Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Efficient Hardware Implementation of Nonlinear Moving-horizon State Estimation with Artificial Neural Networks

Rafael Koji Vatanabe Brunello * **Renato Coral Sampaio** *
Carlos H Llanos * **Leandro dos Santos Coelho** **,***
Helon Vicente Hultmann Ayala ****

* *Department of Mechanical Engineering, UnB, 70910-900 DF Brazil*
** *Industrial and Systems Engineering Graduate Program, PUCPR, 80215-910 PR Brazil*
*** *Department of Electrical Engineering, UFPR, 81531-980 PR Brazil*
**** *Department of Mechanical Engineering, PUC-Rio, 22231-180 RJ Brazil*
(e-mail: rafaelkvb@gmail.com, renatocoral@unb.br, llanos@unb.br, leandro.coelho@pucpr.br, helon@puc-rio.br)

Abstract: In this contribution we investigate the application of radial basis functions artificial neural networks embedded in hardware for real-time moving-horizon state estimation. The solution of the optimal moving-horizon state estimation problem may be faced as the mapping from the inputs and outputs of the system to the state estimates according to the system model. This mapping may be solved offline with the optimal formulation and then approximated by any higher order function approximation algorithm, such as the ones from machine learning. An approximate version with radial basis functions neural networks is developed and implemented in a Field Programmable Gate Array (FPGA) showing good results in terms of accuracy and computational time. We show that the state estimate using the approximate version of the moving-horizon algorithm can be run using a laboratory scale kit of approximately 500 kHz for an inverted pendulum at a clock rate of about 110 MHz. The latency to provide an estimate can be further reduced when FPGAs with higher clocks are used as the artificial neural network architecture is inherently parallel.

Keywords: Estimation and filtering, Nonlinear observers and filter design, Nonlinear predictive control, Neural networks, Embedded computer architectures

1. INTRODUCTION

The state space approach has had a central role in modern control methods (Bennett, 1996), due to, among other reasons, its generality in dealing with complexity of system orders and number of inputs. The state space representation was seminal for the inception of optimal control (Lewis et al., 2012) and filtering (Kalman, 1960a) methods. Model predictive control strategies are regarded as one of the most important recent developments in the control field (Raković and Levine, 2019). Frequently new methods in nonlinear optimal control are derived according to the model predictive control philosophy, and so we met no decrease in computational complexity for methods that ideally run in a deterministic clock frequency. Many methods have been devised in order to embed such control laws, and very practical tools for C code generation, for example, are now available with vast documentation and flexible GPL-3 software licenses (GNU General Public License-3) (Andersson et al., 2019).

On the other hand, state estimation has close relation with the classical regulation problem, as it is in fact mathematically the same (dual) problem (Kalman, 1960b; Bennett, 1996). An analysis on the dual problem for the model predictive control is analysed and used in (Goodwin et al., 2005; Alessandri et al., 2008). The moving-horizon state estimation paradigm uses a sliding window of most recent measurements and a state prediction to infer the state estimates (Rawlings et al., 2017). This is conceptually different than the Kalman original formulation, which uses solely one set of measurements made at a given time instant in order to provide an estimate of the mean and covariance of the states of the system. The advantages in terms of accuracy is generally recognized for moving-horizon approaches for estimation when compared to its Kalman-based counterparts, in spite of the greater computational effort (Haseltine and Rawlings, 2005). In the receding-horizon approach, the algorithm solves an optimization problem at each sampling instant, what makes possible to take into account many measurements and explicitly the bounds in the states. Naturally this advanced state estimation algorithm demands considerable amount of computational resources to run in real-time. This hinders the application of advanced control and estimation methods to embedded solutions and systems that may operate at high frequency rates such as piezoelectric mi-

cromanipulators (Ayala et al., 2015), (Ayala et al., 2018) and vibration mitigation (Zorić et al., 2019).

