24th COBEM - 2017

24th ABCM International Congress of Mechanical Engineering
December 3-8, 2017, Curitiba, PR, Brazil

COBEM-2017-1772
SUPPORT VECTOR REGRESSION BASED NONLINEAR MODEL PREDICTIVE CONTROL ON FPGA

Renato Coral Sampaio
Faculdade Gama - University of Brasília - Brasília - DF
renatocoral@unb.br

Carlos Eduardo da Silva Santos
Federal Institute of Education, Science and Technology of Tocantins - Palmas - TO
carlosedu@ifto.edu.br

Carlos Humberto Llanos
Ricardo Pezzuol Jacobi
Department of Mechanical Engineering - University of Brasília - Brasília - DF
llanos@unb.br
jacobi@unb.br

Leandro dos Santos Coelho
Industrial and Systems Engineering Graduate Program - Pontifical Catholic University of Paraná and Electrical Engineering Graduate Program - Federal University of Paraná - Curitiba - Paraná - Brazil
leandro.coelho@pucpr.br

Helon Vicente Hultmann Ayala
IBM Research Brazil - Rio de Janeiro - Rio de Janeiro - Brazil
helonv@br.ibm.com

Abstract. *Model based predictive control (MPC) is a control technique currently applied in industry which yields very effective control performance. Since it requires solving an optimization problem at each sampling interval, its computation cost is very high when compared to other control techniques. Recently, there is a great focus on applying MPC in real-time embedded systems. In the last decade various methods to achieve fast embedded solutions for Linear MPC were proposed. For more complex nonlinear systems though the problem remains open. This paper proposes a very fast hardware architecture for Nonlinear MPC by approximating an offline solution to a Support Vector Regressor (SVR) and implementing it on a FPGA. The SVR training process is done by the proposed Nature Inspired Optimization Tools for SVM (NIOTS) where various input configurations are explored in search of an optimal solution. The NIOTS tool is based on a Multi-Objective Particle Swarm Algorithm (MOPSO) and is able to generate a set of solutions that balance complexity and precision. The results show three SVR implementations in FPGA based on floating-point operations that are able to compute a control action in two about microseconds.*
Keywords: *Nonlinear Model Predictive Control, Support Vector Regression, Field Programable Gate Arrays.*

1. INTRODUCTION

Model Based Predictive Control (MPC) is a sophisticated and very effective control technique that originated in the 1960's and was first applied in the process industry. Among its main advantages are the ability to deal with complex dynamic systems with multiple inputs and multiple outputs including control and state constrains, which are built in to the control law. The main idea behind MPC is that, at each sampling time, a control action is computed by solving an optimal control problem over a finite horizon based on the system model and its current state.

Due to the computational demanding nature of the algorithm, which includes solving an optimization problem at each sampling time, it is only recently that this method has been used to control systems with fast dynamics. Recent studies such as Mayne (2014) and Gros *et al.* (2016) show many advancements in the application of Linear MPC in real-time in systems with fast dynamics with sampling times below the milliseconds threshold. For nonlinear systems though, the computational requirements are even higher making it more difficult to implement a real-time controller with such high

frequency.

One approach to create a viable Nonlinear MPC (NMPC) solution is through the use of Artificial Neural Networks (ANNs). Since ANNs can be trained to act as a nonlinear system and by nature its computation can be highly parallelized, it becomes a good candidate to solve this problem. As described by Lawryńczuk (2009), most approaches use an ANN to model the nonlinear system and accelerate the prediction step and then use a nonlinear optimization technique to compute the control action. Another approach, which is presented in Ortega and Camacho (1996), computes the control law of the NMPC in various conditions off-line and then trains an ANN to map this behavior and uses it for the on-line controller. This approach can also be observed in Åkesson *et al.* (2005) and Kittisupakorn *et al.* (2009) although in both cases they are applied to slow processes.

An alternative to the use of ANNs that has been growing in the last decade is the use of Support Vector Machines (SVMs) which represents a type of Machine Learning (ML) technique proved to have high generalization capacity and robustness in both data classification and regression. It's main advantage over other techniques is related to its ability of minimizing the generalization error in the training phase through maximizing the distance between the margins on training data. It was modeled to work over convex problems and therefore yields a unique solution, different from ANN models where the solution has several local minima (Theodoridis and Koutroumbas, 2009). SVMs are also capable of condensing information in the training set, representing the function with very small number of data points, called Support Vectors (SVs).

SVM complexity is measured by the number of SVs and its empirical error is evaluated by the Mean Squared Error (MSE) on the training set. The training process can be tuned by the regularization parameter (C) which controls the balance between the empirical error and the number of SVs, and the kernel parameters. This approach is used to create a training tool called NIOTS where MOPSO (Miranda *et al.*, 2012) is used to guide the training process.

Previous efforts presented in Ayala *et al.* (2016) and Santos *et al.* (2017) used the technique of mapping the whole NMPC offline solution to a Radial Basis Function ANN and then to a Support Vector Regression (SVR), respectively, and then generated a dedicated hardware solution in FPGA. In both cases, controller performances with sampling rates of MHz were achieved but for a very limited input reference variation. This work is focused on applying a similar technique to a nonlinear model of a pendulum on a cart and on improving the SVR training method to generalize the solution to a much wider range of reference inputs. It then deals with the generation of a more complex SVR solution in FPGA.

The remainder of this paper is organized as follows. Section 2 presents the main concepts of Nonlinear Model Predictive Control. In Section 3, the the SVM and SVR mathematical models and their respective training parameters are presented. Section 4 presents details about the NIOTS tool to train the SVR along with the multi-objective optimization techniques (MOOP) develop. Section 5 explains the procedures used in defining the data sets for training and validating the SVR solution along with its hardware implementation. Finally, Section 6 presents the results and discussion while Section 7 ends with the conclusions and future research directions.

