FACTA UNIVERSITATIS (NIŠ)
Series: Electronics and Energetics vol. 11, No.1 (1998), 1-24

IDENTIFICATION OF RULES FOR A RULE–BASED CONTROL OF WALKING

Slavica Jonić and Dejan Popović

Abstract. In this paper several nonparametric supervised machine learning (ML) techniques for automatic designing of rules for a rule–based control (RBC) of functional electrical stimulation (FES) assisted human walking are described. The application of an artificial neural network with radial basis functions, which works the best for the required pattern matching, is presented. ML can describe behavior of the system under certain conditions by performing spatio–temporal mapping of input–output variables and store it in appropriate form for use in real–time control. This approach is applicable whenever there is a skilled repetitive action or a process involving human or natural control. RBC relies on a finite state model of walking where the process is described using sensory information and motor activities as state variables. Sensory signals are used as inputs to the ML. Since the muscles are operating as low–pass filters with respect to neural inputs, the prediction of the control signals driving muscles and the prediction of sensory signals are used as outputs from the ML. The inputs and outputs for the learning used for this study are obtained from simulation of a fully customized musculo–skeletal model described in details elsewhere [28]. The supervised learning task for an ML is to extract all invariant characteristics from the relationship between the provided inputs and outputs of the system (examples) and to store them in form of decision tree which can later produce approximated outputs when only inputs are provided. Since MLs generally do not have any limitations regarding the number of inputs and outputs, this approach is appropriate for multi–input–multi–output systems. The results provide good basis for design of robust control systems for FES – assisted walking.

1. Introduction

A review of a series of supervised nonparametric techniques for pattern matching is presented in this paper including the following: 1) Multilayer

Manuscript received February 7, 1998.

The authors are with Faculty of Electrical Engineering, University of Belgrade, Bulevar revolucije 73, 11000 Belgrade, Yugoslavia.

perceptron (MLP) [30] type of Artificial Neural Networks (ANNs); 2) Radial Basis Function (RBF) type of ANNs [6]; 3) Adaptive Logic Networks (ALNs) [2,3,17]; 4) Fuzzy Logic [40]; 5) Inductive Learning (IL) [10,25,31]; 6) Adaptive–Network–Based Fuzzy Inference system (ANFIS) [11,13]; and 7) combination of RBF network and IL [12,24]. The task in all cases was the same: determination of a connectivism between the sensory information describing walking of a human and activities of muscles that must be required to generate joint torques needed for walking. This connectivism can be used as the essence of a knowledge base of a rule–based controller.

Existing controllers for walking frequently use analytical, dynamic model of the human body. Dynamic analysis relies on a detailed and correct model, all parameters of the model must be known with a certain accuracy, and the trajectory has to be known. The body is usually presented as a system of rigid segments connected with rotational joints and powered by a set of joint actuators [39]. However, even the most complicated model proposed is distinct from the reality with respect to the number of degrees of freedom, variability of inertial properties of the system and specific nonlinear effects imposed by muscles and tendons. Simulated actuators typically implement visco–elastic, Hill–based model taking into account specific characteristics of human musculo–skeletal plant [37]. Most models use a simplified model of the foot, including the toe phalange at best. Flexibility of the foot itself, compression of the soft tissue during the heel impact and many other important events are not considered. There is no model for locomotion that takes into account the characteristics of the segmented spinal column, upper body, neck and the head when analyzing locomotion. Designing the control system for functional electrical stimulation (FES) assisted locomotion of humans with spinal cord injury (SCI) requires taking into account the changes caused by the injury.

Since there are many possible solutions when calculating of control signals driving muscles for the known desired joints trajectory, a criterion for selection of a single solution has to be adopted. In engineering field, the appropriate solution is selected by using criterion functions, or performance index of the system. This function may include total energy used, time required for an operation, power and torque requirements, changes in the acceleration of the system, forces acting at specific parts of the body, etc. In most cases the term optimal solution' is used, but actually this is usually a preferred solution. In biological organisms the selection of preferred solution is goal dependent and it comes as a result of self-organization and learning (e.g., when walking over ice the balance and friction forces are optimized to decrease the chance of falling; for walking while carrying cup with a hot coffee that should not spill, jerks of the upper body are minimized; for

marathon running the energy cost per distance is minimized, etc.). Without optimization criterion all solutions are equally possible and there is no unique solution to the problem.

An alternative nonparametric method suggested in literature [34] is to use pattern matching of sensory and motor states, defined as inputs and outputs. The approach is similar to, although it was not influenced by, the hierarchical organization of biological motor control in humans and animals [34]. A nonparametric model of process uses a finite state model and a rule–based control (RBC) [33,34]. The states and rules can be either: 1) "hand–crafted"; or 2) automatically generated using artificial intelligence (AI) tools.

In the "hand–crafting" approach states and control rules are defined heuristically. Such a system is implemented to real FES–assisted locomotion application and the quality of the resulting gait is assessed, necessary adjustments are completed and the process is repeated again until a satisfactory gait is achieved [15]. This "trial and error" method is very time consuming and requires the subject's presence at every iterative cycle. The most important disadvantage of this method is that the performance of the resulting control system depends on an experts abilities to express acquired knowledge explicitly in states and rules.

The transfer of human knowledge about locomotion to the computer knowledge base relies basically on identification and representation of invariant features of functional movements. While the method works satisfactorily in modeling of the gait of non–impaired persons, its application to handicapped persons is not as successful because each case of sensory–motor deficiency is specific in many ways. This may be the reason that, although expert systems based on the transfer of human knowledge to the machine at the conscious level are in wide use today, machine control by skill–based AI systems is still in the stage of early development. The most serious problem on the way to faster development of skill–based control system is the description of human skill in machine comprehensible form. By definition, automatic features of motor skills are expressed as spatio–temporal events. Consequently, capturing of such knowledge requires a new type of identification methods which rely to a large extent on external manifestations of neuromotor control mechanisms.

A complex movement can be improved with training, thus the acquisition of skills can be considered as an optimization process taking place at a number of levels in the central nervous system. A skilled subject or therapist manually controlling FES – assisted locomotion can develop a set of motor control rules that are near optimal. If these rules can be copied and

stored, they can be used to form a controller to reproduce the near optimal movements. This represents a basis for another approach to control rule definition: transfer of the knowledge, stored in form of a skill, from a skilled subject or therapist to a computer, using automatic process that employs machine learning (ML) algorithms.

