Procedia Computer Science

Volume 51, 2015, Pages 60–69

ICCS 2015 International Conference On Computational Science

A Neural Network Embedded System for Real-Time Estimation of Muscle Forces

Gabriele Maria LOZITO, Maurizio SCHMID, Silvia CONFORTO, Francesco RIGANTI FULGINEI, Daniele BIBBO

University of Roma Tre, Italy, Department of Engineering

gabrielemaria.lozito@uniroma3.it

Abstract

This work documents the progress towards the implementation of an embedded solution for muscular forces assessment during cycling activity. The core of the study is the adaptation to a real-time paradigm an inverse biomechanical model. The model is well suited for real-time applications since all the optimization problems are solved through a direct neural estimator. The real-time version of the model was implemented on an embedded microcontroller platform to profile code performance and precision degradation, using different numerical techniques to balance speed and accuracy in a low computational resources environment.

Keywords: Microcontrollers, Neural Networks, Muscle Forces, Cycling, Inverse Dynamics, Inverse Problems Solution.

1 Introduction

In recent times, the increase of custom devices used for sports training raised a great deal of interest on the development of embedded solutions for these applications. In particular, in cycling, as well as in many other sports, the estimation of the exerted muscle forces can support athletes and trainers in monitoring and improving performance.

Concerning the pedaling task it is important to evaluate how the athlete executes it, also to obtain parameters that quantify the performance [1]. This aspect can be studied evaluating the role of muscle activity [2,3,4] or the power exerted, using different techniques [5]. To this aim a biomechanical model based system, able to predict forces acting on each involved joint, can be adopted with the design of an optimization criterion suited to determine the contribution of each muscle to the overall force [6]. This approach has been applied in many contexts, as in gait and running analysis [7, 8] or for the study of upper limb movements [9].

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.196

In previous studies an inverse dynamics approach, based on the solution of optimization problems, was proposed to estimate muscle force patterns [10,11,12,13], by the direct measurement of external forces exerted on the pedal, using for example custom designed devices [14], that were correlated to the muscle activity estimated by surface Electromyography (sEMG) data [15].

In these studies, the real time approach was not defined as the main goal, thus not considering the performance of the adopted solution algorithms. In a recent study [16] a new optimization algorithm based on artificial Neural Networks (NN) was proposed in order to reduce the computational complexity of the deterministic one previously used [10], while maintaining the quality of estimation. In particular in these works both the kinematics and the dynamics required the solution of an optimization problem: the kinematic section required, for the computation of the angles between the elements schematizing the leg, the solution of a transcendental implicit equation; the dynamic section required the solution of a sparse and undetermined linear system, with 3 equations and 9 unknowns, subject to boundaries, through a cost function minimization, in order to estimate the muscular forces time trends. The solution of these problems using a set of Neural Networks for solution-prediction gave optimal results [16] when implementing the required algorithms on a standard workstation (Intel Core i7 with 16Gb RAM running Matlab in Windows 7 64-bit environment).

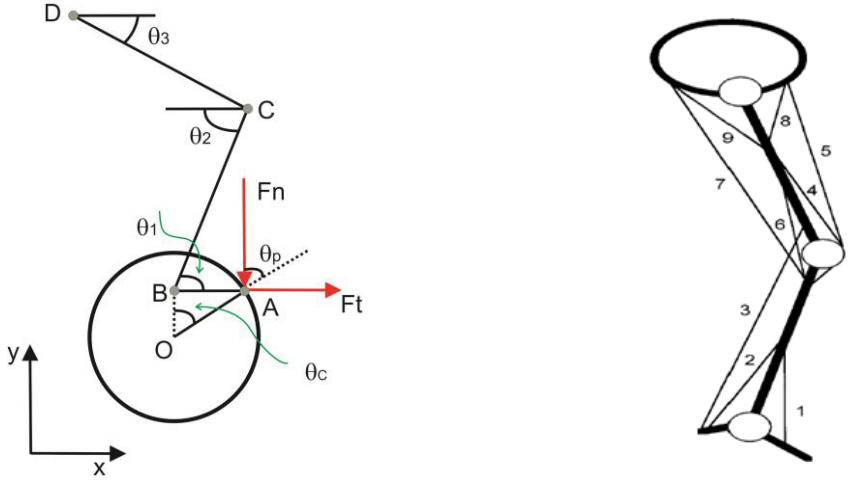
In this work, the possibility of implementing a neural solution in an embedded environment for the muscular force estimation was assessed on the basis of the aforementioned approach. When implementing this algorithm in an embedded environment, the limited computational capabilities calls for a trade-off among precision, memory footprint, and computational cost. Different studies tested the embedded implementation of NNs to achieve optimal results, either by re-arranging the operations required to compute the linear part of the NN [17] to fully exploit pipelining, or by speeding up the costly non-linear activation function through different numerical approximation [18,19,20,21,22,23,24].