2. Nonlinear Model Predictive Control

The MPC is defined as cost function minimization problem that has to be computed at each sampling time. This optimization is computed over a finite horizon of length N and is based on the prediction model of the controlled system. The nonlinear version of the MPC (NMPC) uses a nonlinear prediction model and, according to Grüne and Pannek (2011), can be defined in its discreet form as

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{y}_k = \mathbf{x}_k \tag{1}$$

where $\mathbf{x}_k$, $\mathbf{u}_k$ and $\mathbf{y}_k$ are vectors of the system state, system input and system output, respectively.

The optimal control problem, including the cost function and constrains is defined as

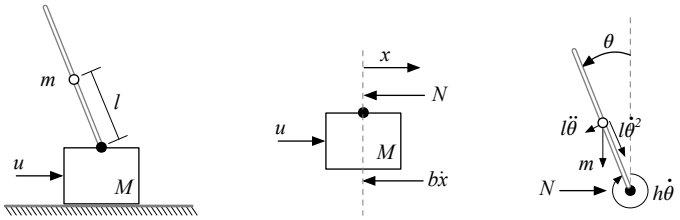
$$\begin{aligned} \min J(\mathbf{x}_k, \mathbf{u}_{k,k+N-1}) \triangleq \quad & \sum_{i=1}^{N} \|\bar{\mathbf{x}}_{k+i} - \mathbf{x}_{k+i}\|^2_{\mathbf{Q}} + \sum_{i=1}^{N} \|\mathbf{u}_{k+i}\|^2_{\mathbf{R}}, \\ s.t. \quad & \mathbf{u}_{min} \leq \mathbf{u}_{k+i} \leq \mathbf{u}_{max} \\ & \Delta\mathbf{u}_{min} \leq \mathbf{u}_{k+i} \leq \Delta\mathbf{u}_{max} \\ & \mathbf{x}_{min} \leq \mathbf{x}_{k+i} \leq \mathbf{x}_{max} \end{aligned} \tag{2}$$

where $\bar{\mathbf{x}}_k$ is the reference of the state $\mathbf{x}_k$ in a given instant k, N is the length of the prediction horizon, $\mathbf{Q}$ and $\mathbf{R}$ are symmetric weighting matrices that penalize deviations from a reference trajectory and a reference control value, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$ are the vectors of control constrains, $\Delta\mathbf{u}_{min}$ and $\Delta\mathbf{u}_{max}$ are the vectors control action variation constrains and finally $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the vectors defining state constrains. The solution to Eq. (2) is the optimal control sequence denoted by $\mathbf{u}_k^{opt}$.

The application of the NMPC follows three steps that are repeated at each sampling time: (a) Measure the system state $\mathbf{x}_k$; (b) Find $\mathbf{u}_k^{opt}$ by solving the finite horizon optimal control problem; (c) Apply the first optimal control action $\mathbf{u}_k^{opt(1)}$ at instant k.

2.1 Inverted Pendulum Model

The system used as a case study for the NMPC is of an inverted pendulum on a cart. The model used was the same as in Mercieca and Fabri (2011) and Alaniz (2004). Figure 1 details the pendulum system along with it's free body diagrams.



(a) (b) (c)

Figure 1: (a) Inverted Pendulum on a Cart; (b) Free body diagram 1; (c) Free body diagram 2.

Equations (3) and (4) describe the system's nonlinear model and are obtained by applying Newton's Laws of Motion to the free body diagrams in Fig. 1. These equations are expressed in terms of the state variables x, $\dot{x}$, θ and $\dot{\theta}$ which are, respectively, the position and velocity of the cart and the pendulum's angle and angular velocity. M is the cart mass, m is the uniformly distributed mass of the ideal pendulum, l is half the length of the ideal pendulum, b is friction damping constant of the surface, h is the rotational friction damping coefficient, g is the gravitational acceleration and u is the control action which is the force applied to the cart.

$$\ddot{x} = \frac{1}{M+m}[u - b\dot{x} - ml\ddot{\theta}\cos(\theta) + ml\dot{\theta}^2\sin(\theta)] \tag{3}$$

$$\ddot{\theta} = \frac{3}{4ml^2}[mgl\sin(\theta) - ml\ddot{x}\cos(\theta) - h\dot{\theta}] \tag{4}$$

The values used for each of the constants are $14.6\,kg$ for M, $7.3\,kg$ for m, l equals $1.2m$, b equals $14.6\,kg/s$, h equals $0.0136\,kg.m^2/s$ and finally g equals $9.81\,m/s^2$.

3. Support Vectors Regression - SVR

The SVMs were designed by Cortes and Vapnik (1995), originally created to classify data and were based on finding an optimal hyperplane with large margins that maximizes distance from classes in the training data set. This technique assures that the error on the training set and its generalization capability can be optimized simultaneously (Burges, 1998; Lima, 2004).

However, to apply the SVM theory to applications having continuous outputs, an extension named Support Vector Regression (SVR) had to be created. SVRs were developed by Drucker *et al.* (1997) to induce the approximated function from a training set $X = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_3) \ldots (\mathbf{x_n}, y_n)\} \subset \mathcal{R}^d \times \mathcal{R}$, where $\mathcal{R}^d$ denotes the space of the input patterns and $\mathcal{R}$ is the label set. Equation 5, shows the general function approximated by SVRs.

$$f(\mathbf{x}) = \sum_{i=1}^{\#SV} (\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}) + b, \tag{5}$$

where α_i and α_i^* are the optimal solutions of the quadratic problem shown in Eq. (6) as defined by Drucker *et al.* (1997), $K(\cdot,\cdot)$ is the kernel function, b is the bias evaluated from α_i and α_i^* and $\#SV$ is the cardinality of the support vector set.

$$\begin{aligned} \max \quad L(\alpha, \alpha^*) = \quad & -\tfrac{1}{2}\textstyle\sum_{i,j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) - \epsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \\ & s.t.\ \textstyle\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C] \qquad i = 1, ..., N. \end{aligned} \tag{6}$$

The quadratic problem in Eq. (6) has three types of parameters, called the hyperparameters, that need to be tuned: C which is the regularization parameter, ϵ which is width of the ϵ-insensitive tube and the kernel parameters. The choice of hyperparameters is closely related to the complexity of the SVR and its generalization capability. In this work, the adopted metrics to evaluate complexity, precision and function fitting of the SVR are the cardinality of the support vectors set, the Mean Squared Error (MSE) and the squared correlation coefficient, respectively. Defining suitable hyperparameters for the SVR is crucial for minimizing these metrics.