In the present work we evaluate the real-time implementation of the approximate filter proposed in (Alessandri et al., 2008, 2011) for the first time, to the best of our knowledge, in FPGAs using new architectures developed to implement an artificial neural network. To this end, we improved a previous implementation of a radial basis function artificial neural network made by the authors in (Ayala et al., 2017), by minimizing the use of resources for calculating the output of each neuron of the artificial neural network and also adapting it to the multiple-output case needed for the moving-horizon state estimation. In summary, the contributions of the present work are (i) the implementation of the moving-horizon state estimation scheme in (Alessandri et al., 2008, 2011) using artificial neural networks and floating-point representation in FPGAs, showing that it is possible to obtain very small sampling times by employing this strategy using ad-hoc hardware implementations and leveraging the inherent parallel architecture of this model class; (ii) the improvement of the hardware implementation of the radial basis function artificial neural network detailed in (Ayala et al., 2017) with respect to the use of resources and adaption to the multiple output case.

The remainder of the paper is organized as follows. The moving-horizon state estimation algorithm is depicted in Section 2 while the design of its approximate version is given in Section 3 together with the hardware architecture proposed for the artificial neural network. In Section 4 we give the results in terms of accuracy and latency obtained for the filter. Finally, Section 5 ends the document with conclusions and perspectives for future work.

2. MOVING-HORIZON STATE ESTIMATION

In the present paper we are concerned with the state estimation problem for system of the type

$$x_{t+1} = f(x_t, u_t) + \xi_t \tag{1a}$$

$$y_t = h(x_t) + \eta_t \tag{1b}$$

where $t \in \mathbb{Z}_+ \triangleq \{0, 1, \ldots\}$ denotes the time sample, $x_t \in \mathbb{R}^n$ describes the continuous state of the system, $u_t \in \mathbb{R}^q$ is the control input and $y_t \in \mathbb{R}^g$ is the system measured output. The quantities $\xi_t \in \mathbb{R}^n$ and $\eta_t \in \mathbb{R}^g$ are additive disturbances affecting the system dynamic states and the measured output, respectively. The former may represent any uncertainty in modeling the system equations and the later eventual disturbances affecting the measurements. Functions $f : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^g$ are in general nonlinear. We employ the moving-horizon state estimation algorithm as described in (Alessandri et al., 2008). The moving-horizon approach for state estimation, also termed receding-horizon, comprehend a set of estimation algorithms in which the most distinctive feature is the use of a sliding window containing a set of most recent measurements. This is in contrast with the Kalman filter-based solutions, which solely use the most recent measurement and a state estimation propagation from one time instant to the next.

The filter in (Alessandri et al., 2008) has asymptotic convergence guarantees, given mainly that the system is state observable according to an ad-hoc definition for the case of moving-horizon state estimation. Briefly speaking, the moving-horizon estimation paradigm uses the most recent amount of information available and it is assumed that a nonlinear state space model is available. As the estimation problem is termed as an optimization problem to be solved online once new measurements become available, it is possible to explicitly take into account the constraints of the system variables.

Consider the cost function J given by

$$J = \mu \| \hat{x}_{t-N,t} - \bar{x}_{t-N} \|^2 + \sum_{i=t-N}^{t} \| y_i - h(\hat{x}_{i,t}) \|^2 \tag{2}$$

where $N + 1$ is the size of the window considered for calculating J, $\bar{x}_t$ is the state prediction at time t, and $\hat{x}_{i,t}$ is the state estimation of the state at time i made at time t, with $N - t \leq i \leq t$. The variable μ is used to weight the confidence we have about the measurements with respect to an initial estimate of the state $\bar{x}_0$ provided by the user. When the cost function J is minimized having $\hat{x}_{t-N,t}$ as the decision variable, that is

$$\begin{aligned} &\hat{x}_{t-N,t} = \arg\min J \\ &\text{s.t. } \hat{x}_{i,t} \in \Omega \end{aligned} \tag{3}$$

where Ω denotes the closed set of admissible state estimates, and $I_t = [y_{t-N}, \ldots, y_t, u_{t-N}, \ldots, u_t]^\intercal$ is the information vector with all the information within a sliding window of the output and input vectors, we may obtain a state estimate in the beginning of the moving-horizon window. Note that for sake of simplicity, when solving Eq. (3) we focus on obtaining the state estimate at $t - N$, that is, at the beginning of the moving-horizon window. Great simplification is achieved in this process by obtaining for the rest of the state estimates within the window by recursively iterating the dynamic state equation as

$$\hat{x}_{i+1,t} = f(\hat{x}_{i,t}, u_t) \tag{4}$$

Summing up, the moving-horizon state estimation algorithm adds up to solving Eq. (3) to obtain the state estimate at the beginning of the window while the rest of the state estimates are obtained by the iterative application of Eq. (4).