Another issue worth being addressed in this implementation is the possibility of solving in real time the unknowns. In the original proposed algorithm data were processed in batch, allowing heavy filtering for noisy signals. In a real-time approach, only a small time-window for the signal is available, thus excluding the possibility of intensive filtering. The inverse model requires, to be computed, several II order numerical differentiations, that naturally introduce an amplification for high frequency noise. Different techniques are used in the literature to obtain a noise-rejecting differentiator [25,26,27,28,29] that can be applied easily in embedded environment.

In the first part of this paper, the biomechanical model along with the equations for the assessment of its parameters will be shortly presented. Then, the real-time implementation of the model will be explained from a systemic point of view, with special attention to the neural estimator and the differentiation techniques. The embedded implementation will be presented along with the performance evaluators considered. Results, conclusions and final considerations will follow.

2 Biomechanical Model

In order to reproduce the cycling task, a biomechanical model of the lower limb was defined based on 3 joints (i.e. ankle, knee, and hip) and 9 muscles, as described in [10]. The model was used to assess the muscular forces using 3 muscular moments, one for each joint, obtained by a previous inverse dynamics approach. To do this, a cost function minimization was proposed, defined using a physiological criterion.



a b

Figure 1: Kinematic chain (a) and correspondent muscular model (b) of the lower limb while cycling.

The adopted kinematic planar model of the lower limb (Figure 1a) was defined considering 3 body segments (AB = foot, BC = shank, CD = thigh) and 2 bicycle elements (DO = frame, OA = crank) constrained by hinge joints, so described by 2 degrees of freedom. On this assumption, the position of each element in the sagittal plane, being their lengths known, can be reconstructed using 2 of the relative angles values θ_C, θ_P, θ_1, θ_2, θ_3. The angles θ_C and θ_P, together with the pedal forces Fn and Ft, along the perpendicular and parallel axis to the pedal load plane, were directly measured on a cycling simulator [14,30], already validated in recent works for monitoring cycling activity in real time [31]. An inverse dynamics model provided muscular moments for each joint. This approach requires the solution of differential and non linear equations, with a non-negligible computational cost.

Subsequently, 9 muscles are identified (Figure 1b) as the ones involved in the task and necessary to set the model: 1) Tibialis anterior (TA); 2) Soleus (SO); 3) Gastrocnemius (GA); 4) Vastii (VA); 5) Rectus femoris (RF); 6) Short head of biceps femoris (BFs); 7) Long head of Biceps Femoris (BFl); 8) Iliacus (IL); 9) Gluteus Maximum (GLM).

The muscular moments and the muscular forces at each joint j are related as:

$$\sum_{i=1}^{N_j} F_i \times d_{ij} = M_j \tag{1}$$

where M_j represents the muscular moment at the j-th joint, N_j is the number of muscles acting on the j-th joint, F_i is the muscular force exerted by the i-th muscle and d_{ij} is the effective moment arm of the i-th muscle from the j-th joint, estimated as a function of the joint angle [32].

Minimizing the cost function:

$$U = \sum_{i=1}^{p} \left(\frac{F_i}{PCSA_i}\right)^3 \quad \text{with} \quad 0 < F_i < F_{imax} \tag{2}$$

where p is the total number of muscles, $PCSA_i$ and F_{imax} are respectively the physiological cross sectional area and the maximum force value for the i-th muscle. Muscular force values can be obtained even if p is greater than the number of equations previously obtained. The chosen cost function is widely used in literature [33,34] as it relies on the co-activation of all the muscles involved in the gesture.

3 Methods and Materials

This section will present the methods applied to reproduce the biomechanical model in embedded environment. First, the real-time version of the biomechanical model will be overviewed. Then, an in-depth explanation of the two main peculiarities of the model (the Neural Estimator and the Noise-Rejecting Differentiator) will be presented. Finally, the actual workbench used to test the algorithm will be discussed.

