, higher moments, linear autocorrelation and other statistical quantities. It is therefore important when analyzing a complex signal to determine whether it results from low–dimensional chaos or just random behaviour. Methods for making this determination include attractor dimension and entropy estimation, reported below.

Prediction of a time series produced by a chaotic system can be split into two parts. First, the present state a of the system can be measured, and leads to a measured quantity $g(a)$; because of assumed determinism, a contains all information needed to produce the state t time units into the future, i.e. a t–lag prediction can be made, which we will call $F_t(a)$. Therefore, we have to *reconstruct* the state of the system: since future states are determined uniquely by past and present states, and it can be shown that the dynamics is bounded (remember the folding operation) so that nearly similar states will lead to nearly similar future states, we can infer future behavior of a system by looking at previous similar states and their consequenses. This method forms a solution to the *function approximation* problem in chaotic prediction.

However, we don't have direct access to the state a of the system, only to a (lower dimensional) measurement $g(a)$ of it, nl. the time series at hand. The solution to this *representation problem* (the second part in chaotic prediction) is to use an m–dimensional vector of *delay coordinates*,

$$b = \left[g(a), g(F_{-\tau}(a)), \ldots, g\left(F_{-(m-1)\tau}(a)\right) \right] \tag{21}$$

i.e. the present value of a time series and its m–1 predecessors, for estimation of the state. It can be shown (based on the famous theorem by Takens) that there exists a one–to–one correspondence between states and delay coordinate vectors, i.e. $D(a) = b$, under quite reasonable conditions on the dynamics F_t of the system, as long as

$$m > 2D_0(A) \tag{22}$$

where $D_0(A)$ denotes the *box–counting dimension* of A, and A is a compact finite–dimensional set of states of the system at hand, $a \in A$.

5.4.3 Attractor dimension.

Invariant sets arising in dynamical systems, such as chaotic attractors, often have structure on arbitrarily fine scale, and determination of their dimension, one of the most basic aspects of a set, is nontrivial. Since the frequency with which orbits visit different regions of a chaotic attractor can have its own arbitrarily fine scaled structure, the assignment of a dimension value gives a quantitative characterization of the geometrical structure of a complicated object. There exist altrenative methods for computing the dimension of sets, e.g. the *box–counting dimension* method, where the number of cubes $\hat{N}(\epsilon)$ of edge length ϵ, needed to cover the set within an N–dimenional Cartesian space, is counted, and the dimension is computed as

$$D_0 = \lim_{\epsilon \to 0} \frac{\ln \hat{N}(\epsilon)}{\ln(1/\epsilon)} \tag{23}$$

The box–counting dimension gives the scaling of the number of cubes needed to cover the attractor. For strange attractors, however, it is commonly the case that the frequency with which different cubes are visited can be vastly different from cube to cube: typical orbits will spend most of their time in a small minority of those cubes that are needed to cover the attractor.

A generalization of the former dimension that takes the various natural measures into account leads to the concept of a D_q–dimension. For $q = 0$, this equals the box–counting dimension, for $q = 1$, it is called the *information dimension,* since the numerator of the associated formula resembles a Shannon entropy form,

$$H_s = \sum_{i=1}^{r} p_i \ln \frac{1}{p_i} \tag{24}$$

We look at the frequencies with which typical orbits visit the various cubes covering the attractor in the limit that the orbit length goes to infinity, called the *natural measures* of the cubes. That is, for a typical x_0 in the attractor's *basin of attraction*, i.e. the set of initial conditions that converge to that particular attractor, the natural measure of a typical cube C_i is

$$\mu_i = \lim_{T \to \infty} \frac{\eta(C_i, x_0, T)}{T} \tag{25}$$

where $\eta(C_i, x_0, T)$ is the amount of the time the orbit originating from x_0 spends in C_i in the time interval $0 \leqslant t \leqslant T$.

Another case of particular interest is the case $q = 2$, called the *correlation dimension*, because it is relatively easy to determine from experimental data. It is defined as

Definition 4: The correlation dimension of a set K equals

$$D_2(K) = \lim_{\epsilon \to 0} \left[\frac{\ln \sum_i \mu^2(C_i)}{\ln \epsilon} \right] \tag{26}$$

It can be computed using the following algorithm:

Algorithm: Correlation Dimension (CD).

Consider a time series $x(n)$. Determine a suitable reconstruction vector dimension k and a range of vector distances *epsrange*. The correlation integrals of $x(n)$ can be estimated using the following algorithm:

(1) determine a (long) evolution $\{x(1), x(2), \ldots, x(N)\}$ in the supposed attractor K, using an arbitrary starting point $x(1)$

(2) extract k–dimensional reconstruction vectors from this sequence, using

$$rv_k(i) = \{x(i-k+1), x(i-k+2), \ldots, x(i)\} \tag{27}$$

(3) **for** each $\epsilon \in$ *epsrange*, determine the number of pairs $\#pairs_\epsilon$ of reconstruction vectors $(rv_k(i), rv_k(j))$, $0 < i < j \leq N$, having distance smaller then ϵ, i.e. $\| rv_k(i), rv_k(j) \|_\infty < \epsilon$

(4) the correlation integrals can be estimated by normalizing $\#pairs_\epsilon$ to the total number of pairs used in the comparance, $\hat{P}_{\epsilon,N}(K) = \dfrac{\#pairs_\epsilon}{N(N-1)/2}$

(5) look at (26); extrapolating the graph of

$$\ln \hat{P}_{\epsilon,N}(K) \text{ vs. } \ln \epsilon, \qquad \text{for } \epsilon \to 0,$$

we obtain an estimate of the correlation dimension $D_{2,k}(K)$ by determining the derivative of the extrapolated graph in $\epsilon = 0$

Note that in the algorithm, the summation–term in the numerator of (26) is computed by determining the *correlation integrals* of an attractor K, i.e. (for $k = 1$):

$$\hat{P}_{\epsilon,N}(K) = \frac{1}{N^2} \sum_{j=1}^{N} \sum_{i=1}^{N} U\left(\epsilon - |z_i - z_j|\right) \tag{28}$$

where

$$U(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{29}$$

and $\sum_i \mu^2(C_i) = \hat{P}_\epsilon(K) = \lim_{N \to \infty} \hat{P}_{\epsilon,N}(K)$. In fact, we are estimating the dimension by using a set of N orbit points $z_1, z_2, \ldots, z_N$ on the supposed attractor K, where N is large and the orbit has been sampled at some fixed interval δt.

From the above algorithm it becomes clear that we now can sketch a way in which chaos and noise can be distinguished. Since a chaotic attractor has an invariant low dimension, embedding into higher–dimensional reconstruction vectors sould yield the same dimension estimation. For a random time series, the dimension of the underlying (stochastic) system will be equal to the dimension of the reconstruction vector, as there is no underlying structure and samples will be spread uniformly throughout the reconstruction dimension.

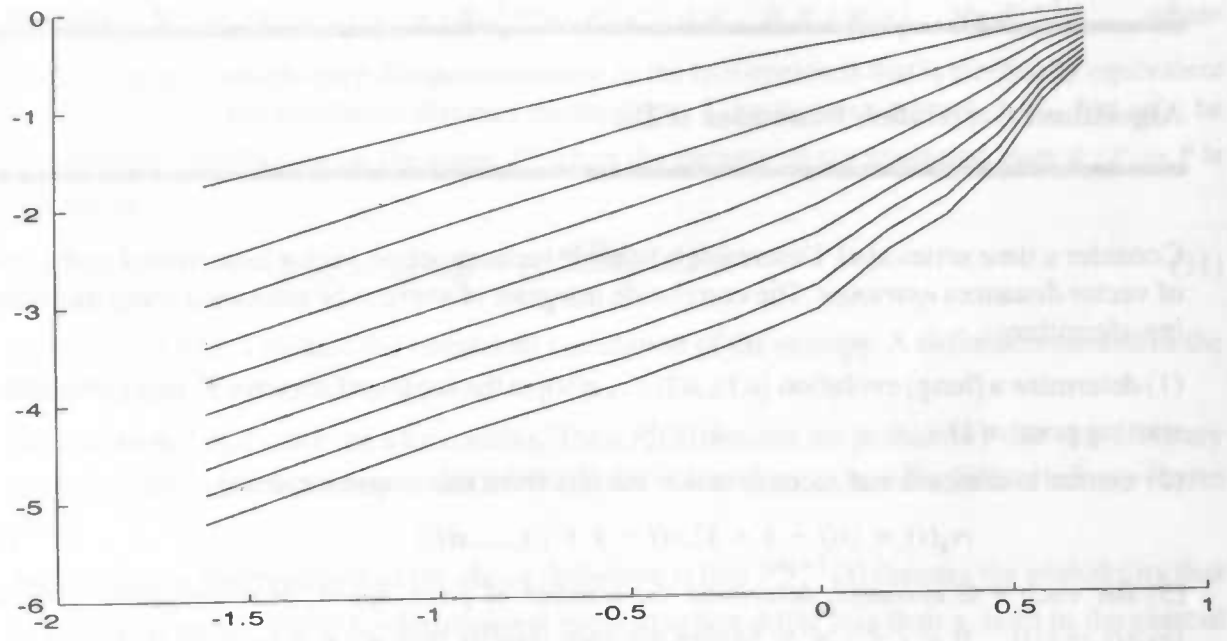

Figure 14. Correlation integrals of Hénon–attractor

Hence, the graphs of *log* (correlation integral) vs. *log* (ϵ) for increasing values of the reconstruction dimension will show **constant derivative** (which is seen to be an estimate of the correlation dimension) with chaotic data, whereas random sequences will show **increasing derivative**. For the chaotic Hénon–attractor, this is demonstrated for consecutive values of the reconstruction dimension in Figure 14..

The following algorithm for distighuising noise from chaos can hence be given:

Algorithm: Noise or Chaos? (NOC).

Consider a stationary time series $x(n)$ of length N. When $N \to \infty$, the first method below can be used to distinguish between chaotic dynamical and random behaviour underlying the time series. For short, finite time series, the second method is more appropriate.

First method.

(1) determine a range of reconstruction vector dimenions *krange*; **for** each $k \in$ *krange* extract k–dimensional reconstruction vectors from the time series, using

$$rv_k(i) = \{x(i-k+1), x(i-k+2), \ldots, x(i)\} \tag{30}$$

These reconstruction vectors define subsets of $\mathbb{R}^k$, and implicitly define a probability measure associated with the relative frequencies on the k dimensions, i.e. a *k–dimensional reconstruction measure* μ_k ;

(2) estimate the correlation integrals of these probability measures, e.g. by performing steps (3) and (4) of the **CD**–algorithm;

(3) estimate the correlation dimension of the k–dimensional reconstruction measures, e.g. by performing step (5) of the **CD**–algorithm;

(4) **if** the time series has *deterministic* dynamics **then**, for k large enough, the correlation dimension of the k–dimensional reconstruction measures μ_k will be independent of the particular value

of k; **else** the time series was produced by a *stochastic* system, **causing** the k–dimensional reconstruction measures to be distributed continuously throughout $\mathbb{R}^k$, **i.e.** $D_{2,k}(\mu_k) = k, \forall k \in krange$.

Second method.

(1) estimate mean $E(x(n))$ and covariances γ_i of the short length time series at hand;

(2) generate several new time series, resembling the original in the former respect, using an (optimal linear predictor based) linear model;

(3) **for** all (original and synthetic) time series, estimate the correlation integrals of the k–dimensional reconstruction measures μ_k ;

(4) **if** the ratio $\dfrac{\Delta\left[\hat{P}_{\epsilon,N}\left(K_{orig}\right), \hat{P}_{\epsilon,N}\left(K_{synth_i}\right)\right]}{\underset{i}{\sigma}\, \hat{P}_{\epsilon,N}\left(K_{synth_i}\right)}$ is small **then** there is a large difference between original time series correlation integrals on the one hand and synthetic time series correlation integrals on the other, so the original time series is likely to be also of stochastic nature **else** the original time series is likely to have deterministic behaviour

---

We have to say a few words on the effects of *noise* and *filtering* on the dimension estimation using the above algorithm. First, if we assume that white noise is present, than it can be regarded as fattening or fuzzing the attractor by an amount of order η, the typical noise amplitude. Thus, for observations of the attractor characteristics on scales ϵ greater than the noise level η, the attractor appears to be fractal, while for scales $\epsilon < \eta$ the attractor appears to be an N–dimensional volume, where N is the dimension of the space in which the attractor lies. White noise effectively limits the smallest size ϵ that can be used indimension determinations.

Second, much digital data is already filtered by the measurement apparatus (e.g. *anti–aliasing filtering*, i.e. low–pass filtering of raw data to prevent violation of the sampling theorem by high signal frequencies, causing aliasing), which can lead to erronuous calculations of dimension. However, the application of a finite impulse response (FIR) filter to delay coordinates does not change, in theory, the number of independent coordinates needed to represent an attractor. When calculating correlation dimension, $m > D_2$ is sufficient (m is the output dimension of the filter, i.e. the numer of FIR–filtered delay coordinates) for a one–to–one correspondence to occur between states and vectors. In a practical situation, when only a fixed amount of finite resolution data is available, an FIR filter may increase the difficulty of calculating dimension, although it can not change the dimension theoretically. The attractor may now be embedded in a much more complicated way, so a significantly larger time series may be needed in order to recognize the underlying structure properly. An infinite impulse response (IIR) filter may change the dimension, even in theory. The exact effect depends on the Lyapunov exponents of the filter. The more an FIR filter resembles an IIR filter (the limiting case of an FIR filter), the more the attractor will be distorted, and the higher the dimension estimate will possibly be.

5.4.4 Estimation of the entropy of a time series.

From Shannon's information theory it is known that the *entropy* measure of (24) indicates the degree of uncertainty in being able to predict the output of a probabilistic event. There are many definitions of entropy, that serve various purposes. We adopt the following definition, by which a clear connection between entropy and Lyapunov exponents (which we will not discuss) can be seen. It is defined for the Thom–map, that is operating on a torus T^2.

Definition 5: Let $D(p,n,\epsilon) = \left\{p' \mid dist\left[\phi^i(p), \phi^i(p')\right] < \epsilon, \quad 0 \leqslant i \leqslant n\right\}, \quad \forall p \in T^2,$ where *dist*[. , .] denotes an arbitrary distance–measure on the two operands that is metrically equivalent to $D(p,n,\epsilon)$, e.g. the Euclidean distance measure $\| \cdot, \cdot \|_2$ or the max–norm $\| \cdot, \cdot \|_\infty$. Let $m(\cdot)$ be a probability distribution on the torus T^2. Then the entropy of the evolution map $\phi : P \to P$ is defined as

$$h(\phi) = \lim_{\epsilon \to 0} \lim_{n \to \infty} - \ln \frac{(m(D(p,n,\epsilon))}{n} \tag{31}$$

We will next give a method for numerical calculation of the entropy. A definition needed in the algorithm is;

Definition 6: Let $X = x(n)$ be a time series. Then $P_\epsilon^k(X)$ denotes the probability that two arbitrary k–dimensional reconstruction vectors from X differ less than ϵ, in a Euclidean infinity–norm sense.

An alternative interpretation of the above definition is that $P_\epsilon^{k_0+1}(X)$ denotes the probability that two points p and p' in the k_0–dimensional reconstruction differ less than ϵ, both in the current situation and after one evolution step of the system underlying the time series. Hence, the entropy of a time series can also be defined as

Definition 7: Let $X = x(n)$ be a time series. Then the entropy of the time series equals

$$h(X) = \lim_{\epsilon \to 0} \lim_{k \to \infty} - \ln \frac{P_\epsilon^k(X)}{k} \tag{32}$$

This definition is valid for every time series, chaotic or random. However, when noise is present, the entropy $h(X)$ will tend to ∞ in the limit for $\epsilon \to 0$. Also, taking the limit for $k \to \infty$ will introduce noise because of the small resulting values of $P_\epsilon^k(X)$; a maximum likelihood estimation seems in order.

Algorithm: Maximum Likelihood Entropy Estimation (MLEE).

Consider a time series $X = x(n)$. (1) estimate the average number of times *#prolongings* a pair of k_0–dimensional reconstruction vectors $\left(rv_{k_0}(i), rv_{k_0}(j)\right)$ of distance less than ϵ has to be prolonged, i.e. the reconstruction dimension k_0 has to be increased, for the vectors to become apart farther than ϵ. This can be done using

$$\#prolongings = \sum_{i=0}^{\infty} P_\epsilon^{k_0+i}(X) \,/ P_\epsilon^{k_0}(X) \tag{33}$$

(2) the maximum likelihood estimation of the entropy of X at distance level ϵ is given by

$$\hat{h}(X,\epsilon) = - \ln\left(1 - \frac{1}{\#prolongings}\right) \tag{34}$$

Next, we have to **interprete** the values computed for the entropy. Heuristically speaking, entropy can be looked upon as a measure for the number of iterations of a map that is necessary for two values initially having distance less than ϵ to diverge farther than ϵ. Thus, it quantifies the extent to which the underlying system shows sensitive dependence to initial conditions.

Random time series will exhibit less prolongings of reconstruction vectors as the distance level ϵ decreases. To see this, imagine two arbitrary reconstruction vectors at distance δ, and let $\delta < \epsilon_1 < \epsilon_2$. The number of prolongings at level ϵ_2 is expected to increase w.r.t. the number of prolongings at level ϵ_1, because of the random character of every next sample added to both reconstruction vectors. The interdependences between samples are all random and of the same order, so slight increase in distance level will cause slight increase in the "initial pool of close vectors". Since an increase of the same order is expected for each of the terms $i = 1, 2, \ldots, \infty$ in the summation of (33), the *average* number of prolongings will increase, hence the entropy will decrease.

With chaotic time series, on the onther hand, sensitive dependence to initial conditions will be present, so there will exist a region of ϵ where the average number of prolongings will be nearly constant, since there exists a large "distance" between states, going from reconstruction dimension k_0 to $k_0 + 1$, so a slight increase in ϵ will not be followed by an increasing number of prolongings. In other words, there will be a region of ϵ where the entropy is almost independent of ϵ.

For very large ϵ, i.e. where it exceeds the maximum distance in the time series, the average number of prolongings will be infinite, so the entropy will tend to zero. For very small ϵ, i.e. where it is exceeded by quantization noise, predictability will decrease, so the entropy will increase with decreasing ϵ.

5.4.5 Local linear prediction.

We can employ the idea of local linear approximation of nonlinear differentiable functions in the prediction of chaotic signals. In effect, we look for the nearest neighbours in state space (as indicated by the sequence of past delay coordinate vectors) and build a model for linear inference with it. The assumption underlying this procedure is, of course, that nearly similar states will lead to nearly similar future evolutions. Local linear prediction can be performed using the following algorithm:

Algorithm: Local linear prediction (LLP) .

Consider a chaotic time series $x(n)$. We can predict the value of x at time N+1 from the values of the series up to time N by employing local linear prediction:

(1) collect all k–dimensional reconstruction vectors $(x(i-k+1),\ldots,x(i))$ from the past that are closer (in a Euclidean distance sense) than r to $(x(N-k+1),\ldots,x(N))$.

(2) determine an optimal linear model of the form

$$\hat{x}(i+1) = a_0 + \sum_j a_{j+1} x(i-j) \tag{35}$$

in order to perform linear prediction *within* this set of reconstruction vectors, i.e. we determine coefficients $a_0,\ldots,a_k$ such that the variance σ of $x(i+1) - a_0 - \sum_j a_{j+1} x(i-j)$ is minimized.

(3) let M be the number of reconstruction vectors used in the linear model of the previous step. The predicted value of $x(N+1)$ equals

$$\hat{x}(N+1) = a_0 + \sum_j a_{j+1} x(N-j) \tag{36}$$

The prediction error variance σ_{all} depends on the number of nearest neighbours in the linear model. For small M, the former value of σ has to be corrected for parameter estimation errors in the determination of the optimal linear predictor model,

$$\sigma_{all} = \begin{cases} \sigma \dfrac{M+k+1}{M-k-1}, & N \text{ small} \\ \sigma, & M \text{ large} \end{cases} \tag{37}$$

(4) keeping parameter k at a fixed value, vary the value of r and look for minimum variance σ_{all} (small and large values of k will both cause large variances, so a minimum does occur).

(5) vary the value of r and again look for minimum variance σ_{all}. Using the optimal values of k and r, perform the prediction of step (3).

The main shortcoming of this algorithm lies in the (not very structured) way the prediction error variances σ_{all} are determined. Finding stronger foundations for the application of this method is an item of future research.

Chapter 6.
Neural networks

Nowadays, neural networks (NN's) are sometimes considered as a kind of "magic potion", "the solution to all your problems", in other words: a universal problem solver. Others consider them a metaphore of the human brain, a means to gain insight into the way its biological counterpart behaves.

We like to think of a NN mainly as just a signal processing tool, of which the overall mechanisms are not yet fully understood. Just as any other method, it has its own specific application areas, advantages and disadvantages. It is a *powerful* tool, though, that is able to (adaptively) perform any nonlinear i/o–mapping and to learn (spatial, temporal, etc.) characteristics of the inputs, thus capable of generalizing the captured features to patterns unseen. It has limited ability in interpreting input–data: when offered conflicting or badly preprocessed data, it has a hard time in establishing any relations between inputs (and outputs) at all, so no learning, or bad generalization takes place. The topic of *preprocessing vs. network architecture* is thus of major importance in neural network design.

In this chapter we will give a summary of neural structures that can be utilized in the solution of the central problem of this thesis: identification/ prediction of inverse dynamics/ muscular activity within the framework of an artificial walking system, based on [33]. Of course, in order to gain more insight in a solution procedure, and for benchmarking purposes, we have to look in the fields of conventional (mathematical) temporal processing (statistical and chaotic dynamical methods for time series analysis) also.

6.1 Neural networks: learning and architectures

In Chapter 4, already some statements were made about the utilization of neural networks in nonlinear system identification, having inherent nonlinear mapping properties. It can be shown, that a neural network of just one hidden layer is capable of approximating any nonlinear continuous i/o–mapping (subject to some constraints), as long as the number of units in the hidden layer can be mae infinitely large. This is called the *Universal Approximation Theorem.* We will not go into details about neural network fundamentals, but assume these known by the reader. When desired, details can be found in [33].

6.1.1 Learning theory.

Supervised neural network training can be used to perform an *empirical risk minimization procedure*, where network learning is looked upon as *minimizing an empirical risk functional*, which is a function of the discrepancy between a desired response vector and the actual network response to the associated input vector. According the method of *Structural Risk Minimization*, it follows that networks having too much degrees of freedom (in other words, the learning problem is underdetermined, or the machine capacity is too large for the amount of training data), will in fact show bad generalization performance.

6.1.2 Neural temporal processing.

One objective of this research is to gain some insight into the way a neural network captures time dependencies. A number of approaches can be described, which have each their own way of doing this:

(1) incorporating input or intra–neural **time delays** in the network (also called: time–delayed neural networks, or *TDNN*'s)

(2) **preprocessing** the inputs

(3) employing **network recurrency**, e.g. the *real–time recurrent neural network*

Of these approaches, the standard multilayered Perceptron, incorporating an input tapped delay–line, will be chosen to use in further investigation and simulation.

6.2 Self–Organized Principal Components Analysis (SOPCA).

A second class of network architectures is formed by the so–called *self–organizing systems*, networks that perform a learning task on their own, without a teacher telling them what to do.

6.2.1 Introduction.

In the previously mentioned networks, learning takes place in a supervised manner, i.e. target outputs are offered along with the input–patterns, so the network is able to learn on the basis of the error that is made w.r.t. these targets. In fact, the targets may take the form of a desired i/o–mapping that the algorithm is required to approximate.

We can imagine that there are situations where it is impossible to provide target–values to a network, e.g. in tasks like *pattern recognition* and *feature extraction*. In these cases, we would like a network to learn a mapping by itself, i.e. to learn in an *unsupervised* manner. The purpose of an algorithm for unsupervised learning is to discover significant patterns or features in the input data, and to do the discovery without a teacher. To do so, the algorithm is provided with a set of rules of a *local* nature, which enables it to learn to compute an i/o–mapping with specific desirable properties; the term "local" means that the change applied to the synaptic weight of a neuron is confined to the immediate neighborhood of that neuron. Note that the modeling of network structures used for self–organized learning tends to follow neurobiological structures to a much greater extent than is the case for supervised learning: the process of self–organization is actually fundamental to the organization of the brain.

6.2.2 Self–organizing systems.

The key question in unsupervised learning is how a useful configuration can finally develop from self–organisation. An observation of fundamental importance to learning in both biological and artificial neural networks aids in answering this key question:

"Global order can arise from local interactions"

In particular, many initially random local interactions between neighbouring neurons of a network may ultimately give rise to coherent behaviour (by a transition to states of global order). This is the essence of self–organisation.

6.2.2.1 Three principles of self–organization.

Network organisation occurs by means of modifications in connection strengths (synaptic weights) between neurons, in response to network activity patterns that are fed back into the network. The invoked activity patterns are produced by the network in response to input signals (and previous activities). Connection strengths are modifiable because of *synaptic plasticity*. We now state the *three principles of self–organisation*:

(1) Modifications in synaptic weigths tend to self–amplify

The feedback between changes in synaptic weights and changes in activity patterns have to be positive to prevent early network stabilization. Since modifications in synaptic weights have to be based on locally available signals (presynaptic and postsynaptic signals), a combination of self–reinforcement and locality is actually being required, which results in a (Hebbian like) mechanism of self–amplification.

(2) Limitation of resources leads to competition among synapses and therefore the selection of the most vigorously growing synapses (i.e. the fittest) at the expense of the others

Ultimately, the network has to stabilize into states of global order. Unrestricted growing of synapses would prevent such a stabilization.

(3) Modifications in synaptic weights tend to cooperate

Favourable events are produced by many cooperating synapses rather than by the sole activity of a single synapse. The presence of a vigorous synapse can enhance the fitness of other (neighbouring) synapses, in spite of the overall competition in the network.

Besides these architectural constraints, also an input data requirement has to be satisfied in order to achieve useful self–organization: there has to be *redundancy* in the activation patterns fed into the network by the environment. This can be clarified by the following reasoning:

Knowledge to be incorporated in the neural network is provided via input pattern redundancy, part of which can be obtained by observations of statistical parameters of the input data (e.g. mean, variance and correlation matrix). This knowledge gives rise to a model of "what usually happens", against which incoming messages are compared, and unexpected discrepancies are thereby identified, a necessary prerequisite of self–organized learning.

6.2.2.2 Approaches to self–organized learning.

Within the paradigm of self–organized learning, there are three (main) approaches. First, there is the standard *Hebbian learning* approach, usually performed with standard feedforward networks. We will adopt this approach in the sequel.

A special class of artificial neural networks is formed by *self–organizing feature maps*, which are based on *competitive learning*; the output neurons of the network compete among themselves to be activated or fired, with the result that only one output neuron, or one neuron per group, is on at any time. This can be established, for example, by using lateral inhibitory connections between the output neurons (e.g. by employing a Mexican hat function). As self–organization proceeds, a topographic feature map of the input patterns is ultimately obtained, i.e. a one– or two–dimensional lattice of output nodes, in which the spatial locations of the neurons correspond to intrinsic features of the input patterns.

Finally, a third approach is rooted in Shannon's *information theory*. The main idea is, to make the synaptic connections of a multilayered Perceptron to develop in such a way as to maximize the amount of information that is preserved when signals are transformed at each processing stage of the network, subject to certain constraints. There is again a direct relationship between this approach and the technique of feature extraction, since it is recognized that encoding of data from a scene for the purpose of redundancy reduction is related to the identification of specific features in the scene.

6.2.3 Principal components analysis (PCA).

We already mentioned the suitability of self–organizing systems to perform feature extraction. Feature extraction (or feature selection) refers to a process whereby a data space is transformed into a feature space, theoretically having the same dimension as the original data space. This transformation is designed in such a way that the data set may be represented by a reduced number of "effective" features and yet retain most of the intrinsic information content of the data. When we establish this dimensionality reduction by boldly truncating the data vector **x** to the (smaller) amount of desired elements, we will cause a mean–squared error equal to the sum of the variances of the elements eliminated from from **x**. Therefore, we are looking for an invertible *linear* transformation **T** such that the truncation of **Tx** is optimum in the mean–squared error sense. This transformation **T** should obviously have the property that some of its components have low variance. Principal components analysis (PCA, or the *Karhunen–Loève transformation* in communication theory), a standard method in statistical pattern recognition, can be considered as such a transform, since it maximizes the rate of decrease of variance, going from one component to another (less principal) component.

6.2.3.1 Introducing PCA.

We now present the formal reasoning underlying PCA, as presented in [33]. We give this detailed derivation, because it clarifies the behavior of the algorithm as a feature extractor, a major topic in NN–design. Later on, it is described how this statistical algorithm can be implemented elegantly by using a self–organizing neural network, illustrating both power and rationality of the neural approach.

Let **x** denote a p–dimensional *random vector* representing the data set of interest. We can safely assume that **x** has zero mean: if this is not already the case, we subtract the mean from it before proceeding with the analysis. Let **u** denote a *unit vector*, also of dimension p, onto which the vector **x** is to be projected. This projection is defined by the inner product of the vectors **x** and **u**:

$$a = x^t u = u^t x \tag{38}$$

and is subject to the constraint

$$|u| = (u^t u)^{1/2} = 1 \tag{39}$$

The projection a is a random variable with mean and variance related to the statistics of the data vector **x**. Under the assumption that the random data vector **x** has zero mean, it follows that the mean value of the projection a is zero too:

$$E[a] = u^t E[x] = 0 \tag{40}$$

Therefore

$$Var[a] = E\left[a^2\right] - E^2[a] = E\left[a^2\right] \tag{41}$$

so we may write

$$\sigma^2 = Var[a] = E\left[a^2\right] = E[(u^t x)(x^t u)] = u^t E[xx^T]u = u^t R u \tag{42}$$

where the p–by–p *correlation matrix* **R** of the data vector **x** is defined by

$$R = E[xx^t] \tag{43}$$

We observe that the correlation matrix **R** is *symmetric*, which means that

$$R^t = R \tag{44}$$

and it follows that if **a** and **b** are any p–by–1 vectors, then

$$a^tRb = b^tRa \tag{45}$$

From (42) we see that the variance $\sigma^2$of the projection a is a function of the unit vector **u**; we may write

$$\psi(u) = \sigma^2 = u^tRu \tag{46}$$

and think of $\psi(u)$ as a *variance probe*.

6.2.3.2 Eigenstructure of PCA.

Now we want to find unit vectors **u** along which $\psi(u)$ has local maxima or minima, subject to a constraint on the Euclidean norm of **u**. The solution to this problem lies in the eigenstructure of the correlation matrix **R**. We already give two important consequences we are about to draw from the eigenstructure of PCA:

- the eigenvectors of the correlation matrix **R** pertaining to the zero–mean data vector **x** define the unit vectors u_j, representing the principal directions along which the variance probe $\psi(u_j)$ have their extremal values
- the associated eigenvalues define the extremal values of the variance probes $\psi(u_j)$

If **u** is a unit vector such that the variance probe $\psi(u)$ has an extremal value ($\psi'(u) = 0$), then for any small perturbation δu of the unit vector **u**, we find that, to a first–order approximation in δu,

$$\psi'(u) = 0 \leftrightarrow \psi(u + \delta u) = \psi(u) \tag{47}$$

From (46) we have

$$\psi(u + \delta u) = (u + \delta u)^tR(u + \delta u) \tag{48}$$

which, when using (45), can also be written as

$$\psi(u + \delta u) = u^tRu + 2(\delta u)^tRu + (\delta u)^tR\delta u \tag{49}$$

Since we are occupied with a first–order approximation, we can discard the second–order term $(\delta u)^tR\delta u$ and can take (48) a step further, using (46):

$$\psi(u + \delta u) = u^tRu + 2(\delta u)^tRu = \psi(u) + 2(\delta u)^tRu \tag{50}$$

Using (47), this leads to

$$(\delta u)^tRu = 0 \tag{51}$$

We have derived a condition the data vector u has to satisfy in order to obtain an extremal variance probe value. Next we restrict the perturbations δu to those perturbations for which the Euclidean norm of the perturbed vector $|u + \delta u|$ is unity; using (39) this means that

$$(u + \delta u)^t(u + \delta u) = 1 \tag{52}$$

and when we again discard the second–order term, and use (38), we actually required that

$$(\delta u)^t u = 0 \tag{53}$$

Note that the left–hand side of the last equation is just the projection of a perturbation δu onto the unit vector **u**, so the permitted perturbations are orthogonal to **u**. In fact, only changes in the direction of **u** are admissible this way.