3. DESIGN OF APPROXIMATE MOVING-HORIZON STATE ESTIMATION

The optimization problem in Eq. (3) may be performed in order to obtain the state estimates within a sliding window. Being so, it is necessary to solve at each sampling instant an optimization problem, what in many cases may be unpractical or demand significant computational resources. In (Alessandri et al., 2008) the authors propose also an approximate version of the algorithm described in Eqs. (3) and (4), which was further analysed for the case of artificial neural networks in (Alessandri et al., 2011). The method for obtaining an approximate version of the moving-horizon estimation algorithm as given in Section 2 can be obtained if the mapping from the state prediction and information vector, which is performed offline solving an optimization problem, is done by any function approximation regression model. In other words, we want to construct an approximate nonlinear mapping for the mathematical operation performed in the optimization in

Eq. (3), which will be much less computational intensive and indicated for an embedded computation platform.

In order to perform the nonlinear mapping from observations to state estimates, we may employ any nonlinear approximation function for regression as we have many examples from machine learning. One of the cases is the artificial neural network, which is particularly interesting in real-time applications due to its inherent parallel architecture.

3.1 Radial Basis Functions Artificial Neural Networks

The radial basis functions artificial neural network may be represented as

$$\hat{\lambda} = F[r] = W \cdot \Phi(r, C, \sigma), \tag{5}$$

where $\hat{\lambda} \in \mathbb{R}^p$ and $r \in \mathbb{R}^{n_r}$ are respectively the i-th network predicted output and the input vector, $C \in \mathbb{R}^{M,n_r}$ and $\sigma \in \mathbb{R}^M$ are respectively the centers and the widths of the radial basis functions artificial neural network, $M \in \mathbb{N}^+$ is the number of neurons in the hidden layer, and the output weights are given by $W \in \mathbb{R}^{p,M}$. The outputs of the hidden layer are given by

$$\Phi(r, C, \sigma) = \begin{bmatrix} \phi(r, C^1, \sigma^1) \\ \vdots \\ \phi(r, C^M, \sigma^M) \end{bmatrix} \tag{6}$$

where $\phi(r, C^m, \sigma^m)$ is the output of the hidden layer for each m-th neuron [1]. The Gaussian activation function is more frequently used in the case of radial basis functions artificial neural networks. They may be calculated as

$$\phi(r, C^m, \sigma^m) = \exp\left[-\frac{\|r - C^m\|^2}{2(\sigma^m)^2}\right]. \tag{7}$$

The squared vector norm calculation may be summarized as

$$\phi(r, C^m, \sigma^m) = \exp\left[-\frac{1}{2(\sigma^m)^2}\sum_{i=1}^{n_r}(r^i - C^{m,i})^2\right] \tag{8}$$

which is more conveniently used for calculation in hardware as will be seen next.

The radial basis function artificial neural network can be employed for state estimation according to the approximate strategy devised previously. We look for constructing a nonlinear mapping $F : \mathcal{X} \times \mathcal{I} \to \mathcal{X}$ which can be constructed by employing an artificial neural network to fit a dataset constructed offline by solving (3) in a simulation environment, where $\mathcal{I}, \mathcal{X}$ represent respectively the function domains for the information vector and the state estimate. Being so, the artificial neural network will deliver a state estimate readily once the information vector and the state prediction are updated. Thus if we set

$$\lambda_t = \hat{x}_{t,t}, r_t = \begin{bmatrix} \bar{x}_{t-N,t} \\ I_t \end{bmatrix} \tag{9}$$

and build a dataset to train the artificial neural network using simulation data and the results of the moving-horizon which is rich enough to represent the system dynamics, it is possible to construct a proxy for the optimal moving-horizon estimator using an artificial neural network. In the following section we devise hardware architectures to implement the radial basis functions artificial neural network as in (5).