3.1 Real-Time Inverse Model Overview

In order to simplify its implementation and subsequent test-debug procedure, the proposed model can be solved considering two different sections: the first, addressed as "kinematic section", is related to the determination of the complete kinematics, considering as inputs the actual angles measured as explained above; the second, addressed as "dynamic section", aims to the determination of the joint reactions and the joint moments, using current and past data obtained from the kinematic section, and of the muscular forces. As displayed in Figure 2, the input data provided to the whole model are the angles θ_c and θ_p and the pedal force components Fn and Ft.

The kinematic section of the model receives as input only the angles and, using trigonometric equations, computes the {x,y} positions for the leg joints. For the computation of the other angles, instead of solving an inverse trigonometry problem, a neural network was used. The input of the neural network is the cosine of both θ_c and θ_p angles while the output is θ_3. The other angles θ_1 and θ_2 are calculated as a function of θ_3. The network is composed by a single neuron for reasons that will be explained in the next section.

In the dynamic section, the joint reactions and the moments must be computed to determine the muscular forces of the leg. The mechanical model, summarized above, is a II order one, which requires a numerical solution for the acceleration resolution of several elements composing it. To compute the second derivative of a quantity with respect to time, at least the current value, and the previous two samples of the actual quantity, must be known. Since this is a real-time model, a buffer system that holds the previous values of the quantity must be interposed between the kinematic and the dynamic parts of the model. The buffer system is composed by a set of three bi-dimensional arrays, two for the {x,y} positions, and one for the angles. Every time the model is computed, the array acts like a shift register, discarding the oldest sample and replacing it with the new one.

Given the experimental nature of the data a noise-rejecting differentiator, which works on more than 3 points, was implemented. More information on the differentiator will follow. Once the joint reactions are computed, the muscular moments can be easily calculated. The final computation of muscular forces is demanded to the Neural System, which receives the three muscular moments Mb, Mc and Md as inputs.

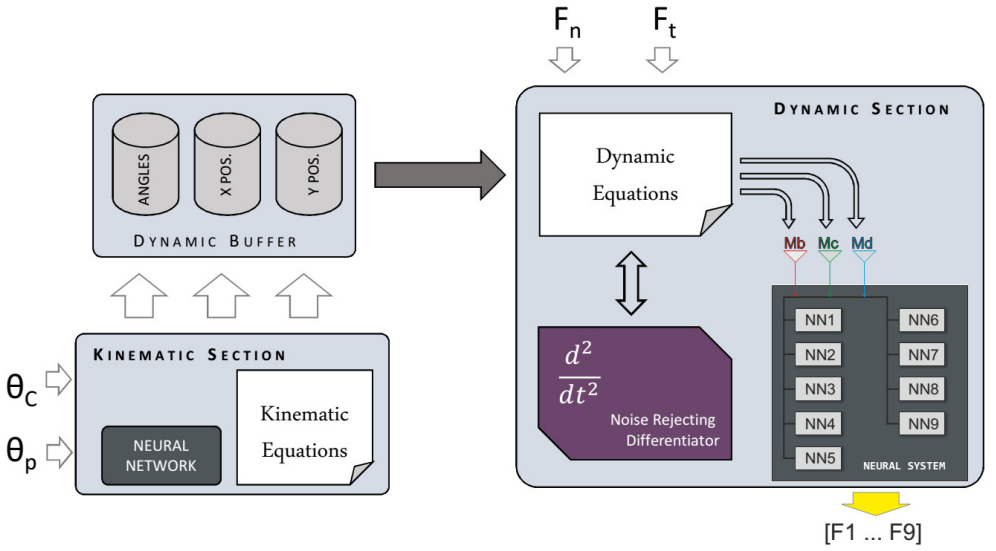


Figure 2 - Inverse Biomechanical Model

3.2 Neural Estimator

The neural estimator implemented in the model was refined starting from the one implemented in [16], and is composed by two parts: a Multiple-Input-Single-Output (MISO) NN, to calculate the θ_3 angle, and a Neural System of nine MISO NN, to assess the muscular forces (one for each force).