Next we can write

$$(\delta u)^t R u - \lambda (\delta u)^t u = 0 \tag{54}$$

or, since the multiplication order of the scalar and the transposed perturbation may be changed,

$$(\delta u)^t (Ru - \lambda u) = 0 \tag{55}$$

For this condition to hold, it is necessary and sufficient that

$$Ru = \lambda u \tag{56}$$

which is the well known *eigenvalue problem,* having nontrivial solutions (i.e. $u \neq 0$) only for special values of λ, the eigenvalues of the correlation matrix **R**. When the eigenvalues are distinct, the associated eigenvectors are unique.

Now that a connection between the unit vectors implying maximum or minimum variance probe on the one hand and the eigenvectors of the correlation matrix on the other has been established, we are looking for a connection between the actual values of the extremal points of the variance probe $\psi(u)$ and the (distinct) eigenvalues of **R**. This way, we could order projection vectors a according to their variance on the basis of these eigenvalues.

Let the eigenvalues of the p–by–p matrix **R** be denoted by $\lambda_0, \lambda_1, \ldots, \lambda_{p-1}$ and the associated eigenvectors by $u_0, u_1, \ldots, u_{p-1}$. Then, (56) can also be written

$$Ru_j = \lambda_j u_j, \quad j = 0, 1, \ldots, p-1 \tag{57}$$

Now we arrange the corresponding eigenvalues in decreasing order

$$\lambda_0 > \lambda_1 > \cdots > \lambda_j > \cdots \lambda_{p-1} \tag{58}$$

such that $\lambda_0 = \lambda_{max}$. Remember that we assumed distinct eigenvalues. We can combine the p equations (57) into one new equation

$$RU = U\Lambda \tag{59}$$

when we construct the p–by–p matrix **U** with eigenvectors, sorted in the same way as their associated eigenvalues (58)

$$U = \left[u_0, u_1, \ldots, u_j, \ldots, u_{p-1}\right] \tag{60}$$

and define

$$\Lambda = diag\left[\lambda_0, \lambda_1, \ldots, \lambda_j, \ldots, \lambda_{p-1}\right] \tag{61}$$

Again reminding the assumption of distinct eigenvalues, the columns of **U**, i.e. the eigenvectors of **R**, can be seen to satisfy the *conditions of orthonormality*

$$u_i^t u_j = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases} \tag{62}$$

We can use this to rewrite (59) in the form

$$U^t R U = \Lambda \tag{63}$$

since (62) leads to

$$U^t U = I \tag{64}$$

and obviously

$$U^t = U^{-1} \tag{65}$$

Arriving then at

$$u_j^t R u_k = \begin{cases} \lambda_j, & k = j \\ 0, & k \neq j \end{cases} \tag{66}$$

and combining this with (46), we conclude

$$\psi(u_j) = \lambda_j, \quad j = 0, 1, \ldots, p-1 \tag{67}$$

We see that variance probes and eigenvalues are equal; thus, when assuming that the correlation matrix **R** has distinct eigenvalues, we can always find a set of unit vectors u_j such that the projections of **x** on the u_j, a_j, exhibit decreasing variance. The fact, that each unit vector probe value reflects a local extremum of the projections variances, guarantees this.

6.2.3.3 Representation and reduction of data.

The previous outcomes can be used to encode a data vector in an efficient way, nl. expressed w.r.t. a basis in feature space, as opposed to its original representation (in data space). This new basis consists of unit vectors such that projections of the data vector onto this basis show some (above stated) desirable property.

With a p–dimensional unit vector, we have to consider p possible projections of the data vector **x** onto **u**,

$$a_j = u_j^t x = x^t u_j, \quad j = 0, 1, ,\ldots, p-1 \tag{68}$$

where the a_j are called the ***principal components***, displaying the same physical dimensions as the data vector **x**. In fact, this embodies an **analysis** of a data vector into its principal components, hence (68) represents a ***principal components analysis***.

We can also go the other way around, i.e. reconstruct a data vector **x** exactly from its projections a_j, using the **synthesis formula**

$$x = Ua = \sum_{j=0}^{p-1} a_j u_j \tag{69}$$

where the original vector is retrieved by means of a linear combination of In effect, the unit vectors u_j represent a *basis* of the data space.

The power of the previously described decomposition lies in the fact that it provides an efficient technique for *dimensionality reduction*: we may reduce the number of features needed for effective data representation by discarding projections with small variance in (69).

Note that it is common practice to look for minimum variance (remember the way to compute linear predictor coefficients: essentially, the coefficients yielding minimum variance of the prediction errors over the train data are chosen as the optimal coefficients); in this case, however, the opposite is true: components with large variance indicate much information ...

When we denote the largest m eigenvalues of the correlation matrix **R** of **x** by $\lambda_0, \lambda_1, \ldots, \lambda_{m-1}$, we may approximate **x** by truncating the synthesis formula (69) after m terms:

$$x' = \sum_{j=0}^{m-1} a_j u_j, \quad m < p \tag{70}$$

The approximation error vector **e**, given by

$$e = x - x' \tag{71}$$

is orthogonal to the approximating data vector **x'**. Its variance can be computed when keeping in mind, that the total variance of the p components of the random vector **x** equals (via (46) and (66))

$$\sum_{j=0}^{p-1} \sigma_j^2 = \sum_{j=0}^{p-1} \lambda_j \tag{72}$$

with σ_j^2 the variance of the jth principal component a_j. The same reasoning applies to the approximating vector **x'**, so using (70), the variance of **e** equals

$$\sum_{j=m}^{p-1} \sigma_j^2 = \sum_{j=m}^{p-1} \lambda_j \tag{73}$$

Since the eigenvalues $\lambda_m, \lambda_{m+1}, \ldots, \lambda_{p-1}$ are the **smallest** *(p–m)* eigenvalues of the correlation matrix **R,** the approximation preserves the most important information contained in the data, and will become more accurate when the eigenvalues discarded in the approximation are closer to zero.

In effect, when performing dimensionality reduction on some input data using a Principal Component Analysis, we compute the eigenvalues and eigenvectors of the correlation matrix of the input data vector, and then *project* the data orthogonally onto the *subspace spanned by the eigenvectors belonging to the largest eigenvalues.* This method is known as **subspace decomposition**.

6.2.3.4 Usefulness of PCA.

Principal Components Analysis can offer a useful unsupervised procedure for achieving good data compression while preserving as much information about the inputs as possible. For example, a PCA may be used for data compression by employing the greatest numerical precision to encode the first principal components of the input, and progressively less precision to encode the remaining $m-1$ components. Also note that a PCA on the input data provides a linear transformation, which is optimum in the sense that it permits reconstruction of the original input data within a mean–squared error. Another useful application of PCA appears in data made up of an aggregate of several clusters: the leading principal axes then will tend to pick projections of clusters with good separations.

We can also point at a situation where a PCA seems a less suitable procedure: where there are meaningless variables (outliers) with a high noise level.

Some researchers look upon a *self–organized PCA* (SOPCA, or the generalized Hebbian algorithm (GHA), presented in Section 6.2.4.2) as a special case of a *hierarchical clustering* algorithm based on competitive learning. Moreover, a SOPCA may be employed as a *preprocessor* for a (un)supervised neural network. This can be seen by noting that the set of projections formed by a PCA are uncorrelated with each other, and that the convergence process in the backpropagation algorithm is typically slow due to interacting effects of a MLPs synaptic weights on the error signal. Applications of this kind include: use of GHA for dimension reduction of speech and acoustic emission signals, leading to a significant reduction in training time without affecting classification accuracy. Remember that in the context of convergence rates with the backpropagation algorithm, is can be stated that large–scale neural network training problems are so inherently difficult to perform that no supervised learning strategy is feasible, and that other techniques like preprocessing may be necessary.

Finally, also nonlinear PCA networks have been proposed, where a sigmoidal activation function is added to the model of a (linear) neuron. This enables the extraction of higher–order statistics

and adds robustness to the expansion. This technique has been succesfully applied to the separation of sinusoidal signals.

6.2.4 Neural network implemented PCA.

We now present a neural implementation of the PCA–technique, based on Hebbian self–organized learning. The specific method for computing eigenvectors of the correlation matrix of the input data is similar to a technique known as *Hotelling's deflation technique*, following a procedure similar to the well–known Gram–Schmidt orthogonalization.

6.2.4.1 A single linear neuron: maximum eigenfilter.

We start the description by stating the following lemma:

Lemma 1. The synaptic weight vector $w(n)$ of a self–organized linear neuron, operating under the modified Hebbian learning rule of equation (77) converges with probability 1 to a vector of unit Euclidean length, which lies in the maximal eigenvector direction of the correlation matrix of the input vector.

Proof.

We will give a brief outline of the (rather extensive) proof of this lemma. Consider a neuron model exhibiting a linear transfer, i.e.

$$y = \sum_{i=0}^{p-1} w_i x_i \tag{74}$$

where the x_i, w_i denote the neuron's p inputs and weights, respectively. With Hebbian learning, a synaptic weight w_i varies with time, and increases when presynaptic and postsynaptic signals (x_i, y, respectively) coincide. This can be expressed by

$$w_{ji}(n+1) = w_i(n) + \eta\, y(n)\, x_i(n), \quad i = 0, 1, \ldots, p-1 \tag{75}$$

Since this learning rule allows the synaptic weight w_i to grow infinitely, a normalization step is incorporated:

$$w_{ji}(n+1) = \frac{w_i(n) + \eta\, y(n)\, x_i(n)}{\left(\sum_{i=0}^{p-1}\left[w_i(n) + \eta\, y(n)\, x_i(n)\right]^2\right)^{\frac{1}{2}}} \tag{76}$$

which can be approximated by a Taylor–series in η (for small η and ignoring second– and higher order terms):

$$\Delta w_{ji}(n) = \eta\, y(n)\left[x_i(n) - y(n) w_i(n)\right] \tag{77}$$

When we recognize the term between squared brackets as the *effective input* of the ith synapse after normalization (remember self–organization Principle (2)), the familiar Hebbian learning rule of (75) emerges. The product term $-\, y_i(n)\, w_i(n)$ is related to a *forgetting factor*, that becomes more pronounced with a stronger response $y(n)$.

Proceeding, we define

$$x(n) = \left[x_0(n), x_1(n), \ldots, x_{p-1}(n)\right]^T \tag{78}$$

and

$$w(n) = \left[w_0(n), w_1(n), \ldots, w_{p-1}(n)\right]^T \tag{79}$$

as input and synaptic weight vector, respectively, typically random vectors. Since (74) can then be written in the inner product form

$$y(n) = x^T(n)\, w(n) = w^T(n)\, x(n) \tag{80}$$

equation (77) transforms via

$$\Delta w(n) = \eta\, y(n)\left[x(n) - y(n)w(n)\right] \tag{81}$$

into

$$\Delta w(n) = \eta\left[x(n)x^T(n)w(n) - w^T(n)x(n)x^T(n)w(n)w(n)\right] \tag{82}$$

which is a recursive, stochastic, time–varying difference equation.

General convergence of stochastic approximation algorithms.

Convergence of this algorithm can be analyzed by looking more generally to the convergence of a class of stochastic approximation algorithms. Here, we look at approximation algorithms, the purpose of which is the recursive computation of synaptic weight vectors of neural networks, as described by

$$w(n+1) = w(n) + \eta(n)h(w(n), x(n)), \quad n = 0, 1, 2, \ldots \tag{83}$$

where n is the number of iterations, $w(\cdot)$ is a sequence of vectors that are the object of interest, and $x(n)$ is an observation vector received at time n (a sample function of a stochastic process), which causes $w(n)$ to be updated to take account of new information. The update function $h(\cdot,\cdot)$ is a deterministic function with some regularity conditions imposed on it. Note that the algorithm of (81) fits into this framework, except for the learning parameter η which is kept constant in it.

The convergence of the algorithm relies on a number of assumptions:

(1) learning–rate η is small enough for the synaptic weights to be treated as stationary on a short term, i.e. $E\left[w(n+1)|w(n)\right] = w(n) + \Delta w(n)$

(2) input vector $x(n)$ is drawn from a stationary stochastic process, with correlation matrix $R = E\left[x(n)x^T(n)\right]$ having distinct eigenvalues

(3) input vector $x(n)$ and synaptic weight vector $w(n)$ are statistically independent

The goal of the procedure in this particular algorithm, is to associate a deterministic nonlinear differential equation with the stochastic nonlinear difference equation (81). The stability properties of the differential equation are then tied to the convergence properties of the approximation algorithm. For this purpose, the learning rate parameter is made time–varying, and $\eta(n)$ is assumed to be a decreasing sequence of positive real numbers, satisfying

(1) $\sum_{n=1}^{\infty} \eta(n) = \infty$

(2) $\sum_{n=1}^{\infty} \eta^p(n) < \infty, \quad p > 1$

(3) $\eta(n) \to 0, \quad n \to \infty$

Condition (1) is necessary for the algorithm to move the estimate to a desired limit, regardless of the initial conditions. The speed of decay to zero (3) is conditioned by (2). Note that for $\eta(n)$ we can choose the function $\eta(n) = \frac{1}{n}$ to satisfy all three conditions.

Requiring some more conditions, e.g. with respect to local asymptotic (Lyapunov) stability of an ordinary differential equation (ODE) derived from (81),

$$\frac{d}{dt}w(t) = \bar{h}(w(t)) \tag{84}$$

and existence of a limit on $h(w,x)$, denoting the term between squared brackets in (82),

$$\bar{h}(w) = \lim_{n\to\infty} E[h(w,x)] \tag{85}$$

the *asymptotic stability theorem* states that

$$\lim_{n\to\infty} w(n) \to q_0, \quad \text{infinitely often with probability 1} \tag{86}$$

where q_0 denotes the solution to the ODE mentioned above. When, for convenience. $w(t)$ in (84) is expanded into the complete orthonormal set of eigenvectors of the correlation matrix **R** of the stochastic process represented by the input vector $x(n)$

$$w(t) = \sum_{k=0}^{p-1} \theta_k(t) q_k \tag{87}$$

where

- q_k is the kth normalized eigenvector of **R**
- $\theta_k(t)$ is the (time–varying) projection of $w(t)$ onto q_k

it can be shown that $\lim_{t\to\infty} w(t) \to q_0$, where q_0 is the normalized eigenvector associated with the largest eigenvalue λ_0 of **R**. Since all six conditions of the asymptotic stability algorithm are then satisfied, the stochastic approximation algorithm of (81) will cause $w(n)$ to converge with probability 1 to the eigenvector q_0 associated with the largest eigenvalue λ_0 of **R**. This is, however, not the only fixed point of the algorithm, but it is the only one that is asymptotically stable.

6.2.4.2 More linear neurons.

At present, we are aiming at a generalization of this learning rule, in order to produce a network that performs a PCA of arbitrary size on the input vector.

Two assumptions with respect to the network's structure are made:

(1) each neuron in the output layer of the network is linear

(2) the network has p inputs and m outputs, both of which are specified. Moreover, the network has fewer outputs than inputs (so $m < p$)

The last assumption in fact induces a dimensionality *reduction* of the input data, which is being performed when the input data is transformed *linearly* to the outputs, ascertained by the first assumption.

In this algorithm, the only part of the network that is subject to training is the set of synaptic weights $\{w_{ji}\}$ connecting source nodes i in the input layer to computation nodes j in the output layer, where $i = 0, 1, \ldots, p-1$, and $j = 0, 1, \ldots, m-1$.

We now give an overview of the algorithm [33]; some comments on its meaning and practical application will be made afterwards.

Algorithm: Generalized Hebbian Algorithm (GHA)

(1) **initialize** the synaptic weights of the network, w_{ji}, to small random values at time $n = 1$. **assign** a small positive value to the learning–rate parameter η.

(2) **for** $n = 1$, $j = 0, 1, \ldots, m - 1$, and $i = 0, 1, \ldots, p - 1$, compute

$$y_j(n) = \sum_{i=0}^{p-1} w_{ji}n)\ x_i(n) \tag{88}$$

$$\Delta w_{ji}(n) = \eta\left[y_j(n)\, x_i(n) - y_j(n) \sum_{k=0}^{j} w_{ki}(n)\, y_k(n) \right] \tag{89}$$

where

- $x_i(n)$ is the ith principal component of the p x 1 input vector **x**(n)
- m is the desired number of principal components

(3) **increment** n by 1, **goto** step (2), and **repeat until** the synaptic weights w_{ji} reach their steady–state values, i.e. $\Delta w_{ji} \approx 0$.

(4) **if** n is large enough **then** the synaptic weights w_{ji} of neuron j converges to the ith component of the eigenvector associated with the jth eigenvalue of the correlation matrix of input vector **x**(n)

6.2.4.3 Practical considerations.

When putting the theory into practice, some additional comments must be made:

We must keep in mind, that this algorithm is intended to be performed on p input–signals of length N each, i.e. there are N different input vectors of length p. Training (organization) of the network will be done by consecutively feeding the same batch of train–vectors into the net. Since the analysis into components is done on each input vector separately, network organization doesn't depend on the interrelations between single vectors, only on the ensemble properties of the batch. Hence, we can randomize the order in which input vectors from the batch are feeded to the network (and prevent erroneous learning of interrelations due to the order in which the train–vectors are fed to the net).

Furthermore, note that step (4) in the algorithm states that, for n large enough, the network extracts the eigenvectors of the current input vector. When we are dealing with a batch of input vectors, i.e. **x**(n) varies with n, it becomes clear that, upon convergence of the network, the *same* eigenvectors are extracted from *all* different input vectors from the batch. The eigenvalues also converge to values that are *uniform* for all input vectors within a batch: the algorithm is designed in such a way, that the variance of the outputs (i.e. the respective eigenvalues) decreases with m, where m denotes the id of the output–neuron.

The correct operation of the algorithm can be clarified by adopting a neuron–to–neuron approach, performed in the next subsection. We must keep in mind, that in practice, all neurons in a net trained by the generalized Hebbian algorithm tend to converge together; it is, however, un-

likely that the second neuron converges correctly until the first neuron is at least part way toward the first eigenvector.

Finally, the choice of the learning–rate η is an issue of importance. In step (1), a "small positive value" is proposed for η, which is an approximation to condition (3) on the time–varying learning–rate parameter $\eta(n)$ used in the convergence analysis above. Its exact size depends on the application of interest.

Extraction of principal components by deflation.

We will now show [33] that the algorithm just given indeed computes the principal components of an input vector **x**(n). A new variable, $x_i'n)$, is introduced, which equals

$$x_i'(n) = x_i(n) - \sum_{k=0}^{j-1} w_{ki}(n)\, y_k(n) \tag{90}$$

When we substitute (90) into (89), we get

$$\Delta w_{ji}(n) = \eta y_j(n)\Big[x'_i(n) - w_{ji}(n) y_j(n)\Big] \tag{91}$$

and replacing the term between squared brackets with $x''_i(n)$, we recognize the familiar form of a Hebbian learning rule.

Now we consider the operation of the algorithm. In that case, we are dealing with an input vector $x(n)$, and (91) can then be written as

$$\Delta w_j(n) = \eta y_j(n) x'(n) - \eta y_j^2(n) w_j(n), \quad j = 0, 1, \ldots, m-1 \tag{92}$$

where the p–dimensional equivalent of $x_i'n)$ is denoted by

$$x'(n) = x(n) - \sum_{k=0}^{j-1} w_k(n)\, y_k(n) \tag{93}$$

This new vector constitutes in fact a modified form of the input vector.

When the algorithm is operating, we know from Lemma 1. that the first neuron of the net mentioned above will discover the first principal component of input vector $x(n)$. This corresponds to the situation where the input vector is not yet modified, i.e. $x'(n) = x(n)$ or, equivalently, to $j = 0$.

Then we look at the second neuron, i.e. $j = 1$, and note that the algorithm modifies the input vector w.r.t. this second neuron to $x'(n) = x(n) - w_0(n)y_0(n)$. Since the subtracted term corresponds to the first term of the synthesis formula (69) (i.e. the eigenvector $w_0(n)$ associated with the largest eigenvalue multiplied by the first principal component $y_0(n)$), application of the same procedure (extraction of the eigenvector associated with the largest eigenvalue of correlation matrix **R** of input vector $x(n)$) to this modified input vector $x'(n)$ leads to extraction of the second principal component of the original input vector $x(n)$ at the second neuron. Proceeding, it is clear that the outputs of the resulting GHA–trained net correspond to the projections of the input vector on the respective eigenvectors of **R**, and that the individual outputs are ordered by decreasing eigenvalue (i.e. the variance of the projections/ outputs, remind equations (46) and (67)).

Chapter 7.
Experimental setup

In Section 4.2.2.1, we have already formulated a hypothesis about network architecture and signal characteristics. We would like to investigate using test data with correlation time and frequency ratio known a priori. Moreover, we want to benchmark EMG predictions with a synthetic version of it. Consequently, the procedures to construct the data will be described.

7.1 Simulation environment.

The experiments, in which the hypothesis from Section 4.2.2.1 is investigated, and the self–organized PCA from Section 6.2, and (neural) predictors of several orders were constructed, couldn't be performed without a proper simulation environment. We used the neural network simulator

- ***InterAct*** by *NeuroTechnology Gmbh*

for neural network experiments. The software was written in the C programming language, using a UNIX platform. Data processing and analysis was done using the MatLab software package.

7.2 Preprocessing and data analysis.

First of all, we have to choose a segment of data which we will use for our prediction. Based on this segment, a signal mimicking its shape will be constructed. A number of choices, concerning the EMG signal for prediction, have to be made:

- the segment length; the segment has to be long enough to yield the necessary signal characteristics, but also short enough to enable generalization
- raw or filtered data; often the EMG is analyzed by looking at the envelope (when the signal is full–wave rectified and low–pass filtered).
- ensemble/ coherent averaged or not; we already mentioned, that the inter– and intrasubject–variability of the EMG is rather large[13]: "the parameters are in general more constant during a follow–up investigation of one subject than between different subjects"). We could cast some doubt on the usefulness of a predictor that is tuned on a particu-

lar segment of EMG of a particular person on a particular measurement.

- normalized to the walking cycle or not; often, in gait analysis, signals are drawn with respect to the phase in the walking cycle. In our case, we could normalize the EMG with respect to the ground reaction forces, which are good indicators of the begin and end of the stance phase. Together with coherent averaging, this would lead to a normal pattern for the EMG as a function of the walking cycle of a particular person at a particular moment in time. Note: it can be expected, that addition of movement data to the network will have a smaller effect when we are predicting the normalized ("windowed") EMG compared to the real time series.

Since normalization to a walking cycle and coherent averaging of instantaneous data is not feasible for on–line utilization, we will ignore the latter issues, and focus on segment length and processing. Since the EMG is expected to be (somewhat) periodic in the walking cycle, one full step would, at least in principle, be adequate to train with. For safety, we choose a segment containing 2 to 3 steps, at a random starting point (Figure 15.).

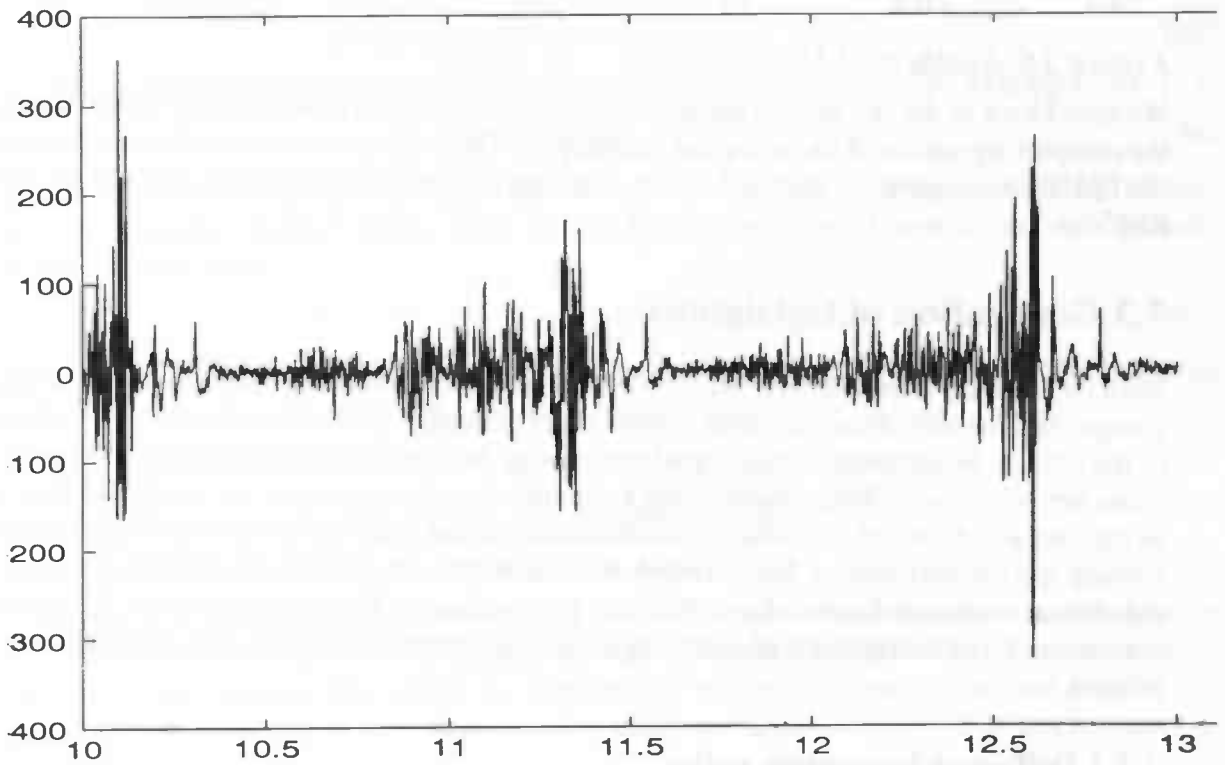

Figure 15. Segment of raw EMG from the right Tibialis Anterior

The third step has to be predicted on basis of the preceding data, i.e. the first 2000 samples may be used for training, whereas the last 1000 samples are not touched upon: they are used for prediction verification solely. We will just deal with the raw EMG and its synthetic counterpart (which is already tailored to a smoothed version of the raw EMG).

7.2.1 Synthetic EMG.

Because of the complexity of the raw EMG–signal, we chose to do experiments with a synthetic signal, chosen in such a way that it resembles the low–pass filtered EMG well (Figure 16.).

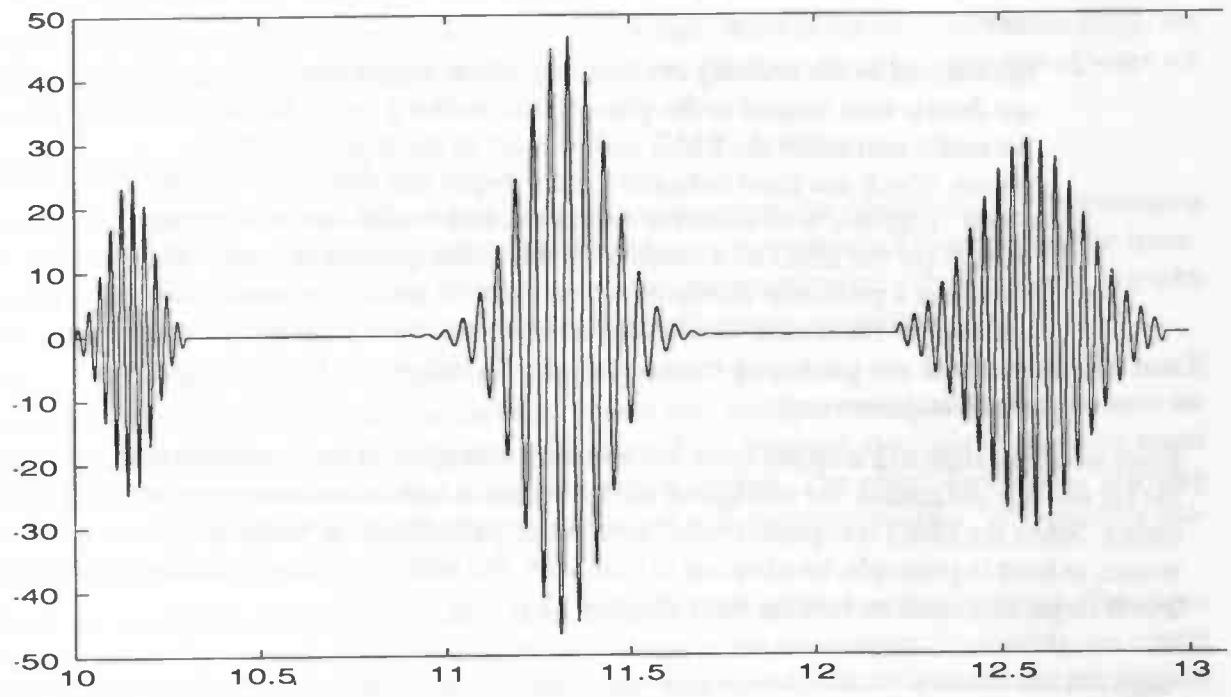

Figure 16. Synthetic EMG

We could look at the last testing signal as a synthetic smoothed EMG–signal, which is lacking the complex dynamics of its biological counterpart. This way, we can assess the contribution of the EMG's basic shape to the final results obtained in predicting the actual raw and smoothed EMG.

7.3 Generation of testsignals.

With the solution of problems in which a neural network can be expected to be a useful system component, we will often encounter complex and difficult sigals to deal with. Possible failures of the NN can be attributed to this signal complexity, besides the obvious artefacts due to inadequate network design. When maintaining a *structured design* approach, we would like to be able to indicate which part of the failure is attributable to signal complexity and which part to preprocessing and network design. Furthermore, we want to gain insight into the relation between signal characteristics and network architecture/ preprocessing. We can observe the NN's performance on a set of testsignals with predefined (suitable) characteristics, in order to investigate this relation.

7.3.1 Different frequency ratios.

The left part of inequality (12) indicates the need for signals showing varying maximum to minimum signal frequency ratio. Hence, the first set of signals was constructed by adding two sines of possible frequencies (in [Hz]) {1, 5, 10, 25, 30, 50 }, leading to signals with varying $f_{max} - f_{min}$ –ratio.

7.3.2 Correlation time and network architecture.

The number of lags during which a signal is autocorrelated is expected to be an upper bound for the number of time delays the TDL in front of an MLP has to possess in order to be able to learn the signal characteristics (and generalize it adequately). Thus, to verify this hypothesis, we are

looking for signals that exhibit predefined autocorrelation. In the sequel, we propose a method for the generation of surrogate data with predefined autocorrelation content.