[1] We denote by A^k the vector composed by the k-th line of a matrix A. Similarly, the k-th component of a vector v is given by v^k.

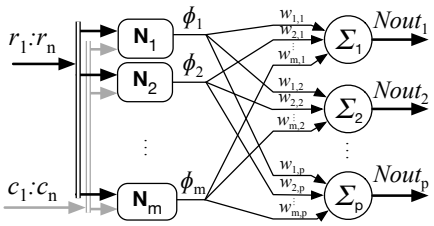

Fig. 1. Radial basis function artificial neural network hardware architecture.

3.2 Approximation Residual Evaluation

It is possible to compare λ_t and $\hat{\lambda}_t$ to evaluate the approximation capability of the artificial neural network. If we define the residual measure as $e_t^i = \lambda_t^i - \hat{\lambda}_t^i$, we can calculate the multiple correlation coefficient for each i-th output as (Schaible et al., 1997)

$$R^{2,i} = 1 - \frac{\sum_{t=1}^{N}\left[e_t^i\right]^2}{\sum_{t=1}^{N}\left[\lambda_t^i - \bar{\lambda}^i\right]^2} \tag{10}$$

where N is the total number of samples and the upper bar denotes the mean of a sequence. The closer R^2 is to 1, the better is the approximation capability of the artificial neural network. It is convenient to use this measure as it allows the comparison of outputs of different units and magnitudes, as we will evaluate the approximate estimation of translation/angular positions/velocities.

3.3 Hardware Architectures for Radial Basis Function Artificial Neural Network

The generalized neural network architecture is presented in Fig. 2 which shows the variable inputs ($r_1 : r_n$), the constant Gaussian centers ($c_1 : c_n$), obtained from the training process, all as inputs of the hidden layer neurons ($\mathbf{N}$). The hidden layer neurons initiate their computations in parallel with a *start* signal and when their output values are done, yield a *ready* signal which initiates the output layer ($\sum$) compute process which uses the hidden layer outputs (Φ) and constant weights ($w_{m,p}$). Finally, the output layer raises a *ready_all* signal when the output values are available.

The hidden neuron module described in Fig. 2 is responsible for implementing Eq. 8 and uses two *FPadd*, a *FPmul* and a *FPexp* units to implement the logic. The process is controlled by an FSM using the start and ready signals for the *FPadd*, *FPmul* and *FPexp* units and synchronizes them using two MUXES with their respective switch signals (sel_1, sel_2) to efficiently minimize hardware area reusing each FP unit.

The output layer neurons follow a similar architecture each using one *FPadd* and one *FPmul* unit and is represented in Fig. 3.

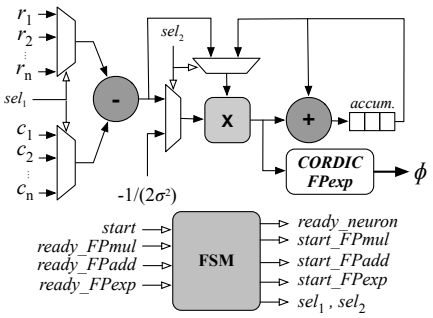

Fig. 2. Gaussian radial basis function neuron architecture on hardware, denoted by **N** in Fig. 1.

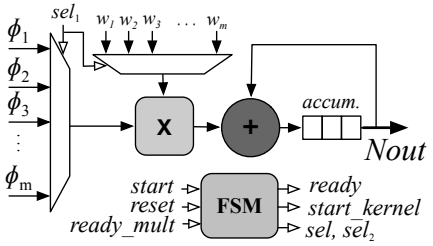

Fig. 3. Radial basis function output layer neuron architecture on hardware, denoted by $\sum$ in Fig. 1.