The main purpose of the original NNs in [16] was to obtain the maximum performance through heavily filtered signals, and for this reason, the criterion to size the NNs was maximizing the accuracy with the smallest number of neurons (to preserve generalization capabilities). Conceiving the problem in an embedded environment, two considerations must be done: first, since the computational capabilities of a MCU is limited, some accuracy should be traded for performance, to ensure the real-time capabilities of the system; then, as it will be shown, a serious problem affecting the embedded implementation of the model lies in the strong noise introduced by the process. From this perspective, a simpler and less accurate NN, with less neurons, can actually enhance the model performance by introducing a natural low-pass filtering of data. For these reasons, the NNs used for the embedded implementation of the model were reduced in complexity. The NN used to compute the θ_3 angle has a single nonlinear (tangent sigmoid) neuron in the hidden layer, and uses as input the cosine of θ_p and θ_c angles. All the NNs used to compute the nine muscular forces have four nonlinear (tangent sigmoid) neurons in the hidden layer, and use the three muscular moments Mb, Mc and Md as inputs. Networks were trained and validated in Matlab® environment according to the methodology described in [16, 35]. To enhance the embedded performance of the NN, according to [18, 19], a speedup for the activation function of the hidden neurons was obtained by computing it through a 2nd degree polynomial interpolating function.

3.3 Noise-Rejecting Differentiator

The dynamic section of the model requires the computation of the second derivative for unfiltered, noisy signals. The magnitude impulse response of an ideal second degree differentiator is $|H(\omega)| = \omega^2$, exalting the high frequency components of the signal, and lowering the signal to noise ratio (SNR) for

signals affected by white Gaussian noise. In the original implementation of the model [16], the problem of noise was solved by using heavy low-pass filters on the signals. However, the filter lengths and the real-time nature of the present implementation discourage the use of intermediate filtering during the process. An alternative approach [25], proposes to use differentiating filters, shaped to have a response proportional to the low-pass filtered second derivative of the excitation. The filters order N is variable, must be odd and larger than 5. From N, the coefficients and the equations for the filtering can be easily derived through Eq. 3 and Eq. 4.

$$f(x_0) \approx \frac{1}{2^{N-3}h^2}\left(s_0 f_{-M} + \sum_{k=1}^{M} s_k (f_{-M+k} + f_{-M-k})\right) \tag{3}$$

Where *M = (N-1)/2* and the $\{s\}_{k=0}^{M}$ coefficients can be calculated by a recursive algorithm for k = [M-1, …, 0].

$$\begin{cases} s_{M+1} = 0 \\ s_M = 1 \\ s_k = \dfrac{[(2N-10)s_{k+1} - (N+2k+3)s_{k+2}]}{(N-2k-1)} \end{cases} \tag{4}$$

The differentiator was implemented in the system in the form of a library, where the coefficients are pre-computed at the beginning of the program execution using a separate function. Theoretically, the order of the filter could be changed in real time by re-computing the coefficients. However, the order of the filter determines the length of the filter itself, i.e. the number of points needed in the dynamic buffer to compute the second derivative. Since the memory for the buffer is allocated statically (through a series of #define directives) the order of the filter can be modified at compile-time, not run-time. Obviously, increasing the filter length yields a smoother signal at the cost of heavily degrading the real-time algorithm performance. A comparison of different options will be presented in the results section.

3.4 Workbench

The model was initially developed in C and tested in x86 Windows environment using the simple CodeBlocks IDE. In this environment, a set of libraries was created for the model: the NNs and the noise-rejecting differentiator. To implement the project in embedded environment, a powerful Cortex M4-F ARM microcontroller was used, the LM4F120H5QR device, mounted on the Stellaris® LaunchPad (Texas Instruments). This microcontroller has a maximum clock frequency of 80MHz, 256KB of Flash / 32Kb SRAM / 2KB EEPROM, dual 12-Bit ADC, a dedicated Floating Point Unit, and the board implements a RS232 interface, through the programming port (In-Circuit Debug Interface ICDI), that can be used for communication. The free IDE Code Composer Studio was used to program the microcontroller board. To test the real-time performance of the model, a control program was created in Matlab, to send a stream of data to the MCU, to recover the results, to assess the error introduced by the microcontroller, and to show the results. The graphical interface of the Matlab utility is shown in Figure 3. Code profiling, however, had to be performed on the MCU itself: the RS232 interface implemented in Matlab library is a harsh bottleneck, introducing a considerable delay between samples, which should not be accounted for when evaluating code performance. For this reason, execution times were measured by clock-cycles using the debug utility of Code Composer Studio.

