A Hybridized Neuro-Genetic Solution for Controlling Industrial R^3 Workspace

E. Irigoyen, M. Larrea, J. Valera, V. Gómez, F. Artaza
Department of Systems Engineering and Automatic Control
Computational Intelligence Group
University of the Basque Country (UPV/EHU),
ETSI, 48013 Bilbao,
{eloy.irigoyen ; m.larrea}@ehu.es

Abstract

This work presents a hybridized neuro-genetic control solution for R^3 workspace application. The solution is based on a Multi-Objective Genetic Algorithm Reference Generator and an Adaptive Predictive Neural Network Strategy. The trajectory calculation between two points in R3 workspace is a complex optimization problem considering the fact that there are multiple objectives, restrictions and constraint functions which can play an important role in the problem and be in competition. We solve this kind of problem using Genetic Algorithms, in a Multi Objective Optimization Strategy. Subsequent, we enhace a training algorithm in order to achieve the best adaptation of the neural network parameters in the controller which is responsible for generating the control action for a nonlinear system. As an application of the proposed hybridized control scheme, a crane tracking control is presented.

Key words: *Hybrid Neuro-Genetic Solution, Optimal Trajectory Generation, Multi-Objective Genetic Algorithm, Nonlinear Neural Control, Adaptive Predictive Control*

Received:20th September 2010
Revised and accepted:13th November 2010

1 Introduction

Nowadays, our aggressive market requires more accurate, reliable, productive, and competitive industrial solutions. This involves a monumental effort from the researchers and technicians in order to solve complex, real-world problems. Among these problems is the industrial kinematic control (where it is necessary to handle raw materials, semi-finished and finished products), which implies a wide number of goals to reach [1]. In sequential industrial processes, for the transportation, handling and machining of materials and products into different manufacturing workplaces, it is more essential than ever to obtain automated and enhanced solutions based on new technologies such as Computational Intelligence.

This work presents a hybrid intelligent solution that solves tracking and movement problems in a R^3 workspace. It uses a complex calculation of a precise trajectory. It also solves accuracy and control action issues for precise and safe tracking operations. Our solution uses different Computational Intelligence Techniques for solving these problems. We initially implemented one device for tracing optimal trajectories as the reference to the control system. Later on, we chose a control scheme based on Adaptive and Predictive Control fields. Previous control loop approaches have been studied as the presented in [2] where a 2D crane anti-swing problem is solved.

The first part of our work was involved into designing of a Multi Objective Genetic Algorithm (MOGA). This MOGA solves a nonlinear and complex problem for calculating R^3 trajectories [3]. This solution takes into account requirements based on the workspace (restricted areas, points of passage, etc.), and constraints on the basis of parameter values (max-min) to preserve the life of actuators and different components. MOGA technique has been used with success in different works such as [4] and [5].

Furthermore, the tracking operation is made by an Adaptive-Predictive Neural Network (APNN) control system, which includes some intelligent strategies to reach the appropriate target. There exist different APNN Control approaches where the performances of different control loop are tested in [6] and [7].

Our system contains two Recurrent Neural Networks (NNARX): The first one provides a nonlinear process model to estimate the process output and derivatives in time, and the second one is involved in the current action calculation at every sample time.

Next in chapter 2 the different elements of the hybridized neuro-genetic system will be presented. In chapter 3 the components of the Multi Objective Genetic Algorithm Reference Generator will be laid out in detail. Then, the Neural Network Adaptive Predictive Control Strategy that was selected as well as the specific NN training algorithms designed will be explained. A case of study with a crane system will be introduced in chapter 5 Finally, the conclusions obtained and some ideas for future work will be commented.

2 Hybridized Neuro-Genetic Strategy

This work deals with the hybridization of different Computational Intelligence Techniques for solving non-trivial real tracking problems. The Genetic Algorithms have performed well in optimal solution calculation within multi-objective problems. In this approach we have designed a MOGA Reference Generator (MOGA-RG) in order to obtain a trajectory within a R^3 workspace. The MOGA-RG takes into account several objectives and different constraints of movement and workspace, which creates a more complex problem, the control of nonlinear systems.

To develop an appropriate control solution, the Neural Network Paradigm has been implemented. Different Neural Network Topologies were designed to perform the identification of the nonlinear system and to generate a nonlinear controller. An Adaptive Predictive Control Strategy was selected for this work as a result of certain needs referring to the control strategy and the use of the NNs as Controllers and Identifiers. This strategy was employed in several different works like [8][6].

The scheme used (Fig.1) has the following four basic blocks: a MOGA-RG, a Neural Network Identifier, a Neural Network Based Controller and the Nonlinear System to be controlled. All these elements have their respective training algorithms. The Identifier can provide an online identified model, which means the scheme has the capability to learn the system dynamics simultaneously to the nonlinear system evolution.