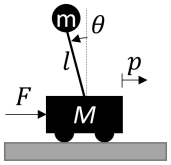

Fig. 4. Representation of the inverted pendulum system, which is used as a case study for state estimation in the present paper.

4. RESULTS

4.1 Case Study

In order to test the moving-horizon state estimation algorithm implementation on hardware, we use an inverted pendulum. See Fig. 4 for details. Let p, θ be respectively the translation of the cart and angular position of the pendulum, m, M be each of the concentrated masses, l and J are respectively the length and moment of inertia of the pendulum and F represents an exogenous force which is the input of the system. If we set the continuous-time state as $z = \begin{bmatrix} p & \theta & \dot{p} & \dot{\theta} \end{bmatrix}$ and the input as $w = F$, we can describe the equations of motion in nonlinear continuous-time state space form as (Aström and Murray, 2008)

$$\dot{z} = f_c(z, w) = \begin{bmatrix} \dot{p} & \dot{\theta} & \ddot{p} & \ddot{\theta} \end{bmatrix}^\intercal \tag{11}$$

with the dynamic equations of the transition of the state in continuous time are given by

$$f_c^3(z,w) = \frac{-mls_\theta\dot{\theta}^2 + m^2gl^2s_\theta c_\theta/J_t - c\dot{p} - mlc_\theta\gamma\dot{\theta}/J_t}{M_t - m^2l^2c_\theta^2/J_t}$$
$$f_c^4(z,w) = \frac{-ml^2s_\theta c_\theta\dot{\theta}^2 + M_tgls_\theta - clc_\theta\dot{p} - \gamma\dot{\theta}M_t/m + lc_\theta u}{J_t(M_t/m) - ml^2c_\theta^2} \tag{12}$$

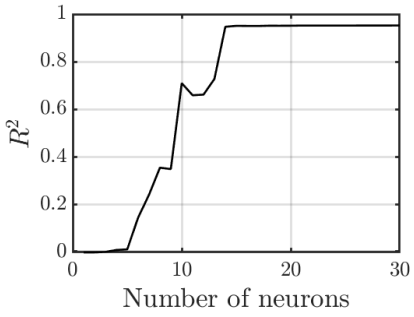

Fig. 5. Multiple correlation coefficient for the worst state estimate obtained for the radial basis functions artificial neural networks varying the number of neurons. We can see that there is a point which represents a good trade-off for accuracy and complexity.

where $s_\theta = \sin\theta, c_\theta = \cos\theta$, $M_t = m + M$, and $J_t = J + ml^2$. We consider to measure both positions of the cart and the pendulum, that is, $y = [p \quad \theta]^\intercal$. The inverted pendulum and its close variants have been used in many recent works with respect to control and state estimation (e.g., to cite a few recent contributions, (Dwivedi et al., 2017; Messikh et al., 2017; Su et al., 2018), due to its switching stable/unstable regimes. As it is a well-known case study, it is a good choice for testing new implementations on hardware for state estimation algorithms.

4.2 Optimal Moving-Horizon State Estimator Design

Data is generated using a 4th order Runge-Kutta solver with sampling time of 10 ms, so that the discrete-time model formulation as in (1) can be employed. Simulated data is generated with a sinusoidal input of 10 rad/s and 50 N amplitude, for 20 seconds in total, and the measurements are corrupted by a white noise signal, so that the mean is the true measurement and with covariance matrix as $\mathrm{diag}(0.05, 0.1)$. The parameters of the physical system used in the simulations are $M = 10$ kg, $m = 80$ kg, $c = 0.1$ N s/m, $J = 100$ kg.m^2/s^2, $l = 1$ m, $\gamma = 0.01$ N.m.s, $g = 9.8$ m/s^2, all with proper units.

The optimal moving-horizon state estimator is then run in order to establish the state estimations as required. After some trial and error, we set $N + 1 = 10$ as the moving-horizon length and $\mu = 0$. We use thus for this example no state prediction and a window containing the 10 most recent measurements. The size of the window has been set so as to guarantee constrained error as $t \to \infty$. By running the optimal estimator, we were able to generate all the inputs necessary for the approximate version of the state estimator. The required data to construct the approximate version are, according to Eq. (9), the time history of the information vector and the state estimate in the beginning of the moving-horizon window, as we are not using the state prediction for sake of simplicity as the focus in on the comparison of execution time on different heterogeneous platforms as we see next.