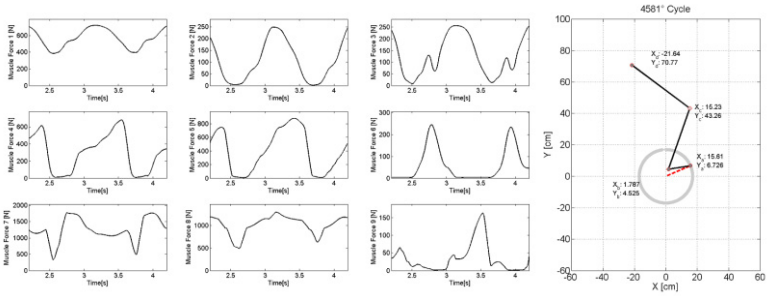


Figure 3: Graphical Interface of Matlab control utility.

4 Results

The algorithm was validated on the test bench previously illustrated and the code was profiled both in terms of precision, memory footprint and clock cycles required for computation. The code was built with different compile-time parameters allowing fine-tuning of performance. In this investigation, two parameters were changed:

- Order (i.e. number of samples used) of the II order differentiator;
- Activation function of the NNs.

In the following table, for differentiators of variable length, the computational cost of the algorithm, in terms of clock cycles and maximum sampling, is shown. Indeed, the actual microcontroller works with a clock of 50MHz, but the TM4C1294NCPDTI model, lately released by Texas Instruments, belonging to the same family of microcontrollers, has a 120MHz internal clock, and yields higher performance.

ORDER	CLOCK CYCLES	PERIOD @50MHZ	PERIOD @ 120MHZ	MAX FREQUENCY @50MHZ	MAX FREQUENCY @120MHZ
5	125632	0.00251264	0.001046933	397.988	955.171
7	172991	0.00345982	0.001441592	289.032	693.678
9	191415	0.0038283	0.001595125	261.213	626.910
11	212707	0.00425414	0.001772558	235.065	564.156

Table 1: Computational cost for different filtering orders.

As displayed, the computational cost for the model rises quickly as soon as the order of the differentiator grows. This is due to the increased number of operations required to compute the dynamic section of the model. However, in terms of overall error, the best results are obtained with an order of 5 or 7. Even if a higher order filter better removes the noise of the signal, the system responds slowly and in proximity of quick variations the error is considerable. Indeed, an order higher than 9 should be considered only if more data processing is needed for which noisiness is less desirable than inaccuracy.

The second parameter that was investigated was the activation function used for the hidden neurons of the NN. Two alternatives were used: the full-precision activation function, computed using *math.h* C library, and a polynomial interpolation of the activation function, pre-computed in Matlab, composed by a fit of five 2nd order polynomials [18]. The second solution is obviously a tradeoff

between precision and performance. By the combination of this parameter and the order, two final solutions are proposed, one for best performance, one for highest precision. Percent Root Mean Square Error (%RMSE) was calculated as the average RMSE (on the 9 forces) between the output in embedded environment and the output obtained from the original algorithm, on a set of 20.000 samples already used in [16]. This comparison shows, in quantitative terms, the degradation introduced by transposing the algorithm from a batch Matlab implementation to a real-time, embedded implementation.

a) Performance Configuration

2nd Degree Differentiator Order	5
Activation Function	Polynomial
% RMSE	10.3%
Clock Cycles	125632
Cutoff Frequency (50MHz Clock)	397 Hz
Cutoff Frequency (120MHz Clock)	955 Hz
Flash Memory Occupation	4.13kB

b) Precision Configuration

2nd Degree Differentiator Order	7
Activation Function	Full-Precision
% RMSE	3.61%
Clock Cycles	212636
Cutoff Frequency (50MHz Clock)	235 Hz
Cutoff Frequency (120MHz Clock)	564 Hz
Flash Memory Occupation	3.59kB

Table 2: Code configurations for best performance and best precision.

As it can be seen in Table 2, error for the Performance configuration is almost 3 times higher than the Precision one, whereas the speedup factor is less than 2. The Performance configuration should only be used if a coarse and rapid guess of the forces is needed.