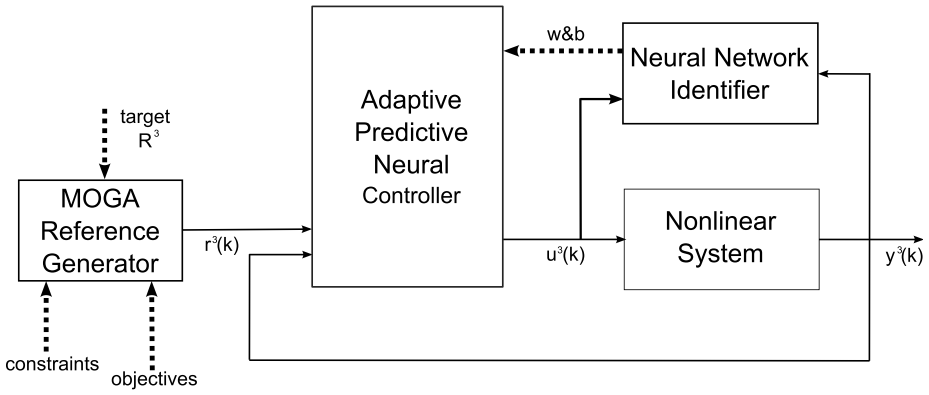

Figure 1: Control Scheme

The block Adaptive Predictive Neural Network Controller (APNNC) is responsible for generating the control action for the nonlinear system calculated in a predefined prediction horizon (Fig.2) where as the block MOGA-RG calculates a path to be tracked by the nonlinear system.

The APNNC performs a simulation of the entire loop and employs a replica of the nonlinear system provided by the NN Identifier in order to do so. This replica provides not only the nonlinear system output estimation ($\hat{y}$) but also the estimation of the identified system derivatives ($\frac{\partial \hat{y}_{k+1}}{\partial u_k}, \frac{\partial \hat{y}_{k+1}}{\partial y_k}, ..$). Those estimations are integrated in the training algorithm to be presented in section (4.1). Once the training algorithm finalizes its work, the NN Controller weight and bias are tuned to generate a control action that will be the output of the block. In figure 3, the different stages in the training process are presented.

One of the advantages that the Adaptive Predictive Control has is the capability to change the controller behavior. This is positive when the nonlinear system to be controlled suffers a modification (e.g. deterioration, wear, use of slightly different parts, etc.) and the nonlinear system model changes to a new

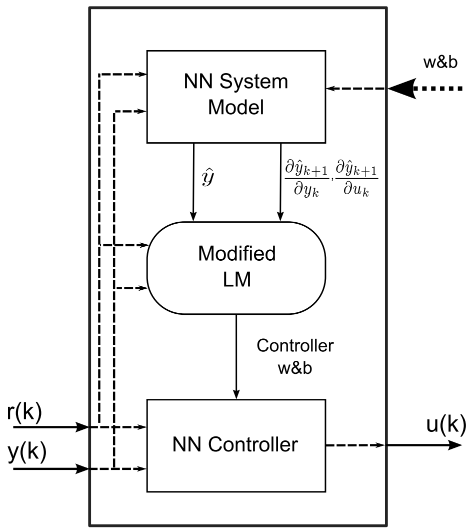
NN System Model
w&b
$\hat{y}$
$\frac{\partial \hat{y}_{k+1}}{\partial y_k}, \frac{\partial \hat{y}_{k+1}}{\partial u_k}$
Modified LM
Controller w&b
r(k)
y(k)
NN Controller
u(k)

Figure 2: Control Scheme

operating regime, causing the Controller change too.

3 MOGA Reference Generator

The trajectory calculation between two points in R^3 workspace is a complex optimization problem considering the fact that there are multiple objectives, restrictions and constraint functions which can play an important role in the problem. The following are some important aspects that have to be considered for an appropriate trajectory reference calculation: minimization of the time employed to travel from the initial to the final point, minimization of the travelled distance between these two points avoiding obstacles and restricted areas, minimum oscillation according to previous acceleration reference calculations, and minimization of mechanical elements wear in movement transition. Consequently, the problem formulation is not trivial especially taking into account that some objectives are not differentiable, so gradient or higher derivatives information are not available when searching for an optimal solution. This kind of problem can be solved using the Genetic Algorithm (GA) [9], in a Multi Objective Optimization Strategy, as previously introduced in Valera et al [3].

Thereby, a possible trajectory reference $r(t)$ between two points in R^3 workspace is given by equation 1.

$$r(t) = [x(t), y(t), z(t)] \tag{1}$$

In industrial processes the R^3 workspace usually has some restricted workspaces, as shown in equation 2.