The second set of surrogate data was constructed out of white, Gaussian noise using a suitable FIR–filter, i.e. a filter, which repeatedly applied in fact represents a new filter displaying Gaussian autocorrelation. Note the following: by varying the σ of a Gaussian and sampling the resulting function, we obtain sequences with varying support. One extreme is a level function, a discrete Gaussian with infinite support, constructed by letting σ run to infinity. This function just happens to be the autocorrelation–function of another level function. The other extreme is the Dirac–delta function, defined by

$$\delta(t) = \begin{cases} \infty, & t = 0 \\ 0, & t \neq 0 \end{cases} \tag{94}$$

which evolves when σ approaches zero. Again, this function happens to be its own autocorrelation function.

7.3.2.1 Heuristical motivation of the above procedure.

The following reasoning underlies the procedure just mentioned:

Consider the first–order moving average scheme

$$y(n) = \epsilon(n) + \lambda\epsilon(n-1) \tag{95}$$

which can also be seen as the convolution of white Gaussian noise $\epsilon(n)$ with a kernel $h(n) = [1, \lambda]$

$$y(n) = h(n) * \epsilon(n) \tag{96}$$

The autocorrelation function of $y(n)$ has the values $\{1 + \lambda^2, \lambda, 0, 0, ...\}$ for lags zero and up. Choosing $\lambda = 1$ yields a kernel which, when repeatedly applied, gives a new kernel with Pascal–triangle coefficients:

$$[1\ 1] * [1\ 1] = [1\ 2\ 1], \quad [1\ 2\ 1] * [1\ 1] = [1\ 3\ 3\ 1], \ \textit{etc.} \tag{97}$$

Since a row in the Pascal–triangle exhibits a bell–shape, and converges to a Gaussian for large row length, and since the autocorrelation $\divideontimes$ of $y(n)$ equals

$$\{h(n) * \epsilon(n)\} \divideontimes \{h(n) * \epsilon(n)\} = \{h(n) * \epsilon(n)\} * \{h_{rev}(n) * \epsilon_{rev}(n)\} \tag{98}$$

which, since $\epsilon(n) * \epsilon_{rev}(n) = \epsilon(n) * \epsilon(n) = \sigma_\epsilon = 1$, also equals

$$\{h(n) * h_{rev}(n)\} = \{h(n) * h(n)\} \tag{99}$$

it follows, that we can generate test–signals $y(n)$ with approximate Gaussian autocorrelation functions by convolving white Gaussian noise with a "Pascal–like" FIR–filter. The more coefficients the filter contains, the greater the support (the number of elements unequal zero) of the autocorrelation function of $y(n)$. More detailed treatments about this procedure can be found in [32] and [49].

7.3.2.2 Choice of autocorrelation test signals.

We constructed two sets of test signals for investigation of the correlation time hypothesis:

I. three signals having correlation time 3, 5 and 7, respectively that were constructed using white noise of 40.000 samples

II. three signals having correlation time 5, constructed using white noise of length 512, 1024 and 2048 samples, respectively

The orders were kept low, to avoid large networks (many inputs ...) leading to slow convergence.

7.3.3 Validation.

Of course, we have to validate the results we will obtain: what is the value of our findings, are they persistent, do they make sense, are they useful, etc. Here we describe a number of ways we can do this.

7.3.3.1 Performance criteria.

In our quest for the best network architecture, a suitable means for comparing the performance of different structures in predicting the EMG–signal has to be established. Since we will be learning in a supervised manner (during training we always have the correct values at hand), we take the error between actual and predicted values as a performance measure.

Next, an important issue is whether we update ("learn") in pattern– or batch mode. In batch mode, the individual errors may (somewhat) cancel out, giving an acceptable performance on the average, while large individual errors still occur. This, however, does not necessarily mean a worse learning performance over a large number of epochs. The effect depends on learning rule. Since we use standard error backpropagation, we will, referring to [Hay, 199] practice pattern mode updating.

Then we would like to define a suitable error measure for a whole batch of patterns. The exact form of the definition will certainly influence the shape of the error curve. Preferably, we would like to see a *smoothly decreasing* error curve evolve: this would indicate smooth and predictable learning. Furthermore, a similar looking curve for a number of network structures would make a comparison easier, so we look for a "*uniform*" curve shape. Also, we would like to see a *repeatable* error curve appear, i.e. when we repeat the experiment with the same network, a similar curve should emerge.

We propose four different error measures:

$$e_j = \hat{\mu}_j \tag{100}$$

$$e_j = \underset{i=1}{\overset{nr\_patterns}{MAX}} |d_i - o_i| \tag{101}$$

$$e_j = \frac{1}{nr\_patterns} \sqrt{\sum_{i=1}^{nr\_patterns} (d_i - o_i)^2} \tag{102}$$

$$e_j = 3\hat{\sigma}_j \tag{103}$$

where

$$\hat{\sigma}_j = \sqrt{\frac{1}{nr\_patterns - 1} \sum_{i=1}^{nr\_patterns} \left[(d_i - o_i) - \hat{\mu}_j\right]^2} \tag{104}$$

$$\hat{\mu}_j = \left[\sum_{i=1}^{nr\_patterns} (d_i - o_i)\right] / nr\_patterns \tag{105}$$

and *nr_patterns* = the number of different patterns in a batch.

Notice that the particular choice of error measure is also dependent on the problem at hand, e.g. for critical applications, where performance must be guaranteed at any instant (outliers are for-

bidden), a max–error is the most appropriate. We will check all four errors for their suitability in tracking the learning process.

Another performance criterion that seems useful is the *learning speed*, which we can express by the number of cycles during which the network has to be trained before the error has attained a certain level ϵ. A suitable value of ϵ has to be determined experimentally.

After having determined a suitable neural predictor of a synthetic EMG–like signal, we have to look at the value of these results in the context of a future artificial walking system. Naturally, a neural predictor will have to deal with real (invoked) EMG from the paraplegic's limbs.

7.3.3.2 Prediction with classical techniques.

A convenient way of evaluating the results of our neural predictors is by comparing them with the results that can be obtained with classical prediction techniques, such as linear prediction on basis of the partial autocorrelations of the signal. This techniques have been described Section 5.3. Once we have discovered an acceptable 1–lag predictor, we can make it generate a full signal by recursively predicting 1 step ahead, and feeding back the prediction, moving the time window, etc. Initially, the prediction is performed recursively on the synthetic EMG–approximations. We compare the predicted signal with the actual approximation, hence gaining insight into the prediction capabilities over a longer period of time or, in other words, the predictor's *generalization performance*. We will also compare the results with those obtained from recursive prediction with the best 1–lag conventional predictor.

Chapter 8.
Experiments & results

Now that we have attained a thorough understanding of the actual problem, meaning and interrelations of the data, and various (neural network based and classical) techniques for prediction, data characterization and dimensionality reduction, we have arrived at the point where we will investigate the assumptions concerning data characteristics and neural architecture, using the method for construction of surrogate data and the simulation environment described in the previous chapter. Based on the results obtained with preprocessing and hypothesis testing, a starting point is provided for our final objective: prediction of the human EMG. Again, surrogate data is used for benchmarking purposes; to be able to validate the method properly, neural results are compared to results obtained with conventional methods for time series prediction.

8.1 Preprocessing the data.

Our attempts to predict the EMG start with an analyis of the characteristics of the available signals. First, we use some standard techniques, when we look at its temporal and spectral behavior. Also, we check the EMG for underlying determinism. When we have developed some understanding of the data, we proceed with a neural network implemented PCA, which aims at an efficient encoding of the available body signals, describing the walking cycle using their most important constituents; this could provide us both features for a neural solution and a more thorough understanding of the signal.

8.1.1 Signals for prediction.

We start with a presentation of the signals we used in our prediction experiments. First of all, an EMG–signal was selected. We already mentioned in Chapter 4 that we chose the EMG–signal that is caused by a contraction of the right human Tibialis Anterior muscle. Because its functionality in the walking cycle is clear and, moreover, already succesful attempts have been made at FES–induced correction of walking pathologies (correction of *dropfoot* by FES–induced contraction of the Tibialis Anterior muscle), this muscle seems a good starting point for predictive control within the framework of an artificial walking system. Out of a recording of 20 seconds of normal human walking (Figure 6.), we chose the following segment for prediction:

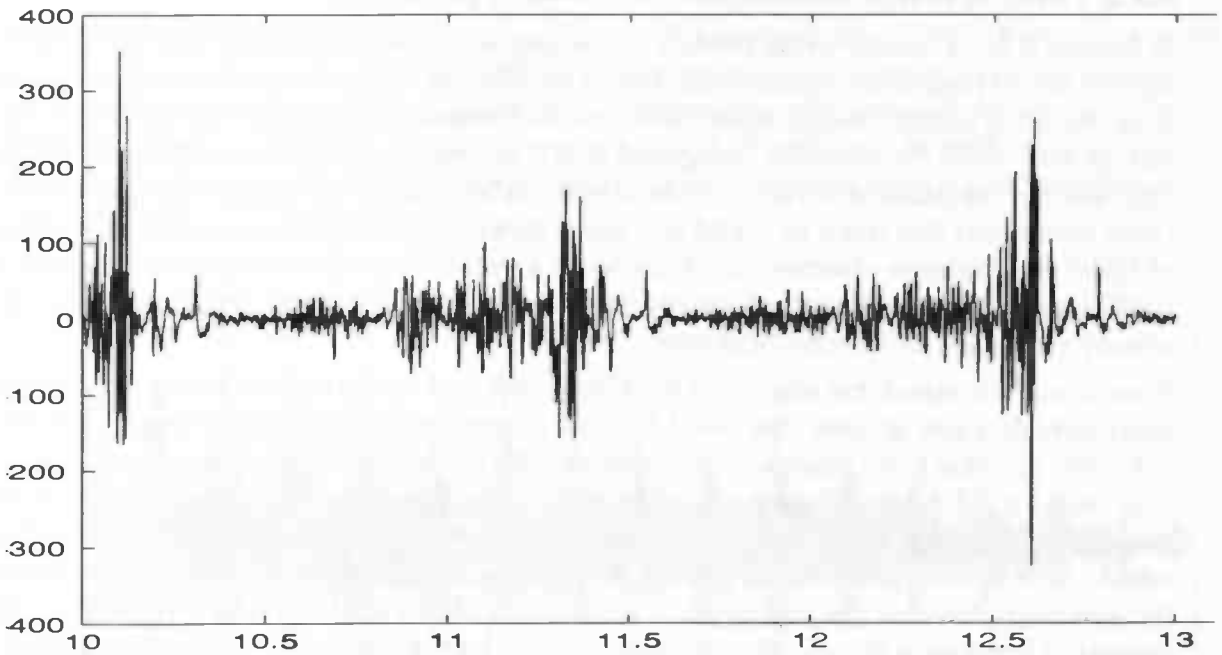

Figure 17. Segment of raw EMG from the right Tibialis Anterior

The data is sampled at a rate of approximately 1000 Hz, so the original recording consisted of some 20.000 samples. The step time of the subject under investigation was about 2 seconds per step, so the segment above (containing 3000 samples) represents 2.5 step. During sampling, some anti–aliasing low–pass filtering is probably performed by the recording apparatus, which might affect the estimation of the correlation dimension (see below).

For benchmarking purposes, a synthetic EMG (Figure 16.) was constructed, that resembles the above segment, but lacks complex or stochastic behaviour. It consists of three sines, that are multiplied by a Gaussian window, separated by zeros. It can be shown to fit the shape of the above segment fairly well.

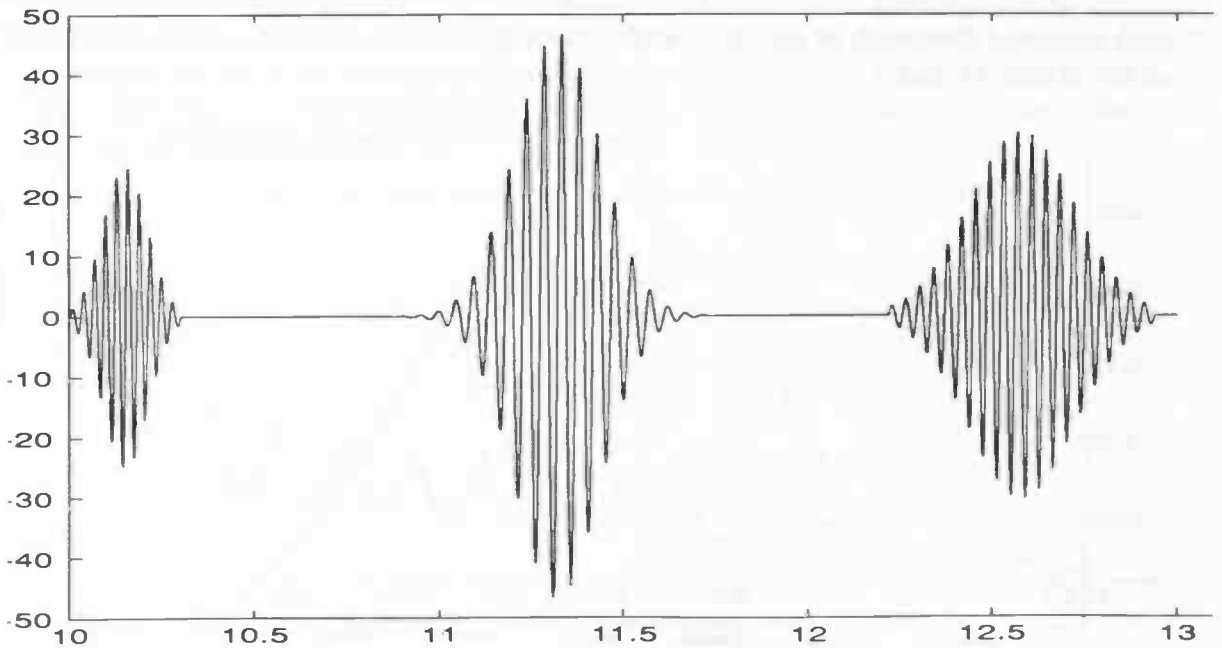

Figure 18. Synthetic EMG

8.1.2 Time, spectral and amplitude domain parameters.

In Section 3.2.3, a number of parameters for the quantification of the EMG is given. Of special interest are the parameters that are indicative of the walking cycle, that can be computed on–line from incoming measurements: these could serve as features for a neural predictor. It was stated, that the area under the curve (or: integrated EMG) is proportional to the exerted muscle force; equivalently, the standard deviation of the signal can be used for this purpose (Section 3.2.3.4). Other parameters that could be useful are: mean, maximum and minimum values of a segment of EMG. The ultimate objective is, of course, to construct a predictor that is able to track the EMG very well. For practical utilization, reliable prediction of the area under the curve would already yield a useful system component.

Since it was also stated that analysis of the EMG is not very useful without taking the associated movement data into account (Section 3.3.1), we expect the incorporation of angles and forces to be very valuable in the construction of a neural EMG–predictor. To be more specific: the periodic character of the walking cycle is reflected in these signals much more directly, compared to the EMG. The signals could give the network a hint about the position in the walking phase, which could be very useful in the timing of activity bursts and silent periods. We will look for the movement features that contain most information of this kind: we want to avoid too many degrees of freedom in the resulting net, since this can lead to bad generalization (Section 6.1.1), hence to bad prediction.

8.1.2.1 Time domain: autocorrelating the EMG.

First, we want to look at the temporal properties of the EMG, and its temporal relation to movement data. Also, an estimate of the correlation time can be obtained from the autocorrelation of a signal by looking at the lag at which the first zero–crossing occurs. This could provide an indication of the size of the tapped delay–line in front of the neural predictor for a specific signal. We verified this claim by looking at plots of the ACF from the surrogate data of Section 7.3.2, where the correct correlation time is known beforehand. In all cases, the procedure turned out to be right.

A snapshot of the autocorrelation function of the EMG segment under investigation is shown in Figure 19. In fact, a 'biased' ACF (it is not normalized to 1 at lag 0) is shown, but this does not matter when inspecting its shape and zero–crossings. Since the autocorrelation function is symmetric around the length of the signal under investigation, the correlation time of the segment of raw EMG is about 3.

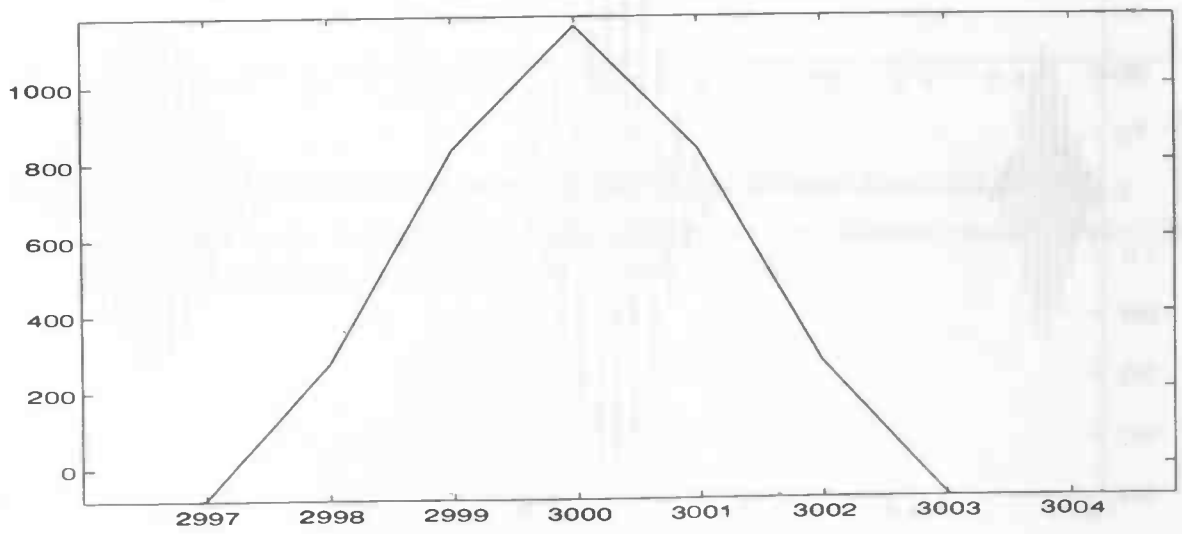

Figure 19. EMG segment correlation time

Similarly, the synthetic EMG shows a correlation time of approximately 11.

The full (biased) ACF plot for the original (20 seconds) raw segment of EMG (Figure 20.) seems to indicate a random character of the EMG. This can be seen, when keeping in mind that the ACF of white noise is a Dirac delta–function [49]. Furthermore, with practical random waveforms, like white Gaussian noise in an electrical system, the ACF will have its peak value at zero lag and will reduce to a random fluctuation of small magnitude about zero for lags greater than about unity [31].

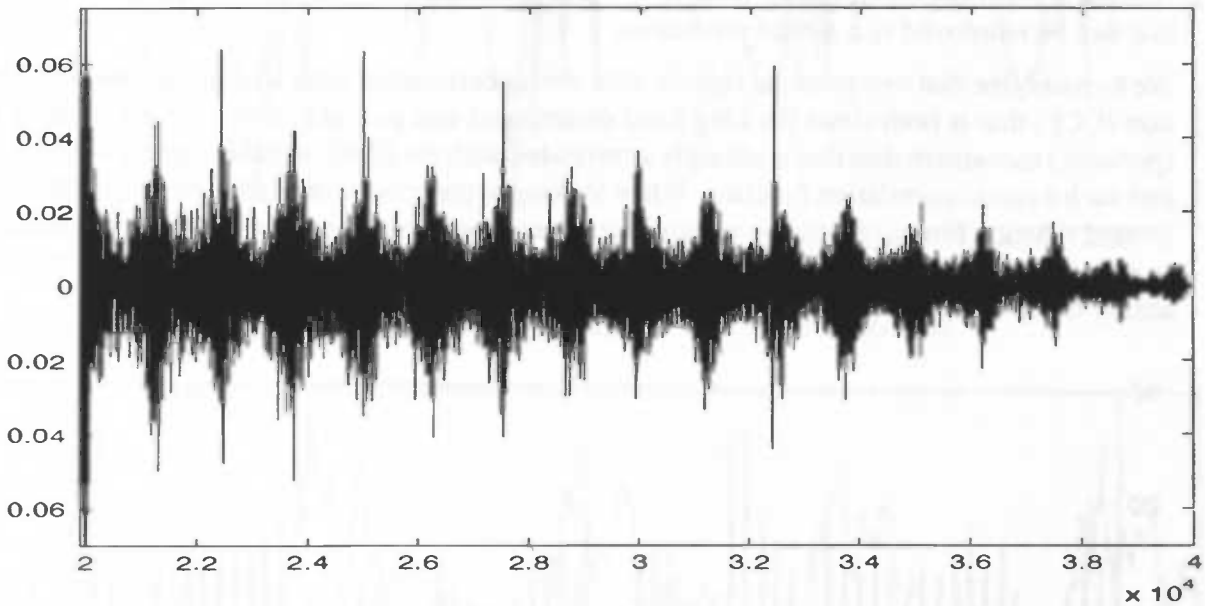

Figure 20. Autocorrelation function raw EMG (right Tibialis Anterior)

Looking more closely into the ACF, however, a periodic character of the signal can also be observed (remember that the ACF of a periodic signal is also periodic [31]). In Section 3.2.3.3 it was already reported that the EMG has stochastic appearance, while it can be seen to be constituted from deterministic components at the same time. Surely, a more detailed investigation of the deterministic or random character of the EMG signal seems in order.

The hypothesized existence of high–frequency noise in the EMG motivates low–pass filtering of the signal as one of the fundamental processing steps (Section 3.2.3.2). Moreover, the EMG is often full–wave rectified before further processing. The autocorrelation function of the modulus of the raw EMG looks like:

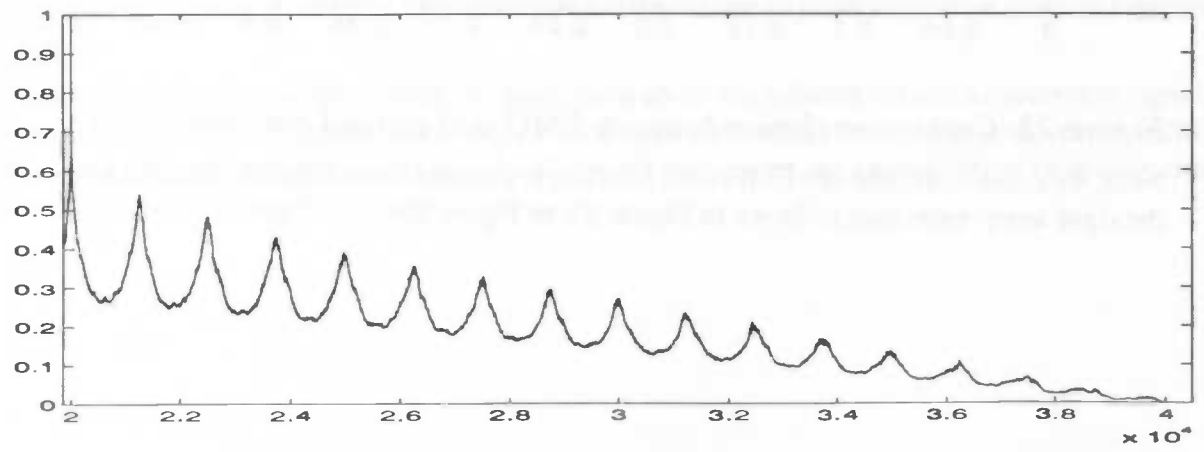

Figure 21. Full–wave rectified EMG autocorrelation function

An offset can be observed: because of the modulus–operation, the mean value of the signal now exceeds zero. Note also the striking cleansity of the figure, rectifying seems to smoothen the autocorrelation function of the signal, which is probably the purpose of this step.

8.1.2.2 Time domain: cross–correlating EMG and movement data.

Inspired by Section 3.3, we will relate EMG to associated movement data. A straightforward way to do this, is by looking at the cross–correlations of muscle and movement signals. Later on, we will use a Principal Components Analysis to determine the most important movement features that can be employed in a neural prediction.

We hypothesize that two periodic signals with strong correlation have a cross–correlation function (CCF) that is both clean (lacking local distortions) and periodic. Since we are looking for (periodic) movement data that is strongly interrelated with the EMG, suitable signals would yield just such a cross–correlation function. When looking at the cross–correlation between EMG and ground reaction forces (aggregate values over all sensors from a foot) of the right lower extremity, we see a wildly fluctuating, nonstationary function appear (Figure 22.).We question the suitability of force data to serve as feature within a neural predictor of the EMG.

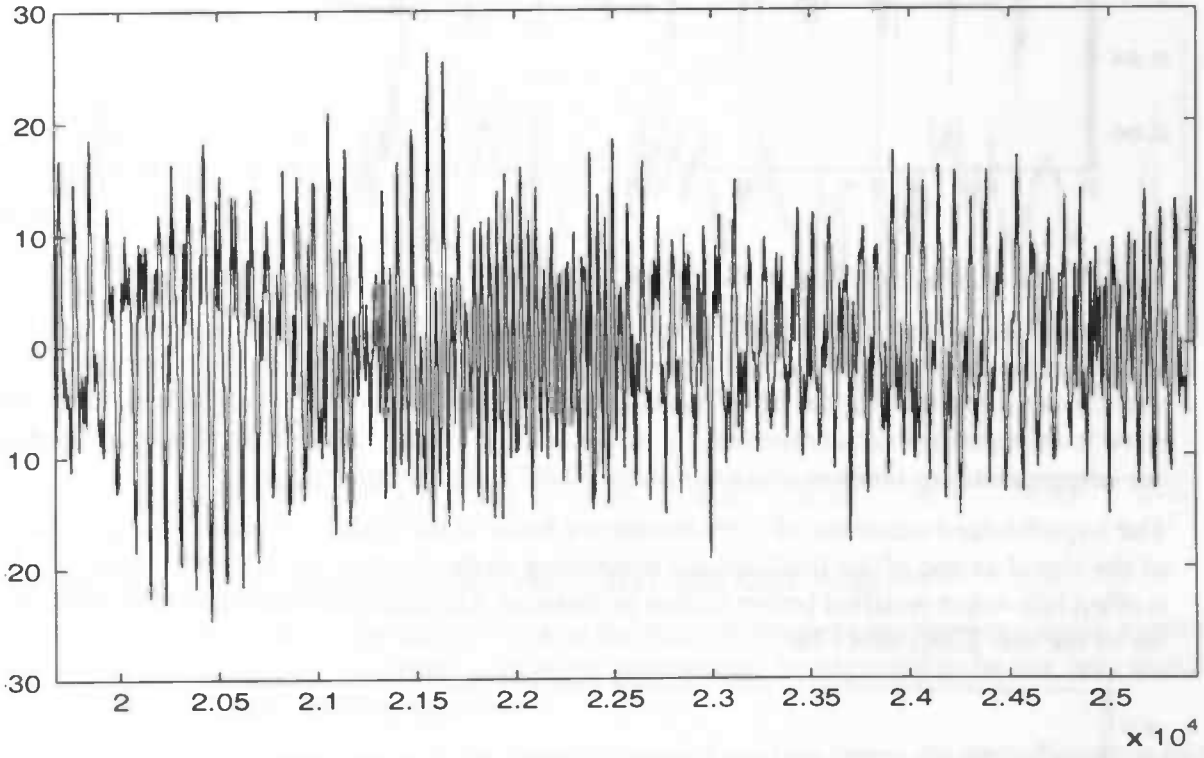

Figure 22. Cross–correlation between EMG and ground reaction forces

Next, we cross–correlate the EMG with joint angle patterns from different rotation angles (from the right lower extremity), shown in Figure 23. to Figure 26.

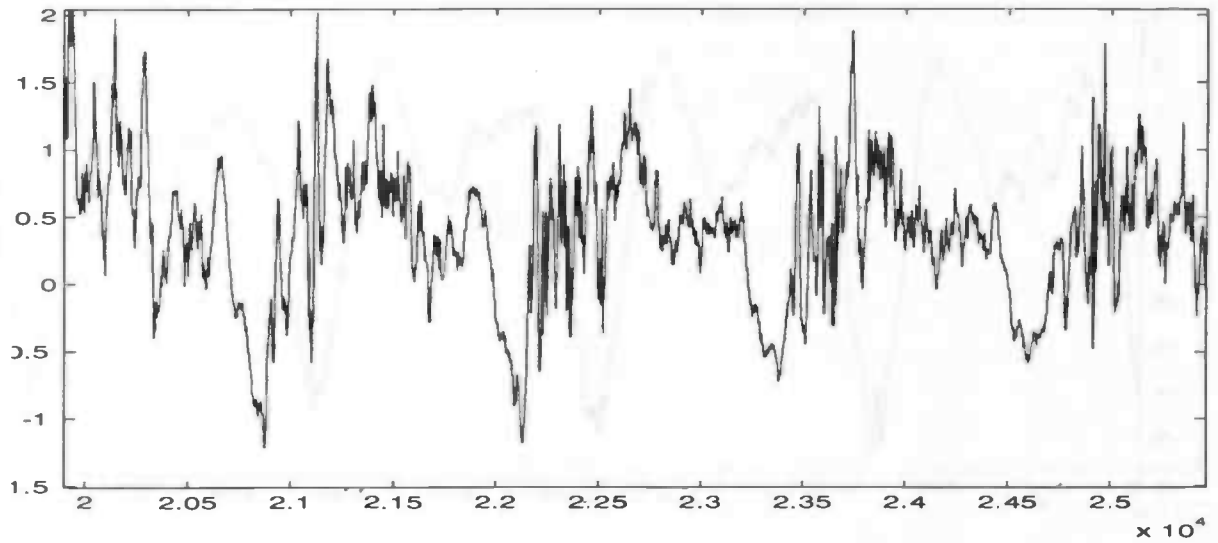

Figure 23. Cross–correlation between EMG and hip abduction

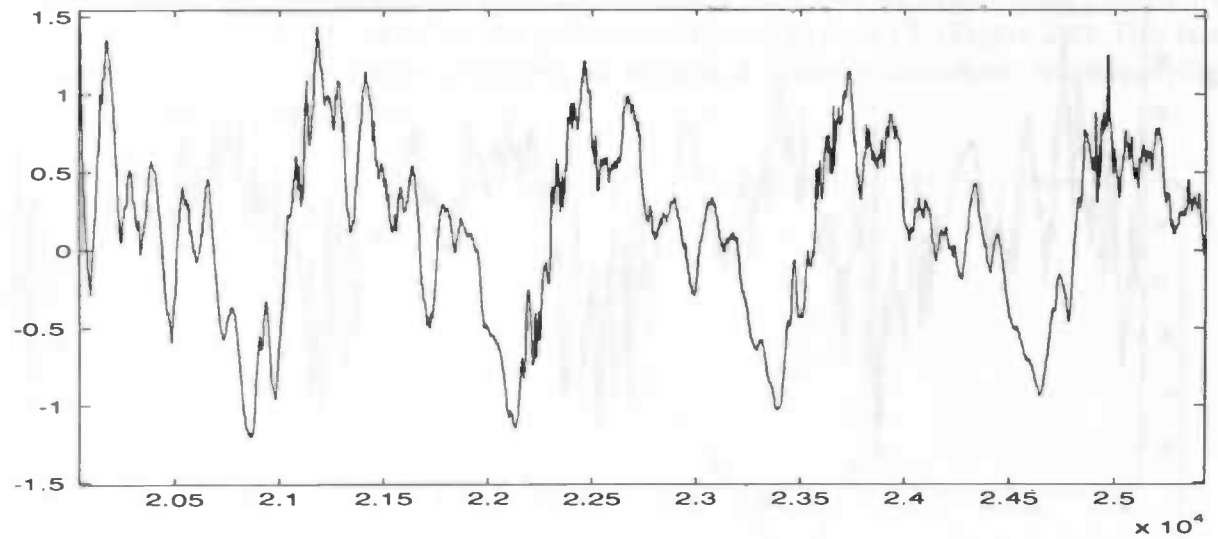

Figure 24. Cross–correlation between EMG and hip extension

It becomes clear that only the joint excursions in the sagittal plane (i.e. hip extension and knee flexion) are correlated with the EMG in a manner that seems to justify its use as a neural network feature.