5 Conclusions and Future Developments

In this work, a real-time embedded implementation for a biomechanical model of the leg during cycling activity was created. The original model, created in Matlab environment using batch data, allowed heavy filtering and data conditioning. The new model was developed to run in embedded environment in real time, thus using only a limited buffer of data, and with the idea of the tradeoff between accuracy and performance. This model was first tested in Matlab, to validate the on-line implementation of the mechanical model. Then, an x86-based C implementation of the model was developed and included in a set of libraries. The libraries were used to implement the model in embedded environment using a microcontroller unit, receiving a stream of data from a Matlab control application. Achieving useable results without the filtering capabilities of a batch algorithm, required complex numerical considerations: the implemented libraries have numerous tools that can be activated through pre-compiler directives. Different configurations of the model were tested and two optimal configurations were proposed, one precision-oriented and one performance-oriented. Both configurations largely satisfy minimum requirements for frequency, since for the biomechanics of the studied gesture, the force information is contained between 0 and 40 Hz. Such a large gap in terms of

computational time can be used either to increase precision further, through intermediate filtering, or to include additional data elaboration, like sEMG correlation algorithms [15].

References

[1] Castronovo A.M., Conforto S., Schmid M., Bibbo D., D'Alessio T. How to Assess Performance in Cycling: the Multivariate Nature of Influencing Factors and Related Indicators. Frontiers in Physiology 2013, 4,116.
[2] De Marchis C.; Schmid M.; Bibbo D.; Castronovo A.M.; D'Alessio T.; Conforto S. Feedback of mechanical effectiveness induces adaptations in motor modules during cycling. Frontiers in Computational Neuroscience 2013, 7(35), 1-12.
[3] De Marchis C.; Schmid M.; Bibbo D.; Bernabucci I.; Conforto S. Inter-individual variability of forces and modular muscle coordination in cycling: A study on untrained subjects. Hum Mov Sci, 2013, DOI: 10.1016/j.humov.2013.07.018.
[4] De Marchis C.; Castronovo A.M.; Bibbo D.; Schmid M.; Conforto S. Muscle Synergies are Consistent when Pedaling under Different Biomechanical Demands. Proceedings of the 34th IEEE-EMBS Conference, San Diego, California (USA), 28 Aug-1 Sep/2012.
[5] Watson M.; Bibbo D.; Duffy C.R.; Riches P.E.; Conforto S.; Macaluso A. Validity and Reliability of an Alternative Method for Measuring Power Output During 6 s All Out Cycling. J Appl Biomech, 2014, Jun, doi:10.1123/jab.2013-0317. PubMed PMID: 24977624.
[6] Erdemir, A.; McLean, S.; Herzog, W.; van den Bogert, A.J. Model-based estimation of muscle forces exerted during movements. Clinical Biomechanics 2007, 22(2),131-154.
[7] Pandy M.G.; Andriacchi T.P. Muscle and joint function in human locomotion. Annual review of biomedical engineering 2010 12,401-433.
[8] Hughes R.E; An K.N. Monte Carlo simulation of a planar shoulder model. Medical and Biological Engineering and Computing 1997, 35.5, 544-548.
[9] Dorn T.W.; Schache A.G.; Pandy M.G. Muscular strategy shift in human running: dependence of running speed on hip and ankle muscle performance. The Journal of experimental biology 2012, 25, 1944-1956.
[10] Bibbo D.; Conforto S.; Gallozzi C.; D'Alessio T. Combining electrical and mechanical data to evaluate muscular activities during cycling. WSEAS Trans Biol Biomed 2006, 5, 339-346.
[11] Prilutsky B.I. Coordination of two-and one-joint muscles: functional consequences and implications for motor control, Motor control 2000, 4.1, 1-44.
[12] Prilutsky B.I.; Zatsiorsky V.M. Optimization-based models of muscle coordination, Exercise and sport sciences reviews 2002, 30.1, 32.
[13] Bottasso C.L.; Prilutsky B.I.; Croce A.; Imberti E.; Sartirana S. A numerical procedure for inferring from experimental data the optimization cost functions using a multibody model of the neuro-musculoskeletal system. Multibody System Dynamics 2006, 16.2, 123-154.
[14] Bibbo D.; Conforto S.; Schmid M.; D'Alessio T. A wireless integrated system to evaluate efficiency indexes in real time during cycling. Proceedings of ECIFMBE IFMBE 2008.
[15] D'Alessio T.; Conforto S. Extraction of the envelope from surface EMG signals: an adaptive procedure for dynamic protocols, IEEE Engineering in Medicine and Biology Magazine 2001.
[16] Cecchini G.; Lozito G. M.; Schmid M.; Conforto S.; Riganti Fulginei F.; Bibbo D. Neural Networks for Muscle Forces Prediction in Cycling. Algorithms, 2014, 7.4, 621-634.
[17] LAUDANI, Antonino, et al. An Efficient Architecture for Floating Point Based MISO Neural Neworks on FPGA. In: Proceedings of the 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation. IEEE Computer Society, 2014. p. 12-17.