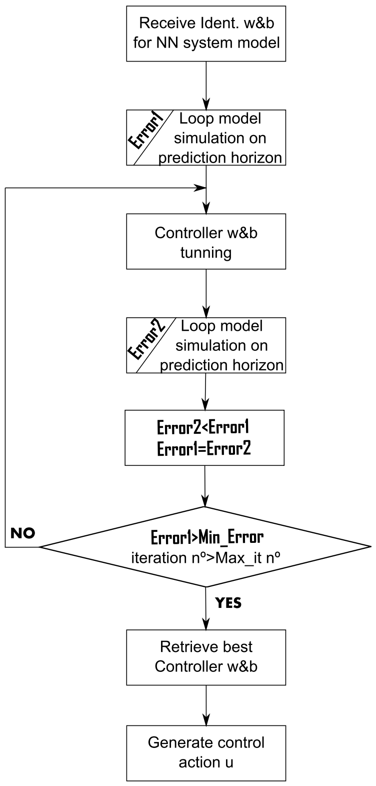
Receive Ident. w&b for NN system model
Error1
Loop model simulation on prediction horizon
Controller w&b tunning
Error2
Loop model simulation on prediction horizon
Error2<Error1
Error1=Error2
NO
Error1>Min_Error
iteration nº>Max_it nº
YES
Retrieve best Controller w&b
Generate control action u

Figure 3: Flowchart of adapting parameters and predicting errors

$$\begin{aligned} 0 \leq x(t) \leq X_{lim} \\ 0 \leq y(t) \leq Y_{lim} \\ 0 \leq z(t) \leq Z_{lim} \end{aligned} \tag{2}$$

Furthermore, this optimization problem has two main objectives to reach: minimize the $r(t)$ length or distance travelled, and minimize the required path time to travel from one point to the other. In addition, the trajectory has to satisfy the following constraints and restriction functions:

- Electromechanical component related constraints [10]: Speed $v(k)$ and Acceleration $a(k)$ on each axis or movement must not exceed the thresholds determined by the device manufacturers.
- Mechanical transmission elements and the useful life of the system: The Acceleration or Torque Gradient $j(k)$ of each movement must not exceed a

certain value to avoid so-called “impact torques” in the mechanical transmission elements which cause jerky movements and vibrations, reducing their useful life.

- Constraints related to avoid obstacles in the workspace: Any point of this trajectory cannot be included in the space defined by the constrained limited surface: $z = f_1(x;y)$, $y = f_2(x)$, and $z_p = f_3(x;y)$.

In our work, a Multi Objective Reference Generator based on Genetic Algorithms (MOGA) has been developed in order to satisfy all the objectives and constraints presented above. Figure 4 schematically represents the different components that perform the R^3 optimal trajectory within the MOGA Reference Generator.

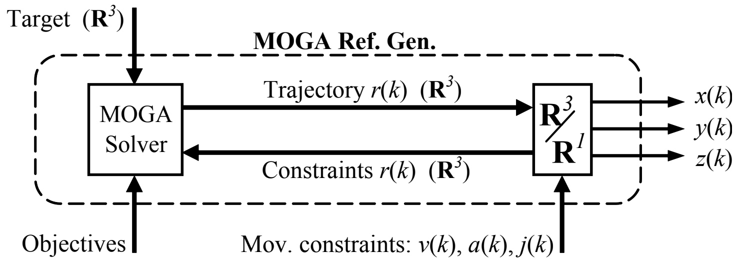
Target ($\mathbf{R}^3$)
MOGA Ref. Gen.
MOGA Solver
Trajectory $r(k)$ ($\mathbf{R}^3$)
Constraints $r(k)$ ($\mathbf{R}^3$)
$\mathbf{R}^3/\mathbf{R}^1$
$x(k)$
$y(k)$
$z(k)$
Objectives
Mov. constraints: $v(k)$, $a(k)$, $j(k)$

Figure 4: MOGA Reference Generator

The MOGA core is the solver that generates the values of the optimal trajectory $r(k)$, in each sample time. For this calculation, the solver takes into account the constraints related to the working restricted areas and the movement constraints $[v(k), a(k), j(k)]$, and try to minimize the travelled time and the trajectory length as objectives.

In order to have a smooth trajectory, a bounded acceleration reference and a bounded acceleration gradient [11], we divided the positioning time into six intervals taking into account the speed reference shown in Valera et al. [3].

To find three smooth position references ($x(k)$, $y(k)$, and $z(k)$), we used a nonlinear search method based on the Multi Objective Genetic Algorithm (MOGA) presented before, resulting in a R^3 combined trajectory (R^3 workspace) that simultaneously minimizes the distance travelled, the time used, and the final position error. The formulated objectives for MOGA execution can also be found in Valera et al. [3].