Two methods are in common use to determine the input–output mapping. Induction relies entirely on the ability to transfer human expertise into form understood by a machine and results in expert systems. This method has been applied extensively for design of artificial limbs [1,4,27]. The expertise required for designing a knowledge–base (expert system) for real–time control is gained by analyzing the sensory patterns acquired while able–bodied subjects, or amputees walked at different speeds and under various conditions. The sensory patterns are coded (e.g., single threshold, multi threshold, timing, local vs. absolute minimum or maximum, etc.), and the rules defining the relationship between sensory patterns and necessary activities are built into the controller.

The fast development of artificial and computational intelligence techniques, such as artificial neural networks, fuzzy logic and genetic algorithms, brings new approach to the control system design problem formulation and solution. Following the early work of Michie and Chambers [22] on an algorithm known as "Boxes" implemented in the "pole balancing" paradigm, Kirkwood et al. [14] proposed the use of inductive learning technique for the upper level controller for FES–aided walking of subjects with incomplete SCI. They evaluated the use of induction and traditional transducers for automatic generation of control rules by cloning the skill of the subject with SCI in manually controlling a simple two–channel–per–leg FES–system for paraplegic walking, such as one described by Kralj and Bajd [18]. In a continuation of this work, Heller [10] evaluated the use of inductive learning technique in controlling the swing-through walking of paraplegic subjects. Veltink et al. [35] have used a backpropagation multilayer perceptron network for reconstructing muscle activation patterns in the walking cycle on the basis of signals recorded from external sensors (goniometers and foot–switches).

The most important question of a generalization from ML technique training to a real–time control application remained unanswered in most of the works described above. After preliminary results demonstrated possibility of using neural networks for designing control rules for FES–assisted walking of subjects with incomplete SCI [16,25], a similar approach was adopted as the basis for this study.

The muscles are operating as low-pass filters with respect to neural inputs.

Dynamics of muscles can be characterized by rise time of approximately 50 to 100 ms, depending on the muscle type. Muscle activity is delayed after the neural signal for about 30 to 50 ms. Those dynamic features of a musculoskeletal system impose that a command signal precede the required muscle activity when a real–time control is to be implemented. The recognized sensory combination which precedes the muscle activity allows sufficient time for turning on the stimulation, and for the muscle to contract. A RBF network is applied in this study as an example of predicting of muscle activation pattern and joint angle from preceding sensory data.

2. Artificial neural networks for determination of rules

ANN elements are inspired by biological nervous systems. The ANN is composed of many simple elements working in parallel, and the network function is determined largely by the connections between them. During the learning process to perform a particular function the values of connections between elements are adjusted. Complex functions in various fields including identification, control system, pattern recognition, and vision can be performed by an ANN. Neural net classifiers are nonparametric and make weak assumptions about shapes of underlying distributions.

In the areas of biomedical engineering, the feedforward networks, such as the MLP, ALN and RBF networks, are often used. The function of a feed–forward network with single output can be described as the weighted combination of basis functions:

$$f = \xi_0 + \sum_{q=1}^{N} \xi_q \phi_q(\nu)$$

where ν is the input vector and $\phi_q(\cdot)$ represents the q-th basis function connected to the output by weight ξ_q (ξ_0 is the bias of output node).

2.1 Multilayer perceptron

Investigations of the MLPs have been intensified since the formulation of the backpropagation (BP) learning algorithm [30]. It was found that this algorithm represents a simple and powerful tool for adjusting the connections between elements of the networks with arbitrarily complex architecture. Typically MLP consists of several layers of nonlinear processing nodes called hidden layers with a linear output layer. Processing node takes as input only the outputs of the previous layer, which are combined as a weighted sum and then passed through a nonlinear processing function known as the activation function. This activation function is typically sigmoidal in shape.

A MLP with three hidden layers can form arbitrarily complex decision regions and can separate the classes that are meshed together. It can form regions as complex as those formed using mixture distributions and nearest neighbor classifiers [20]. A MLP usually used has a single linear output node and single hidden layer with sigmoidal activation functions. Following the feed-forward network structure, the basis functions for this MLP could be written in terms of activation function $\psi(\cdot)$:

$$\phi_q(\nu) = \psi(d_q^T \nu + \theta_q)$$

where ν is the input vector, d_q is the input weight vector and θ_q is the scalar bias for the q-th node in the hidden layer. All the weights and biases of the network are called network parameters.

The BP learning algorithm is a generalization of a gradient descent algorithm. It uses a gradient search technique to minimize a cost function equal to the sum square difference between desired and estimated net outputs. Derivatives of error (called delta vectors) are calculating for the network's output layer, and then backpropagated through the network until delta vectors are available for each hidden layer of the network. The BP algorithm may lead to a local, rather than a global error minimum. If the local minimum found is not satisfactory, use of several different sets of initial conditions or a network with more neurons can tried.

Simple BP algorithm is very slow because it requires small learning rates for stable learning. There are ways to improve the speed and general performance of BP algorithm. It can be improved in two different ways: by heuristics and by using more powerful methods of optimization. Speed and reliability of BP can be increased by techniques called momentum and adaptive learning rates. The momentum technique helps the network to get out, if stacked in shallow minimum. By the use of adaptive learning rates it is possible to decrease the learning time. By using Levenberg–Marquardt optimization the learning time can be shortened. Its update rule is:

$$\Delta\omega = \left(J^T J + \mu I\right)^{-1} J^T e$$

where $\Delta\omega$ is column matrix whose number of rows matches the number of network parameters, J is the Jacobian matrix of derivatives of network function error, that is difference between desired and estimated net outputs, for each learning pattern to each network parameters. The number of rows in J matches the number of learning patterns, and the number of columns matches the number of network parameters. e is a column matrix of errors for each learning pattern. The number of rows in e matches the number

of learning patterns. I is identity matrix whose number of rows as well as columns matches the number of network parameters. μ is scalar. If μ is very large, the above expression approximates gradient descent, while if μ is small this expression becomes the Gauss-Newton method. The coefficient μ is changed in such a way to join good features of both algorithms: gradient descent algorithm (it does not request that initial values of parameters are good chosen), and Gauss–Newton algorithm (it has quadratic convergence near an error minima). As long as the error gets smaller, μ is made bigger, but, once the error starts increasing, μ is getting smaller. The Levenberg–Marquardt is much faster than the gradient descent algorithm, on which standard BP algorithm is based. However, it requires more memory than the gradient descent algorithm.