Figure 2 depicts the SVR fitting process based on the training data set (black dots). The function adjusting is guided by the ϵ-insensitive that is defined by the ϵ constant. Data outside of the ϵ-insensitive tube is penalized by the regularization parameter. Data points that remain out of the ϵ-insensitive tube are considered support vectors as depicted in Fig. 2.

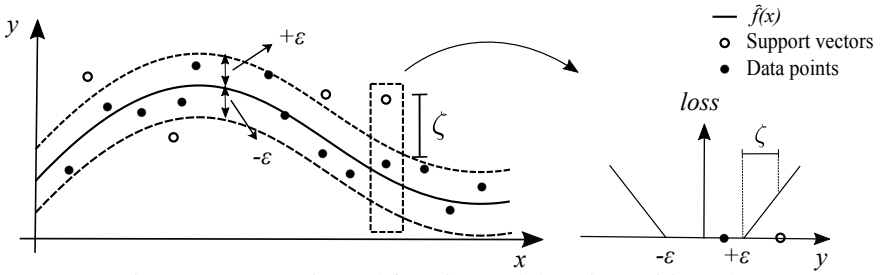


Figure 2: Approximated function and the ϵ-insensitive tube.

The function kernel is an inner product in the high dimensional space, in which the original space data can be approximated by a linear function. In this work, we use the exponential Radio Basis Function (RBF) as kernel function shown in Eq. (7).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\gamma}\right) \tag{7}$$

where $\mathbf{x}_i$, $\mathbf{x}_j$ are the data in original space and the γ is the parameter to be tuned.

4. Nature Inspired Optimization Tools for SVM - NIOTS

The Nature Inspired Optimization Tools for SVM is a system developed to find a solution set as close as possible to the Pareto set. To execute this task NIOTS uses the Particle Swarm Optimization algorithm described in subsection 4.3 , but in its Multi-objective optimization (MOOP) form.

4.1 SVM selection parameters problem as a MOOP

Real problems usually are presented with several criteria that have to be optimized at the same time. The best SVM is able to classify unseen data with maximum accuracy and minimum complexity. However, precision and complexity are conflicting criteria and this is tightly related to the definition of the hyperparameters which is a multimodal problem called SVM selection parameters problem (Yi *et al.*, 2016). It belongs to a class of problems defined as multi-objective, multicriteria optimization, multiperformance or vector optimization problem (Carlos A. Coello Coello and Veldhuizen, 2007). These problems have contradictory criteria functions otherwise, they could be converted into a single function problem without loss information. Equation (8) shows a SVM parameters selection problem modeled as a MOOP.

$$\begin{array}{rl} \min\ \mathbf{F}(C, \gamma, \epsilon) & = \left(MSE, \#SV, 1 - R^2\right) \\ s.t. & 2^C, 2^\gamma \in \left[2^{LB},\ 2^{UB}\right] \\ & \epsilon \in [0,\ UB_\epsilon] \end{array} \tag{8}$$

where MSE is the mean squared error in the validation set, $\#SV$ is the cardinality of the support vector set, R^2 is Pearson's correlation coefficient (Spuler *et al.*, 2015), these two last criteria are evaluated by Eq. (9), Eq. (10) respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{9}$$

the variable n is the cardinality of the validation set, $\hat{y}$ is the prediction of the SVR while y are the labels of the validation set.

$$R^2 = \left(\frac{\sum_{i=1}^{n}(y_i - m)(\hat{y}_i - \hat{m})}{\sqrt{\sum_{i=1}^{n}(y_i - m)^2 \sum_{i=1}^{n}(\hat{y}_i - \hat{m})^2}}\right)^2, \tag{10}$$

where m is the average y_i and $\hat{m}$ is the SVR prediction average. The metric R^2 was modified in the objective function to transform the maximization problem into a minimization problem.

In these problems, the notion of dominance is an important concept to evaluate which solutions are better, because every objective function in MOOP has the same importance. The dominance concept is defined as: A vector $\mathbf{u} = (u_1, u_2, ..., u_k)$ is said to dominate another vector $\mathbf{v} = (v_1, v_2, ..., v_k)$ (denoted by $\mathbf{u} \preceq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e. $\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$.

However, as consequence of the dominance concept, in this produce a set of vector solutions where each of them do not dominate one another is generated and is known as the non-dominated set. In the non-dominated set every solution

has the same importance and the optimal non-dominated set is known as the Pareto set. The image of the Pareto set is a curve in the objective space called the Pareto Front.

Evolutionary algorithms have been used to solve MOOP due to its complexity and multi-modal characteristic. In this work, we use the modified Particle Swarm Optimization to treat multi-objective SVM selection parameters problem.