The trajectory generation is a non-trivial problem, due to some objectives are in competition. It has therefore been necessary to select the optimal solution by using the Pareto set optimal solutions technique. As seen in figure 5, the MOGA calculates a set of non inferior solutions that we represented in the Pareto front. By analyzing these solutions, we are able to find a solution that optimizes the R^3 movement depending on the actual working point and the objectives priorities previously defined. In figure 5, the time required for the trajectory (objective 1) is represented on the x axis, the error of travelling near the point $[xp, yp, zp]$ (objective 2) is represented on z axis, and the total distance (objective 3) is represented on y axis. In future works this selection will be solved by Computational Intelligence Techniques, as a fuzzy system recording the actions of an experienced operator.

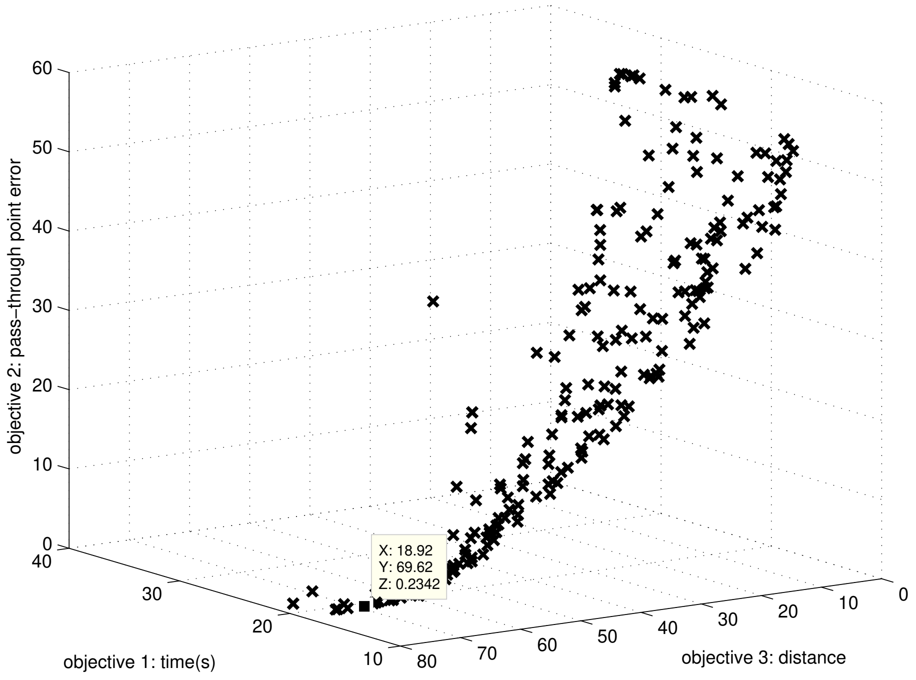

Figure 5: Set of non inferior solutions. Pareto frontier

4 Neural Network Adaptive Predictive Control

In this section the one dimensional Adaptive Predictive Control will be introduced. Using this strategy, the two NNs employed are MultiLayer Perceptrons (MLP). The MLP are known as Universal Approximators because of their capacity to approximate any function of interest (both linear and nonlinear) as well as its derivatives [12]. The latter one is of great importance in the implementation of the Identifier, since the derivatives that it provides will be integrated into the training algorithm. The topology of the NN Controller and the NN Identifier are correspondingly presented in figure 6 and figure 7.

The NN Controller (Fig.6) is a NN AutoRegressive with eXogenous input (NNARX) that gives output feedback (control action).

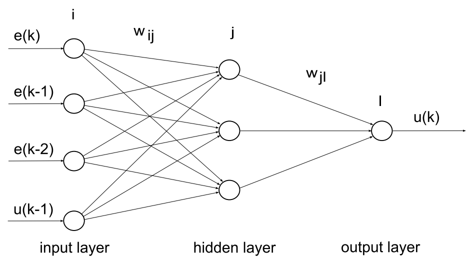

Figure 6: NN Controller.

The NN Identifier (Fig.7) obtains the nonlinear system model based on the system input/output relation. Once the model is obtained, it can be used to emulate the nonlinear system behaviour and extract its derivatives. Both the NN Controller and the NN Identifier can be trained online or offline.

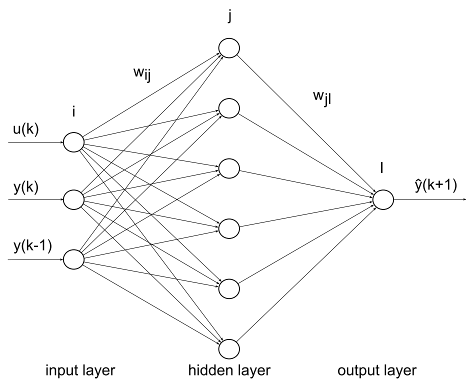

Figure 7: NN Identifier.