2.2 Radial basis function type of artificial neural network

RBF network [6,24] usually used has a single output node and single hidden layer which contains as many neurons as are required to fit the function within the specifications of error goal. The transformation from the input space to the hidden–unit space is nonlinear, whereas the transformation from the hidden–unit space to the output space is linear. Following the feed–forward network structure, the basis functions are given in the form:

$$\phi_q(\nu) = \zeta(\| \nu - c_q \|)$$

where ν is the input vector, and $\zeta(\cdot)$ is activation function (for RBF network known as radial basis function). Theoretical investigations and practical results show that the type of nonlinearity $\zeta(\cdot)$ is not crucial to the performance of RBF network [29], and it is usually taken to be bell–shaped function. The $\| \cdot \|$ denotes a norm that is usually taken to be Euclidean. The c_q are known as vectors of radial basis function centers.

A common learning algorithm for RBF networks is based on first choosing randomly some data points as radial basis function centers and then using singular value decomposition to solve for the weights of the network. An arbitrary selection of centers may not satisfy the requirement that centers should suitably sample the input domain. Furthermore, in order to achieve a given performance, an unnecessarily large RBF network may be required. Since a performance of an RBF network critically depends upon the chosen centers, an alternative learning procedure based on the orthogonal least squares (OLS) learning algorithm [6] is often used. By providing a set of the inputs and corresponding outputs, the values of weights ξ_q, bias ξ_0, and radial basis function centers can be determined using OLS algorithm in one

pass of the learning data so that a network of an adequate size can be constructed.

When an input vector ν is presented to such a network, each neuron in the hidden layer will output a value according to how close the input vector is to the centers vector of each neuron. The result is that neurons with centers vector very different from the input vector will have outputs near zero. These small outputs will have a negligible effect on the linear output neurons. In contrast, any neuron whose centers vector is very close to the input vector will output a value near one. If neuron has an output of one, its output weights in the second layer pass their values to the neuron in the second layer. The width of an area in the input space to which each radial basis neuron responds can be set by defining a spread constant for each neuron. This constant should be big enough to enable neurons to respond strongly to overlapping regions of the input space. The same spread constant is usually selected for each neuron.

The RBF network has some advantages over the MLP: 1) RBF network with supervised learning of cluster centers as well as network weights has characteristic fast training; often it can be designed in a fraction of the time it takes to train MLP with BP learning algorithm, even RBF network may require more nodes than MLP; 2) RBF network using this learning algorithm is able to exceed the generalization performance of MLP with BP algorithm substantially [8], and 3) the spread constant is the only element which has to be selected for RBF network with the described learning algorithm.

2.3 Adaptive logic network

An ALN can be considered a special type of the feedforward MLP in which the signals in the network are restricted to be boolean (binary) after a layer of processing units that act on whatever other types of signals are present (reals etc.) to produce boolean values. The two versions of learning algorithms for ALNs were evaluated and implemented by Kostov [17].

Atree versions 2.7 and earlier [2] deal with binary numbers and to deal with continuous quantities are used fixed operators such as threshold units to encode real numbers as bit strings. Previously Kostov [16] used "random walk encoding", but then switched to a so called "thermometer code" which was even less sensitive to the input noise. It is a type of linear encoding where the number of bits corresponds to the number of encoding threshold levels and all bits under the encoding level closest to the real number to be encoded are ones while the other bits are zeroes. To obtain real results the output vector has to be decoded, which is the inverse process of the encoding. The nodes of the ALN tree are two types: adaptive elements and

leaves. Each adaptive element is a two–input logic gate, which can be any one of the following four boolean functions: AND, OR, LEFT and RIGHT, i.e., $g(x,y) = xy,\ x+y,\ x,\ y$ respectively [2]. Leaves are the nodes of the first layer of an ALN tree used to connect binary inputs from the encoder to the tree. The leaves of the tree are connected to input variables either in a one-to one fashion (the disjoint case) or with a multiplicity of connections going to the same variable of the binary tree and its inverse (the nondisjoint case). The adaptation procedure involved selecting the node functions of an ALN based on sequences of presentations of inputs and outputs. The critical parameter of an adaptive learning algorithm is training time which may become very long if the encoding and training parameters are not properly chosen.

In the Atree ver. 3.0 [3], the logic trees containing AND and OR operators have been furnished with input operators in the form of linear threshold elements (LTEs). The logic gates (AND and OR) may have an arbitrary number of logical inputs, and produce a logical output. The LTE is the basic element of approximation and it is very similar to the Perceptron developed in the 1950's. The LTE has a logical output value of one for input points $(x_1, x_2, \ldots, x_n)$ that satisfy the inequality: $w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n > 0$ and a logical value of zero otherwise. The LTE fits the data presented to it during training by performing a least–squared error fit, similar to linear regression. The direct consequence of new ALN design is possibility to apply ALNs directly to real values and thus define piecewise linear approximations of functions. This approach is not restricted to approximating functions, but can approximate relationships of more general types represented as sets of data points, which is important for pattern recognition applications.

The fundamental difference between Atree 2.7 and 3.0 versions is the way ALNs are used to solve problems. Instead of computing an output as a function of some inputs, ALNs in Atree 3.0 are used to represent relations among inputs. The Atree 2.7 "output", in the functional sense of the word, now becomes just another input to the ALN, which computes whether or not all of its inputs are related in a certain way. Therefore, the inputs to the LTEs are the inputs and outputs of the problem.

If ALN with an Atree 3.0 algorithm and MLP with BP algorithm are compared, it becomes obvious that the LTE (called hard–limiter) replaces the activation function in the MLP. The advantage of using a hard–limiter is in speed of evaluation: a comparison operation is much faster than computing e.g. sigmoid or even doing a table lookup to find a precalculated sigmoid value. The not so obvious advantage is that not all inputs of a logic gate have to be evaluated when using a hard–limiter. As soon as a zero input to

an AND gate is found, we know the AND gate output is zero. As soon as a one input to an OR gate is found, we know that the OR gate output is one. The effect of using hard–limiters is that the entire ALN does not need to be evaluated to determine the output of the tree. This contrasts with BP algorithm, where every input of every layer must be evaluated before calculating the output.

3. Rule–based learning techniques

Learning methods like IL and fuzzy logic generate set of "if–then–else" decision rules which are both explicit and comprehensible, contrary to an ANN which determines rules implicit within its structure and not easily comprehensible.