4.3 Approximate Moving-Horizon State Estimator Design

An approximate filter has been obtained for optimal moving-horizon estimator, using a radial basis function

Table 1. Accuracy of the hardware implementation of the moving-horizon state estimation implemented on hardware. In the table below, the state estimates of the optimal filter performed offline, the FPGA implementation, and artificial neural network implemented on MATLAB are described respectively as $\hat{x}^{\text{opt}}_{t-N,t}$, $\hat{x}^{\text{FPGA}}_{t-N,t}$, and $\hat{x}^{\text{ANN}}_{t-N,t}$.

Metric \ No. of neurons	10	11	12	13	14	15	16
$\text{MSE}_{\text{dB}}\left(\hat{x}^{\text{opt}}_{t-N,t}, \hat{x}^{\text{FPGA}}_{t-N,t}\right)$	7.53	11.12	9.82	5.27	-6.87	-8.18	-9.84
$\text{MSE}_{\text{dB}}\left(\hat{x}^{\text{FPGA}}_{t-N,t}, \hat{x}^{\text{ANN}}_{t-N,t}\right)$	-178.26	-170.36	-167.38	-157.00	-170.90	-183.05	-184.52

Table 2. FPGA Synthesis Results.

Neurons	ALMs (251.680)	REGs	Mult. (156)	Freq. (MHz)	Cycles
14	45,986	14,509	72	111.4	178
15	48,596	15,470	76	109.2	179
16	51,383	16,435	80	108.8	182

Table 3. FPGA Timing Results.

Neurons	FPGA (μs)	ARM (μs)	Speed up
14	1.60	50.58	31.6 x
15	1.64	54.19	33.0 x
16	1.67	57.80	34.6 x

artificial neural network as in Eq. (5). We set the inputs of the network as the information vector (I_t) containing all inputs and outputs within the receding-horizon window, as we have set $\mu = 0$. We trained the network using the data from the optimal estimation procedure using a 2-steps procedure (Haykin, 2009), by selecting the centers using the k-means algorithm, fine tuning the spread by testing many different values and getting the output layer weights with the Penrose-Moore pseudo-inverse with QR factorization (Moody and Darken, 1989).

The training has been done for a range of $1/\sigma^2$ in $[0.1, 100]$ with step of 0.1 between two consecutive values. The impact of the number of neurons on the estimation accuracy was also evaluated by varying it in the range $[1, 30]$. As the k-means algorithm depends on random initial solutions, we tested all these configurations 5 times. Testing 5 times, for the range of $1/\sigma^2$ and number of neurons set amounts to 150,000 different artificial neural networks tested. Fig. 5 depicts the multiple correlation coefficient as in (10) for the least accurate state (among the four) evaluated for a given number of neurons in the artificial neural network. It can be seen that there is a point which augmenting the number of neurons does not improve significantly the accuracy of the artificial neural network in estimating the states. As the hardware resource use, energy consumption and latency are directly affected by the number of neurons, which represent the computational complexity of the state estimator, we favor the solution that represents the best compromise between accuracy and complexity. We then chose the artificial neural networks around the 14 neurons solution, which is the inflection point of the curve with greatest R^2.