4.1 Neural Network Training

The NN Controller is trained in the "Adaptive Predictive Neural Controller" block that can be seen in figure 1. As previously mentioned, inside this block a simulation of the control loop is performed. This simulation creates the possibility to simulate the control loop evolution for a prediction horizon, and to simulate it for different control actions. The NN Controller needs to know, or estimate, the error produced on its output in order to be trained. As the desired control action (u) is unknown, the error produced during the output of the NN Controller is also unknown. The only known error is the one produced on the output of the nonlinear system ($y(k) - r(k)$ in Fig.1), which can be related to the NN Controller Weight and Bias through the NN System Model. This way, the equation 3 [13] can be used to train the NN Controller in a K prediction horizon.

$$\sum_{k=1}^{K} \frac{\partial E_k}{\partial w_{lij}} = \sum_{k=1}^{K} \sum_{k'=1}^{k} \sum_{k''=0}^{k'-1} \frac{\partial E_k}{\partial y_{k'}} \cdot \frac{\partial y_{k'}}{\partial u_{k''}} \cdot \frac{\partial u_{k''}}{\partial w_{lij}} \tag{3}$$

Equation 3 is made up of three terms. The first one relates the error committed in the control loop output with the nonlinear system output. The second one relates the nonlinear system output to the control action. Finally, the third term relates the control action to the NN Controller weights and biases. The first and the third terms are known terms. The first is the one that depends on the error function used and the third is the one that can be calculated by backpropagation. The second term represents the model of the nonlinear system to be controlled. A general representation of a nonlinear system can be expressed by using the following equation 4.

$$y(k') = M[y(k'-1), ..., y(k'-n), u(k'-1), ..., u(k'-m)] \tag{4}$$

where n is the nonlinear system order that must satisfy $m \leq n$. Deriving $y(k')$ from $u(k'')$ the unknown term $(\frac{\partial y_{k'}}{\partial u_{k''}})$ can be obtained. This term can in turn be broken down in the following equation 5 [13].

$$\frac{\partial^+ y_{k'}}{\partial u_{k''}} = \sum_{i=1}^{n} \frac{\partial y_{k'}}{\partial y_{k'-i}} \cdot \frac{\partial^+ y_{k'-i}}{\partial u_{k''}} + \sum_{j=1}^{m} \frac{\partial y_{k'}}{\partial u_{k'-j}} \cdot \frac{\partial^+ u_{k'-j}}{\partial u_{k''}} \tag{5}$$

Previous work [14][13] has shown that the reduction of the computational times can be achieved by neglecting some of these terms ($(\frac{\partial u_{k'-j}}{\partial u_{k''}} = 0$ when $k' - j \neq k'')$. By neglecting these terms, the second term of the equation 5 results in the following equation 6.

$$\frac{\partial^+ y_{k'}}{\partial u_{k''}} = \sum_{i=1}^{n} \frac{\partial y_{k'}}{\partial y_{k'-i}} \cdot \frac{\partial^+ y_{k'-i}}{\partial u_{k''}} + \frac{\partial y_{k'}}{\partial u_{k''}} \tag{6}$$

Now the three terms of equation 5 can be found. These three terms are known on the basis of "NN System Model" input/output relations. The "Universal Approximator" property has been applied in [15] to obtain the derivatives of the identified system using the equations 7,8,9 to do so, being the NN represented in figure 7.

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{j=1}^{n} w_{1j} o_j (1 - o_j) w_{j1} \tag{7}$$

$$\frac{\partial \hat{y}(k+1)}{\partial y(k)} = \sum_{j=1}^{n} w_{2j} o_j (1 - o_j) w_{j1} \tag{8}$$

$$\frac{\partial \hat{y}(k+1)}{\partial y(k-1)} = \sum_{j=1}^{n} w_{3j} o_j (1-o_j) w_{j1} \tag{9}$$

where w_{1j} represents the weight that links input 1 with the neuron j of the hidden layer, w_{j1} represents the weight that links the output of the neuron j of the hidden layer to the neuron of the output layer, o_j represents the output of the neuron j of the hidden layer and the n of the summation represents the number of neurons in the hidden layer.