3.1 Fuzzy logic

Fuzzy sets are a generalization of conventional set theory. They were introduced by Zadeh [40] as a mathematical way to represent vagueness in everyday life. A formal definition of fuzzy sets that has been presented by many researchers is following: a fuzzy set A is a subset of the universe of discourse X that admits partial membership. The fuzzy set A is defined as the ordered pair $A = \{x, m_A(x)\}$, where $x \in X$ and $0 \leq m_A(x) \leq 1$. The membership function $m_A(x)$ describes the degree to which the object x belongs to the set A, where $m_A(x) = 0$ represents no membership, and $m_A(x) = 1$ represents full membership.

One of the biggest differences between conventional (crisp) and fuzzy sets is that every crisp set always has unique membership function, whereas every fuzzy set has an infinite number of membership functions that may represent it. This is at once both a weakness and a strength; uniqueness is sacrificed, but this gives a concomitant gain in terms of flexibility, enabling fuzzy models to be "adjusted" for maximum utility in a given situation. One of the questions, that is still asked most often, concerns the relationship of fuzziness to probability. The fuzzy models and the statistical models possess different kinds of information: 1) fuzzy memberships, which represent similarities of objects to imprecisely defined properties, and 2) probabilities, which convey information about relative frequencies. Moreover, interpretations about and decisions based on these values also depend on the actual numerical magnitudes assigned to particular objects and events.

The typical steps of a "fuzzy reasoning" consist of: 1) Fuzzification: comparison of the input variables with the membership functions of the premise parts (the if–part of the rule is called the antecedent or premise) in order to obtain the membership values between 0 and 1, 2) Weighing: applying

specific fuzzy logic operators (e.g., "AND" operator – minimum, "OR" operator – maximum etc.) on the membership values of the premise parts to get a single number between 0 and 1, that is the firing strength of each rule, 3) Generation: creation of the consequent (the then–part of the rule is called the consequent or conclusion) relative to each rule, 4) Defuzzification: aggregation of the consequent to produce the output.

There are several kinds of fuzzy rules used to construct fuzzy models which can be classified into the following three types according to their consequent form [19]:

1) fuzzy rules with a crisply defined constant in the consequent:

R_i: IF x_1 is A_{i1} and ... and x_m is A_{im}
THEN y is c_i

2) fuzzy rules with linear combination the system's input variables in the consequent:

R_i: IF x_1 is A_{i1} and ... and x_m is A_{im}
THEN y is $g_i(x_1, \ldots, x_m) = b_0 + b_1 x_1 + \ldots + b_m x_m$

3) fuzzy rules with fuzzy set in the consequent:

R_i: IF x_1 is A_{i1} and ... and x_m is A_{im}
THEN y is B_i

where R_i is the i-th rule of the fuzzy system, x_j $(j = 1, 2, \ldots, m)$ are the inputs to the fuzzy system, y is the output from the fuzzy system. The linguistic terms A_{ij} and B_i are fuzzy sets, c_i and b_j denote crisply constants.

The so–called zero–order Sugeno, or Takagi–Sugeno–Kang fuzzy model [32] has rules of the first type, whereas the first–order Sugeno fuzzy model has rules of the second type. The easiest way to visualize the first-order Sugeno fuzzy model is to think of each rule as defining the location of a "moving singleton" (single spike from the consequent) depending on what the input is. Sugeno models are similar to the Mamdani model [21] which has rules of the third type, and which is more intuitive, but computationally less efficient. Fuzzification and weighing, are exactly the same, but generation and defuzzification are different [19]. For type of fuzzy rules used in Mamdani model various methods are available for defuzzification: the centroid of area, bisector of area, middle of maximum, largest of maximum etc. [9], but all of these methods are based on the calculation of the two–dimensional–shape surface, that is on the integration. The Sugeno–style enhances the efficiency of the defuzzification process because it greatly simplifies the computation, that is, it has to find just the weighted average of a few data points. The implication method (generation) is simply multiplication, and aggregation

operator just includes all of the singletons. For the first–order Sugeno fuzzy model, usually used, defuzzified value y_0 is:

$$y_0 = \frac{\sum_i g_i\left(a_1, a_2, \ldots, a_m\right) \prod_j \mu_{A_{ij}}\left(a_j\right)}{\sum_i \prod_j \mu_{A_{ij}}\left(a_j\right)}$$

where $\mu_{A_{ij}}\left(a_j\right)$ is the membership degree of input a_j $(j = 1, 2, \ldots, m)$ to antecedent linguistic term A_{ij} for the i–th rule of the fuzzy system.

Membership functions are subjective and context–dependent, so there is no general method to determine them. Currently, when fuzzy set theory is applied in control systems, the system designers are given enough freedom to choose membership functions and operators, usually in a trial–and error way. After a hand–tuning process, the system can function effectively. However, the same methodology is hardly applicable when the system is a general purpose one, or when the context changes dynamically. This suggests an explanation for why the most successful applications of fuzzy logic happen in control systems, rather than in natural language processing, knowledge base management, etc.

The approach usually used to extract the rules for first–order Sugeno type of fuzzy inference system (FIS) is based on replacing identification of membership functions of input variables with identification of the centers of cluster–like regions. Fuzzy c–means technique introduced by Bezdek [5] as an improvement on earlier data clustering methods requires that number of clusters is known. If there is not a clear idea how many clusters there should be for a given set of data, it can be used subtractive clustering method [7] for estimating the number of clusters and their centers in a set of data. It is an extension of the Mountain clustering method proposed by Yager [38]. It assumes each data point is a potential cluster center and calculates a measure of the potential for each data point based on the density of surrounding data points. The algorithm selects the data point with the highest potential as the first cluster center and then destroys the potential of data points near the first cluster center. The algorithm then selects the data point with the highest remaining potential as the next cluster center and destroys the potential of data points near this new cluster center. This process of acquiring a new cluster center and destroying the potential of surrounding data points repeats until the potential of all data points falls bellow a threshold. The range of influence of a cluster center in each of the data dimensions is called cluster radius. A small cluster radius will lead to finding many small clusters in the data (resulting in many rules) and vice versa. The cluster information obtained by this method are used to determining the number of rules and

antecedent membership functions, that is to identifying FIS. Then, the linear least–squares estimation is using to determine consequent for each rule. The result is fuzzy rule base. However, for different initial values, this method may give different results, because the identification algorithm depends on an optimization procedure.

3.2 Entropy minimization type of inductive learning technique

An IL method described in [25] is based on an algorithm called "hierarchical mutual information classifier" [31]. A program Empiric described in [10] implements this algorithm. This algorithm produces a decision tree by maximizing the average mutual information gain at each partitioning step. It uses Shannon's entropy as a measure of information.