Note that this is in accordance with the statements about the relation between movement signals made in Section 3.3.2 (hip and knee angles show most variability; the hip has a major role wihtin human walking; since walking is mainly a forward movement, we expect this to be mainly true for the forward, or sagittal, plane).

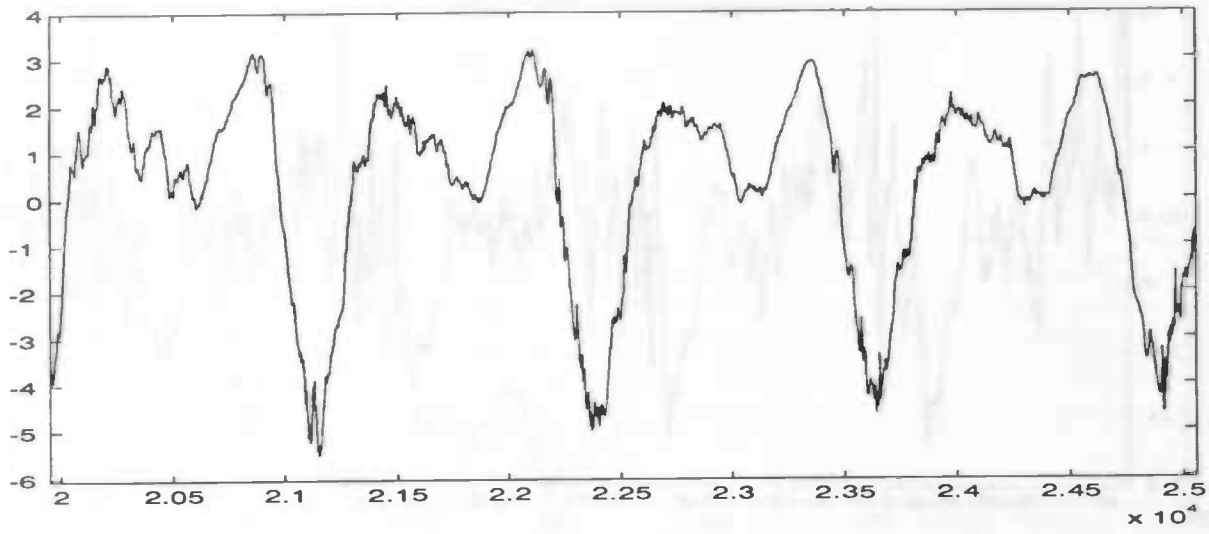

Figure 25. Cross–correlation between EMG and knee flexion

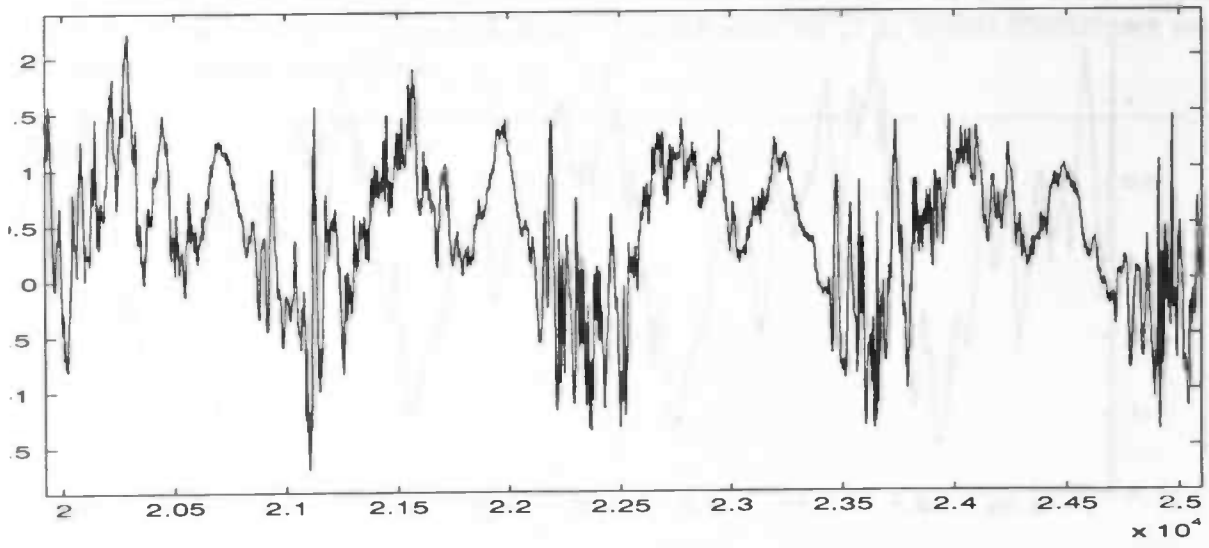

Figure 26. Cross–correlation between EMG and knee rotation

We want to verify the above by taking a look at the relations between movement signals: signals that are weakly related to important joint angle patterns and be expected to be not very useful as features for a neural EMG–predictor.

From Figure 27. it becomes once again clear that ground reaction forces do not indicate the gait cycle very well. At the lags shown from the cross–correlation function, several steps take place, which is not visible from the plot. Note that the CCF has negative values. This is not important, since its *magnitude* is actually indicative of the cross–correlation at a certain lag.

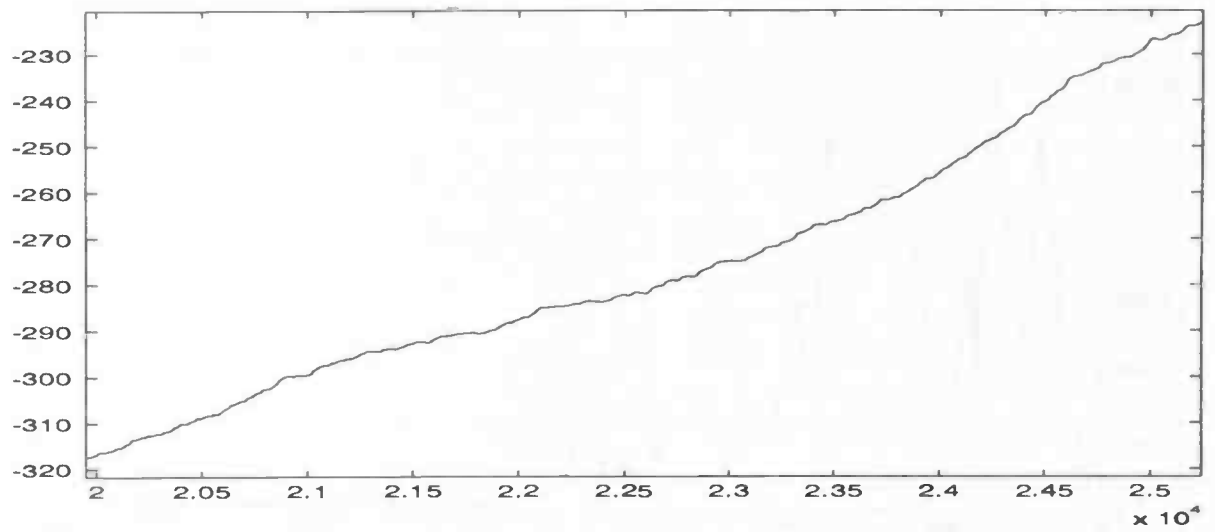

Figure 27. Cross–correlation between hip extension and ground reaction forces

When we repeat this for two strongly related movement signals, like we expect angle excursions in the sagittal plane to be, we do see the gait cycle reflected in the CCF (Figure 28.). This is a further motivation for our choice of features. We will check it later on once more, when applying a PCA to different joint angles.

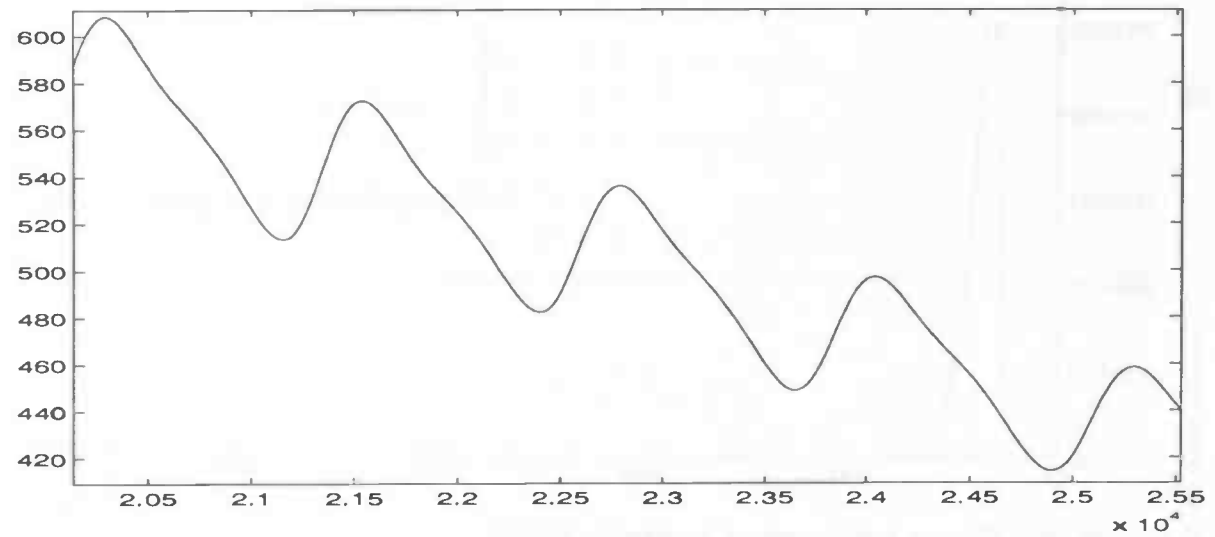

Figure 28. Cross–correlation between knee flexion and hip extension

8.1.2.3 Spectral domain: which (main) periodicities are present?

Remember the fact that autocorrelations and power spectrum of a signal are in fact equivalent [50]. It seems instructive to check the power spectra of the signals to be predicted in order to get a clear view of the important periodicities that are present.

Now, look at the spectrum (or, equivalently, the power spectral density) of the original segement (Figure 29.). In Section 3.2.3.2 it was stated that most of the EMG power is at the frequencies below 150 Hz. This seems indeed the case for the segment under investigation. Most of the energy is even below 80 Hz.

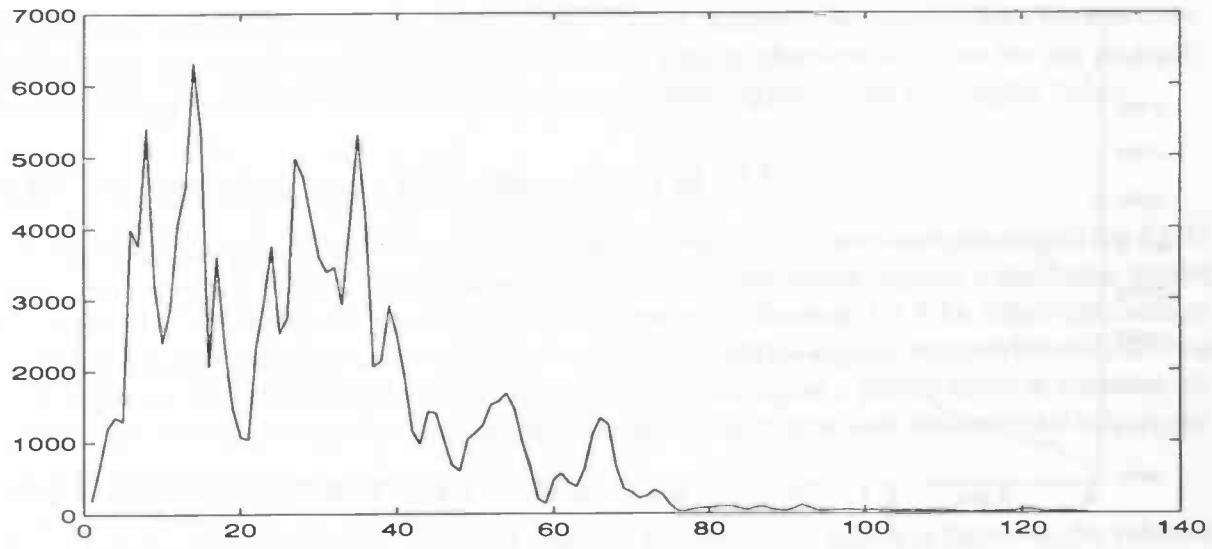

Figure 29. EMG segment power spectrum

Comparing this to the frequency content of the synthetic EMG (Figure 30.), the much simpler dynamics of the latter is obvious. It is, however, in the same range as its real counterpart, so it constitutes a suitable benchmark.

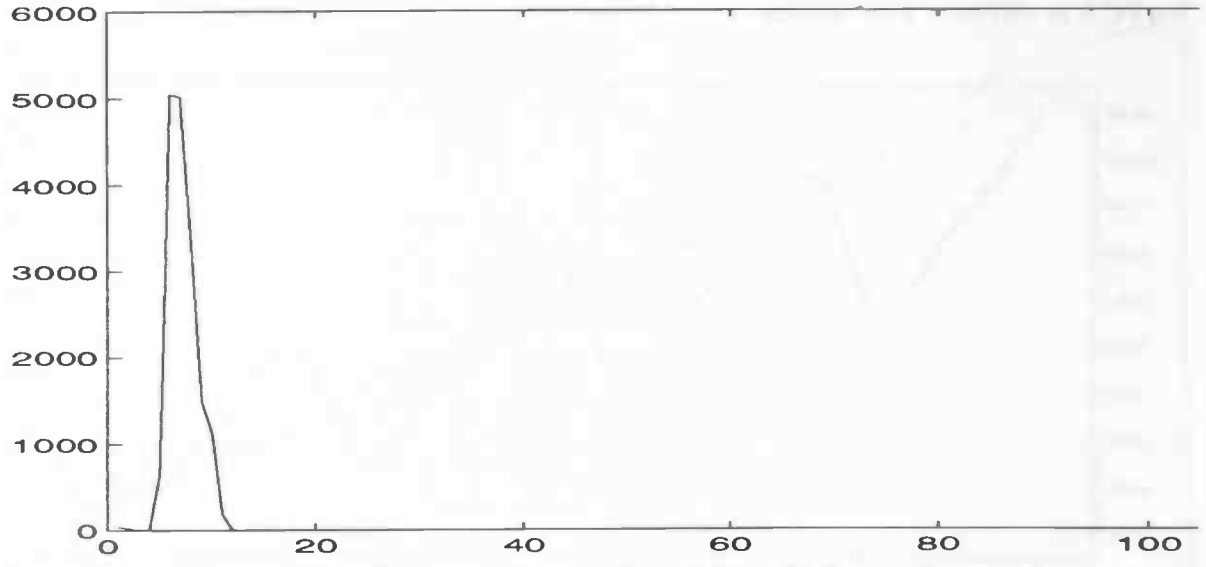

Figure 30. Power spectrum of synthetic EMG

8.1.2.4 Amplitude domain: is the theory right ???

Then we investigate the amplitude domain properties of the signal. Based on Section 3.2.3.4, we expect the amplitude histogram, i.e. the approximate amplitude distribution function of the signal, to be almost Gaussian shaped (a slight skew can indicate a small walking disorder, such as reported in the literature). When we plot it for the original 20 seconds record of EMG, we observe

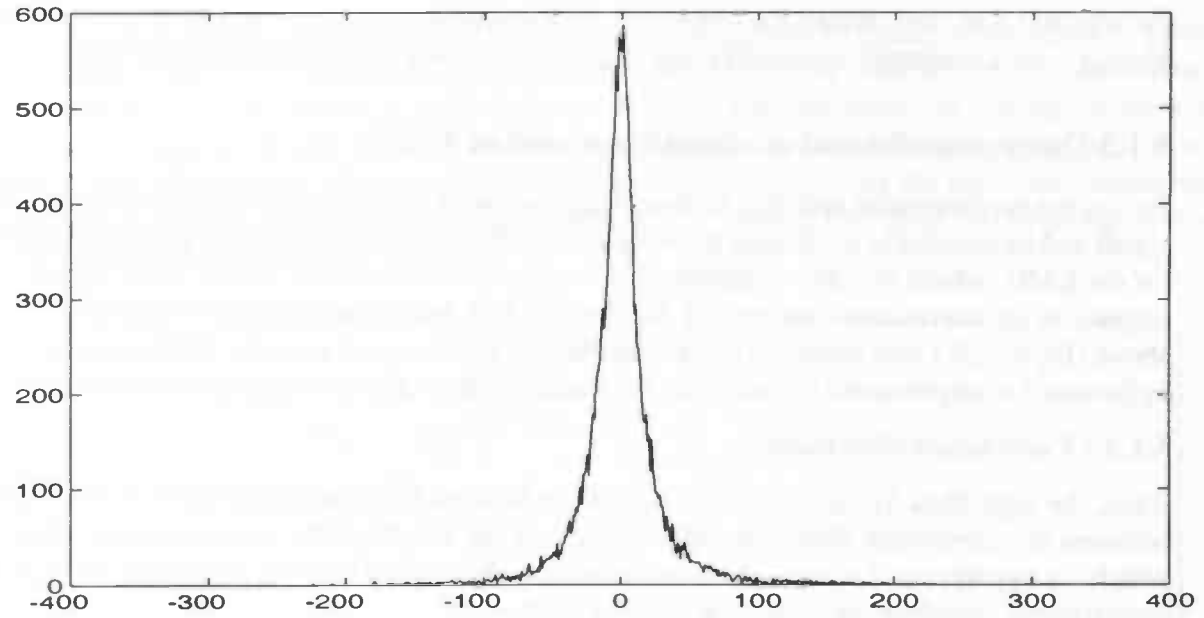

Figure 31. Amplitude histogram of full EMG record

which looks almost Gaussian, only slightly skewed to the right. In order to make the distribution more symmetric, we lead the signal into saturation below or above some level a using

$$thresh\,(y) = \begin{cases} y, & |y| \leq a \\ a, & y > a \\ -a, & y < -a \end{cases} \tag{106}$$

while choosing $a = 30\,\mu V$ (Figure 32.).

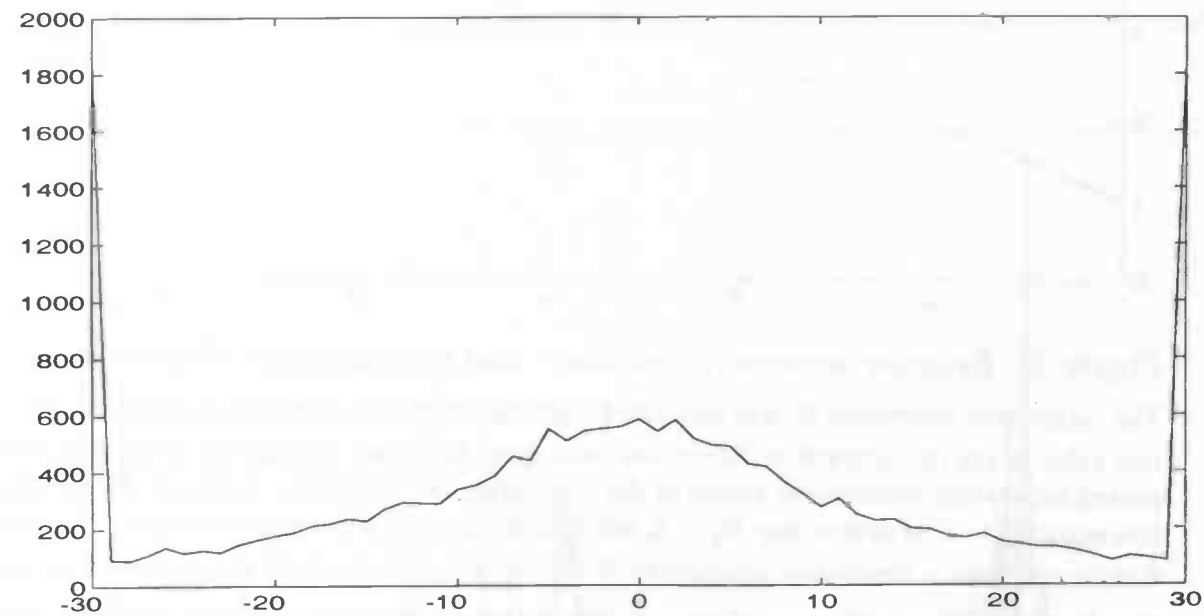

Figure 32. Thresholded amplitude histogram

When large amplitude magnitudes have been removed, a symmetric histogram occurs, just as we expect: high–frequency (Gaussian) noise is probably of small magnitude, and will now dominate the histogram. Again, a stochastic model of the EMG is suugested. Note also that the person

this EMG was obtained from may have a small walking disorder (Section 3.2.3.4). Furthermore, the segment that was chosen for prediction shows similar characteristics, so we are probably dealing with a stationary signal. This assumption is investigated more thoroughly below.

8.1.3 Deterministic and stochastic content of EMG,

During the preprocessing and data analysis stage, we want to gain an understanding of the EMG signal and its relation to associated movement data. Previous results suggest a stochastic model for the EMG, which was also established in the literature (Section 3.2.3.4). Other researchers emphasize its deterministic nature ([9]; Section 3.2.3.3), and the autocorrelation function shown above (Figure 20.) also indicates more determinism in the signal a purely random counterpart. In Section 5.4 we presented two methods for distinguishing random and deterministic behaviour.

8.1.3.1 Correlation dimension.

First, the algorithm *Noise_Or_Chaos* is given in Section 5.4.3, which is based on the relation between the correlation dimension of a signal and the length of the reconstruction vector, in which the hypothesized attractor is to be embedded. We applied it to the segment of EMG under investigation, yielding the following result (Figure 33.):

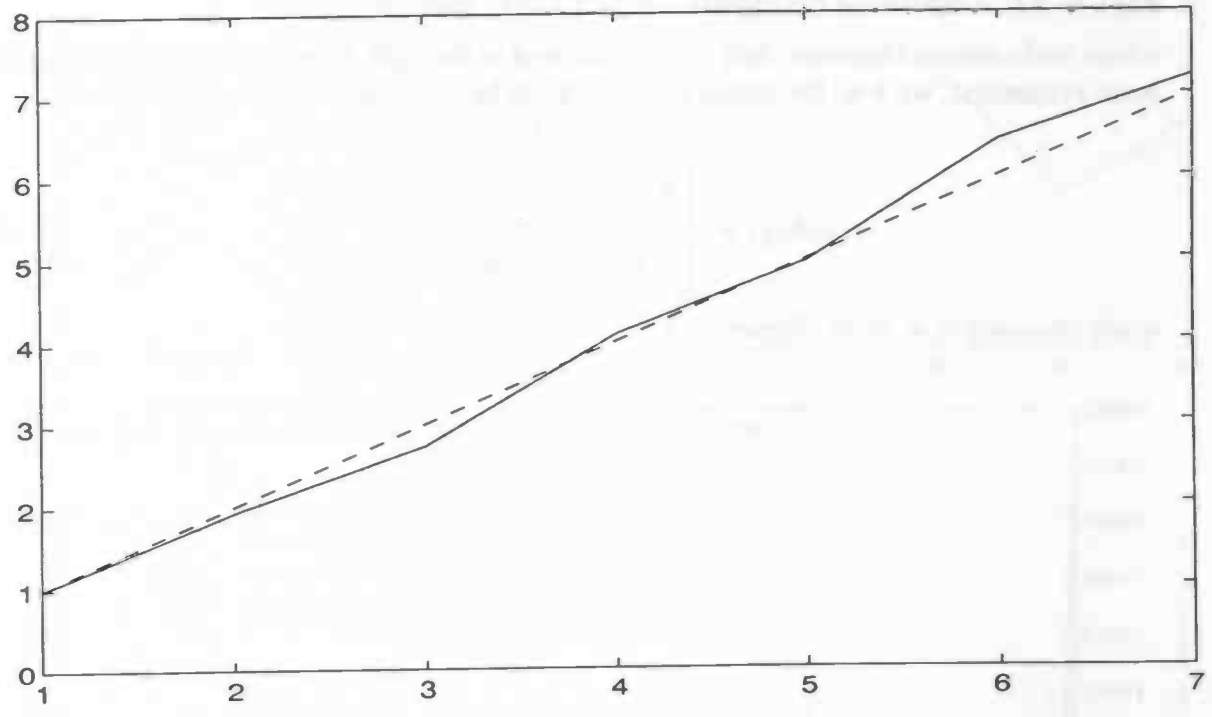

Figure 33. Relation between correlation – and reconstruction dimension

The correlation dimension D_2 was estimated from the correlation integrals associated with a certain value of the reconstruction dimension (analogous to Figure 14.) and the procedure was repeated for several consecutive values of the reconstruction dimension. In Figure 33. the dashed line equals $y = x$, or in this case $D_2 = k$, where k denotes the reconstruction dimension. We see that the correlation dimension increases uniformly with k, which is indicative of a random signal.

Note that anti–aliasing filtering of the signal by the acquisition apparatus may affect the dimension estimantion. This seems, however, not a serious problem, since we are not interested in the magnitude of the dimension, just in its dynamics w.r.t. the reconstruction dimension. It can be expected that a particular increase in the dimension estimation is nearly similar at all values of k.

8.1.3.2 Entropy.

Another phenomenon that can be used to decide whether a signal is chaotic or stochastic, is given in Section 5.4.4: when the entropy of a signal increases uniformly with decreasing distance between reconstruction vectors ϵ, without the existence of a region where the average number of prolongings (hence also the entropy) will be nearly constant, we are probably dealing with a random signal. The entropy of the segment of EMG was estimated using the algorithm Maximum Likelihood Entropy Estimation for several values of ϵ, leading to Figure 34.. Once again, the EMG turns out to be a random signal.

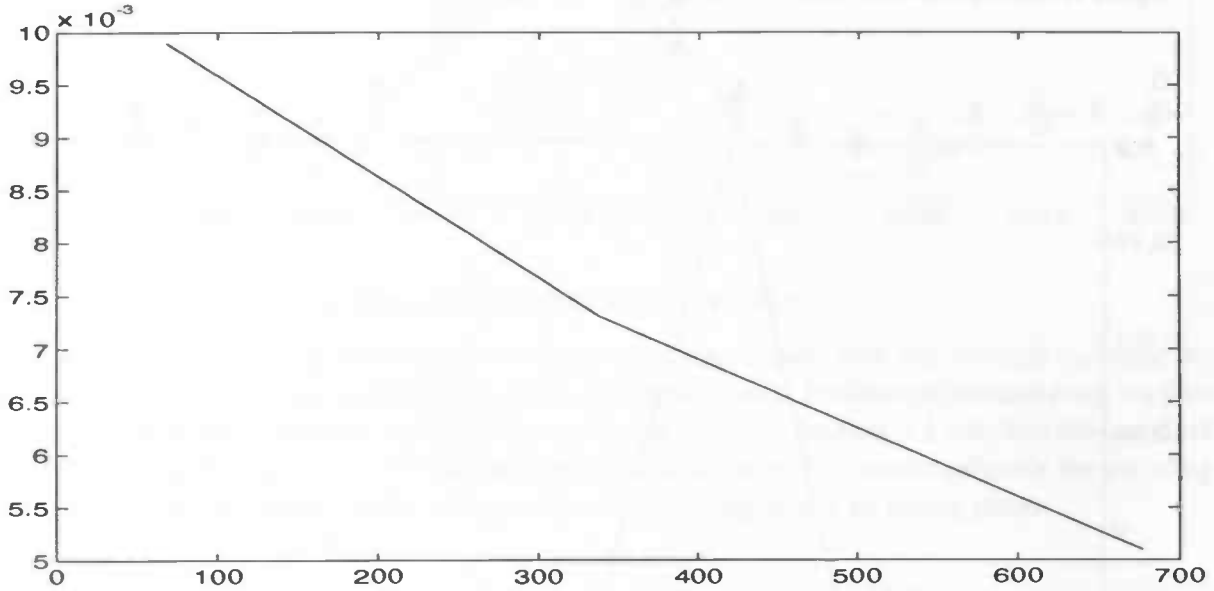

Figure 34. Relation between vector distance and entropy

8.1.3.3 Envelope: rectifying and smoothing.

The standard method for preprocessing the raw EMG is by full–wave rectifying and smoothing the signal (Section 3.2.3.2). Performing this on the signal at hand, we obtain the following signal (Figure 35.):

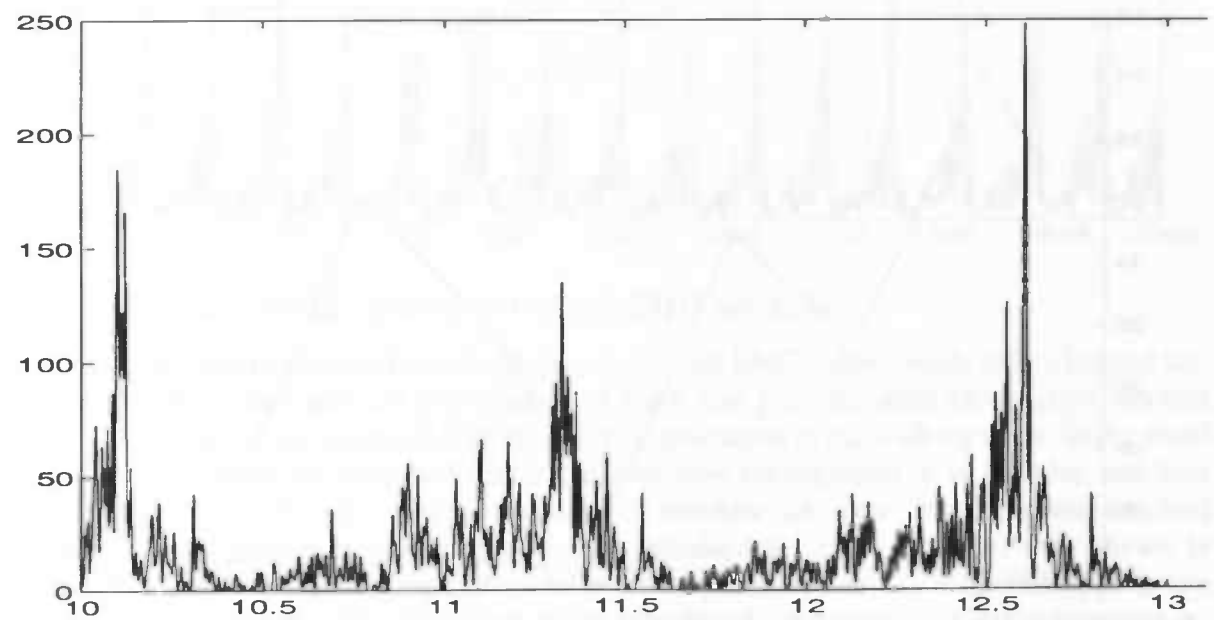

Figure 35. Envelope of EMG segment

We will use this signal for validation of our prediction experiments: when a network can predict this processed version of the raw EMG well, then, because of its adequacy in describing muscle activity duting the walking cycle, stimulation can be based on this prediction.

8.1.3.4 Stationarity check.

Since we are dealing with a physiological signal, we could question its stationarity. Therefore, we split the segment into 10 subsegments (Section 5.3.1), and determine mean, standard–deviation and correlation time of all segments (Figure 36. to Figure 38.). This test indicates that the signal is indeed stationary, so no further processing is needed before prediction.