4.4 FPGA synthesis results

The radial basis function artificial neural network as reported in Subsection 3.3 is implemented using VHDL code. This code was simulated using ModelSim for logic validation synthesized on Intel® Quartus Prime 17.1 for the FPGA implementation. The target FPGA was an Intel Arria 10 (10AS066N3F40E2SG). The mean squared errors (MSE) obtained by the estimation schemes implemented are summarized by using a modified mean square error metric for the case of state vectors, as

$$\text{MSE}_{\text{dB}}\left(x^{[1]}_{t-N,t}, x^{[2]}_{t-N,t}\right) = 20\log\left[\frac{1}{N_t - N}\sum_{t=N+1}^{N_t}\|\hat{x}^{[1]}_{t-N,t} - \hat{x}^{\text{FPGA}}_{t-N,t}\|^2\right] \quad (13)$$

where N_t is the total amount of samples and $x^{[1]}_{t-N,t}, x^{[2]}_{t-N,t}$ are two different state estimates at t made at $t - N$ one wishes to compare. Being so, the MSE_{dB} presents an overall metric of how well the estimates at the beginning of the sliding window are made. Minimizing MSE_{dB} means that the difference of $x^{[1]}_{t-N,t}, x^{[2]}_{t-N,t}$ are minimized. For example, if we compare the optimal solution with the approximate implemented in hardware, minimizing MSE_{dB} means improving the accuracy of the FPGA solution. Table 1 summarizes the MSE_{dB} for the cases in the inflection point of the R_2 curve shown in Fig 5, namely the solutions around the 14 neurons case. In this table we see the comparison of estimates made on the FPGA, the artificial neural network run offline, and the optimal moving-horizon state estimation. From the results presented in the table, we can see that (i) for a number of neurons greater than 13 the estimates do not present great improvements in terms of accuracy, what corroborates the information given in Fig 5 and (ii) that the implementation on hardware is accurate when compared to the MATLAB offline solution, as the errror is virtually zero (please note that the values are given in dB). Thus, we look further for the synthesis results for number of neurons greater than 13, as we shall analyze in the following, as they present a good compromise in terms of accuracy, hardware consumption.

Table 2 shows the synthesis results including the number of Adaptive Logic Modules (ALMs), Registers (REGs), dedicated hardware multipliers (Mult.), maximum frequency (Freq.) and the number of cycles needed to compute each solution. Three architectures with a varying number of neurons from 14 to 16 where synthesized to illustrate the trade-offs between accuracy and hardware use. All floating-point operations are synthesized with double precision arithmetic (64 bits) since lower precision operators incur in high computation errors for these radial basis function artificial neural network configurations.

In Table 3 we see the timing results of the approximate moving-horizon state estimator. The time to process an state estimate of the artificial neural network have been calculated by $T_i \times$ no. of cycles, using information from the last two columns of Table 2 (T_i is the respective period in

each artificial neural network). It is possible to see that the approximate filter implemented on FPGAs achieves considerably high frequency rates, in the order of 500 kHz. The result is compared to a software solution running on an ARM processor of a Raspberry Pi 3 to showcase the speed up that can be achieved by the dedicated hardware solution. A speed up rate higher than 31 times is achieved.

5. CONCLUSION

We have used the inherent parallel structure of artificial neural networks and FPGAs, what is a synergistic combination since the FPGAs allow the ad-hoc implementation of fast parallel computations directly on hardware. The implications of the work herein presented confirm the expectations of the approximate filter presented in (Alessandri et al., 2008, 2011), for the first time implemented in hardware in the present work. It is possible to calculate the estimation offline and accurately approximate the state estimates by means of an artificial neural network online, which performs very fast due to its inherent parallel architecture. This has been explored in the present paper by the implementation of the approximate filter directly on hardware. In future work we plan to test the approximate moving-horizon state estimator implemented in FPGAs in real-world systems with fast dynamics (Ayala et al., 2018), to propose a didactic test-bench with the inverted pendulum (Magana and Holzapfel, 1998) for receding-horizon estimation and control in FPGAs with approximate solutions and artificial neural networks (Zoppoli et al., 2020).

ACKNOWLEDGEMENTS

The authors would like to thank National Council of Scientific and Technologic Development of Brazil - CNPq (Grants: 409274/2016-0-Univ, 400119/2019-6-Univ) for its financial support of this work.

REFERENCES

Alessandri, A., Baglietto, M., Battistelli, G., and Gaggero, M. (2011). Moving-horizon state estimation for nonlinear systems using neural networks. *IEEE Transactions on Neural Networks*, 22(5), 768–780.