Mutual information is a measure of the amount of information that one random variable contains about another random variable. It is a reduction of the uncertainty of one random variable due to the knowledge of the other. Consider two random variables X and Y with a joint probability density function $p(x, y)$ and marginal probability density functions $p(x)$ and $p(y)$. Mutual information $I(X, Y)$ is the relative entropy between the joint distribution and the product distribution $p(x)p(y)$, i.e.,

$$I(X,Y) = \sum_x \sum_y p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

The mutual information can also be written as:

$$I(X,Y) = S(X) - S(X/Y)$$

where $S(X)$ is Shannon's entropy and it is defined as:

$$S(X) = -\sum_x p(x) \log_2 \ p(x)$$

and $S(X/Y)$ is conditional entropy and it is defined as:

$$S(X/Y) = -\sum_x \sum_y p(x,y) \log_2 p(x/y)$$

where $p(x/y)$ is conditional probability density function.

An effective method of integrating results of a mutual information algorithm into a production rule formalism, following the original work of Pitas [26] and Watanabe [36] is shown in [25]. While generating the decision tree,

the algorithm performs a hierarchical partitioning of the domain multidimensional space. Each new node of the decision tree contains a rule based on a threshold of one of the input signals. Each new rule further subdivides the example set. The learning is finished when each terminal node contains members of only one class. An excellent feature of this algorithm is that it determines threshold automatically based on the minimum entropy [25,26]. This minimum entropy method is equivalent to determination of the maximum probability of recognizing a desired event (output) based on the information from an input.

4. Combination of neural networks and rule–based learning techniques

The advantage of a rule–based learning method (e.g., IL method and fuzzy logic) compared to an ANN (e.g., MLP, ALN, RBF network) is that the rules determined are both explicit and comprehensible, whilst the rules used by the ANN are implicit within its structure and not easily comprehensible. There are methods which extract approximate classification rules from a trained ANN, and they help evaluating the learned knowledge [23]. Furthermore, ANNs are computationally intensive. In view of the versatility of ANN and rule–based learning method, their combination can be expected to exhibit many advantageous features like: 1) the parameters of the system have clear physical meanings, which they do not have in general ANN, 2) a network structure facilitates the computation of the gradient for parameters of the system, and 3) human linguistic descriptions or prior expert knowledge can be directly incorporated, for example, into fuzzy neural network structure. On the other hand, the disadvantage is that the network structure requires a large number of term nodes and there is no efficient process for reducing the complexity of combined neural network with rule–based method.

ANFIS is an example of often used combination of FIS and ANN [11,13]. Training procedure usually has two steps. In the first step is used subtractive clustering method for initial identification of a first–order Sugeno–type FIS [7]. In the second step is used an adaptive–network with BP and least squares algorithms for tuning of initial identified linear and nonlinear parameters of FIS respectively [11]. Adaptive–network corrects the rules determined by initial identification of FIS. The result is FIS which corresponds to the minimum training error.

An example where combination of the minimum entropy IL and the RBF network is readily used is the estimation of the muscle activation. It is presented in [12]. The muscle timing is estimated using minimum entropy IL, and the level of muscle activation is estimated using RBF network with

OLS learning algorithm [12]. The RBF network estimates the level very well because it gives a "continuous" output. However, it does not work so well for the muscle timing estimation. Hence, the rules designed using IL method correct the muscle timing estimated using RBF network. The parameters of RBF network were calculated based only on data from the input and output training sets which falls in the interval in which a muscle was estimated to be active using IL method.

5. RBF network for real–time control of walking – an example

5.1 Task formulation and preparation of data

In this section an example of usage of ML for design of the rules for a controller of the FES-assisted human walking is presented. The task was to predict: 1) the activation pattern of a muscle; and 2) the joint angle from preceding sensory data. The term muscle relates to the simplification introduced that all the muscles contributing to the activity at a joint are represented with a single muscle. A RBF network with OLS training algorithm was selected for the pattern matching because of its characteristics: 1) fast training; 2) good generalization; and 3) easy application because it requires only the spread constant to be assumed, while everything else is automatic.

The data used for training and testing of the network was prepared using simulation of walking of a fully customized model of a human body assisted with FES [28]. The sampling rate for all data was 100 Hz. The inputs for the network were following sensory data: 1) knee joint angle; 2) horizontal ground reaction force; and 3) vertical ground reaction force (Fig. 1). The desired outputs from the network were: 1) knee flexor muscle activity (Fig. 1); and 2) knee joint angle, predicted from the network inputs by 60 ms.

In the training phase the number of the nodes and the parameters of the network were tuned on the basis of the provided inputs and desired outputs of the system, called the examples in the supervised learning. For the testing of the obtained network only inputs were provided, and the goal was to generate outputs. The quality of matching was evaluated by comparing the desired outputs with the predicted ones. The testing was done in both cases: 1) using data used for the training; and 2) using data that was not used for training, being different in their amplitude, frequency content and timing.

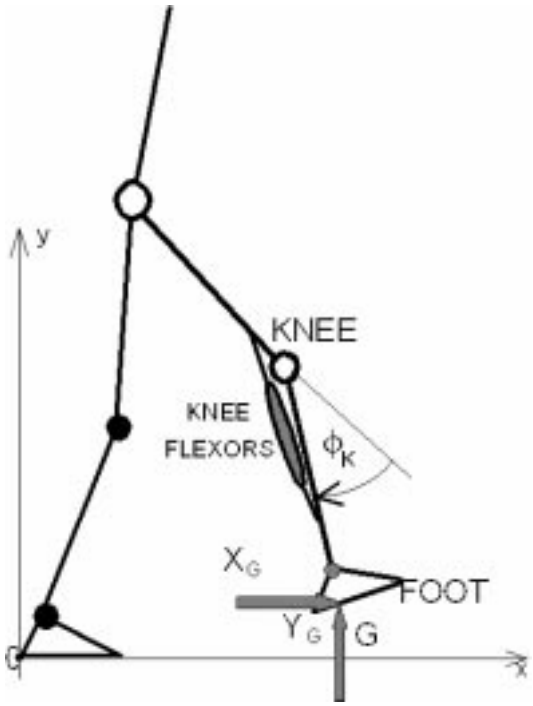

Figure 1. Model of a human body showing the knee joint angle, ground reaction forces, and the knee flexor muscle. The model is used for calculating data that is used as inputs and outputs for the pattern matching.