4.2 Multi-objective Particle Swarm Optimization

Bio-inspired algorithms belong to a class of meta-heuristics based on collective behavior of swarms and fish shoals, these algorithms are often used to solve complex problems. The Particle Swarm Optimization spreads several particles simultaneously over the search space, these particles share information about each other without a central management. Each particle moves over the search space influenced by its current position and the best global position of the swarm. This movement , in iteration t, is directed by a velocity vector evaluated by Eq. (11).

$$v_{ij}^{(t+1)} = wv_{ij}^{t} + c_1 U_{1j}[0,1](y_{ij}^{t} - x_{ij}^{t}) + c_2 U_{2j}[0,1](y_{sj}^{t} - x_{ij}^{t}), \tag{11}$$

where v_{ij} is the velocity of particle i, w is the inertia weight, c_1 and c_2 are the cognitive and social acceleration constants respectively, U_{1j} and U_{2j} are random numbers with uniform distribution on interval $[0, 1]$, y_{ij} is the best position of particle i and y_{sj} is the best particle of swarm. The Eq. (12) defines the position of particle in iteration $t + 1$.

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t}, \tag{12}$$

where t is the iteration counter, x_{ij} is the position of particle i and j is the j-th component of the particle. Equations (11) and (12) drive the particles through the search space based on global and individual bests. However, in MOOP, we have a set of best solutions called non-dominated set, therefore, only one particle is randomly chosen from this set to be the global best.

To pick the individual best, a history of individual non-dominated solutions is kept for each particle, so from this history set, one is randomly chosen to determine the velocity v_{ij}. This procedure ensures better diversity and improves the non-dominated set at each iteration.

In iteration t, the non-dominated set is extracted of the current swarm by a *truncate* function and stored in a matrix, so the new swarm of iteration $t + 1$ is concatenated in the same matrix and again the function *truncate* is applied to update the non-dominated set. The stop criteria is the maximum number of iterations.

4.3 NIOTS

The NIOTS system uses the meta-heuristic algorithm MOPSO described in subsection 4.2 to find a non-dominated set of hyperparameters, as close as possible to the Pareto set. These solutions allow the user to choose the best solution for a specific application. Figure 3 shows the graphical user interface of NIOTS. This system has four environments: (a) Optimization, (b) Prediction, (c) Statistics and (d) Demo.

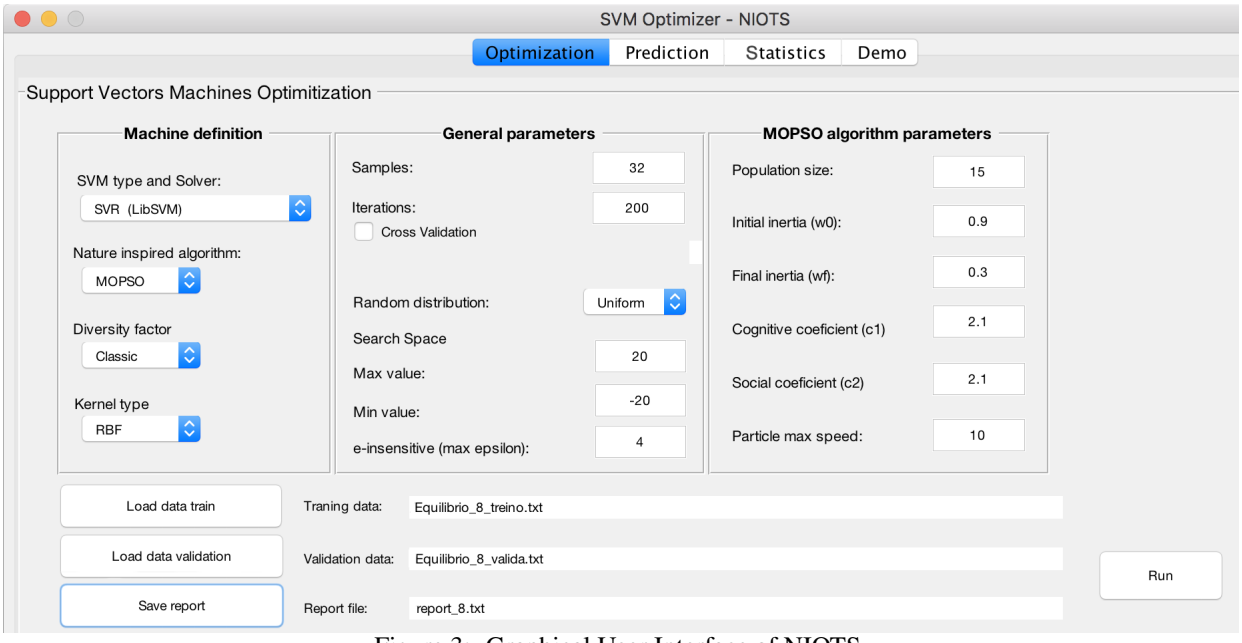


Figure 3: Graphical User Interface of NIOTS.

In the Optimization environment, the *machine definition* panel is used to define the SVM type and its solver, the nature inspired algorithm, a diversity factor and the kernel type. The *general parameters* panel is where the number of samples and iterations of the MOPSO are defined along with the random distribution type and search space parameters. The last panel, *MOPSO algorithm parameters*, is where every MOPSO parameter is defined. In the bottom, the training data used to create the quadratic problem to be solved by LibSVM (Chang and Lin, 2011) and the validation data files are chosen along with the path for the output reports. Finally the *run* button executes the program.

The MOPSO decision variables for Support Vectors Regression are C, γ and ϵ. The search space of the two first variables is defined with the same interval $I = [2^{LB} \;\; 2^{UB}]$. The usual interval for ϵ is $I_\epsilon = [10^{-3} \;\; 4]$.

The non-dominated sets generated by the MOPSO algorithm are evaluated at each iteration by the *space*, *hypervolume* and *cardinality* metrics which are used to analyze the convergence and performance of the MOPSO. The Spacing metric indicates how much the solutions are spread in the Pareto frontier, the Hypervolume evaluates how far the solutions are from the a predefined worst point, and the Cardinality measures the number of solutions in the Pareto set.