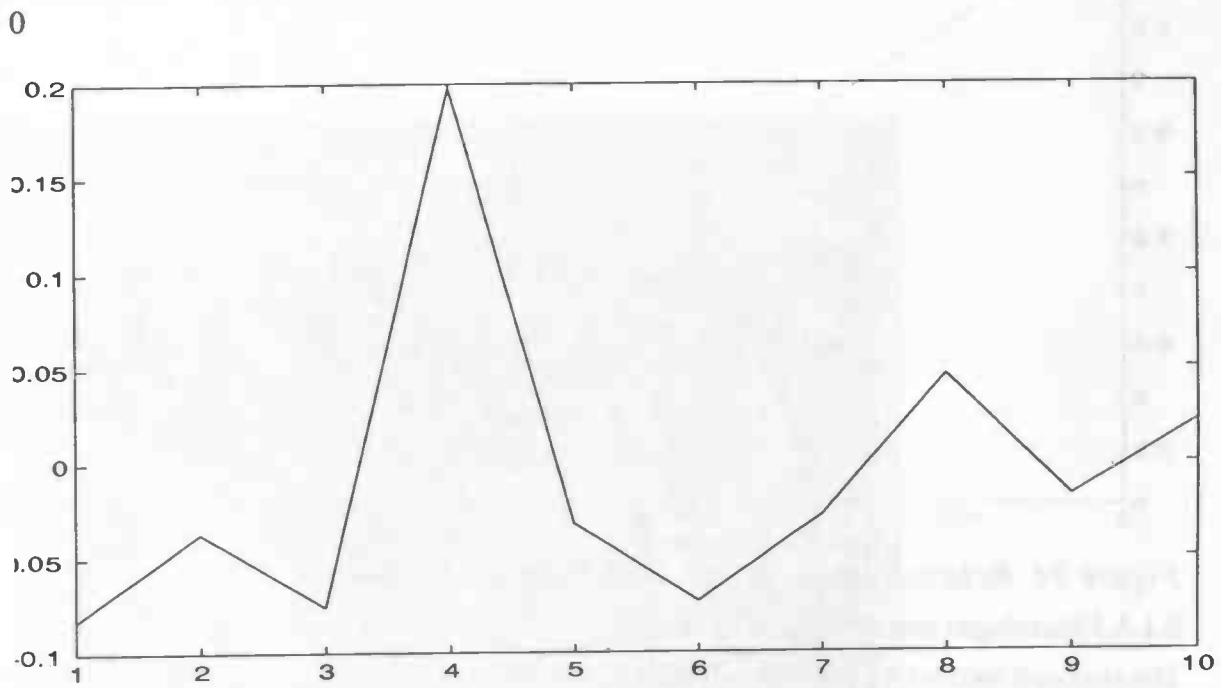

Figure 36. Mean values for different EMG sections

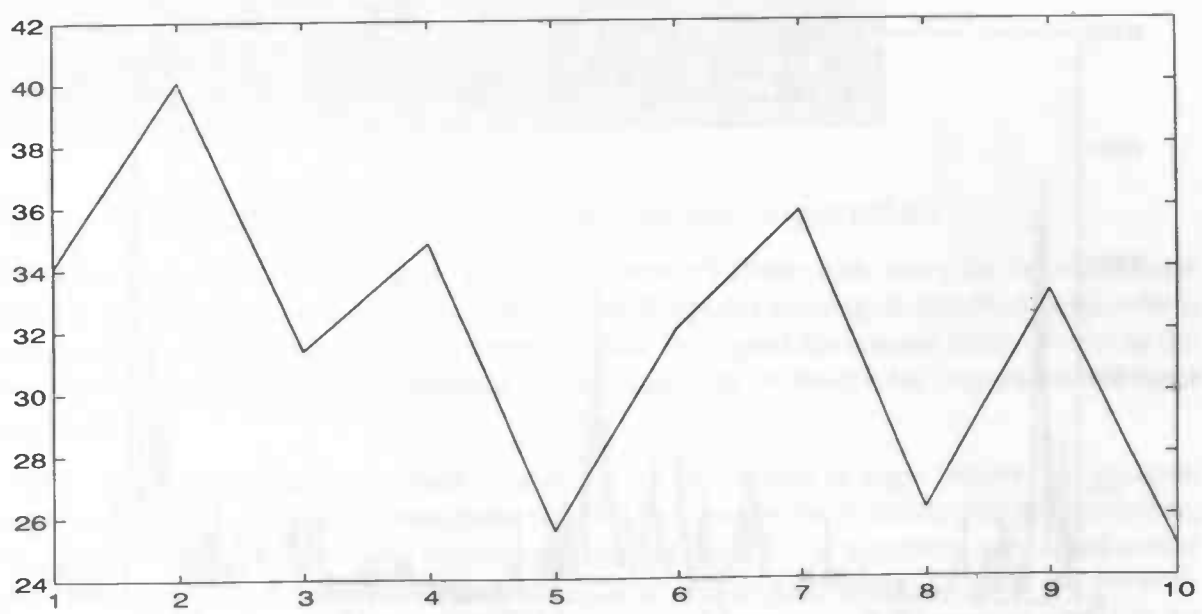

Figure 37. Standard deviations for different EMG sections

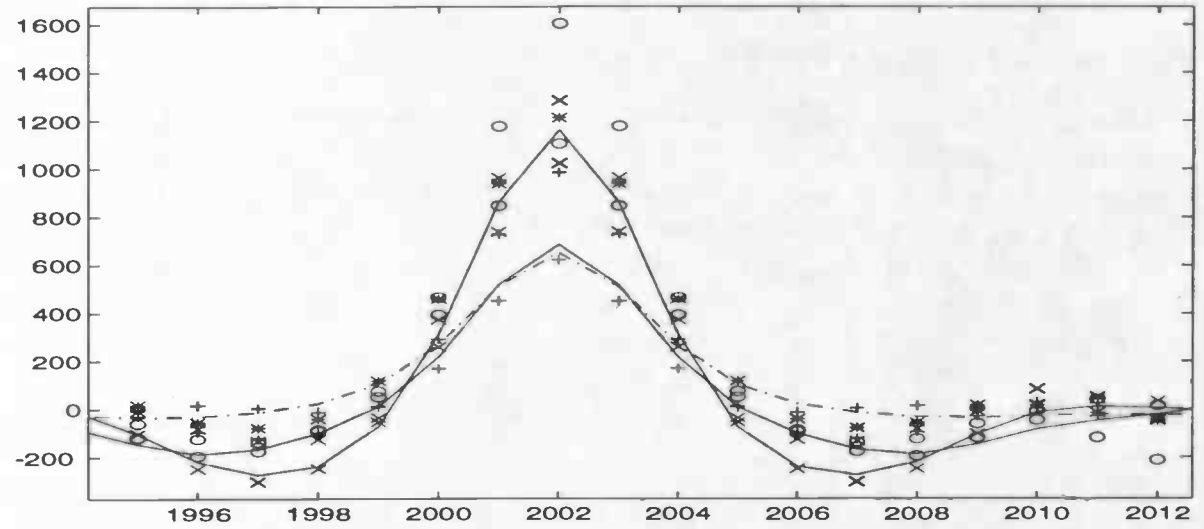

Figure 38. Correlation time of different EMG sections

We would like to include features in our neural predictor, since they can contain valuable information to direct the networks its way to the right predictions. Besides movement data, we also will apply statistical features for this purpose. It was stated in Section 3.2.3.4, that the standard deviation of the EMG is indicative of the exerted muscle force. This seems valuable for deciding whether the instantaneous EMG values come about during stance or swing phase.

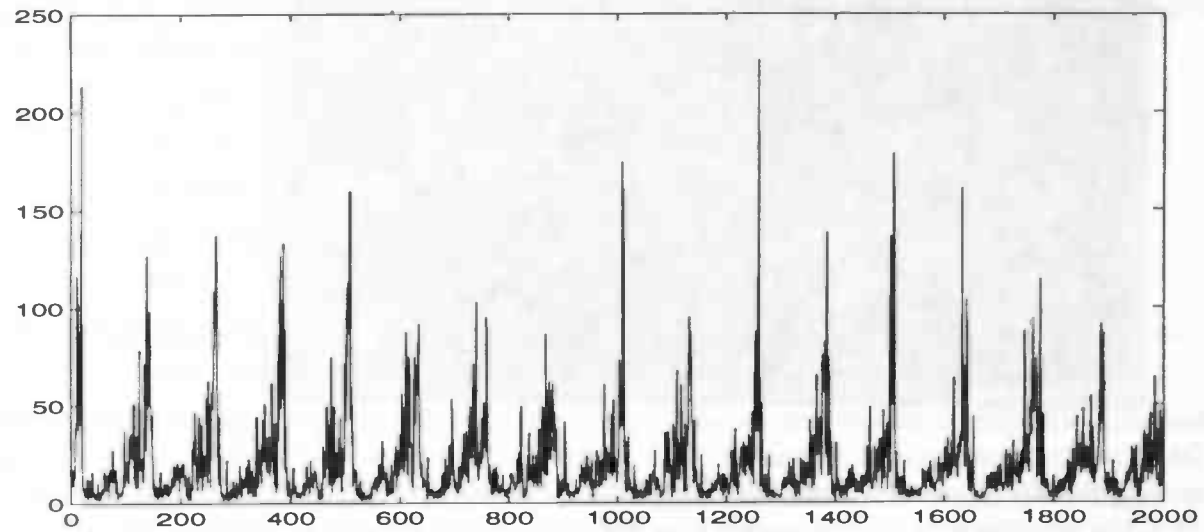

Figure 39. Standard deviations for small EMG sections

Moreover, on–line prediction has to be done using recent EMG values, while only a limited history can be taken into account (for reasons of learn and generalization capabilities, Section 6.1.1). We have to check potential features for their indication of the walking cycle, using small time windows. Hence, we split the EMG into consecutive subsegments of 10 samples, and look at the course of the "filters" we apply to each time window (i.e. mean, max, min and standard deviation of the samples in a window). For the statndard deviation, the results are shown in Figure 39. We see that the walking cycle is reflected very well by them, so they seem appropriate for utilization as features. The minimum and maximum of a window yield less convenient results, hence they are ignored in the sequel.

8.1.4 Principal Components Analysis.

Self–organized neural network implementation.

In Section 6.2.4 a neural realization of a Principal Components Analysis was presented, using the generalized Hebbian learning rule. The algorithm was implemented in the *C* programming language, using the neural network simulator *InterAct*. From the practical considerations concerning the implementation of the algorithm (Section 6.2.4.3), convergence of the algorithm can be seen to depend on the particular choice of the learning parameter η. We experimented with a few values, and finally fixed it at 0.0001. Convergence of the net leads to the following situation (Figure 40.):

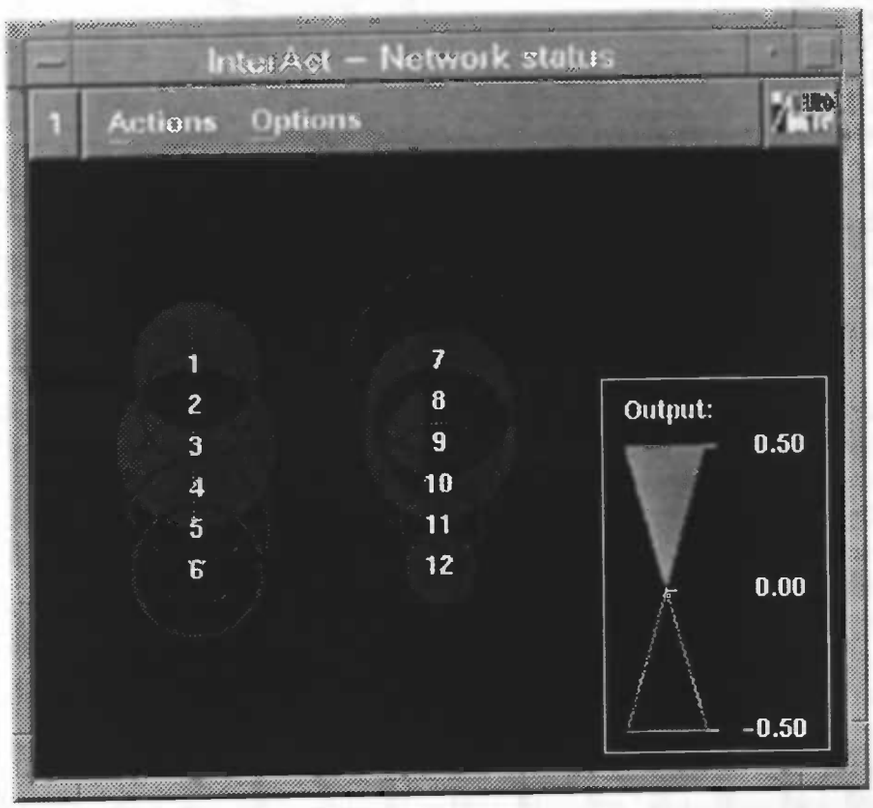

Figure 40. Neural Principal Components Analysis: eigenvalues

The plot was obtained by making an *xgrab* from a network observation using the *InterAct* neural network simulator, at a random moment after some 50 epochs training. In this PCA, angles from all three planes (except knee flexion) were included. We expect the outputs to be ordered in decreasing magnitude, as indeed indicated by the magnitude of the circles (representing output magnitudes).

Another decision that has to be made, concerns the exact change in eigenvectors (i.e. synaptic weights in the network) ε below which the network is considered to be stable. After some tuning, we set it to a value of 0.001. Early experiments showed also that the algorithm converged within some 100 learn cycles (epochs), so this was used as another stop–criterion. Once the network reaches a stable configuration, its weights denote the eigenvectors of the correlation matrix of the inputs (Figure 41.)

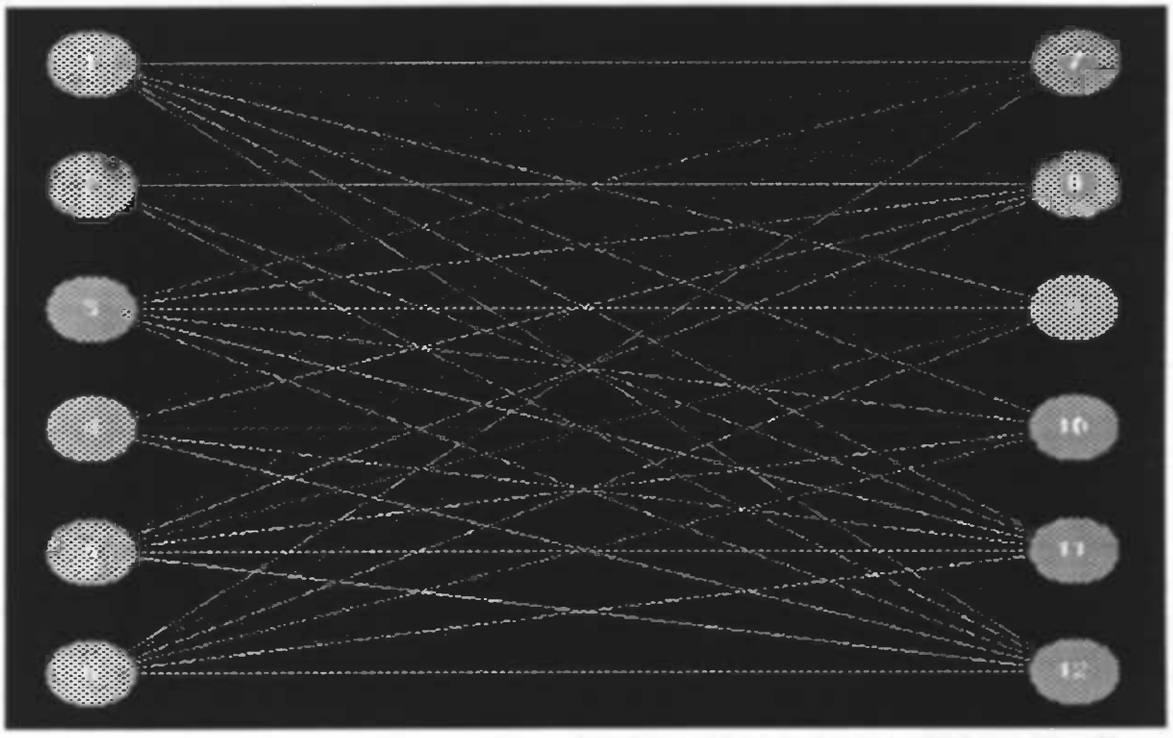

Figure 41. Neural Principal Components Analysis: eigenvectors

Which angles take into account?

In order to investigate the assumptions concerning the selection of joint angle patterns to serve as network features, we applied the algorithm above to three different sets of input vectors:

- I. joint angles from the frontal and transversal plane, i.e. only knee rotation and hip abduction were included
- II. joint angles from all three planes, i.e. hip extension was added
- III. all joint angles from all three planes, i.e. knee flexion was also included

We included both right and left lower extremity angles in the PCA; since walking is a symmetric activity, similar conclusions can also be drawn when dealing with just one extremity side.

In the former discussion (Section 8.1.2.2) we assumed that angles from the sagittal plane are most useful as features for EMG–prediction, since they show the strongest correlation with the EMG. This can be verified by looking at the effect of adding sagittal plane joint angles to the PCA–network on the resulting eigenvalues: when large new eigenvalues emerge after addition of hip extension or knee flexion, these can be expected to contain most of the information of all inputs, i.e. of all angles.

The results from all three simulations are given in Table 1. Displayed are the values of the eigenvalues corresponding to the output neurons (we used networks containing a matching number of input and output neurons, hence the increasing number of eigenvalues).

Set I.	Set II.	Set III.
0.699668	1.236558	1.277148
0.516665	0.675620	1.076043
0.026351	0.184718	0.446925
0.011675	0.133915	0.245274
	0.010417	0.098868
	0.021777	0.018919
		0.030322
		0.014968

Table 1 Application of self–organized PCA to joint angles

It is clear, that with every angle from the sagittal plane a new, very large eigenvalue occurs. Hence, our former assumption is confirmed.

8.1.5 Tuning the network.

In Section 7.3.3.1 we mentioned the issue of choosing a suitable error measure, in order to be able to track network learning adequately. We assumed that the 3σ–error would yield the smoothest decay, whereas the mean error (the error between target and output values over a batch of input patterns having normal distribution with zero mean) is expected to make random fluctuations around zero. This is indeed the case. In Figure 42. a typical learning curve is shown, that emerged when training neural networks to predict the correlation time test signals of Section 7.3.2. The solid line represents the 3σ–error, while the dashed lines denote maximum error (at about the same level as the 3σ–error), mean (random fluctuations around zero) and root mean squared (rms) error. Notice that the rms–error has a (smooth) course that is similar to the 3σ–error, but its magnitude is smaller. The max–error shows an acceptably smooth decay. When looking closer, however, its course appears to be actually quite jagged. In other experiments, this effect was yet stronger than this. The max–error can be used conveniently, however, when we look for *unlearning* to take place, i.e. a uniform increasing test error, after a minimum has been reached (Section 6.1.1): the large errors that emerge when proceeding with the unlearning experiment show up in a much more direct way.

We chose to use the 3σ–error for further experimentation, because they seem to employ all good properties of the rms– and max–error, hence it is more versatile then each of these.

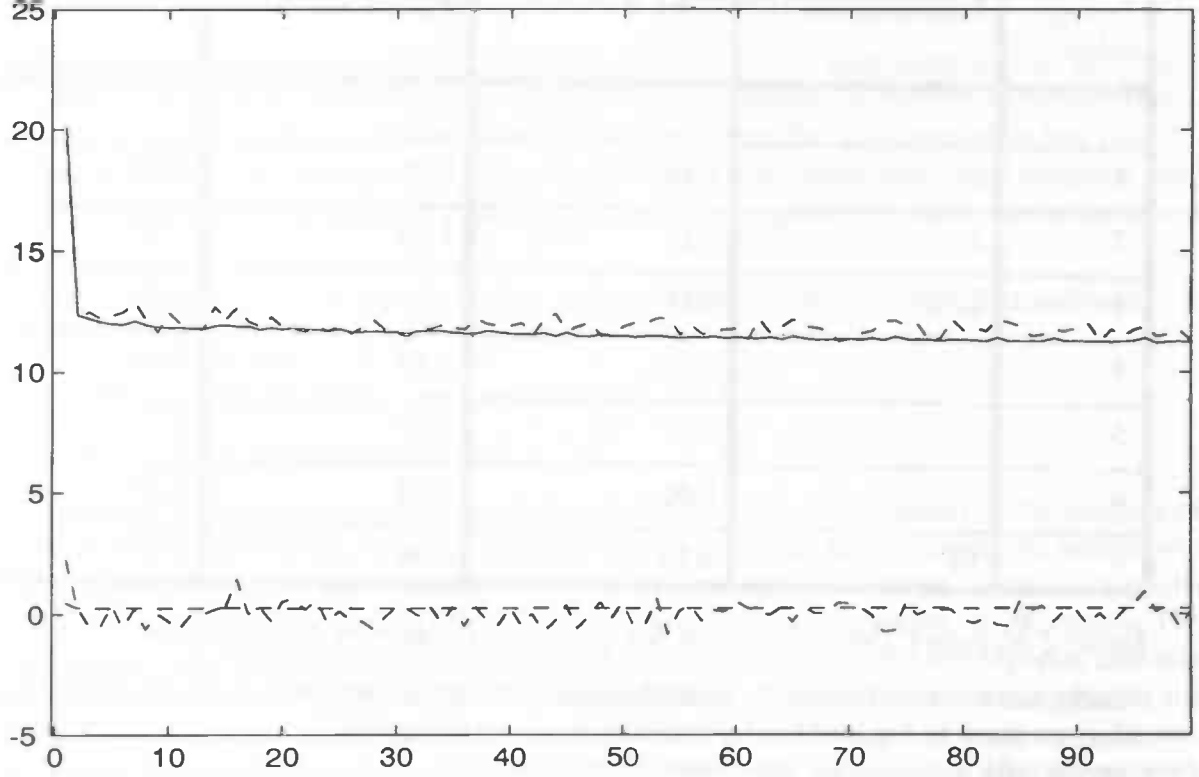

Figure 42. Typical learning curve (train error) for autocorrelation test–signals

8.1.6 Investigation of the frequency hypothesis.

For the investigation of the frequency hypothesis, we constructed test signals with different ratios of minimum and maximum frequencies (Section 7.3.2). Next, all signals are consecutively offered to a neural network, varying the number of taps in the tapped delay–line in a similar way for each signal. We used delay–lines of length $ND = 3,\ldots,5$, where the signals exhibited frequency ratios 2,...10. When the hypothesis is correct, we expect the signals with frequency ratios greater than the size of the delay–line (which equals the number of inputs to the network), to exhibit bad generalization. For signals where the number of inputs is varied on both sides of its frequency ratio, a sudden decrease in the steady–state test error (test error after 7000 learn cycles) is expected, when passing the critical value. In Table 2 the steady–state test errors are shown for all test signals and different values *ND*.

Test signal no.	Frequency ratio	ND = 3	ND = 4	ND = 5
1	5	1.12	1.10	1.25
2	2	1.25	1.20	1.14
3	1.7	1.25	1.25	1.30
4	2	2.20	1.40	1.60
5	5	0.80	0.70	0.5
6	6	1.60	1.60	1.35
7	10	3.20	1.60	1.70

Table 2 Network generalization for different frequency test signals

Clearly, there is no effect on the generalization performance whatsoever. To illustrate this even more, we show in Figure 43. a typical learning curve (course of the *test* error, in this case), that emerges with almost every experiment from Table 2.

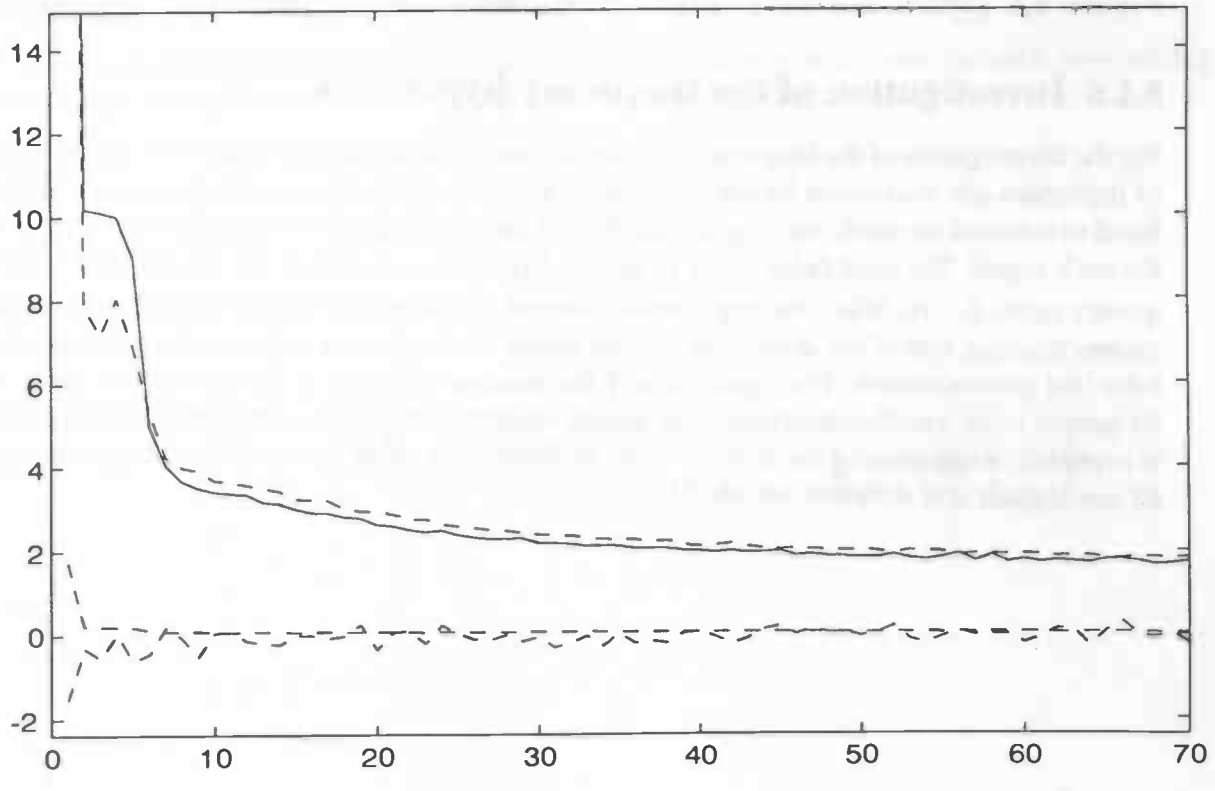

Figure 43. Typical learning curve (test error) for frequency test–signals

This seems to indicate that the hypothesis about the relation between frequency ratio and delay–line size is wrong.

8.1.7 Investigation of the correlation time–hypothesis.

We hypothesized that the size of the tapped delay–line in a TDNN used for time series prediction depends on the temporal relations within the signal to be predicted, i.e. the correlation time (Section 4.2.2). This hypothesis willl now be investigated using the autocorrelation test data described in Section 7.3.2: when it doesn't even hold for data with simple correlation properties that are known a priori, we can certainly discard it for complexer signals (exhibiting nonlinear dependencies).

We chose a number of network parameters, that were held constant during all experiments:

- learning rate $\mu = 0.7$
- momentum term $\alpha = 0.5$
- number of hidden units = 10

Furthermore, we have tried to overcome the usual critique w.r.t. the reliability of the outcome of neural network experiments, by looking only at *trained* networks, and averaging the results over 5 repetitions of the same experiment. A network was observed to be converged (previous section) after it has been trained for some 7000 cycles. For safety, we chose to train for 10.000 epochs. The error measure that seemed most suitable for tracking the learning process was reported above to be the 3σ–error, i.e. the value within which some 95% of all errors (i.e. their magnitudes, the errors are normally distributed having zero mean) from a batch of patterns are positioned. Hence, we defined the mean steady state error (MSSE) to be: the mean 3σ test–error after the network had been trained during 10.000 cycles, averaged over 5 separate identical experiments.

8.1.7.1 Using signals with different correlation time.

The first three test signals exhibit correlation times of 3, 5 and 7, respectively. We vary the number of inputs (i.e. the size of the tapped delay–line) in about the same range. Other network parameters (learning rate, momentum term, number of hidden units) are chosen identical to the frequency hypothesis experiments, reported in the previous section. The results for the first three autocorrelation test signals are shown in Figure 44. to Figure 49. The circle within a rectangular bar denotes the MSSE–value over 5 experiment repetitions, whereas the horizontal bars on the outside of the vertical bar indicate minimum and maximum steady–state test error over 5 experiments; the *plus*–signs denote the mean steady–state train error.

We see a uniformly decreasing train error in all three graphs, something we would expect from Section 6.1.1. The generalization error, however, seems to have its minimum in the experiment with the delay–line size that equals the test signal's correlation time. For the first signal, this is obvious.

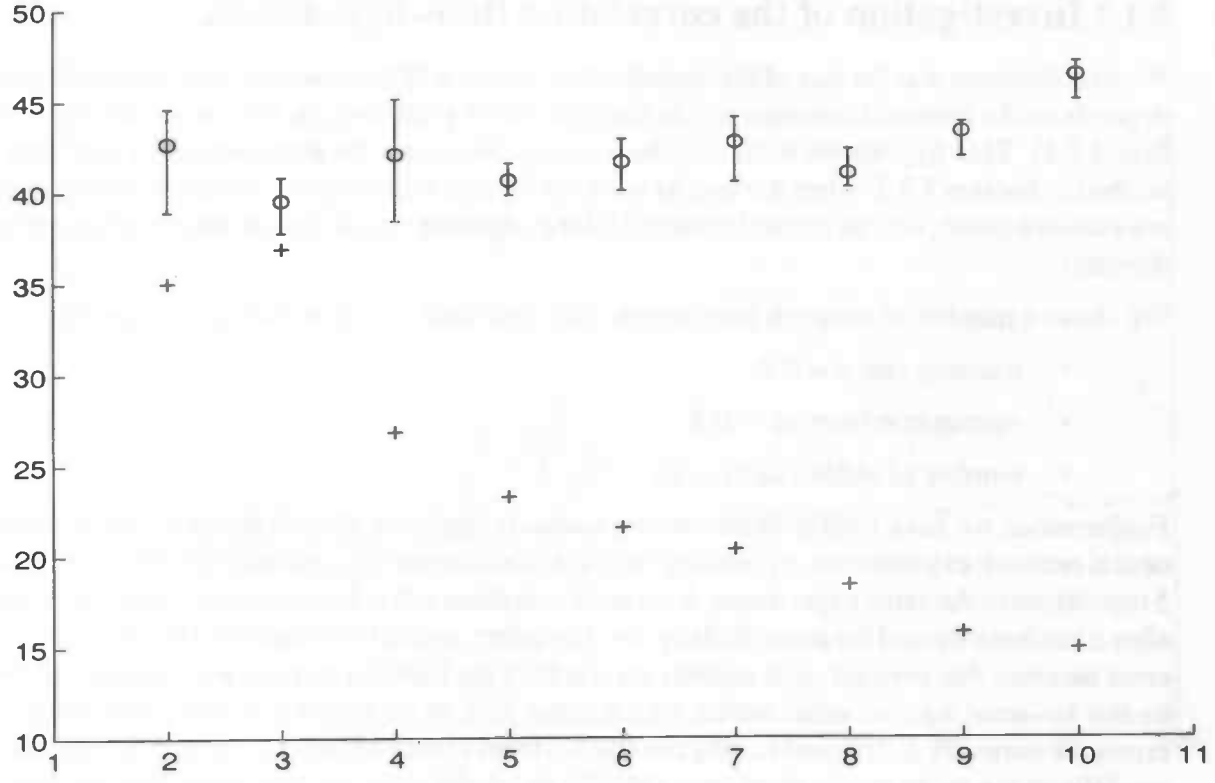

Figure 44. Network generalization test–signal 1

The second signal, however, exhibits a large increase in steady–state error variance, represented by a large vertical bar, when the critical value is surpassed (Figure 45.)