The example shown includes twelve consecutive strides because of the simplicity, even though we trained and tested the network using longer sequence of level walking with up to 50 successive strides and climbing up and down the stairs, and the results obtained are very similar to the one presented here.

The set of inputs and outputs used for the training and the testing is shown in Fig. 2. The set of inputs and desired outputs representing a similar walking was used for the testing, and not for the training (Fig. 3). Note the difference in the amplitude of the joint angles, ground reaction forces and activations of the knee flexor muscle. The top panels in Figs. 2 and 3 show the knee joint angle, the middle panels show the ground reaction forces, while the bottom panels show the activity of the knee flexor muscle. The sequence presents a series of strides that lasted for 15 seconds.

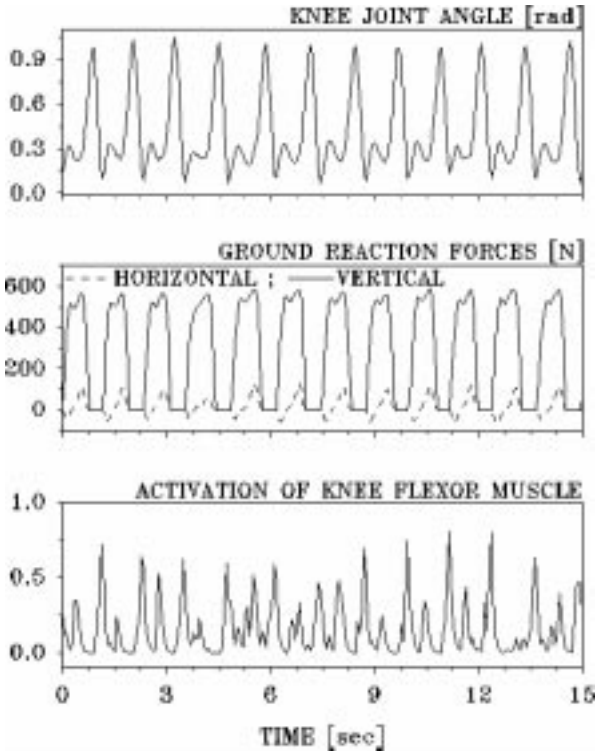

Figure 2. Data obtained from the simulation and used as input and desired output sets for training and testing of RBF network. The top panel shows the angle at the knee joint, the middle panel shows the ground reaction forces, and the bottom panel shows the activation patterns of the knee flexor muscle. The activation is normalized at 1.

The correlation can be graphically observed in Figs. 4 and 5, but the cross–correlation between the desired outputs and the predicted outputs by the network was selected as a measure of the generalization:

$$cor = \frac{\sum_{i=1}^{r} s(i)s_{des}(i)}{\sqrt{\sum_{i=1}^{r} s^2(i)}\sqrt{\sum_{i=1}^{r} s_{des}^2(i)}}$$

where $s_{des}(i)$ and $s(i)$ are the i-th sample of the desired output and the network generated output, and r is the number of samples.

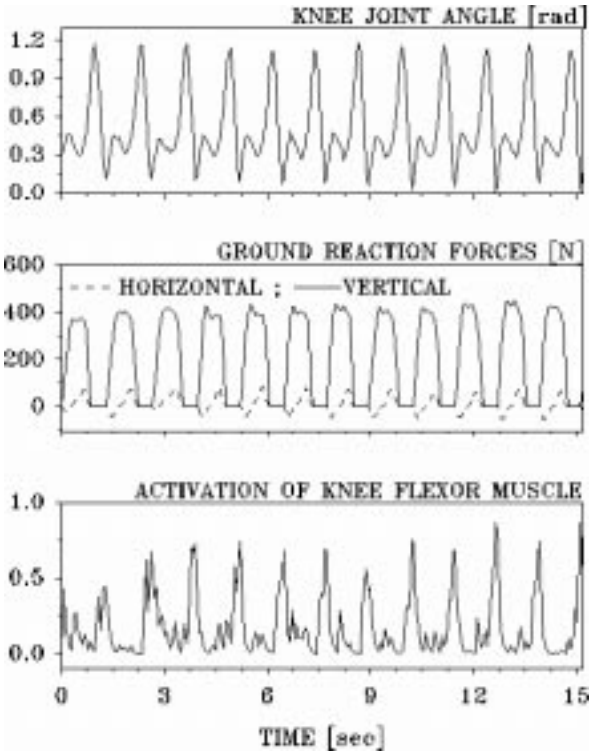

Figure 3. Data obtained from the simulation and used as input and desired output sets for testing of RBF network, and not used for the training. For details see Fig. 2.

5.2 Results of pattern matching by RBF network

Fig. 4 shows the results of the RBF network tested on a set of inputs that was previously used for the training. The desired (full line) and network generated (dashed line) outputs are superimposed in order to show the quality of mapping. It can be observed that the network results are somewhat delayed and do not reach the maximum of the signals which have to be matched. The top panel shows the muscle activation, while the bottom panel shows the joint angle.

Fig. 5 shows the results of the network tested using a set of inputs that was not used for the training. The desired (full line) and network generated (dashed line) outputs are superimposed to show the correlation. The top panel shows the muscle activation, while the bottom panel shows the joint

angle. The agreement of the network generated outputs and the desired outputs is better for the joint angles which can be explained by noticing that the muscle activations are much less repetitive from stride to stride compared to the joint angles. The inputs for these pattern matchings were the knee joint angle and the ground reaction forces delayed for the 60 ms from the outputs.

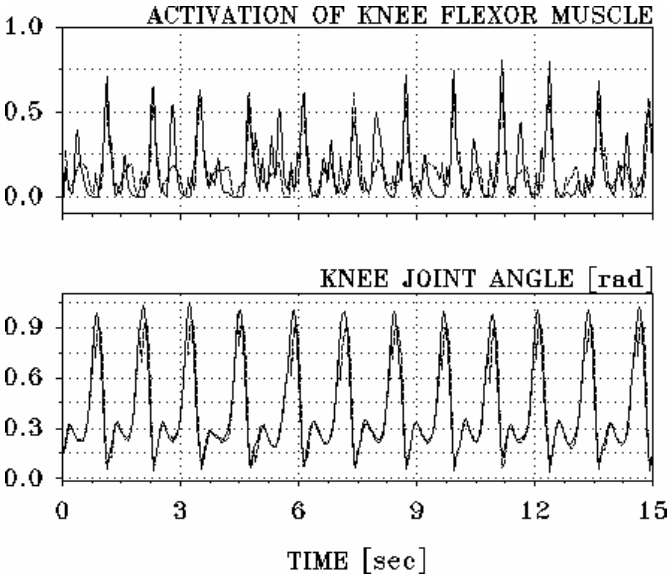

Figure 4. Results of the RBF network when using the data previously used for the training. The top panel shows the network generated (dashed line) and desired (full line) knee flexor muscle activations. The bottom panel shows the network generated (dashed line) and the desired (full line) knee joint angles. The inputs for the pattern matchings were the knee joint angle and the ground reaction forces delayed for the 60 ms from the outputs.