5. Experimental Procedure

As described in (Santos *et al.*, 2017), in order to apply the NMPC technique described in Section 2 to the inverted pendulum model using the SVR, the NMPC problem has to be first solved offline to generate the training data sets. To that end, a NMPC algorithm using numerical simulation was used in Matlab. A 4th Order Runge-Kutta method was used to compute the one step prediction of the nonlinear system and the control action was computed by a nonlinear solver. Parameters used for the NMPC include a prediction horizon and a control horizon of 20, $u_{max} = 50N$, $u_{min} = -50N$, the diagonal values of weighting matrix $Q = [100 \;\; 1 \;\; 100 \;\; 1]$ and the control penalty of the cost function $R = 10$. The sampling time used for the control system was 100 ms.

The desired reference trajectory consists of maintaining the inverted pendulum in equilibrium while varying its target position. Figure 4 depicts this trajectory with the offline solution. The three plots from top to bottom are the cart's position, pendulum angle and the control actions. All continuous (black) lines represent the simulated system behavior while the dotted (red) lines represent references trajectories for the state variables and upper and lower bounds for the control action. To measure controller performance the Mean Square Error (MSE) metric was used. In this case, the resulting MSE value is the sum of the MSE of all four system states weighted by the coefficients of matrix Q. So, the position and angle MSE values are multiplied by 100. The resulting MSE for the system in Fig. 4 is 30.04.

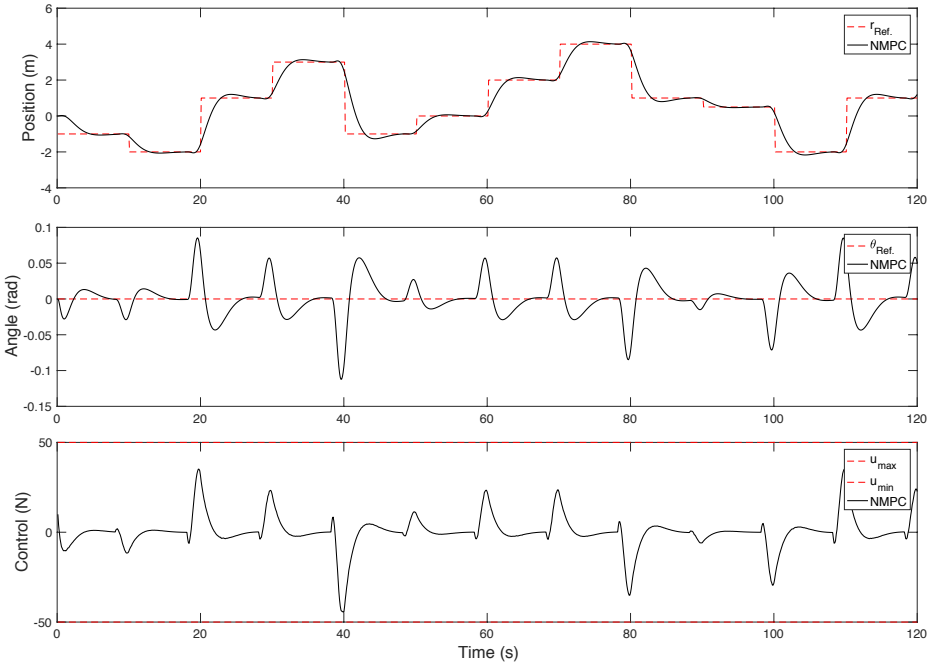
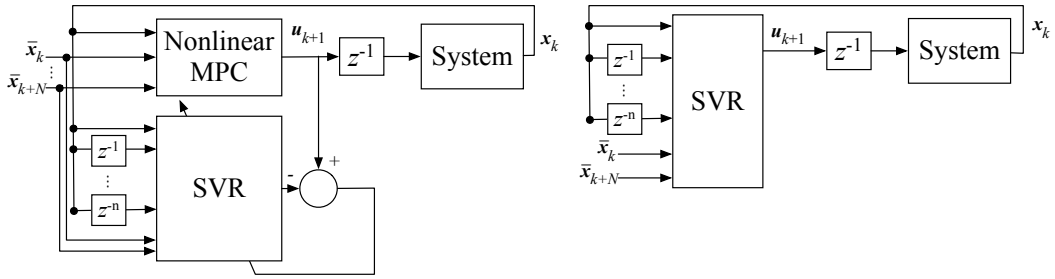


Figure 4: Reference trajectory with the offline solver outputs.

5.1 SVR training

Figure 5a depicts the generation of the training data followed by the SVR training itself and in Fig. 5b the SVR solution controlling the system.

In order for the SVR to correctly capture the nonlinear control dynamic a proper set of inputs have to be chosen. Since the NMPC scheme involves knowing present and past state information and present and future reference, these are natural



(a) MPC offline solution and SVR training (b) Real-time SVR implementation

Figure 5: Schematic model to generate training data and SVR control system.

candidates. Assuming all states are observable, the current state is readily available and past values of the state can be easily stored. Future references for each state are also known from the desired trajectory. What needs to be decided is how much past state samples and future references are enough to capture the system's behavior. In this case, experiments where conducted using states x_k, x_{k-1} upto x_{k-10} and future references equal to the prediction horizon. Considering all these states and future references this configuration would use 124 inputs (1 current state, 10 past states and 20 future references all multiplied by 4 state variables) which can be considered a high number for a FPGA implementation. After a few experiments with trial and error five input configurations where considered as described by Table 1.

Also, to improve the generalization of the solution, it can be observed that the pendulum movements in both directions are equivalent, so a relative position is enough capture the dynamic. So, instead of the raw value of each state x_k, a relative value was calculated as $\bar{x}_k - x_k$. Also, since this problem deals only with step references, using $\bar{x}_{k+1}$, $\bar{x}_{k+N}$ is enough to capture future reference changes.

Table 1: SVR Input Configurations.