4.2 NN Controller Training algorithm modification

The LM algorithm calculates the updated term for the weights and biases on the basis of the equation ΔW in [16]. The modification proposed, which includes the dynamics of the nonlinear system, affects the term on the output layer to be backpropagated (Δ^M presented in [16]).

$$\Delta^M = -\dot{F}^M(\underline{n}^M) \cdot \frac{\partial y_{k'}}{\partial u_{k''}} \tag{10}$$

Applying this formula and following the development presented in [16] the dynamics of the nonlinear system and the ones of the NN Controller are backpropagated. Therefore all the Jacobian terms are calculated so the weight adaptation term (ΔW) can be obtained. Finally we emphasize the different meaning of the term $e'(\underline{w})$ in equation 11 for this work. If the original work represented $e(\underline{w})$ as the error committed in the NN output, this work uses $e'(\underline{w})$ as the error committed in the output of the control loop.

$$\Delta W = \left[J^T(\underline{w}) \cdot J(\underline{w}) + \mu \cdot I \right]^{-1} \cdot J(\underline{w}) \cdot e'(\underline{w}) \tag{11}$$

This equation is used in the same manner as the traditional LM algorithm in [16]. $J(\underline{w})$ is the Jacobian Matrix which is composed of the partial derivatives of the errors in the NN output ($e(\underline{w})$) on the weights ($\underline{w}$)(12).

$$J(\underline{w}) = \begin{pmatrix} \frac{\partial e_1(\underline{w})}{\partial w_1} & \frac{\partial e_1(\underline{w})}{\partial w_2} & \cdots & \frac{\partial e_1(\underline{w})}{\partial w_N} \\ \frac{\partial e_2(\underline{w})}{\partial w_1} & \frac{\partial e_2(\underline{w})}{\partial w_2} & \cdots & \frac{\partial e_2(\underline{w})}{\partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_K(\underline{w})}{\partial w_1} & \frac{\partial e_K(\underline{w})}{\partial w_2} & \cdots & \frac{\partial e_K(\underline{w})}{\partial w_N} \end{pmatrix} \tag{12}$$

To calculate the Jacobian Matrix [16], the term ($\frac{\partial^+ y_{k'}}{\partial u_{k''}}$) of equation (6) is backpropagated through the layers of the NN controller.

5 Application to Crane Position Control

The Adaptive Predictive Control Strategy is applied in the control of a travelling crane. The load trajectory calculation in the R^3 workspace is a complex optimization problem considering the multiple objectives, restrictions and constraint functions. The nonlinear problem of the swinging angle control is considered as a good exercise for the proposed NN control system. The crane model used consists of a Matlab/ Simulink block provided by the company Inteco with a real model of the crane (Fig.8). See [1] for the mathematical model.

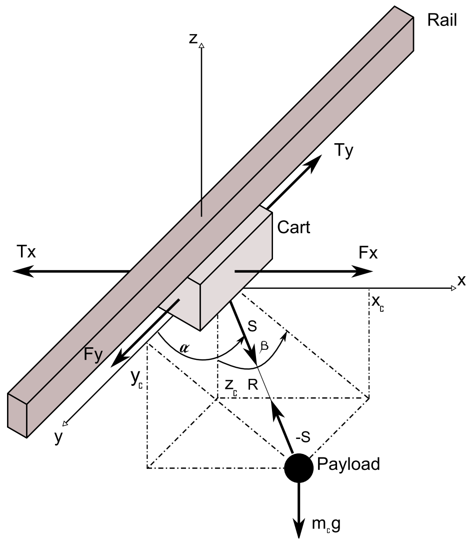

Figure 8: Crane model.

The following information pertains to the trajectory of the MOGA; Initial Pos. $(0, 0, 0)$, Final Pos.$(30, 80, 10)$ with Crossing Point $(50, 50, 50)$cm. The constraints that the MOGA must respect on these 3 axes are; Max. Acceleration $5cm/s^2$, Max. Speed $10cm/s$ and Max. Jerk $0.5cm/s^3$. The objectives applied to the MOGA are: to pass through the specified crossing point ($error < 0.5cm$), the minimization of the travel time required and minimization of the distance travelled. The resultant trajectory is shown in figure 9.

The x-axis control behavior has been observed in a preliminary test. The main objective of the test has been to control the crane position while minimizing the swing of the load. Offline identification of the crane was performed to extract the model to be used in the control loop. The identification was carried out applying random entries (both positive and negative steps inside the work range) to the NN identifier. The training has been performed with the following parameters: Training Vector Length = 4001, Validation Vector Length = 1000, number of epochs = 1000, Initial Weights randomly generated within an interval calculated as in the work [17]. The identification results for the training stage and validation stage are presented in figure 10.