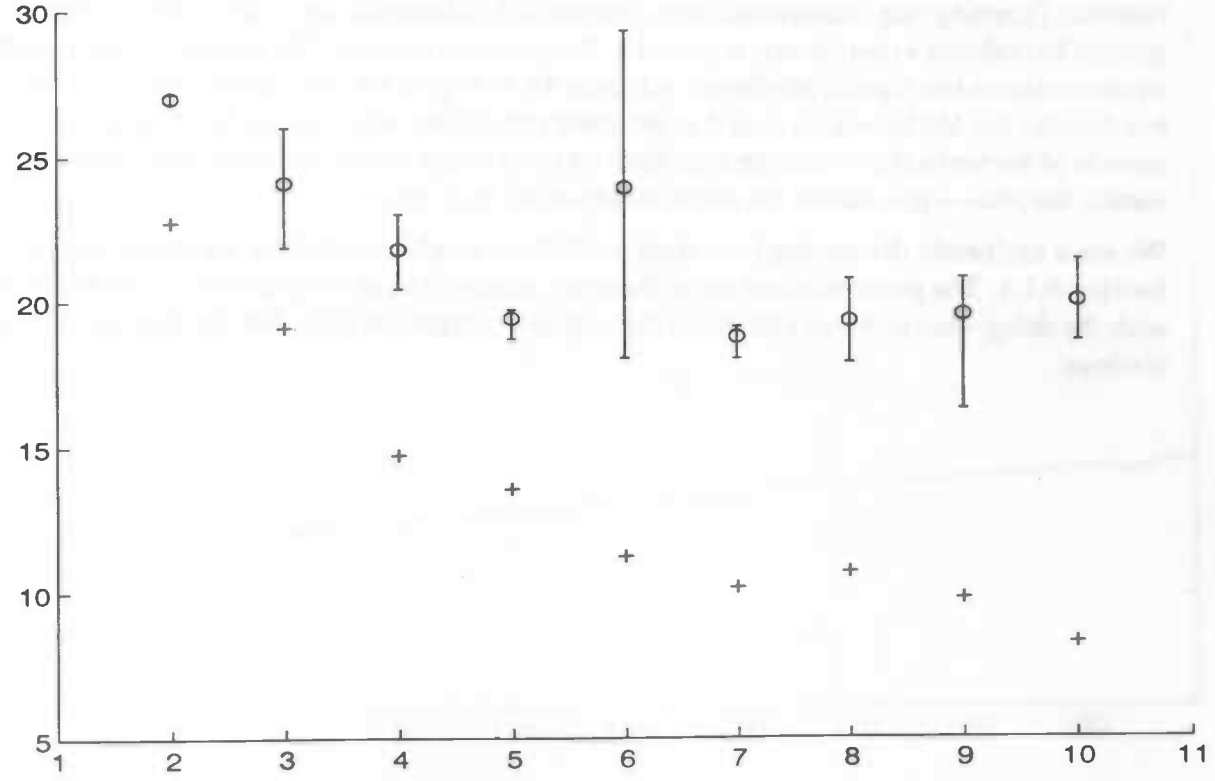

Figure 45. Network generalization test–signal 2

In Figure 46., the experiments are repeated for another 5 times (i.e. the above procedure was repeated) using delay–lines of size 5 to 7. The outcome is nearly the same for the experiment with a number of network inputs that equals the correlation time of 5, whereas for larger number of inputs, very different results emerge: averaged over 10 independent repetitions, the variance for the latter experiments is much higher than in the former case. This indicates (Section 6.1.1) that the network has reached a stable minimum energy level at this number of inputs; at delay–lines *shorter* than the input signal correlation time, there is still "much to be learned", so on average similar generalization capability is shown (i.e. small variance in MSSE); at *larger* delay–lines, the minimum level has already been reached, so addition of inputs in fact means: increasing the number of degrees of freedom, while no extra information is added. This will worsen generalizability.

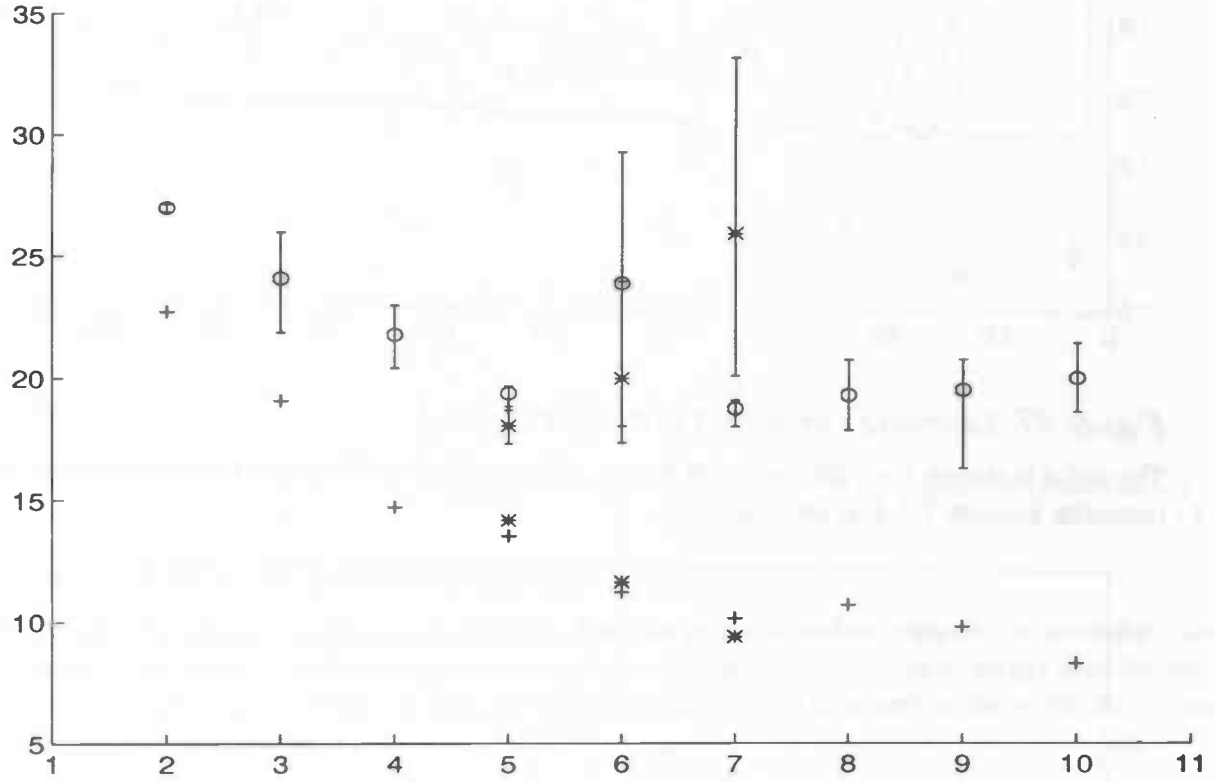

Figure 46. Generalization test–signal 2: revisited

This *unlearning* can also be shown by looking into the error curvatures. In Figure 47., the learning process for a network that is trying to learn the second test signal (using a TDL of size 5) is shown. The dashed line denotes the max–error, whereas the solid line represents the 3σ–error.

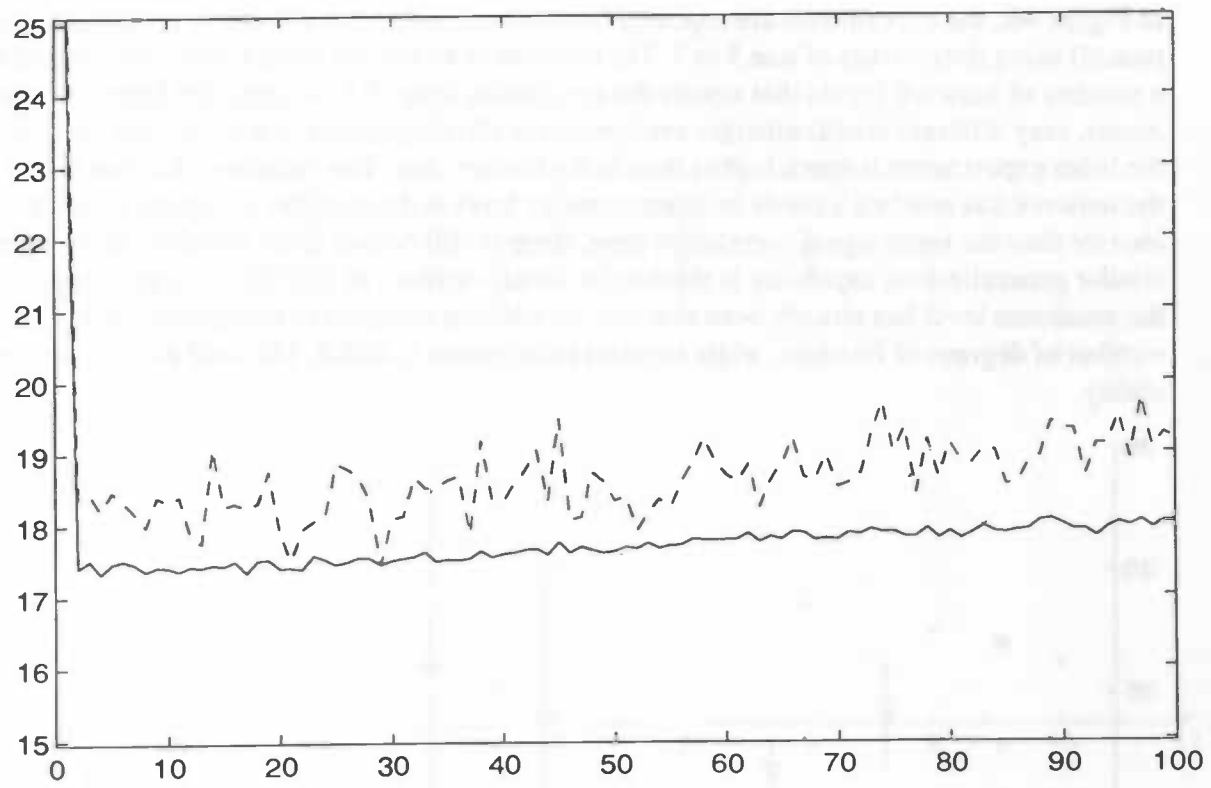

Figure 47. Learning curve (test error) for ND = 5

The same is shown for a delay–line of length 7 (Figure 48.), while all four errors that were mentioned in Section 7.3.3.1. are displayed.

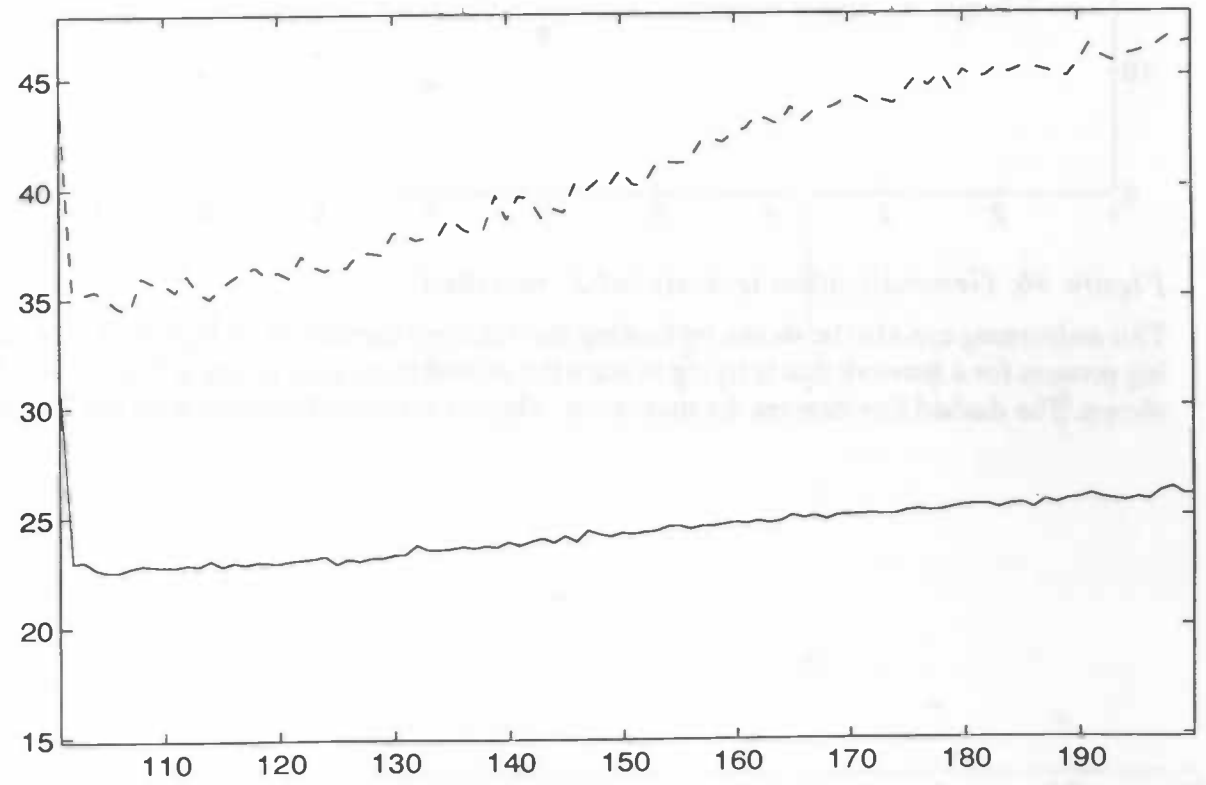

Figure 48. Learning curve (test error) for ND = 7, second run

A dramatic increase in test error can be observed in the latter case, while in the former the error remains almost at a constant level.

The *generalization level* from the correlation time number of inputs and up, is best shown by the third testsignal (Figure 49.):

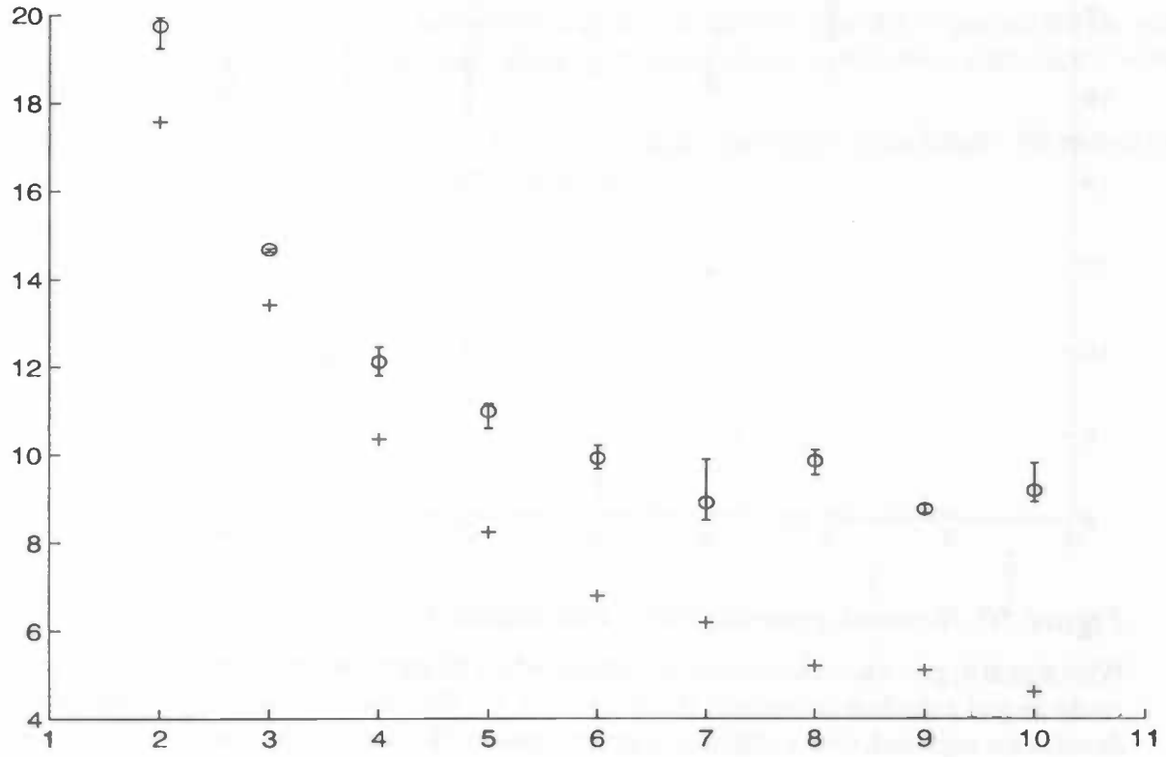

Figure 49. Network generalization test–signal 3

We think that we may safely conclude that a **level in generalization capacity** is entered as the *number of previous samples*, used for one–lag time series prediction, is taken **larger** than the signal's *correlation time*, at least as long as we are dealing with a time series that originates from a linear process.

8.1.7.2 Using signals with different realization length.

Next, we come up with the idea that test signals, constructed with the same convolution kernel, but with different white noise realization length may show different behaviour compared to the above experiments. As the noise process, that is used for the construction of the signals, is of longer length, correlation characteristics are more likely to be exactly such as they are intended to be. For short noise lengths, artefacts may be introduced. The results (obtained in the same way as the former) for the second set of test signals are given in Figure 50. to Figure 52.

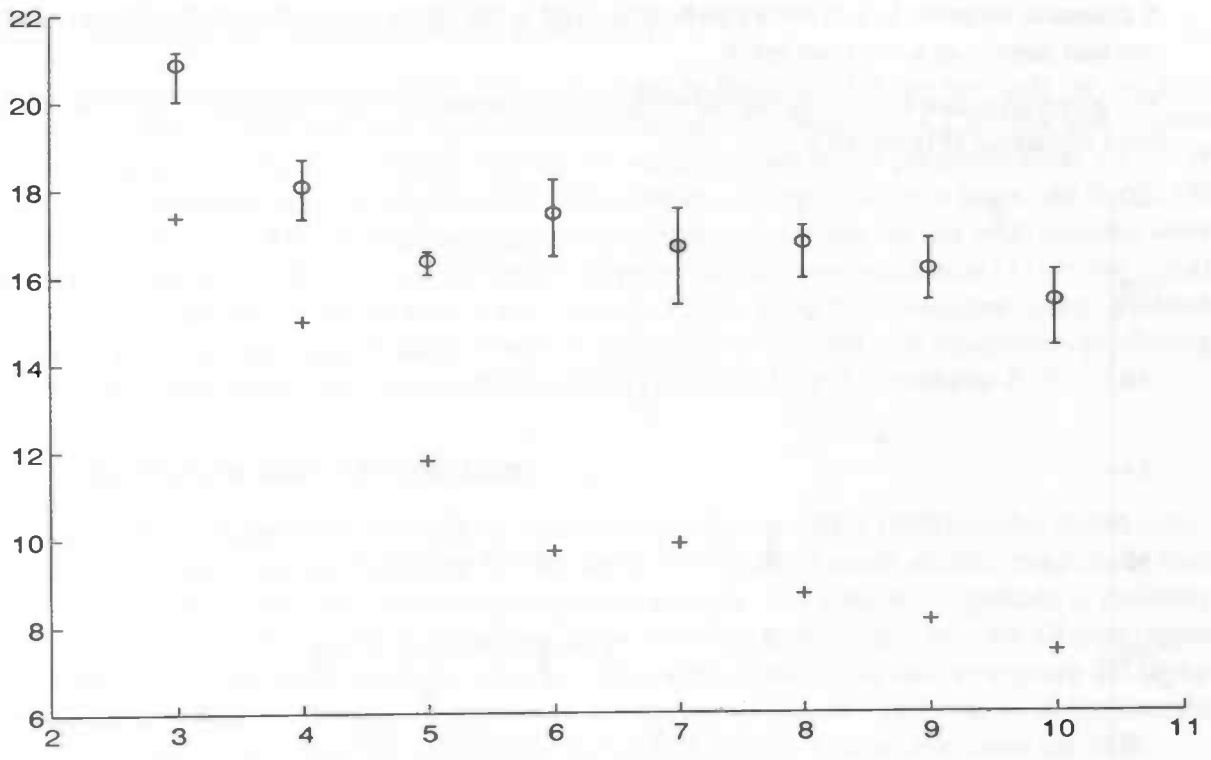

Figure 50. Network generalization test–signal 4

With signal 4, generalization seems to improve when 10 inputs are included in the network. This could be just a random movement about the level, but also indicate that new (erroneous) dependencies are captured, hence improving generalization. When this is the case, we will expect this effect to be the strongest for the signal of shortest realization length, i.e. signal 4.

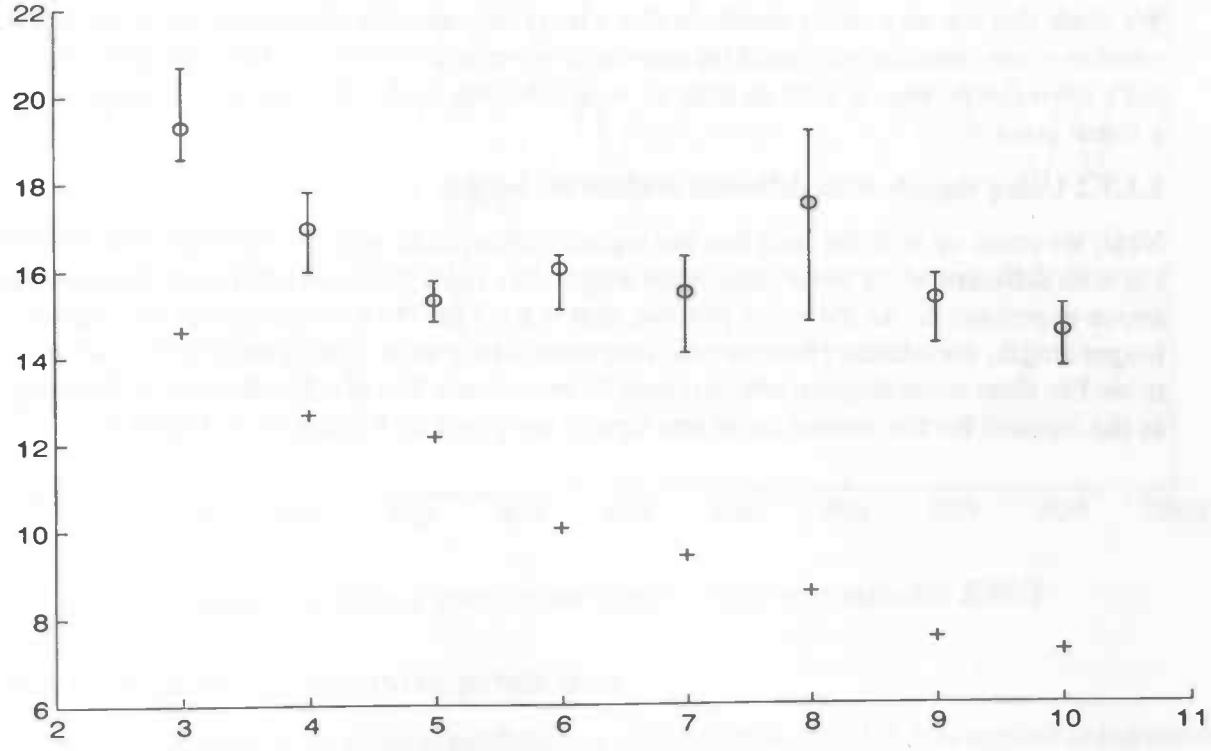

Figure 51. Network generalization test–signal 5

Looking at Figure 51., the previously stated assumption could still to be correct: again, a minimum level of MSSE–values is seen to start at correlation time 5. The MSSE–value at 10 network inputs is positioned below the value at 5 inputs, but the variabilities beyond 5 inputs are also higher, indicating worsened generalization. Hence, the unlearning (e.g. look at the variability of the MSSE at 8 inputs) seems to get stronger with increasing noise length realization level, i.e. our above assumption seems to apply better with longer realization length (the results for the second test signal represent the "limiting case", since the first set of test signals were constructed with whitenoise length 40.000).

Finally, the last test signal is inspected, having still larger associated noise length. We expect to see a similar phenomenon as in the previous case.

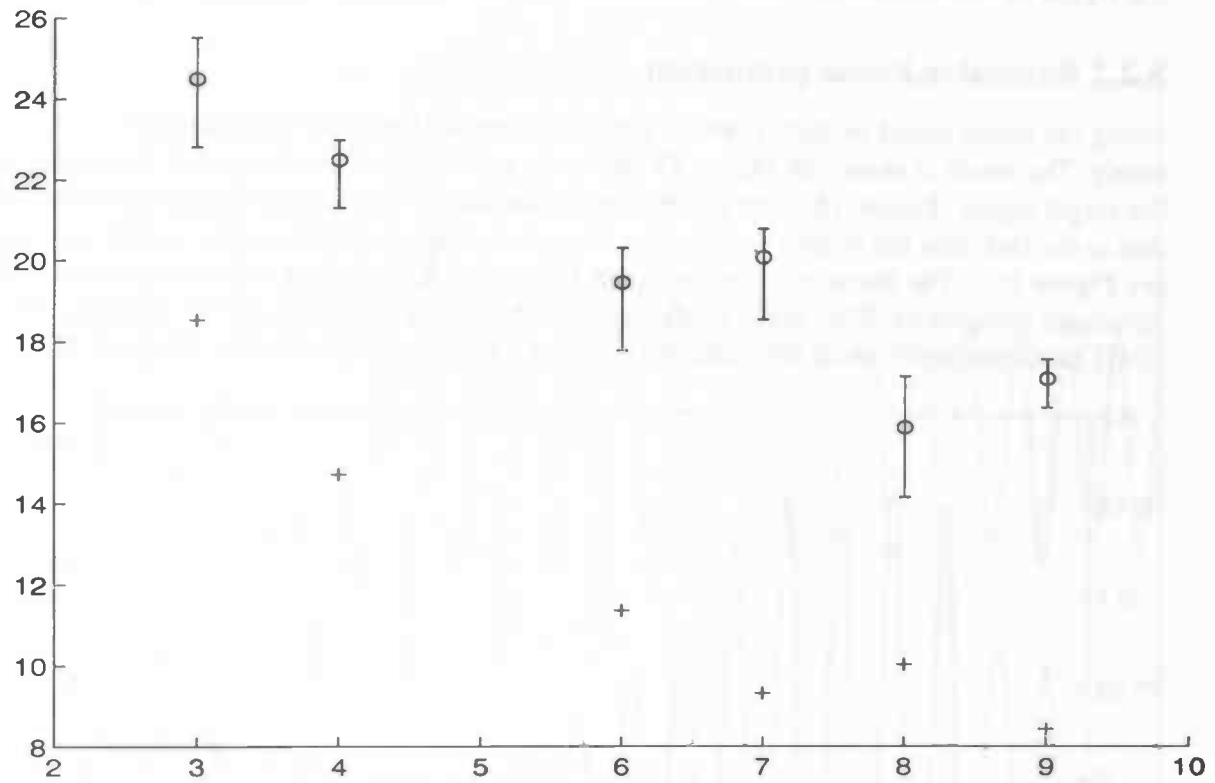

Figure 52. Network generalization test–signal 6

However, the opposite seems to be true: the value of MSSE seems to decrease with increasing number of inputs. Unfortunately, we didn't check the results for delay–lines of lengths 5 and 10, which played such an important role in the previous cases. Until the onset of the minimum level is determined, final conclusions about the effect of noise realization length on network generalization cannot be drawn. The last results seem to contradict our earlier assumption, but this is merely an indication.

8.2 Prediction of synthetic EMG.

In our *structured* approach to neural systems design, we want to make decisions w.r.t. network architecture and inferences about the network's performance on rational grounds. Hence, we use test data, both for the investigation of two assumptions about delay–line size (in other words, the *order* of the neural predictor) and for benchmarking the neural results with. For the latter reason, a synthetic EMG was constructed, and used for prediction. Capability of a neural network to predict this simple signal, based on previously stated assumptions about predictor order and correlation criteria, can shed a new light on neural (dis)abilities in the prediction of the real EMG segment (and some derivatives).

8.2.1 Finding the optimal linear predictor.

First, we have to determine a suitable predictor order. In Section 5.3.3 two methods for this purpose are shown. A third approach is, motivated by the results in Section 8.1.7, to look at the signal correlation time. Using this criterion and the AC–criterion, we found predictor order values of 11 and 115, respectively. The segment of 3000 samples was split into two parts: the first 2000 samples could be used for coefficient calculation or training, whereas the last 1000 samples were used for prediction solely. In fact, we applied to principle of *cross–validation* [33] to the signal at hand. **One–lag linear prediction** of the synthetic EMG, using high predictor orders, **yielded already quite accurate predictions**. Since we are mainly concerned with recursively predicting the signal farther ahead, only this method will be considered for the synthetic EMG.

8.2.2 Recursive linear prediction.

Using the above stated predictor orders, we tried to predict the EMG 1000 samples ahead recursively. The result is shown in Figure 53. We see a prediction of much smaller magnitude than the target signal (Figure 16.), but exhibiting similar shape. The smaller magnitude is probably due to the fact, that the targets surrounding the point where prediction starts, are all zero (again see Figure 16.). The linear predictor of order 120 seems indeed to be able to capture the signal dynamics adequately. This gives confidence for the use of the AC–criterion to determine the EMG predictor order, since the predictor order of 120 is very close to this value (of 115).