Input Config.	N. of Inputs	Inputs
1	8	$\mathbf{x}_k$,, $\bar{\mathbf{x}}_{k+1}$, $\bar{\mathbf{x}}_{k+N}$
2	14	$\mathbf{x}_k, \mathbf{x}_{k-4}, \mathbf{x}_{k-8}, \bar{\mathbf{x}}_{k+N}$
3	16	$\mathbf{x}_k, \mathbf{x}_{k-2}, \mathbf{x}_{k-4}, \bar{\mathbf{x}}_{k+1}, \bar{\mathbf{x}}_{k+N}$
4	20	$\mathbf{x}_k, \mathbf{x}_{k-3}, \mathbf{x}_{k-6}, \mathbf{x}_{k-9}, \bar{\mathbf{x}}_{k+1}, \bar{\mathbf{x}}_{k+N}$
5	28	$\mathbf{x}_k, \mathbf{x}_{k-2}, \mathbf{x}_{k-4}, \mathbf{x}_{k-6}, \mathbf{x}_{k-8}, \mathbf{x}_{k-10}, \bar{\mathbf{x}}_{k+1}, \bar{\mathbf{x}}_{k+N}$

From the system simulation shown in Fig. 4, a training set was generated for each input configuration. Each set corresponds to a simulation of 120 seconds with a 0.1 s sampling time yielding 1200 data points. To prepare this data for the NIOTS tool, each data set was randomly separated in three subsets: training, validation and test sets with 70%, 15% and 15% of the of the original samples, respectively.

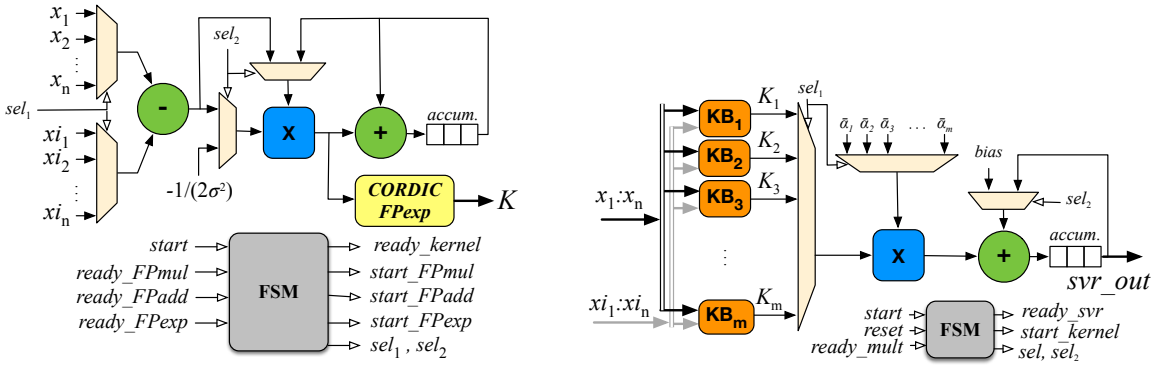
To proceed with the training, the NIOTS tool parameters were configured as is shown in Fig. 3. Although the tool is capable of using other types of kernel, the RBF was chosen as it yields better results to this problem.

5.2 Hardware Architecture

The hardware architerture used to implement the SVR on FPGA is a combination of the RBF kernel described in Ayala *et al.* (2016) and the optimized SVM architecture developed in Santos *et al.* (2017). The RBF kernel can be observed in Fig. 6a while the top level SVR module is shown in Fig. 6b. In both cases, all arithmetic operations are implemented using customizable floating-point libraries based on the IEEE 754 (IEEE, 1985) standard, developed and described in Muñoz *et al.* (2009) and Muñoz *et al.* (2010). The floating-point representation provides a larger dynamic range to solve various problems yielding better overall precision. All floating-point modules used in this work are configures with a bit width of 32, the regular float precision.

6. Results and Discussion

The first results are from the NIOTS training process. Each training provided a Pareto frontier with a series SVR solutions with varying number of support vectors and error metrics. Table 2 shows the 10 best results achieved for the SVR with 28 inputs. Columns 2 to 4 represent the SVR parameters while columns 5 to 7 represent the NIOTS metrics to evaluate the solutions which compares the training results with the input data test set. For the NMPC problem, a final step was added in which the SVR solution is simulated controlling the nonlinear system and a MSE is computed between each state output and its reference an then added together. In this case, columns 8 to 10 represent the MSE values of each



(a) RBF Kernel architecture. (b) Top level architecture of the SVR.

Figure 6: Hardware architecture implemented on FPGA.

SVR solution for a different reference trajectory than the one originally used for the training process and the last column sums all three MSE values. The solution with the lowest value on the last column is usually the one that was able to better generalize the controller dynamics.

Table 2: Pareto frontier Solution for the SVR with 28 inputs.

Index	C	γ	ϵ	Error	SV	$1-R^2$	MSE 1	MSE 2	MSE 3	MSE Sum
1	16.22	-11.87	2.36	0.320	730	3.72E-03	91.10	68.87	78.97	238.94
2	12.33	-8.45	2.57	0.893	13	8.83E-03	43.98	26.89	36.19	107.06
3	15.04	-10.05	1.47	0.354	39	4.29E-03	32.92	19.87	28.62	81.41
4	15.71	-11.51	2.26	0.778	20	5.10E-03	62.56	51.85	60.20	174.61
5	15.00	-10.41	2.19	0.574	18	5.29E-03	111.90	101.76	110.48	324.15
6	13.57	-8.99	2.51	0.693	16	5.85E-03	33.95	20.91	29.27	84.13
7	15.82	-9.53	2.12	0.498	23	5.67E-03	2,468.68	2,485.53	2,533.75	7,487.96
8	12.47	-7.66	1.75	0.450	26	6.00E-03	32.41	19.88	28.00	80.29
9	10.75	-6.23	1.83	0.464	24	6.16E-03	31.96	18.33	27.46	77.75
10	12.03	-6.97	2.22	0.524	21	6.39E-03	31.71	19.10	27.53	78.35

Next, the NIOTS tool was used to generate solutions for all five input configurations and the best results for each one in presented in Table 3. In this case, one selection criteria was solutions with a total MSE (last column) below 80 since above this value the control behavior presented much more noise. A second criteria was a number of support vectors below 30, since above that size it becomes very difficult to fit the solutions in the low cost FPGA used in this work. That last criteria could not be met but solutions with 14 and 16 inputs, so a solution with the lowest number of support vectors that could meet the first criteria was chosen. To put the MSE values in perspective, the first line of Table 3 also presents the MSE values for the original NMPC algorithm used to generate the training data sets.