Figure 11 shows the control of the x-axis position, being the dotted line the path generated by the MOGA and the solid line the tracking performed by the controller. The other lines represent the smooth control action and the low

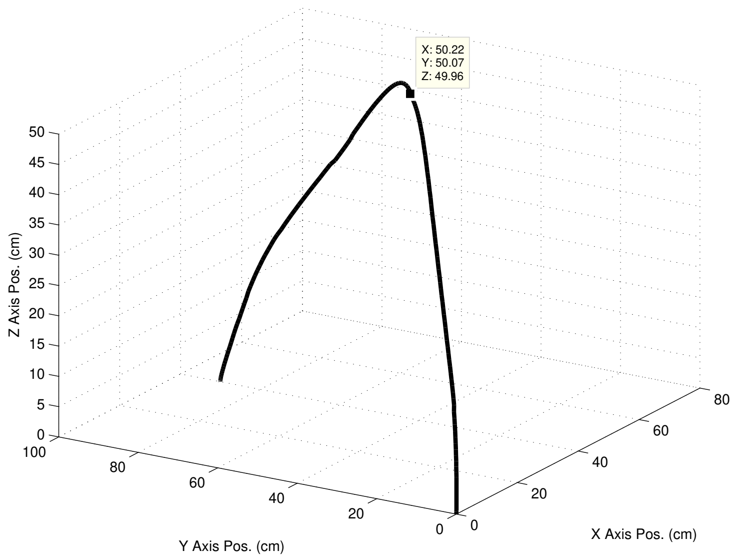

Figure 9: R^3 Trajectory.

swinging of the load.

6 Conclusions

This work tackles the problem of R^3 Multiobjective reference generation and the system control under these circumstances. With an intelligent search algorithm based on MOGA the solution is stable, robust and it is a fast way to find optimal solutions when real-time requirements are not needed and when the problem involves many objectives.

Moreover, the present paper shows the use of NNs in an Adaptive Predictive Control Strategy. The simulation results show a correct online adaptation of the NN controller and the validity of the modification made to the LM Training Algorithm. This modification allows the integration of the nonlinear system dynamics into the training algorithm, thus being able to train the NN Controller despite not knowing the nonlinear system. The NN Identifier estimates the dynamics of the nonlinear. The use of restrictions to control action has been tested on various works such as [18], where first order training algorithms are used. These restrictions may be of interest when implementing a Controller Training that penalizes abrupt changes in the control action. We will also take into account hierachical issues [19].

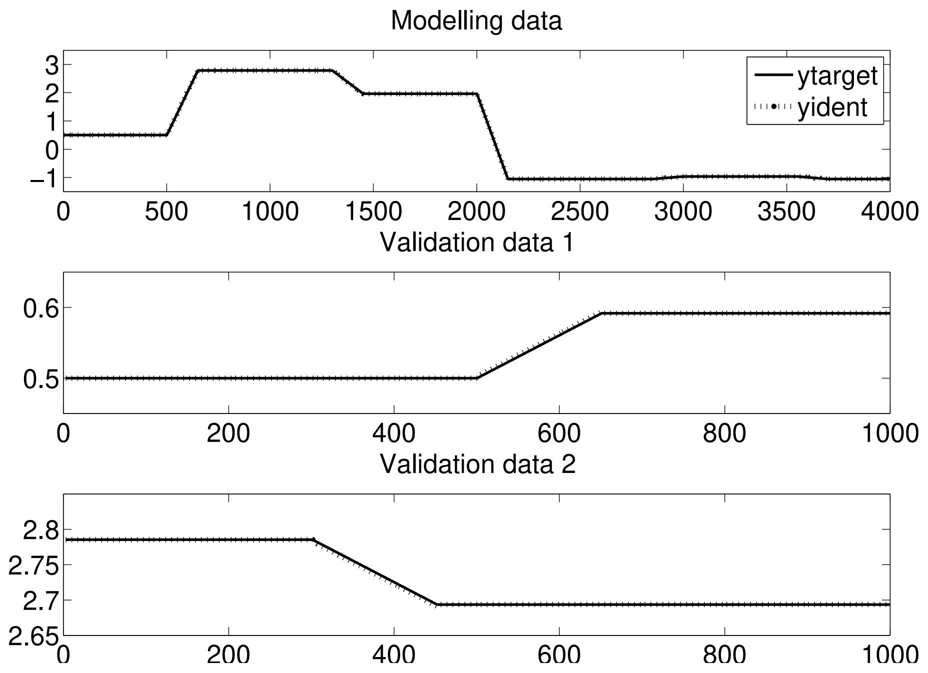

Figure 10: Crane identification

Acknowledgement

This work comes under the framework of the ATICTA research project with reference SAIOTEK08 granted by Basque Government, and the BAIP2020 research project granted by CDTI of the Spanish Government, with permission of INGETEAM for the paper's publication.

References

[1] M. Pauluk, A. Korytowski, A. Turnau, and M. Szymkat, "Time optimal control of 3d crane," *proceedings of the 7th Inter. Conference on Methods and Models in Automation and Robotics*, pp. pp.927–936, 2001.

[2] G. Liu and I. Mareels, "Advantages of smooth trajectory tracking as crane anti-swing schemes," in *IEEE International Conference on Robotics and Biomimetics*, (Piscataway, NJ, USA), pp. 1486 – 90, 2008.