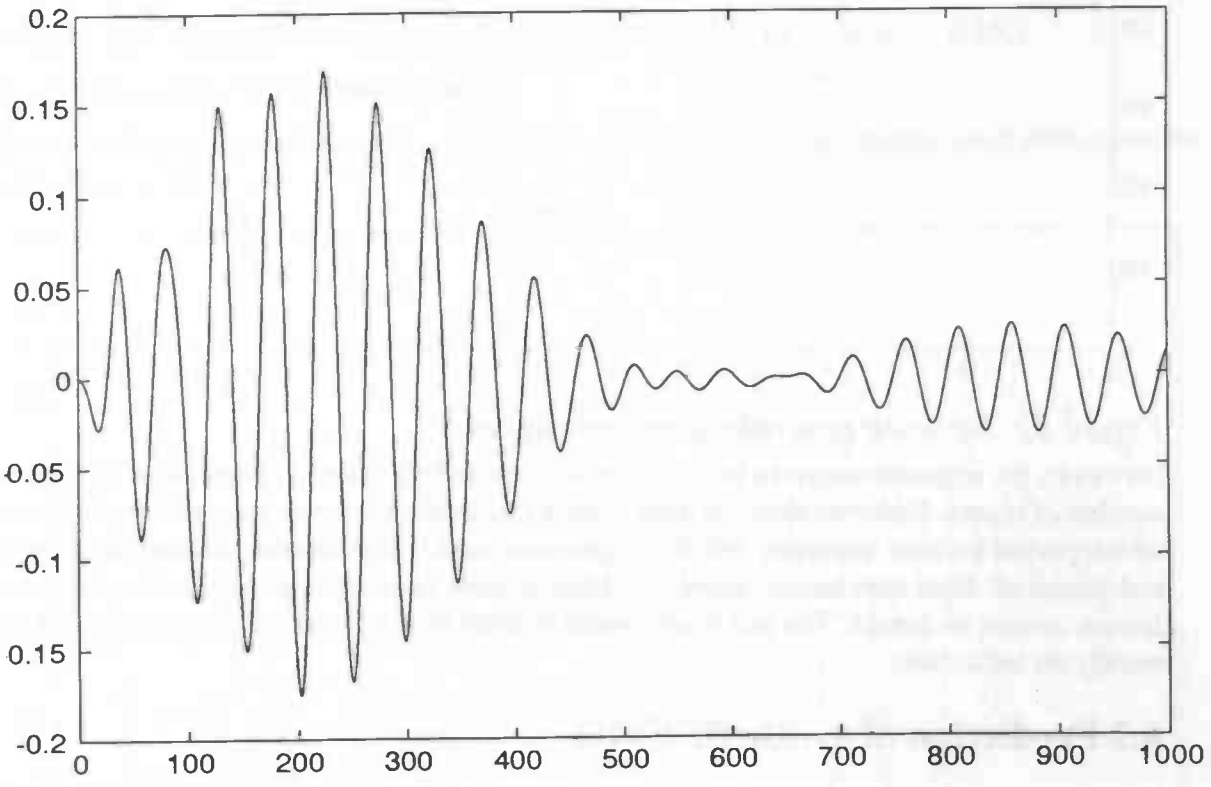

Figure 53. Recursive linear prediction (order 120) of synthetic EMG

8.2.3 Local linear recursive prediction.

Next, the technique of local linear prediction, explained in Section 5.4.5, is applied to the above signal. Since the synthetic EMG is certainly not chaotic, we expect local linear prediction to be unsuitable for it: the method uses the fact, that similarly looking vectors of consecutive samples

will show similar continuations (for a short continuation period). This might be the case occasionally, but certainly isn't guaranteed for all nearest neighbours. For stochastic signals, like we classified the EMG, the former will be very unlikely, so the technique is definitely unsuited.

Care has to be taken in the choice of the local linear predictor order: when the order becomes high, very few reconstruction vectors will be closer than a suitable value of vector distance ϵ, and the algorithm becomes unreliable (containing a very small number of vectors to base a linear predictor on). We chose (experimentally) an order of 10. Results can be seen in Figure 54. A two–period signal of very small magnitude emerges, fading out to zero in the long run. The dynamics of the synthetic EMG was hardly captured; combined with the statement about stochastic signals, mentioned above, we decided to ignore this method for further EMG prediction.

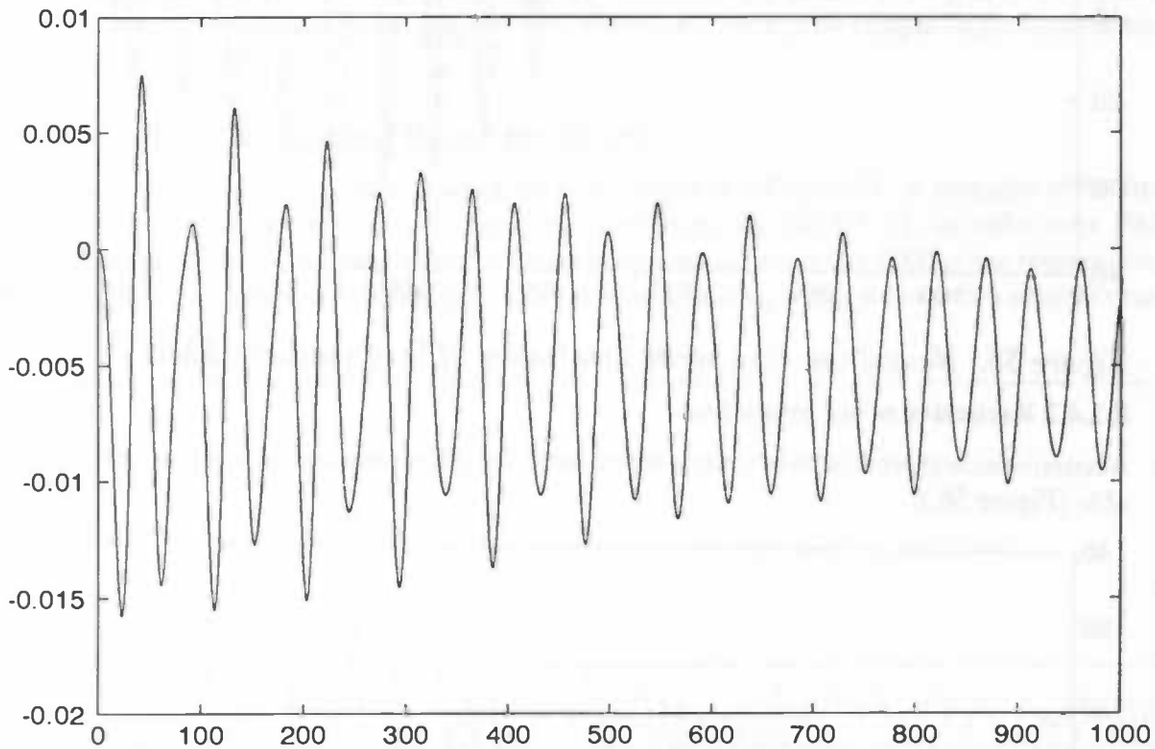

Figure 54. Recursive local linear prediction (order 10)

8.2.4 Neural prediction.

Then, we turn to a neural network for recursive prediction of the test signal. We used standard multilayer Perceptrons, having n inputs (the predictor order), 5 hidden units and 1 output. Furthermore, we trained a network for 5000 cycles. Other parameters were set at values equal to the experiments in Section 8.1.7.

8.2.4.1 One–lag prediction.

In Section 8.2.1, we mentioned that the synthetic EMG proved to be one–lag predictable quite well. Best results for linear and neural predictors were comparable; as an illustration, a neural one–lag prediction at the AC–order of 115 is shown in Figure 55.

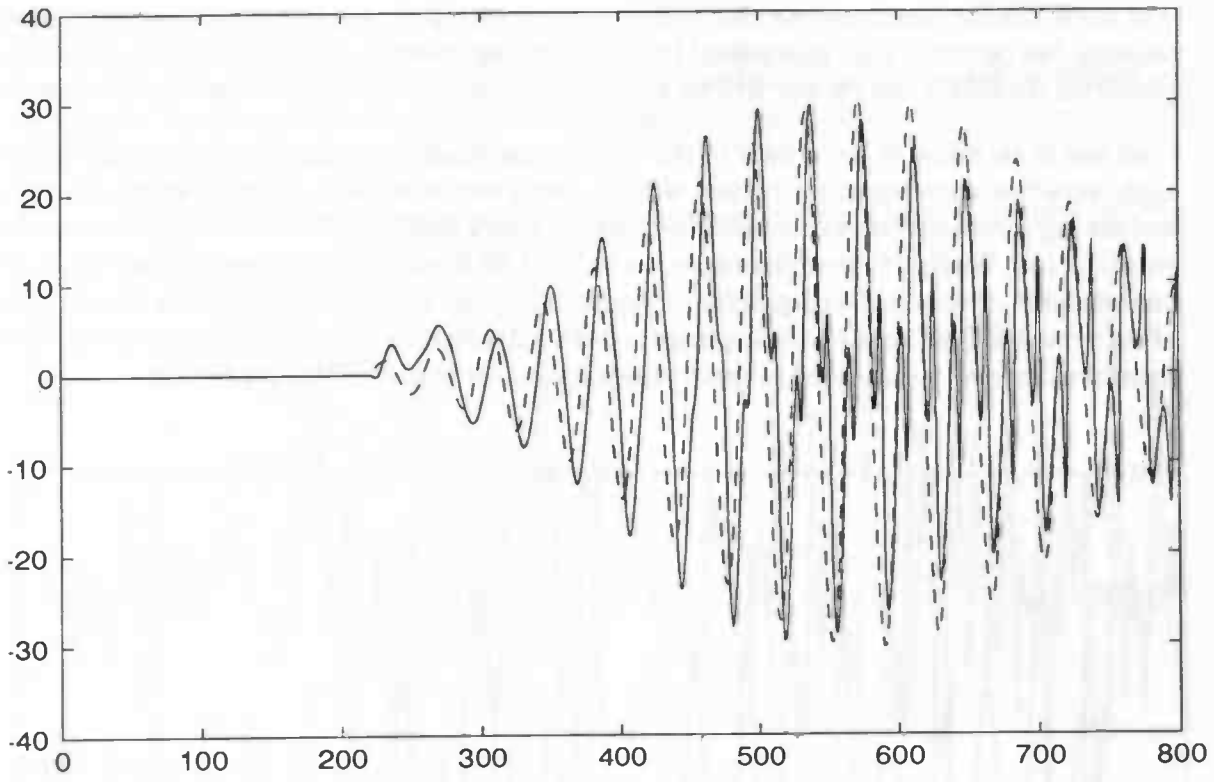

Figure 55. Neural one–lag prediction (order 115) of synthetic EMG

8.2.4.2 Recursive n–lag prediction.

Recursive neural prediction of the test signal using the AC–criterion seems much more questionable (Figure 56.):

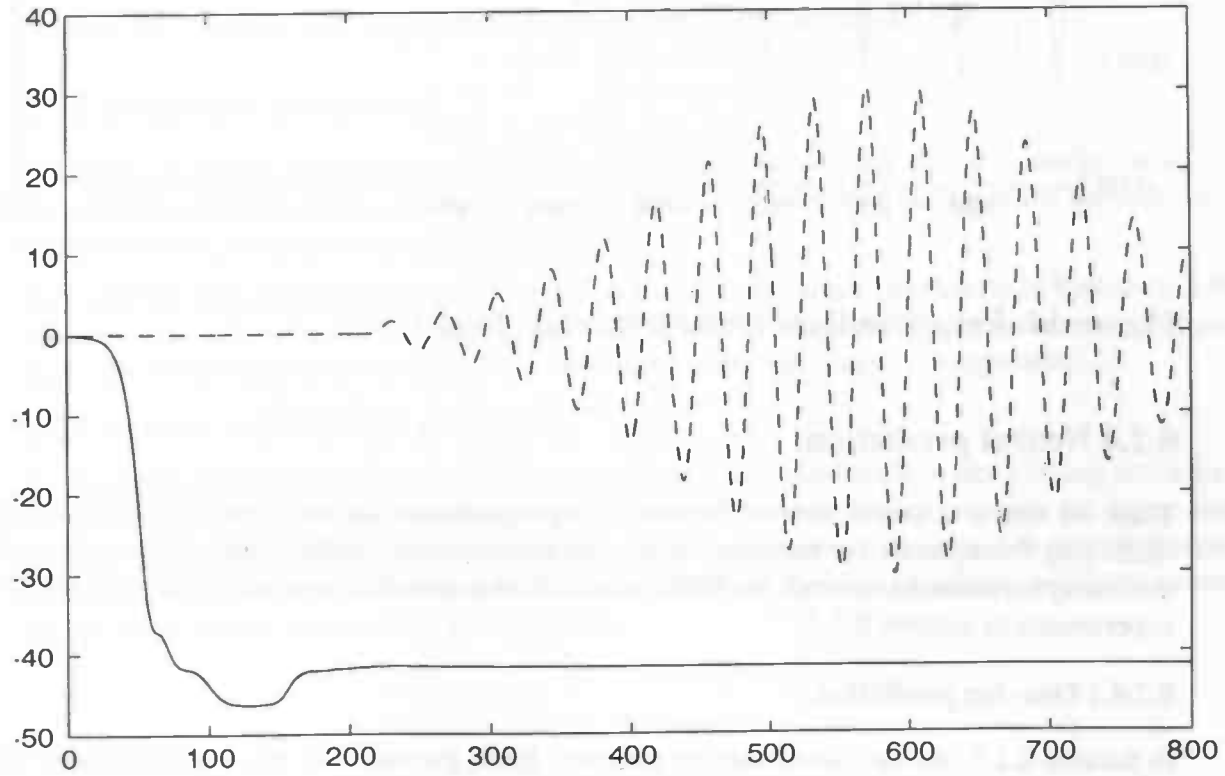

Figure 56. Neural n–lag prediction (order 115) of synthetic EMG

Hence, an overall decision for the AC–criterion to be the most useful, cannot be made yet. We will have to investigate all three criteria in the prediction of the real EMG.

8.3 Prediction of the human EMG.

Next, we proceed towards our final goal: on–line neural prediction of a segment of human EMG, using statistical and movement features. First, we have to determine three parameters that could indicate the predictor order: correlation time, AC–criterion and visual inspection of the partial autocorrelations of the segment. Then we will experiment with networks with a tapped delay–line that equals all three predictor orders, benchmarking with linear predictors of the same order. Later on, statistical features and movement data are added, using the same predictor order.

Note that the predictor order doesn't change when features are added, as long as these features do not contain information from samples on a lag from the prediction sample larger than the size of the tapped delay–line.

8.3.1 Finding the optimal linear predictor.

We plotted the partial autocorrelations of the EMG segment in Figure 12.. A snapshot of the first 150 values is given in Figure 57. Using the method from Section 5.3.3.1, an order over 1000 would be appropriate. Since the 'burst' at about partial autocorrelations 1000 is just representing the correlation of a sample with samples in a previous (walking) step, we expect a smaller order to be appropriate.

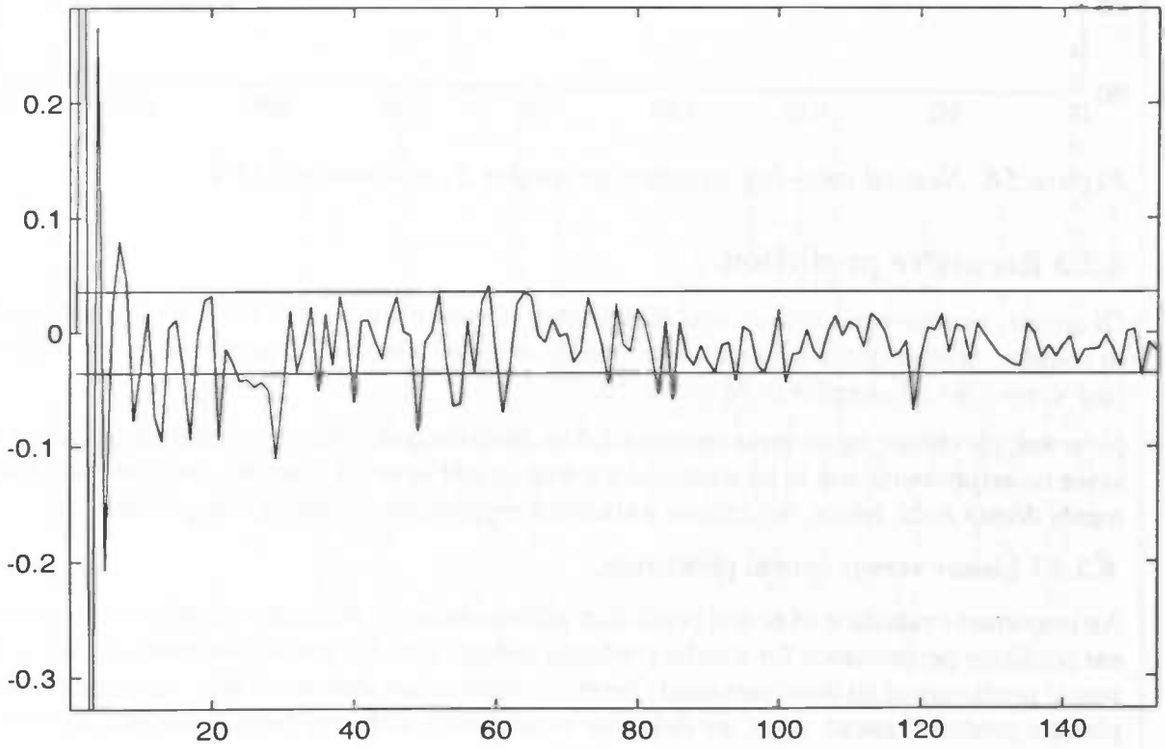

Figure 57. EMG segment partial autocorrelations: snapshot

Based on Figure 57., we chose an order of 62, since, from there on, most of the partial autocorrelations are within the two boundaries (hence ignoring the burst at about order 1000). Determination of the AC–value, yielded an order of 137. We already know the segment's correlation time to be 3.

8.3.2 One–lag prediction.

For verification with the synthetic EMG, we want to look briefly at the one–lag prediction capabilities of a neural network. For small orders (e.g. at the correlation time), predictions turned out to be already quite accurate (Figure 58.).

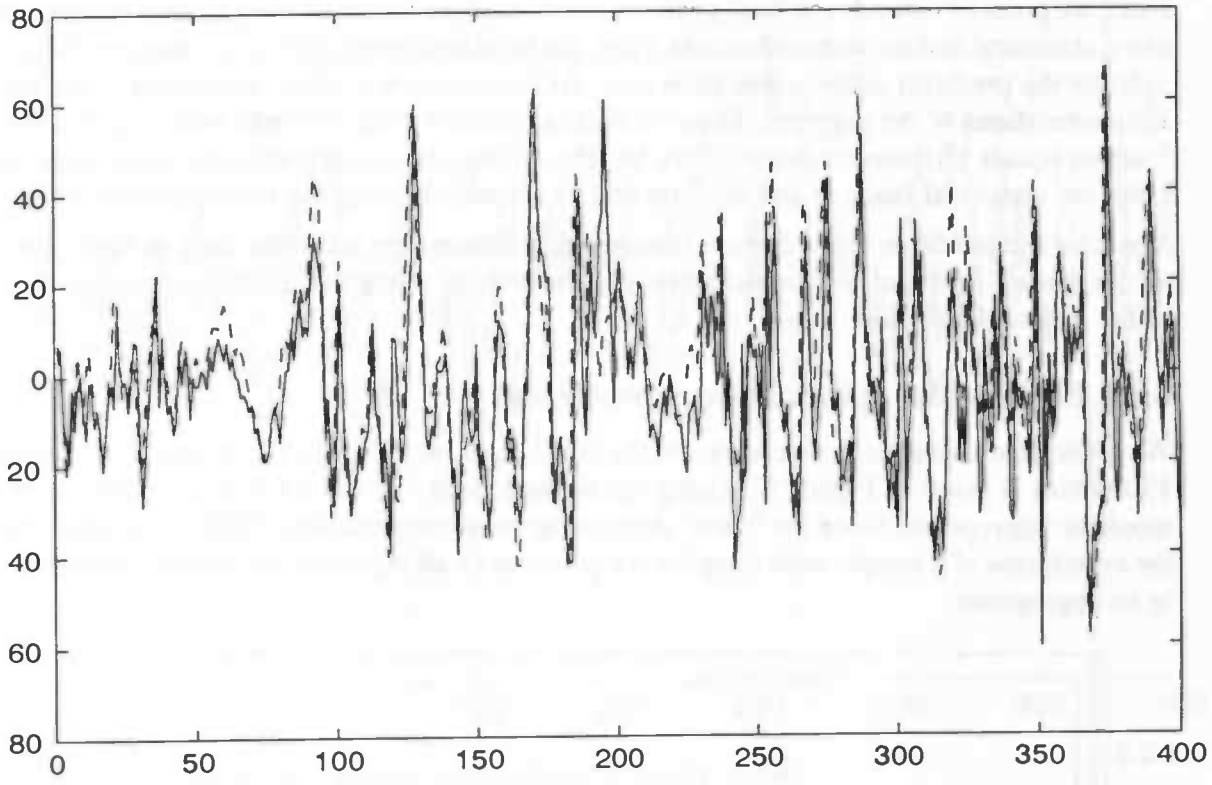

Figure 58. Neural one–lag prediction (order 3) of human EMG

8.3.3 Recursive prediction.

Of course, recursive prediction capabilities are much more interesting. Since it must be possible to compare predictions in a quantitative manner, we used a prediction performance measure: the rms–error over all samples to be predicted.

Note that the choice for an error measure differs from the one made previously in Section 8.1.5; since no error course has to be tracked, like with neural network learning, the mentioned arguments do not hold; hence, we choose a standard measure that is easily computable.

8.3.3.1 Linear versus neural prediction.

An important evaluation of neural prediction performance can be made, when looking at the linear predictor performance for similar predictor orders. The rms prediction errors for linear and neural predictors of all three previously mentioned orders are shown in Table 3. Again, 400 samples are predicted ahead. Also, we did some experiments with very large linear predictor order, the results of which are included into the table.

Predictor order	Linear prediction	Neural prediction
3	1.02	1.54
62	1.02	1.05
137	1.02	1.23
150	1.02	
1500	0.99	
2000	0.97	

Table 3 Recursive neural vs. linear prediction

Interpreting the results, we see the recursive linear prediction become better for increasing predictor order. Even when the predictor order is far greater than the AC–order (which had the largest magnitude of all three criteria), we see the fit become better and better (Figure 59. and Figure 60.).

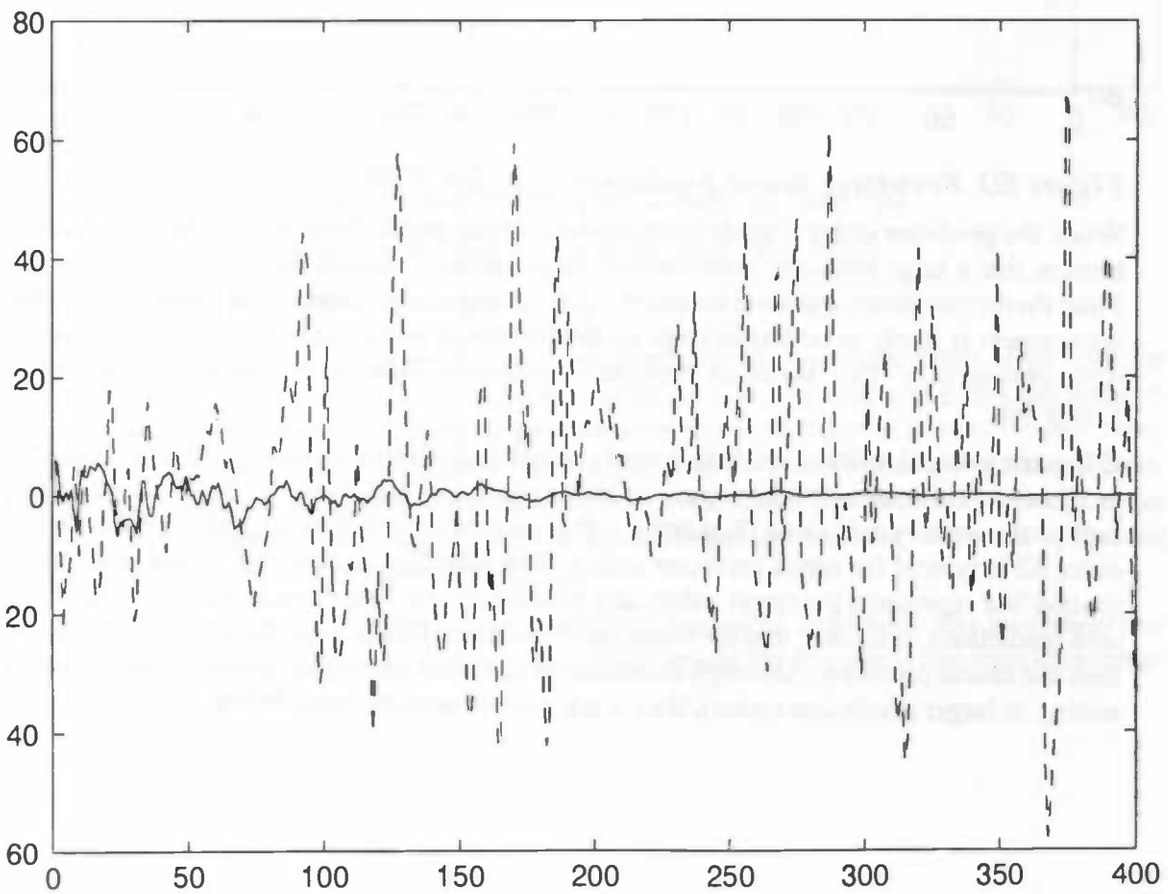

Figure 59. Recursive linear prediction of order 137

This suggests, that we were wrong in our ignorance of bursts above partial autocorrelations 1000.

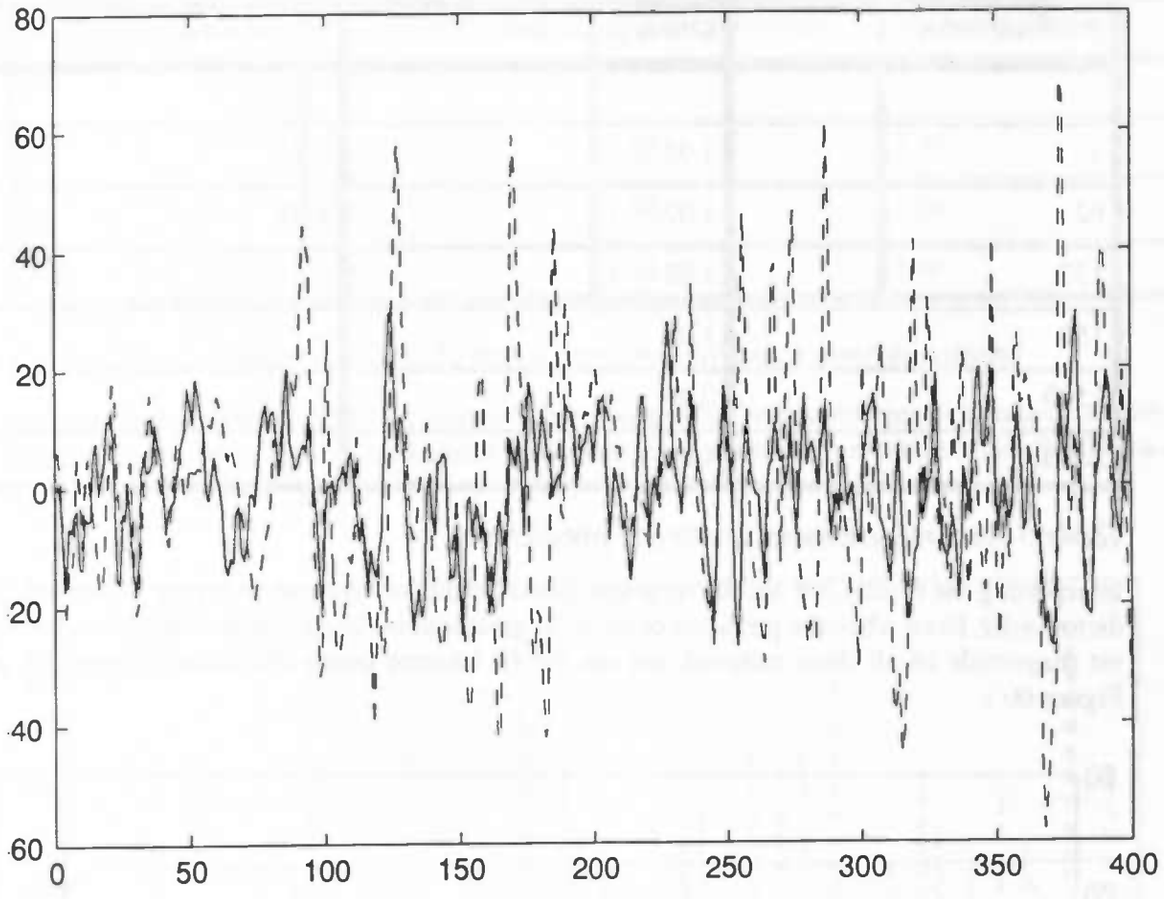

Figure 60. Recursive linear prediction of order 2000

When the predictor order is made large enough, linear predictions seem to become better. It is known, that a large predictor order implies large variance (hence, the AC–criterion, or Akaike Final Prediction Error, was once invented ...), so tuning a large order linear predictor to a particular segment is likely to results in large prediction errors when the predictor is confronted with new, unseen data. This sheds an important (different) light on the really neat prediction in Figure 60..

Comparing neural to linear prediction errors numerically (Table 3), we see the latter perform best in all cases. This numerical advantage of linear predictors can, however, be weakened when we look at the actual predictions (Figure 61.): this snapshot of the first 80 predictions for predictor order 62 is typical for much predictor orders. The solid line denotes the neural prediction, the dashed line represents the target value, and the dashed–dot line corresponds to the linear recursive predictions. It is clear, that the linear predictor much less follows the actual signal dynamics than the neural predictor. Although the neural predictions are frequently out of phase (hence resulting in larger prediction errors), they track muscle activity much better.

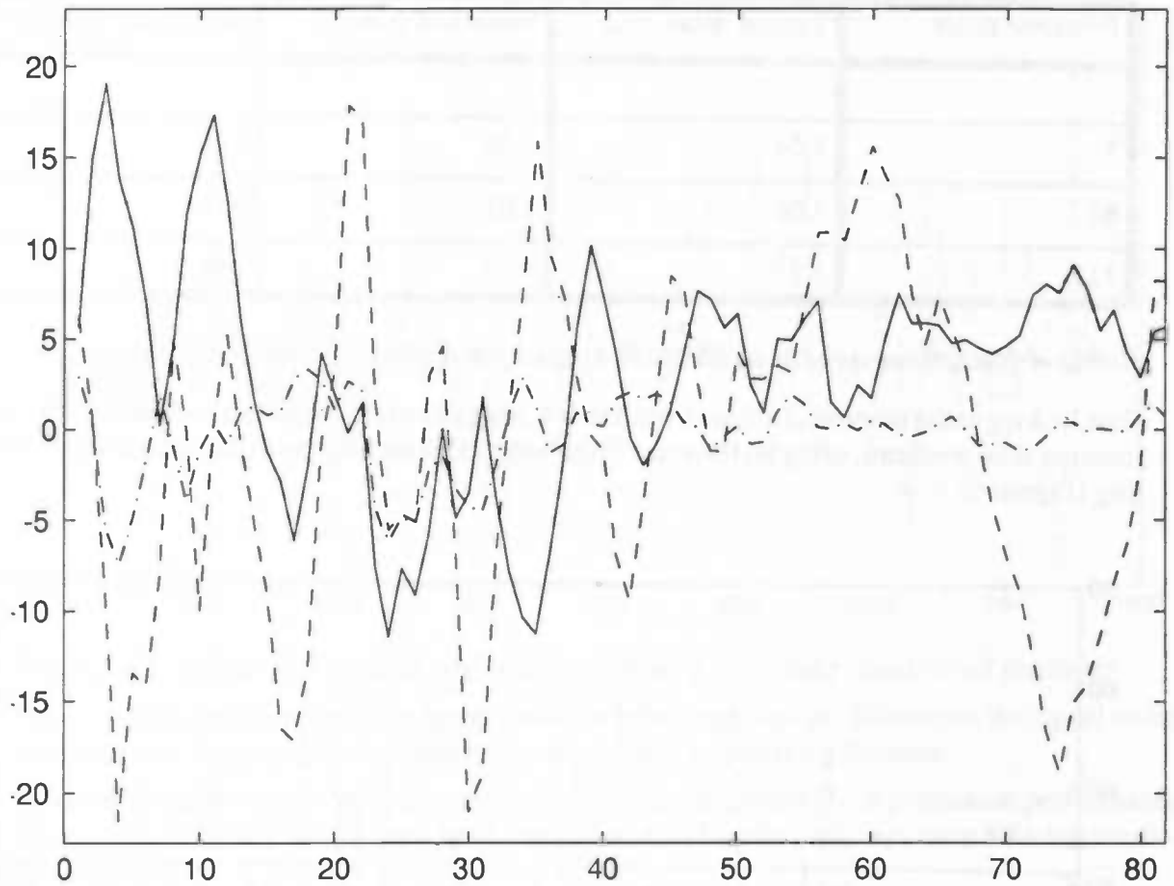

Figure 61. Recursive linear and neural predictions of order 62

Hence, the numerical values from Table 3 have to be treated cautiously.