[18] LOZITO, Gabriele-Maria, et al. FPGA Implementations of Feed Forward Neural Network by using Floating Point Hardware Accelerators. AEEE, 2014, 12.1: 30
[19] LOZITO, Gabriele Maria; BOZZOLI, Ludovica; SALVINI, Alessandro. Microcontroller based maximum power point tracking through FCC and MLP Neural Networks. (EDERC), 2014 6th European Embedded Design in. IEEE, 2014. p. 207-211.
[20] AVCI, Mutlu; YILDIRIM, Tulay. Generation of tangent hyperbolic sigmoid function for microcontroller based digital implementations of neural networks. In: International Turkish Symposium on Artificial Intelligence and Neural Networks. 2003. p. 1-5.
[21] NASCIMENTO, Ivo; JARDIM, Ricardo; MORGADO-DIAS, Fernando. A new solution to the hyperbolic tangent implementation in hardware: polynomial modeling of the fractional exponential part. Neural Computing and Applications, 2013, 23.2: 363-369.
[22] Keegstra, H.; Jansen, W.J.; Nijhuis, J.A.G.; Spaanenburg, L.; Stevens, H.; Udding, J.T., "Exploiting network redundancy for low-cost neural network realizations," Neural Networks, 1996., IEEE International Conference on , vol.2, no., pp.951,955 vol.2, 3-6 Jun 1996
[23] ZAMANLOOY, Babak; MIRHASSANI, Mitra. Efficient VLSI implementation of neural networks with hyperbolic tangent activation function. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 2014, 22.1: 39-48.
[24] Pedro Ferreira, Pedro Ribeiro, Ana Antunes, Fernando Morgado Dias, A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function, Neurocomputing, Volume 71, Issues 1–3, December 2007, Pages 71-77, ISSN 0925-2312
[25] HOLOBORODKO Pavel, Smooth Noise Robust Differentiators 2008
[26] MBOUP, Mamadou; JOIN, Cédric; FLIESS, Michel. Numerical differentiation with annihilators in noisy environment. Numerical Algorithms, 2009, 50.4: 439-467.
[27] CHARTRAND, Rick. Numerical differentiation of noisy, nonsmooth data. ISRN Applied Mathematics, 2011, 2011.
[28] KNOWLES, Ian; RENKA, Robert J. Methods for numerical differentiation of noisy data. 2014.
[29] HE, Zijun, et al. A low-pass differentiation filter based on the 2nd-order B-spline wavelet for calculating augmentation index. Medical engineering & physics, 2014, 36.6: 786-792.
[30] Conforto S.; Sciuto S.A.; Bibbo D.; Scorza A. Calibration of a measurement system for the evaluation of efficiency indexes in bicycle training. Proceedings of ECIFMBE IFMBE Proceedings Antwerp, Belgium 23-27/11/2008, 22, 106-109.
[31] Bibbo D.; Conforto S.; Bernabucci I.; Carli M.; Schmid M.; D'Alessio T. Analysis of different image-based biofeedback models for improving cycling performances, Proceedings of SPIE 2012-The International Society for Optical Engineering, art. no. 829503.
[32] Prilutsky B.I.; Gregor R.J.; Ryan M. Coordination of two-joint rectus femoris and hamstrings during the swing phase of human walking and running. EBR 1998, 120.4, 479-486.
[33] Crowninshield R.D.; Brand R.A. A physiologically based criterion of muscle force prediction in locomotion. Journal of biomechanics 1981, 14.11, 793-801.
[34] Dul J.; Johnson J.E.; Schiavi R.; Townsend M.A. Muscular synergism - II. A minimum-fatigue criterion for load sharing between synergistic muscles. Journal of biomechanics 1984.
[35] F. Riganti Fulginei, A. Laudani, A. Salvini, and M. Parodi, “Automatic and parallel optimized learning for neural networks performing MIMO applications,” Adv. Electr. Comput. Eng.(AECE), vol. 13, no. 1, pp. 3–12, 2013."