Alessandri, A., Baglietto, M., and Battistelli, G. (2008). Moving-horizon state estimation for nonlinear discrete-time systems: New stability results and approximation schemes. *Automatica*, 44(7), 1753–1765.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Aström, K.J. and Murray, R.M. (2008). *Feedback systems: an introduction for scientists and engineers*. Princeton university press, Princeton, New Jersey.

Ayala, H.V.H., Rakotondrahe, M., and d. S. Coelho, L. (2018). Modeling of a 2-DOF piezoelectric micromanipulator at high frequency rates through nonlinear black-box system identification. In *American Control Conference*, 4354–4359. Milwaukee, Wisconsin, USA.

Ayala, H.V.H., Habineza, D., Rakotondrabe, M., Klein, C.E., and Coelho, L.S. (2015). Nonlinear black-box system identification through neural networks of a hysteretic piezoelectric robotic micromanipulator. *IFAC-PapersOnLine*, 48(28), 409 – 414.

Ayala, H.V.H., Muñoz, D.M., Llanos, C.H., and Coelho, L.S. (2017). Efficient hardware implementation of radial basis function neural network with customized-precision floating-point operations. *Control Engineering Practice*, 60, 124 – 132.

Bennett, S. (1996). A brief history of automatic control. *IEEE Control Systems Magazine*, 16(3), 17–25.

Dwivedi, P., Pandey, S., and Junghare, A.S. (2017). Stabilization of unstable equilibrium point of rotary inverted pendulum using fractional controller. *Journal of the Franklin Institute*, 354(17), 7732 – 7766.

Goodwin, G.C., Doná, J.A.D., Seron, M.M., and Zhuo, X.W. (2005). Lagrangian duality between constrained estimation and control. *Automatica*, 41(6), 935 – 944.

Haseltine, E.L. and Rawlings, J.B. (2005). Critical evaluation of extended kalman filtering and moving-horizon estimation. *Industrial & Engineering Chemistry Research*, 44(8), 2451–2460.

Haykin, S.S. (2009). *Neural networks and learning machines*. Prentice Hall, Upper Saddle River, 3rd edition.

Kalman, R.E. (1960a). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82(1), 35–45.

Kalman, R. (1960b). On the general theory of control systems. *IFAC Proceedings Volumes*, 1(1), 491 – 502. 1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR, 1960.

Lewis, F.L., Vrabie, D., and Syrmos, V.L. (2012). *Optimal control*. John Wiley & Sons, Hoboken, New Jersey.

Magana, M.E. and Holzapfel, F. (1998). Fuzzy-logic control of an inverted pendulum with vision feedback. *IEEE Transactions on Education*, 41(2), 165–170.

Messikh, L., Guechi, E., and Benloucif, M. (2017). Critically damped stabilization of inverted-pendulum systems using continuous-time cascade linear model predictive control. *Journal of the Franklin Institute*, 354(16), 7241 – 7265.

Moody, J. and Darken, C.J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2), 281–294.

Raković, S.V. and Levine, W.S. (2019). *Handbook of model predictive control*. Birkhäuser, Cham, Switzerland.

Rawlings, J.B., Mayne, D.Q., and Diehl, M. (2017). *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing, Madison, WI.

Schaible, B., Xie, H., and Lee, Y.C. (1997). Fuzzy logic models for ranking process effects. *IEEE Transactions on Fuzzy Systems*, 5(4), 545–556.

Su, X., Xia, F., Liu, J., and Wu, L. (2018). Event-triggered fuzzy control of nonlinear systems with its application to inverted pendulum systems. *Automatica*, 94, 236 – 248.

Zoppoli, R., Parisini, T., Baglietto, M., and Sanguineti, M. (2020). *Neural Approximations for optimal control and decision*. Springer, Cham, Switzerland.

Zorić, N.D., Tomović, A.M., Obradović, A.M., Radulović, R.D., and Petrović, G.R. (2019). Active vibration control of smart composite plates using optimized self-tuning fuzzy logic controller with optimization of placement, sizing and orientation of pfrc actuators. *Journal of Sound and Vibration*, 456, 173 – 198.