8.3.3.2 Neural prediction.

Now we focus on different configurations for neural prediction: what is the effect of the addition of statistical and movement features? First, we train a network to predict the EMG just on basis of previous samples (so **only a tapped delay–line** is employed for the provision of inputs). Next, we add statistical features to the network, i.e. we compute the **mean value** and **standard deviation** of the samples in the tapped delay–line, while also the **linear prediction** for those samples is provided as a feature. Finally, **hip extension, knee flexion** and their **derivatives** are provided to the network, along with the previous inputs.

Combined results from all runs (including *no* features, *statistical*, and finally also *movement* features) for all criteria are shown in Table 4. The entries denote the rms prediction errors for recursive 400–lag prediction.

Predictor order	Tapped delay–line	Statistical features	Movement features
3	1.54	2.93	2.56
62	1.05	1.07	1.06
137	1.23	1.04	5.79

Table 4 Recursive neural prediction errors for different configurations

Just looking at the numerical values from Table 4 suggests a reasonable performance of the correlation time predictor, using no features. Visual inspection into the prediction yields the following (Figure 62.):

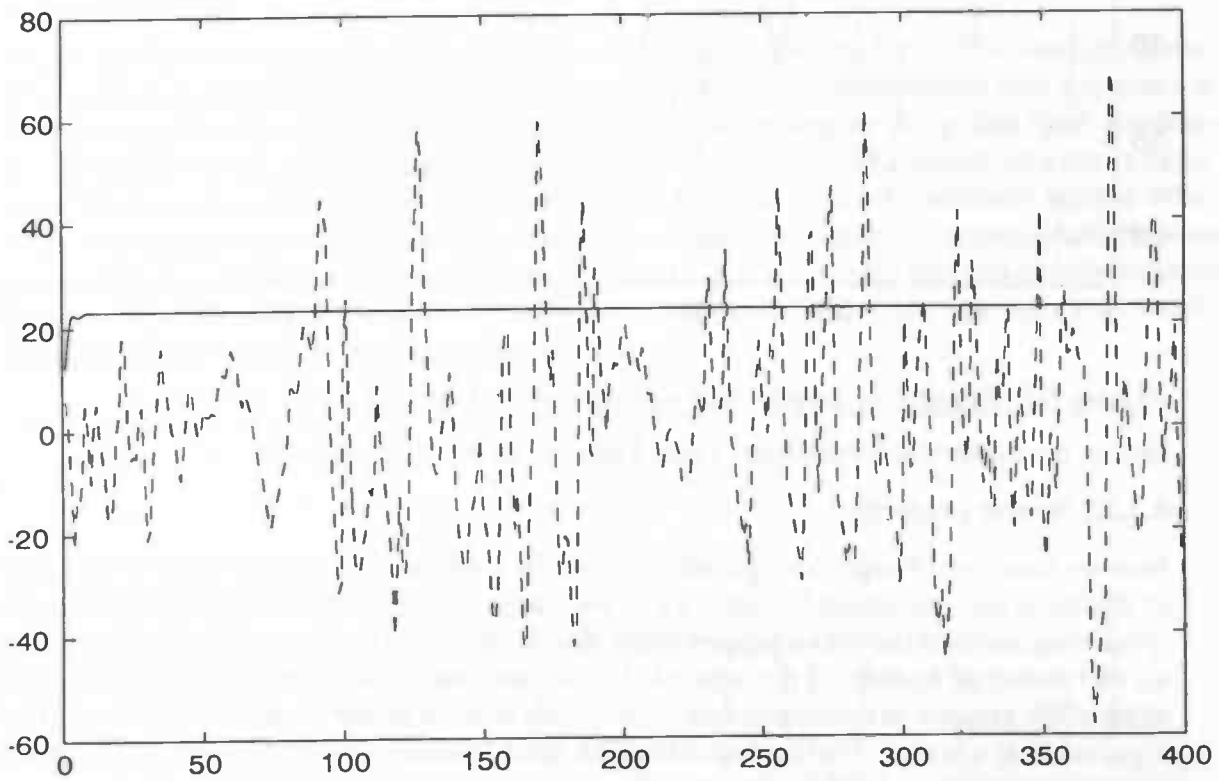

Figure 62. Recursive neural n–lag prediction (order 3) of human EMG

Obviously, using short delay–lines, all signal dynamics cannot be captured properly. Hence, we may question the value of the correlation time criterion, when dealing with signals with (probably) complexer properties. Clearly, the EMG contains much more interesting dynamics than suggested by the former classification as a random signal.

For larger order, neural predictions become better. However, for orders that are too large, artefacts emerge (shown by the increase in rms prediction error for two configurations). For the configuration incorporating just previous samples and statistical features, the opposite is true: tracking becomes significantly better for large order (in this case: the AC–order of 137). This is shown in Figure 63.

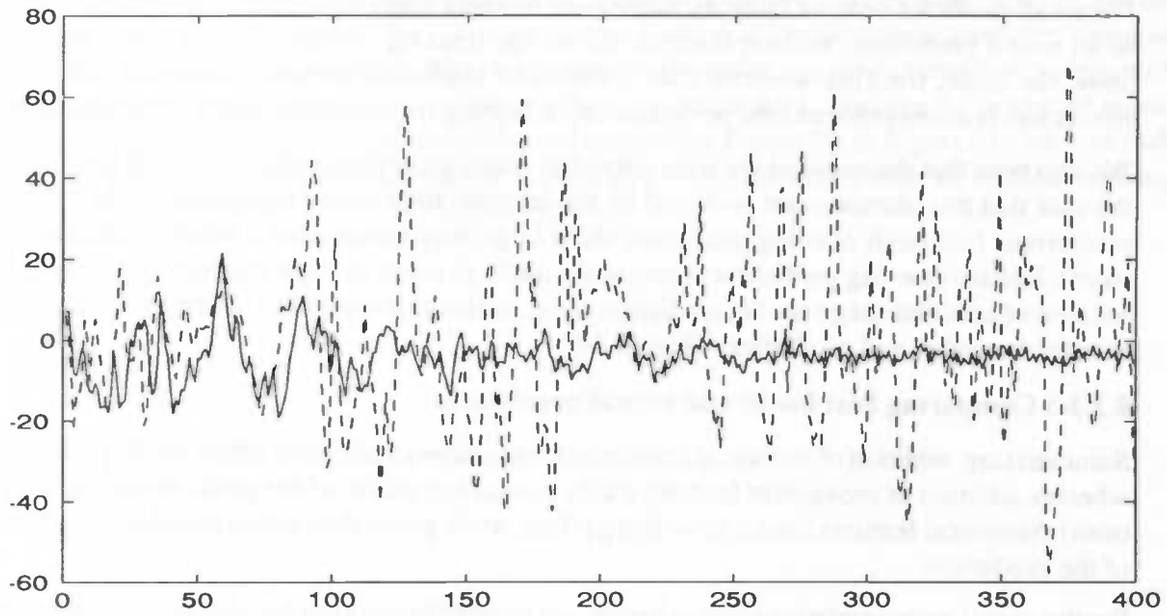

Figure 63. Recursive neural prediction (order 137) using statistical features

We already know that increase in linear predictor order improves predictions on the signal under investigation. Hence, statistical features seem to have a stabilizing function.

When **movement data** is added, consecutively, a dramatic decrease in prediction performance can be observed. It is shown for a predictor order of 137 performing recursive 800–lag prediction, in Figure 64.

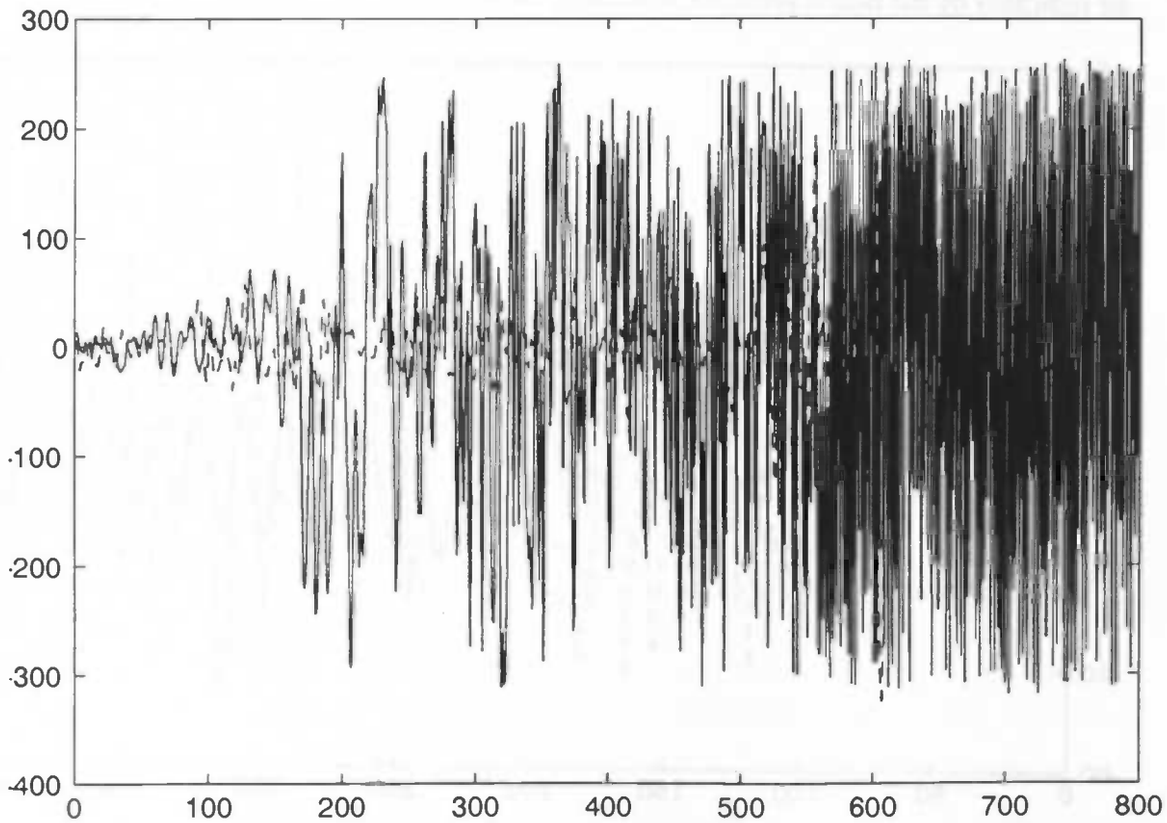

Figure 64. Recursive neural prediction of order 137 using all features

For small predictor orders (Table 4), movement features seem to enhance performance: small–order neural prediction, without features, shows bad tracking. As small–order linear predictors show the same, tracking worsens after addition of statistical features. However, addition of movement features enhances the performance, indicating improved dynamics of the predictions.

We also note that the oscillations arise only after many recursive predictions, so it may well be the case that this phenomenon is caused by the accumulative error propagation in the recursive predictions (although one–lag prediction show large rms–errors over a whole prediction segment). Recursive *n*–lag prediction is obviously likely to result in large artefacts on the long run, because of accumulating one–lag prediction errors, ultimately leading to saturation of the neuron transfer functions and oscillations (Figure 64.)

8.3.3.3 Comparing best linear and neural predictors.

Summarizing, addition of statistical features seems to have a stabilizing effect on the predictions, whereas addition of movement features easily causes instability of the predictions (i.e. oscillations). Statistical features have a smoothing effect, while gonio data seems to enhance dynamics of the prediction.

For the partial autocorrelation order criterion, we observe best results for all three configurations. This is likely to be the best choice for a neural predictor, since it incorporates both dynamics in the same (amplitude) range as the target signal, while also keeping the delay–line short. Remember from Section 6.1.1, that a large number of degrees of freedom in a neural network is likely to result in worsened generalization. A predicted signal that shows dynamically similar behaviour as the target signal, but doesn't track it very accurately, can be said to approximate the *area under the curve* of the original signal. From Section 3.2.3.4, it is known that this already suffices for an estimation of exerted muscle force. Hence, the position within the walking cycle should be indicated by the neural predictor reasonably well.

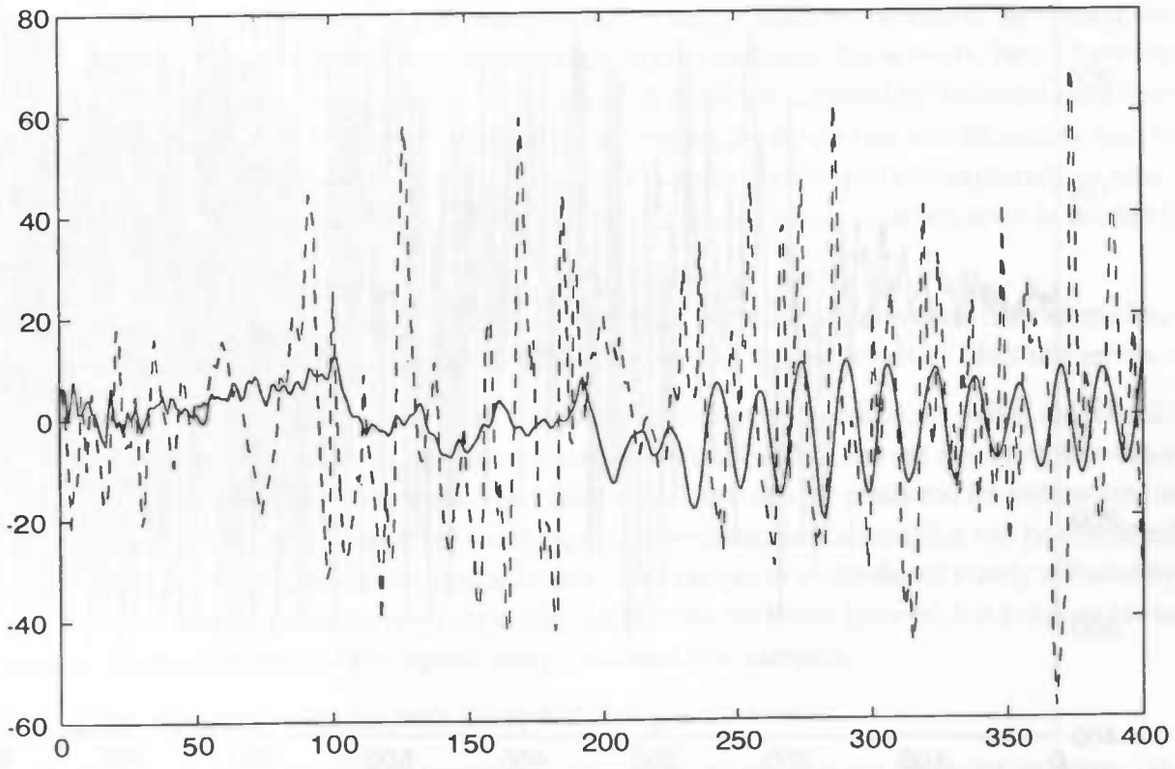

Figure 65. Recursive neural prediction of order 62 using all features

Finally, we can compare best linear and neural predictors, keeping the predictor order within the range of 137, i.e. we compare the linear predictor of order 137 to the neural predictor, using all features, of order 62 (Figure 65.). We see a (somewhat) smaller rms prediction error emerge with the linear predictor; however, remembering the previous remarks concerning predicted signal dynamics and timing of the walking cycle, and comparing Figure 59. to Figure 65., we conclude that the optimal neural predictor can be applied most conveniently within an artifical walking system.

Chapter 9.
Conclusions

Summarizing our experimental results, we can state the following:

1. For signals with simple (linear) dynamics, the correlation time can be used to determine the number of delays in the input tapped delay–line (in temporal neural processing).

2. Delay–lines of length shorter than this number do not capture the dynamics adequately, which is reflected in an increasing mean steady–state error when shortening the delay–line in this region.

3. Longer delay–lines do not provide additional temporal information, since the samples to be predicted and the additional ("older") samples are no longer linearly correlated. Increasing the delay–line also yields an increasing number of degrees of freedom in the network, hence have the potential of worsening generalization. When not all dynamics is captured by the correlation time length delay–line (e.g. with complexer signals), increasing the delay–line will ultimately lead to improved generalization (when important temporal features can suddenly be captured); in practice, the correlation time hypothesis will only reflect a (short) nearly constant level in the MSS test error.

4. This level proved to be longer, as the correlation time of the signal increased. Shorter realization lengths of the correlation time test signals showed also shorter levels of MSS test errors.

5. The criterion is less suitable for signals containing complexer dynamics, i.e. when samples do not only depend on (relatively few) previous ones in a linear fashion, but are also related to other samples (farther ahead) in other ways. The EMG–signal, that is to be predicted for utilization for FES–control, shows such complexer (walking cycle) periodicities, although it can be classified as a random signal. Its correlation time indicates that a sample to be predicted is only affected by very few previous samples; the walking cycle periodicities are hence ignored. It is not possible to capture all the dynamics of the signal, using just these few samples.

6. The last statement holds for both linear and neural prediction:

i. the linear recursive prediction becomes better, when the order of the predictor increases (although the variance of the predictor will become worse); it is better to look at the order that is

computed using the Akaike AC–criterion, than to use the correlation time for this purpose; for small order linear predictors, recursive prediction leads to "averaging and fading out to zero" of the prediction;

ii. the neural recursive prediction gets disturbed when the size of the tapped delay–line (comparable to the order of the linear predictor) is chosen too large; adding statistical features, like EMG standard deviation and the linear prediction of the previous samples, has the effect of stabilizing the neural prediction, probably due to the improved accuracy of a high order linear prediction. For small orders of the neural predictor, addition of movement features can be helpful in restoring the dynamics of the signal; for larger delay–lines, gonio–features cause major deviations, and can bring the network into oscillation;

7. Best neural and linear recursive predictions shown root mean square prediction errors that are comparable. However, the linear predictors do not lie within the same amplitude range of the signal to be predicted, but quickly fade out to zero. Neural predictions, on the other hand, much more reflect the same dynamical behaviour (i.e. show time courses of similar amplitude and appearance) as the prediction signal, but do not track the signal all too accurate (hence causing large rms prediction errors). For the application at hand, where only the "amount of activity" is to be known for adequate stimulation, neural prediction seems to be much more valuable than linear prediction: because of large time constants in the system to be controlled (the human motor control system), predictions have to be made many samples ahead. At large prediction horizons, linear predictors have already settled to zero, whereas neural predictors still show EMG–like dynamics;

8. When the prediction horizon is taken too large (e.g. 800 samples ahead), neural prediction is likely to cause large prediction errors, due to oscillations. This is especially true when large delay–lines are chosen, or when movement features are included;

9. The frequency hypothesis turned out to be wrong. This is probably due to the fact, that the power spectrum of the original signal can be affected by the sampling operation, hence yielding wrong ratios.

10. Recursive n–lag prediction brings about the accumulated propagation of one–lag prediction errors. When dealing with signals that slowly "drive" towards maximum and minimum magnitudes, neuron transfer function saturation and rounding errors can cause oscillations, which casts some doubt on the usefulness of this way of n–lag forecasting.

Chapter 10. Discussion

Finally, a brief reflection on the methods used and approaches taken in our research seems in order. Moreover, we have some ideas concerning future research into the solution of the problem of artificial human walking.

10.1 Evaluation of utilized methods.

Probably due to the very small amount of muscle activity–rest transitions in the train segment (i.e. 2 transitions, Figure 15.), the transition from large to small activity (in the raw EMG prediction segment, this transition occurs at a prediction horizon of about 500 samples, NO TAG) could not be learned and generalized by the network very well. The effect of movement features seems to be, to enhance and maintain the wildly fluctuating time course of the predictions, hence leading to oscillations. Using more activity–rest transitions could have a stabilizing effect, just as the addition of statistical features is seen to have.

In some early experiments (not reported here), we observed oscillating predictions for small order (5 and 10) recursive neural predictors, having all (statistical and gonio) features included. The effect seemed to be stronger when the network is trained longer: a 500 cycles trained neural predictor of order 5 showed oscillations in the region of heavy muscle activity, and was fading out to zero as the resting period was reached. Trained for 2000 cycles, a neural predictor of order 100 showed increasing and decreasing oscillations, showing the same "period of activity" as the original EMG. When a predictor of order 10 was trained for 10000 cycles, a neverending oscillation was reached. In other words: gonio features and the recursive prediction methodology probably cause potential oscillations (prediction instability), but the effect may depend heavily on the number of training cycles that is used.

The statement w.r.t. the small amount of transitions that are included in the training data, could also be made concerning the (equally frequent) opposite transitions; however, this transition comes about much smoother and is easier to generate (during prediction), using only a small amount of previous samples.

It is necessary for practical utilization of a neural EMG–predictor to predict many samples ahead, due to large time constants in the human motor control system. Hence, we focused on the 400–lag recursive prediction capabilities, rather than the task of one–lag prediction (which was performed relatively easy by both neural and statistical predictors).

10.2 Neural time series prediction.

We would like to emphasize our approach to neural time series prediction: the main advantage of a neural predictor over a classical linear predictor, besides the possibility to add features of the input data along with previous time samples, lies in its capability to draw nonlinear temporal relations (Sections 4.1.2, 5.2.2). Similar ideas, concerning the establishment of nonlinear dependencies between consecutive samples, emerge in chaotic time series analysis [46]. The idea, that the fields of chaotic dynamics and neural networks have to be bridged, has gained momentum since the appearance of the proceedings of the Santa Fe contest in time series prediction [44].

Hence, we used methods from chaotic dynamics to serve as an alternative benchmark to conventional linear prediction. New insights from the field of chaotic dynamics could also turn out to be very useful when looking for a more structured approach to neural time series prediction. That there it is indeed reason to expect a rational, structured approach to neural system design to be feasible, is proven most elegantly by the implementation of a statistical Principal Components Analysis by a self–organizing neural network, described in Section 6.2.

The temporally *local* nature of a TDL–neural network enables a straightforward comparison to (non)linear classical predictors. They also exhibit, however, the same disadvantages w.r.t. the inability to capture all input data dynamics, using only a limited number of previous samples. A more *global* approach to neural time series prediction, e.g. by employing FIR–MLPs and recurrent neural networks, could probably be used succesfully to overcome these problems.

10.3 Applicability in a control system for artificial human walking.

It seems right to describe the position of a future neural predictor within a total system for FES–control. We could be asking ourselves the following question: "Why should we perform time series prediction when stimulation is performed open loop? It is already known which muscles will be active at a given instant!" This is indeed the case. We think of the avoidance of excessive muscle fatigue as the main motivation for EMG–based predictive control of FES: both decisions, as to which muscle(s) (groups) to select from the pool of possible alternatives ("don't stimulate a muscle in order to force a certain, e.g. stabilizing, movement, when it is already to be stimulated a short time ahead, use its neighbour instead ...") and when to compensate for predicted muscle fatigue can be made using EMG–prediction.

Moreover, accurate prediction of *one* muscle's activity is nice; practical utilization, however, requires the (simultaneous) prediction of multiple EMG–activities from a limited set of movement data and EMG–history. For this much more complex (and mathematically underdetermined) problem, a flexible and extendable optimization technique like neural networks technology is likely to be very well suited. We are just at the beginning ...

10.4 Future prospects.

The artificial walking system component we focused on, is but a small part of a complete neural FES–controller. Much work has to be done w.r.t. on–line modification of muscle stimulation patterns, e.g. based on neural EMG–predictions. In fact, accurate prediction of muscle activity many milliseconds ahead is expected to be of major importance in the compensation for muscle fatigue arising with FES.

Finally, we noticed the formidable multidisciplinary nature of the problem of artifical human walking from our literature survey (Chapter 2). We believe that a "holistic view" to the problem is to be adopted: many initiatives have to be taken to glue all related research(ers) in several subdiciplines of the problem together, hence benefiting from each other's expertise and enthusiasm, and hopefully leading to the state of synergism that seems indispensable for (ultimately) solving the problem.

Chapter 11. Bibliography

[1]Adaptive feedforward control of cyclic movements using artificial neural networks – Åbbas/ Chizeck, IJCNN '92, Vol. 2, 832 – 837 (1992)

[2] Trajectory formation of arm movement by cascade neural network model based on minimum torque–change criterion – Kawato/ Maeda/ Uno/ Suzuki, Biol. Cybernetics 62, 275 – 288 (1990)

[3] Advancing step by step – Kobetic, IEEE Spectrum (1994)

[4] Control of locomotion in handicapped humans – Popovic, Neural Networks – concepts, applications & implementations Vol. 2, Ch. 6 (1990)

[4]a Evaluation of Adaptive Logic Networks for control of walking in paralyzed patients – Kostov/ Stein/ Armstrong/ Thomas, 14th Ann. Int. Conf. IEEE EMBS, Vol. 4, 1332 – 1334 (1992)

[4]b Learning of EMG–patterns by Adaptive Logic Networks – Kostov/ Popovic/ Stein/ Armstrong, 15th Ann. Int. Conf. IEEE EMBS, Vol. 15 Prt. 3, 1135 – 1136 (1993)

[4]c Sensory nerve recording for closed–loop control to restore motor functions – Popovic/ Stein/ Jovanovic/ Dai/ Kostov/ Armstrong, IEEE Tr. BME, Vol. 40, No. 10, 1024 – 1031 (1993)

[5] Neural network generation of muscle stimulation patterns for control of arm movements – Lan/ Feng/ Crago, IEEE Tr. Rehab. Eng, Vol. 2, No. 4, 213 – 224 (1994)

[6] A neural network solution for bipedal gait synthesis – Lee/ ElMaraghy, IJCNN '92, Vol. 2, 763 – 768 (1992)

[7] A neural network approach to electromyographical signal processing for a motor control task – Lester/ Fernandez/ Gonzalez/ Barr, ACC '94, 2548 – 2552 (1994)

[8] Combined statistical study of joint angles and ground reaction forces using Component and Multiple Correspondence Analysis – Loslever/ Laassel/ Angue, IEEE Tr. BME, Vol. 41, No. 12, 1160 – 1167 (1994)

[9] Dimension–reduction mapping techniques: an application to electromyogram cluster analysis– Matthews, Scientia Electrica Vol. 35, No. 2, 1 – 60 (1989)

[10] Real–Time neural network control of a biped walking robot – Miller, IEEE Control Systems Mag. Vol. 14, No. 1, 41 – 48 (1994)

[11] A neural network representation of electromyographic and joint dynamics in human gait – Sepulveda/ Wells/ Vaughan, J. Biomechanics Vol. 26, No. 2, 101 – 109 (1991)

[12] Control of voluntary limb movements by using a fuzzy system – Young/ Fan, 32nd CDC '93, 1759 – 1764 (1993)

[13] The clinical use of surface EMG – Hermens/ Boon/ Zilvold, Medica Physica No. 9, 119 – 130 (1986)

[14] Effects of electromyographic processing methods on computer–averaged surface electromyographic profiles for the Gluteus Medius muscle – Kleissen, Physical Therapy Vol. 70, No. 11, 716 – 722 (1990)

[15] Simultaneous measurement of surface EMG and movements for clinical use – Kleissen/ Hermens/ den Exter/ de Kreek/ Zilvold, Med. & Biol. Eng. & Comput., No. 27, 291 – 297 (1989)

[16] Control of FES–induced cyclical movements of the lower leg – Veltink, Med. & Biol. Eng. & Comput., No. 29, NS8 – NS12 (1991)

[16]a Adaptive neural network control of FES–induced cyclical lower leg movements – Stroeve/ Franken/ Veltink/ van Luenen, IFAC AI in RT Control, 25 – 30 (1992)

[17] Automatic stance – swing phase detection from accelerometer data for Peroneal nerve stimulation – Willemsen/ Bloemhof/ Boom, IEEE Tr. BME Vol.37, No. 12, 1201 – 1208 (1990)

[18] Identification and control of dynamical systems using neural networks – Narendra/ Parthasarathy, IEEE Tr. Neural Networks Vol. 1, No. 1, 4 – 27 (1990)

[19] A theory for neural networks with time delays – de Vries/ Principe, Advances in Neural Information Processing Systems 3, Morgan Kaufmann Publ., San Mateo, California, 162 – 168

[20] Surface EMG – Hermens, abstract of Ph. D. thesis, J. Rehabilitation Sciences 5, No. 3, 90 – 91 (1992)

[21] The 3D–analysis and prediction of human walking – Koopman, Ph.D. thesis, Un. of Twente, NL (1989)

[22] Control system design for walking neuroprostheses – Franken, Ph.D. thesis, Un. of Twente, NL (1994)

[23] Finite State Control in functional neuromuscular stimulation – Mulder, Ph.D. thesis, Un. of Twente, NL (1991)

[24] Modeling, control and optimization of (FES–induced) paraplegic gait: Literature Review – van de Belt/ Franken, Report no. EL/BIO 91/16 – WB/BW 91/1, Un. Twente (1991)

[25] De toepasbaarheid van het CDG–systeem bij het evalueren van steunzolen ter correctie van platvoeten – Nawijn, Master's thesis, Un. Nijmegen (1992)

[26] Infotronic product information package – Infotronic Medical Engineering (1995)

[27] Disorders of posture and gait – 10th Symp. Soc. Post. Gait Res. (1990)

[28] Electrophysiological kinesiology – Wallinga/ Boom/ de Vries, eds., 7th cong. IS Electr.Phys. Kines. '88, Int. Congr. Ser. No. 804 (1988)

[29] Praktische biomedische signaalanalyse I, II, lab–manual – Rutten, Lecture Notes, Univerity of.Twente (1989)

[30] Biomedical signal processing Vol. I – Cohen, CRC Press Inc., Boca Raton, Florida (1986)

[31] Digital Signal Processing: a practical approach – Ifeachor/ Jervis, Addison Wesley (1993)

[32] Fundamentals of statistical signal processing: estimation theory – Kay, Prentice Hall Int. (1993)

[33] Neural Networks – Haykin, MacMillan (1994)

[34] Artificial neural systems – Zurada, West

[35] The physiological basis of Rehabilitation Medicine – Downey et al. (1994)

[36] Rehabilitation Medicine: principles and practice – de Lisa ed. (1993)

[37] Clinical Electrotherapy – Nelson/ Currier, Appleton & Lange (1991)

[38] Human Motor Control – Rosenbaum

[39] Dynamics of human gait – Vaughan/ Davis/ O' Connor, Human Kinetics Publ., Champaign Illinois (1992)

[40] The biomechanics and motor control of human gait – Winter, Un. of Waterloo Press (1987)

[41] Modern control systems – Dorf

[42] Time series: theory and methods – Brockwell/ Davis (1987)

[43] Time series – Kendall/ Ord, Edward Arnold, Wiley (1990)

[44] Time series prediction: forecasting the future and understanding the past, Weigend/ Gershenfeld eds., Addison Wesley (1993)

[45] Chaos in dynamical systems – Ott, Cambridge University Press (1993)

[46] Coping with chaos – Ott/ Sauer/ Yorke, Wiley (1995)

[47] Artificial neural networks: forecasting time series – Rao Vemuri/ Rogers, IEEE Computer Society Press Nr. 5120–05 (1994)

[48] Nonlinear time series: a dynamical system approach – Tong, Oxford Science Publ. (1990)

[49] Beeldbewerking – Moddemeijer, Syllabus Rijks*universiteit* Groningen (1995)

[50] Chaos en Tijdreeksanalyse – Takens, Syllabus Rijks*universiteit* Groningen (1994)