Note that the agreement in patterns and the timing between the joint angle is good, but the network outputs are not crossing about 0.9 radian, while the actual joint angles reach 1.2 radian. This discrepancy can be explained by analyzing the training set. The network was never "shown" a joint angle bigger than 0.9; hence, there is no reason that the bigger output is predicted. The networks are not meant to generalize if the data does not belong to the group that was used for the training. The results obtained are

acceptable for the purpose of this study, because there is no need to match the joint angle and activation of a muscle ideally. This statement follows the gait analysis which clearly shows that the variability form stride to stride is the normal behavior of walking.

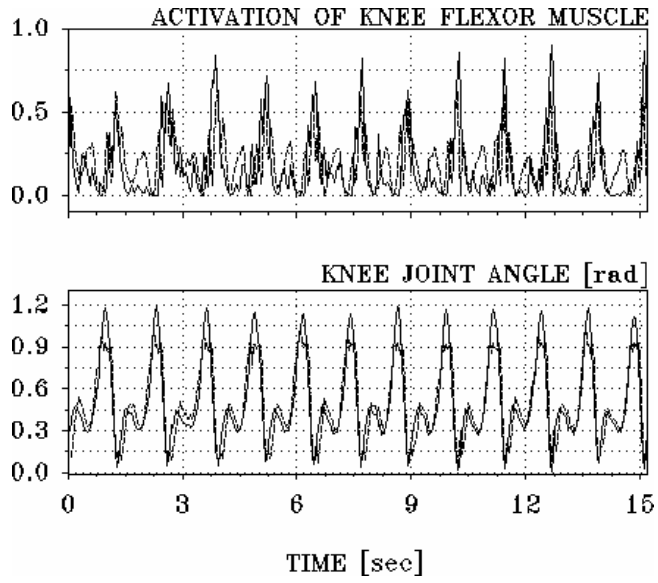

Figure 5. Results of the RBF network applied to the set of inputs that was not previously used for the training. For details see Fig. 4.

The spread constant for the RBF network matching muscle activity was selected at 112. The spread constant for the RBF network matching the joint angle was selected at 28. The spread constant was selected to get as good as possible pattern matching based on the input and desired output data not used for the training. This choice is based on trial and error gained experience. In both cases the obtained network had 792 nodes in the hidden layer, and the number of training epochs was one. The cross–correlation between the desired and network generated muscle activity using the data used previously for the training was 0.90, and it dropped down to 0.75 when applied to the set of data which was not used for the training. The cross–correlation between the desired and the network generated knee joint angle using the data previously used for the training was 0.99, and dropped down to 0.98 when applied to data previously not shown to the network.

6. Discussion

In this paper several supervised nonparametric techniques for pattern matching are described. These are: MLP, RBF network, ALN, Fuzzy Logic, IL, ANFIS, as well as the combination of RBF network and IL. An example of usage of RBF network with OLS learning algorithm is presented. This ML technique was selected because of its characteristics: fast training, and ability for good generalization. The training procedure is not too complicated because it requires that only the spread constant is chosen, while the remaining elements in the network are determined automatically.

By this type of ANN is obtained a large number of the rules which are not explicit and not easily comprehensible. However, there are methods which extract approximate classification rules from a trained ANN, and they help evaluating the learned knowledge [23].

The data for pattern matching was prepared using the results of the simulation of a human walking with an FES system. The goal was to predict: 1) an activation pattern of muscle (command signal), and 2) a joint angle (feedback signal) from preceding sensory data. This example is presented to show ability of the used ML technique to determine the connectivism, that is if-then rules which will be implemented in a controller for walking. There is no need of getting perfect mapping since the results obtained will be used only as a first approximation during an interactive and iterative procedure [34]. The simulation used to get the inputs and outputs for pattern matching provides only one plausible sensory-motor map, and its accuracy is questionable because it utilizes a very simplified biomechanical model of a human body. Therefore, the obtained map of desired signals only represents a starting point for fitting an FES controller to a person with disability.

Acknowledgement

This study has been partly supported by the research grant from the Ministry for Science and Technology of Serbia, Belgrade. We would like to acknowledge the help from Dr. R.B.Stein and Dr. N.M.Oğuztöreli from the University of Alberta, Edmonton, Alberta during studying biomechanical aspects of locomotion.

REFERENCES

1. B. Aeyels, L. Peeraer, J. Van der Sloten and G.Van der Perre: *Development of an above–knee prosthesis equipped with a microprocessor–controlled knee joint: first test results.* J. Biomed. Eng., 14, pp. 199–202, 1991.
2. W.W. Armstrong and J. Gecsei: *Adaptation Algorithms for Binary Tree Net works.* IEEE Trans Systems, Man and Cybern, SMC 9, pp. 276–285, 1979.