[3] J. Valera, E. Irigoyen, V. Gómez-Garay, and F. Artaza, "Application of neuro-genetic techniques in solving industrial crane kinematic control problem," *IEEE International Conference On Mechatronics*, pp. 231–237, 2009.

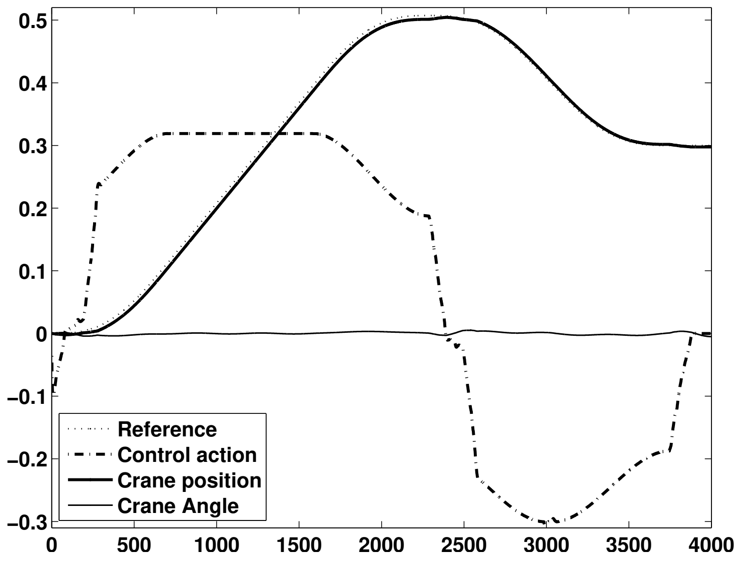

Figure 11: Simulation results

[4] B. Pathak, H. Singh, and S. Srivastava, "Multi-resource-constrained discrete time-cost tradeoff with moga based hybrid method," in *IEEE Congress on Evolutionary Computation*, (Piscataway, NJ, USA), pp. 4425 – 32, 2007.

[5] X. Xing, D. Yuan, and J. Yan, "A novel moga-based method of flight control law design for a helicopter and its application," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 7128, (USA), pp. 71282K (6 pp.) –, 2008.

[6] C.-H. Lu and C.-C. Tsai, "Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine," *IEEE Transactions on Industrial Electronics*, vol. 55, pp. 1366–1375, March 2008.

[7] S. S. Ge, C. Yang, and T. H. Lee, "Adaptive predictive control using neural network for a class of pure-feedback systems in discrete time," *IEEE Transactions on Neural Networks*, vol. 19, pp. 1599–1614, Sept. 2008.

[8] K. K. Tan, T. H. Lee, S. N. Huang, and F. M. Leu, "Adaptive-predictive control of a class of siso nonlinear systems," *Dynamics and Control*, vol. 11, pp. 151–174, apr 2001.

[9] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing.* SpringerVerlag, 2003.

[10] J.-H. Suh, J.-W. Lee, Y.-J. Lee, and K.-S. Lee, “An automatic travel control of a container crane using neural network predictive pid control technique,” *International Journal of Precision Engineering and Manufacturing*, vol. 7, no. 1, pp. 35 – 41, 2006.

[11] V. B. Anand, *Computer Graphics and Geometric Modeling for Engineers.* New York, NY, USA: John Wiley & Sons, Inc., 1993.

[12] K. Hornik, M. Stinchcombe, and H. White, “Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks,” *Neural Networks*, vol. 3, pp. 551–560, 1990.

[13] E. Irigoyen, J. Galván, and M. Pérez-Ilzarbe, “Neural networks for constrained optimal control of nonlinear systems,” *IJCNN*, vol. vol.4, pp. 299 – 304, 2000.

[14] J. Gálvan, “Tuning of optimal neural controllers,” *Proc. Int. Conf. on Engineering of Intelligent Systems*, pp. 213–219, 1998.

[15] T. Fujinaka, Y. Kishida, M. Yoshioka, and S. Omatu, “Stabilization of double inverted pendulum with self-tuning neuro-pid,” *IJCNN*, vol. 4, pp. 345–348, 2000.

[16] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the marquardt algorithm,” *IEEE Transactions on Neural Networks*, vol. 5, pp. 989–993, nov 1994.

[17] E. Irigoyen and M. Pinzolas, “Numerical bounds to assure initial local stability of narx multilayer perceptrons and radial basis functions,” *Neurocomputing*, vol. 72, no. 1-3, pp. 539 – 547, 2008.

[18] E. Irigoyen, J. Galván, and M. J. Pérez-Ilzarbe, “A neuro predictive controller for constrained nonlinear systems,” *IASTED International Conference Artificial Intelligence and Applications*, 2003.

[19] M. Grana and F. Torrealdea, “Hierarchically structured systems,” *European Journal of Operational Research*, vol. 25, no. 1, pp. 20 – 26, 1986.