Table 3: Best SVR solutions for various input configurations.

Solution	Inputs	C	γ	ϵ	Error	SVs	$1-R^2$	MSE 1	MSE 2	MSE 3	MSE Sum
NMPC	-	-	-	-	-	-	-	30.04	19.67	26.21	75.92
SVR Config. 1	8	15.20	-7.97	1.76	33.24	18	0.099	33.22	19.34	28.14	80.7
SVR Config. 2	14	12.91	-9.23	0.38	29.18	76	0.089	29.29	17.64	25.8	72.73
SVR Config. 3	16	14.79	-11.18	0.80	29.06	37	0.089	29.55	17.53	25.79	72.87
SVR Config. 4	20	16.24	-8.08	1.56	0.15	14	0.002	32.07	18.94	27.66	78.68
SVR Config. 5	28	10.75	-6.23	1.83	0.46	24	0.006	31.96	18.33	27.46	77.75

To demonstrate the controlling performance, Fig. 7 shows the simulation results for the SVR with a different trajectory than the one used on the training process. All five SVR configurations are presented along with, the reference NMPC controller in Matlab and the reference trajectory. From the third plot, which shows the control actions, it can be noticed that the signal for the SVR solutions with 8 and 28 inputs present much more noise than the rest.

Finally, Table 4 shows the synthesis results of the SVR solutions for the Altera Cyclone V (5CSXFC6D6F31C6N) FPGA. Solutions 2 and 3 achieve the best MSE values as seen in Table 3 but at a cost of higher number of SVs which in turn result in the use of more hardware resources and cannot fit into the FPGA used. The solution with 20 inputs is

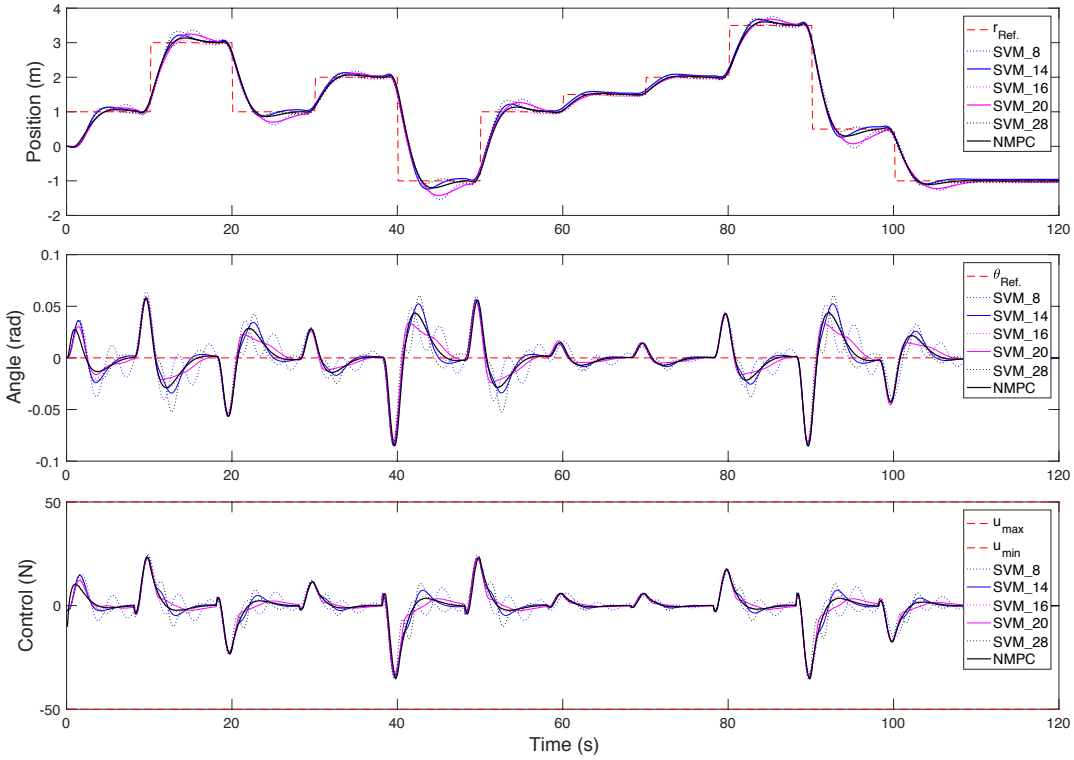


Figure 7: Simluation with the SVR solution following a reference trajectory.

chosen as the best result since not only it follows the reference with a control signal closer to the NMPC controller but also uses the least amount of hardware resources. There is also a performance comparison between the FPGA solutions and a SVM software simulator running on an ARM Cortex-A53 at 1.2 GHz. In the general, the FPGA solutions are about 5 to 10 times faster than the ARM embedded processor.

Table 4: FPGA Synthesis Results and Performance.