3. W.W. Armstrong and M.M. Thomas: *Dendronic decisions Atree 3.0 beta release 1.* ALN Theory, A Practical Guide to Approximating Relations, 1994.
4. A. Bar, P. Ishai, P. Meretsky and Y. Koren: *Adaptive microcomputer control of an artificial knee in level walking.* J. Biomed. Eng., 5, pp. 145–150, 1983.
5. J.C. Bezdek: *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.
6. S. Chen, C.F.N. Cowan and P.M. Grant: *Orthogonal least Squares learning Algorithm for Radial Basis Function Networks.* IEEE Trans. Neu. Net., 2, pp. 302–309, 1991.
7. S. Chiu: *Fuzzy Model Identification Based on Cluster Estimation.* Journal of Intelligent & Fuzzy Systems, vol. 2, 1994.
8. S. Haykin: *Neural Networks, A Comprehensive Foundation.* Macmillan College Publishing Company, New York, 1993.
9. H. Hellendorn and C. Thomas: *Defuzzification in fuzzy controllers.* Journal of Intelligent & Fuzzy Systems, vol.1, pp. 109–123, 1993.
10. B. Heller, P.H. Veltink, N.J.M. Rijkhoff, W.L.C. Rutten and B.J. Andrews: *Reconstructing Muscle Activation During Normal Walking: a Comparison of Symbolic and Connectionist Machine Learning Techniques.* Biolog. Cybern., 69, pp. 327–335, 1993.
11. J.S.R. Jang: *ANFIS : Adaptive–Network–based Fuzzy Inference Systems.* IEEE Trans. Sys. Man and Cybern., vol. 23, pp. 665–685, 1993.
12. S. Jonić and D. Popović: *Rule–based controller for locomotion – use of radial basis function ANN.* Proc. of the Neurel'97, Belgrade, 1997, pp. 49–52.
13. S. Jonić, T.Janković, V.Gajić and D.Popović: *Three machine learning techniques for automatic determination of rules to control locomotion.* IEEE Trans. Biomed. Eng., 1997. (submitted)
14. C.A. Kirkwood, B.J. Andrews and P. Mowforth: *Automatic detection of gait events: a case study using inductive learning techniques.* J. Biomed. Eng., 11, pp. 511–516, 1989.
15. R. Kobetic and E.B. Marsolais: *Synthesis of Paraplegic gait with multichannel functional neuromuscular stimulation.* IEEE Trans. Rehabil. Eng., TRE 2, pp. 66–78, 1994.
16. A. Kostov, R.B. Stein, D.B. Popović and W.W. Armstrong: *Improved Methods for Control of FES for Locomotion.* Proc. IFAC Symp. Modeling in Biomed. Eng., Galveston, Texas, pp. 422–427, 1994.
17. A. Kostov: *Machine learning techniques for the control of FES–assisted locomotion after spinal cord injury.* Ph.D. Thesis, University of Alberta, Ed monton, Alberta, 1995.
18. A. Kralj and T. Bajd: *Functional Electrical Stimulation: Standing and Walking after Spinal Cord Injury.* CRC Press, Inc., Boca Raton, Florida, 1989.
19. K.M. Lee, D.H. Kwak and H. Lee–Kwang: *Fuzzy Inference Neural Network for Fuzzy Model Tuning.* IEEE Trans. Sys. Man and Cybern., vol. 26, pp. 637–645, 1996.
20. R.P Lippmann: *An introduction to computing with neural nets.* IEEE ASSP Mag, vol.3, pp. 4–22, 1987.

21. E.H. Mamdani and S. Assilian: *An experiment in linguistic synthesis with a fuzzy logic controller.* International Journal of Man–Machine Studies, vol. 7, pp. 1–13, 1975.

22. D. Michie and R.A.Chambers: *Boxes: an experiment in adaptive control.* In Dale E. and Michie D. eds: Machine Intelligence 2, Edinburgh University Press, Edinburgh, pp. 137–152, 1968.

23. H. Narazaki, T. Watanabe, and M. Yamamoto: *Reorganizing Knowledge in Neural Networks: An Explanatory Mechanism for Neural Networks in Data Classification Problems.* IEEE Trans. Sys. Man and Cybern., SMC 26, pp. 107–117, 1996.

24. Z. Nikolić: *Automatic rule determination for finite state model of locomotion.* Ph.D. Thesis, University of Miami, Miami, Florida, 1995.

25. Z. Nikolić and D. Popović: *Automatic detection of production rules for locomotion.* J. Autom. Control, 6, pp. 81–94, 1996.

26. I. Pitas, E. Milios, and A.N. Venetsanopoulos: *Minimum Entropy Approach to Rule Learning from Examples.* IEEE Trans. Sys. Man and Cybern., SMC 22, pp. 621–635, 1992.

27. D.B. Popović, R. Tomović, D. Tepavac and L. Schwirtlich: *Control aspects an active A/K prosthesis.* Intern. J. Man – Machine Studies, 35, pp. 751–767, 1991.

28. D. Popović, R.B. Stein, M.N. Ouztöreli, M. Lebiedowska and S. Jonić: *Optimal Control of Walking with Functional Electrical Stimulation: a computer simulation study.* IEEE Trans. Rehabil. Eng., TRE 5, 1997. (in press)

29. M.J. Powell: *Radial basis functions for multivariable interpolation: A review.* in Algorithms for Approximation, J.C. Mason and M.G. Cox , Eds. Oxford, pp. 143–167, 1987.

30. D.E Rumelhart, G.E.Hinton and R.J.Williams: *Learning interval representation by error propagation.* in Parallel distributed processing, Ch 8, pp. 318–361, MIT Press, Cambridge, MA, 1986.

31. I.K. Sethi and G.P.R. Sarvarayudu: *Hierarchical classifier design using mutual information.* IEEE Trans. Pattern Anal. Mach. Intel., 4, pp. 441–445, 1992.

32. M. Sugeno and K.Murakami: *Fuzzy Parking Control Using a Model Car.* in Industrial Applications of Fuzzy Control, M.Sugeno (ed.), pp. 125–138, 1985.

33. R. Tomović: *Control of Assistive Systems by External Reflex Arcs.* in D.Popović (Ed.) Advances in External Control of Human Extremities VIII, Belgrade, pp. 7–21, 1984.

34. R. Tomović, D.Popović, and R.B.Stein: *Nonanalytic Methods for Motor Control.* World Scientific, Singapore, 1995.

35. H.P. Veltink, H.J. Chizeck, P.E. Crago, and A. El-Bialy: *Nonlinear joint angle control for artificially stimulated muscle.* IEEE Trans. Biomed. Eng., BME 39, pp. 368–380, 1992.

36. S. Watanabe: *Pattern Recognition.* New York: Wiley Interscience, 1985.

37. J.M. Winters: *Hill–based muscle models: a systems engineering perspective.* In: Multiple Muscle Systems – Biomech. and Movement Organiz., eds. J.M. Winters and S.L.Y. Woo, pp. 69–73, Springer–Verlag, New York, 1990.

38. R. Yager and D. Filev: *Generation of Fuzzy Rules by Mountain Clustering.* J. Intel. Fuzzy Syst., 2, pp. 209–219, 1994.

39. G.T. Yamaguchi and F.E. Zajac: *Restoring Unassisted Natural Gait to Paraplegics Via Functional Neuromuscular Stimulation: A Computer Simulation Study.* IEEE Trans. on Biomed. Eng., BME 37, pp. 886–902, 1990.

40. L.A. Zadeh: *Fuzzy sets.* Information and Control, vol. 8, pp. 338–352, 1965.