SVR Config.	Inputs	SVs	LEs (41.910)	DSPs (112)	Freq. (MHz)	Cycles	FPGA time (us)	ARM time (us)
1	8	18	24.882	37	118.34	125	1.05	5.84
2	14	76	83.391	77	-	335	-	57.25
3	16	37	52.117	38	-	222	-	28.48
4	20	14	21.352	15	111.16	236	2.12	13.25
5	28	24	37.265	25	108.42	263	2.42	25.31

7. Conclusions

In the present work, an approximate hardware solution for a NMPC controller was shown using a SVR architecture. The SVR training was accomplished by a novel tool called NIOTS which uses MOOP technique to solve a regression problem and generate a set of possible solutions. Five sets of training data using different input configurations containing present and past states from the nonlinear system and also current and future references where explored. The NIOTS tool was able to generate adequate solutions for all training sets. All solutions were synthesized on FPGA, but only three of them could fit. The best one in terms of control performance and hardware resource usage was the configuration with 20 inputs and 14 SVs. In terms of embedded hardware performance, this solution is 6.2 times faster than an equivalent algorithm running in software on an ARM processor.

In future works, we expect to apply this same technique to a system with stronger nonlinearities and with state constrains.

8. ACKNOWLEDGEMENTS

This work was supported by CNPq (National Counsel of Technological and Scientific Development), IFTO (Federal Institute of Education, Science and Technology of Tocantins) and FAPDF (Fundação de Apoio à Pesquisa do DF).

9. REFERENCES

Åkesson, B.M., Toivonen, H.T., Waller, J.B. and Nyström, R.H., 2005. "Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process". *Computers and Chemical Engineering*, Vol. 29, No. 2, pp. 323–335.

Alaniz, A., 2004. *Model predictive control with application to real-time hardware and guided parafoil*. Master's thesis, Massachusetts Institute of Technology. Dept. of Aeronautics and Astronautics.

Ayala, H., Sampaio, R., Muñoz, D.M., Llanos, C., Coelho, L. and Jacobi, R., 2016. "Nonlinear model predictive control hardware implementation with custom-precision floating point operations". In *24th Mediterranean Conference on Control and Automation (MED)*. Athens, Greece, pp. 135–140.

Burges, C.J., 1998. "A tutorial on support vector machines for pattern recognition". *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 121–167.

Carlos A. Coello Coello, G.B.L. and Veldhuizen, D.A.V., 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer Science, USA.

Chang, C.C. and Lin, C.J., 2011. "Libsvm: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology*, Vol. 2, pp. 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

Cortes, C. and Vapnik, V., 1995. "Support-vector networks". *Machine Learning*, Vol. 20, No. 3, pp. 273–297.

Drucker, H., Burges, C.J., Kaufman, L., Smola, A., Vapnik, V. *et al.*, 1997. "Support vector regression machines". *Advances in neural information processing systems*, Vol. 9, pp. 155–161.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A. and Diehl, M., 2016. "From linear to nonlinear MPC: bridging the gap via the real-time iteration". *International Journal of Control*, Vol. 7179, No. October, pp. 1–19.

Grüne, L. and Pannek, J., 2011. *Nonlinear Model Predictive Control*. Springer.

IEEE, 1985. "IEEE standard for binary floating-point arithmetic". Technical Report ANSI/IEEE Std. 754 754, The Institute of Electrical and Electronic Engineering, NY.

Kittisupakorn, P., Thitiyasook, P., Hussain, M.A. and Daosud, W., 2009. "Neural network based model predictive control for a steel pickling process". *Journal of Process Control*, Vol. 19, No. 4, pp. 579–590. ISSN 09591524.

Lawryńczuk, M., 2009. "Neural Networks in Model Predictive Control". In *Intelligent Systems for Knowledge Management*, Springer, Vol. 252, pp. 31–63.

Lima, C.A.M., 2004. *Comitê de máquinas: uma abordagem unificada empregando máquinas de vetores suporte*. Tese.

Mayne, D.Q., 2014. "Model predictive control: Recent developments and future promise". *Automatica*, Vol. 50, No. 12, pp. 2967–2986.

Mercieca, J. and Fabri, S.G., 2011. "Particle swarm optimisation for non-linear model predictive control". *International Conference on Advanced Engineering Computing and Applications in Science*, Vol. 5, No. 1, pp. 88–93.

Miranda, P.B., Prudencio, R.B., De Carvalho, A.C.P. and Soares, C., 2012. "Multi-objective optimization and meta-learning for svm parameter selection". In *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Brisbane, QLD, Australia, pp. 1–8.

Muñoz, D., Sanchez, D., Llanos, C. and Ayala-Rincon, M., 2009. "Tradeoff of FPGA design of floating-point transcendental functions". In *17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*. Florianopolis, SC, Brazil, pp. 239–242.

Muñoz, D., Sanchez, D., Llanos, C. and Ayala-Rincon, M., 2010. "FPGA based floating-point library for cordic algorithms". In *Programmable Logic Conference (SPL), VI Southern*. Ipojuca, Brazil, pp. 55–60.

Ortega, J.G. and Camacho, E.F., 1996. "Mobile robot navigation in a partially structured static environment, using neural predictive control". *Control Engineering Practice*, Vol. 4, No. 12, pp. 1669–1679.

Santos, C.E., Sampaio, R.C., Ayala, H., dos S. Coelho, L., Jacobi, R. and Llanos, C.H., 2017. "A SVM optimization tool and FPGA system architecture applied to NMPC". In *30th Symposium on Integrated Circuits and Systems Design*. Fortaleza, CE, Brazil. In print.

Spuler, M., Sarasola-Sanz, A., Birbaumer, N., Rosenstiel, W. and Ramos-Murguialday, A., 2015. "Comparing metrics to evaluate performance of regression methods for decoding of neural signals". In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. Milan, Italy, pp. 1083–1086.

Theodoridis, S. and Koutroumbas, K., 2009. *Pattern Recognition (Fourth Edition)*. Academic Press, Boston, fourth edition edition.

Yi, W., Gerasimov, I., Kuzmin, S. and He, H., 2016. "An intelligent algorithm of support vector regression parameters optimization in soft measurements". In *Soft Computing and Measurements (SCM), 2016 XIX IEEE International Conference on*. IEEE, St. Petersburg, Russia, pp. 404–406.

10. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.