01 Oct 2008

Real-Time Implementation of Intelligent Modeling and Control Techniques on a PLC Platform

Curtis Alan Parrott

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Recommended Citation

C. A. Parrott and G. K. Venayagamoorthy, "Real-Time Implementation of Intelligent Modeling and Control Techniques on a PLC Platform," *Proceedings of the IEEE Industry Applications Society Annual Meeting, 2008. IAS '08*, Institute of Electrical and Electronics Engineers (IEEE), Oct 2008.
The definitive version is available at https://doi.org/10.1109/08IAS.2008.164

Real-Time Implementation of Intelligent Modeling and Control Techniques on a PLC Platform

Curtis Parrott, *Student Member, IEEE* and Ganesh K. Venayagamoorthy, *Senior Member, IEEE*

Real-Time Power and Intelligent Systems Laboratory
Missouri University of Science and Technology, Rolla, MO 65401 USA
cap9qd@mst.edu and gkumar@ieee.org

***Abstract* – Programmable logic controllers (PLCs) have been used for many decades for standard control in industrial and factory environments. Over the years, PLCs have become computational efficient and powerful, and a robust platform with applications beyond the standard control and factory automation. Due to the new advanced PLC's features and computational power, they are ideal platforms for exploring advanced modeling and control methods, including computational intelligence based techniques such as neural networks, particle swarm optimization (PSO) and many others. Some of these techniques require fast floating-point calculations that are now possible in real-time on the PLC. This paper focuses on the Allen-Bradley ControlLogix brand of PLCs, due to their high performance and extensive use in industry. The design and implementation of a neurocontroller consisting of two neural networks, one for modeling and the other for control, and the training of these neural networks with particle swarm optimization is presented in this paper on a single PLC. The neurocontroller in this study is a power system stabilizer (PSS) that is used for power system oscillation damping. The PLC is interfaced to a power system simulated on the real time digital simulator. Real time results are presented showing that the PLC is a suitable hardware platform for implementing advanced modeling and control techniques for industrial applications.**

***Index Terms* – Computational intelligence, modeling, neurocontrol, particle swarm optimization, programmable logic controllers, power system stabilizer.**

I. Introduction

Programmable logic controllers (PLCs) have been used extensively in industrial applications for control for decades due to their high reliability and robust architecture [1]. The newest PLCs have moved past just a robust platform into a new realm of high computational power and processor speed. These, along with the PLC's highly expandable layout, makes it an ideal platform for far beyond the classical applications. These new applications include implementing computational intelligence based modeling, optimization and control techniques that require fast processing power to be executed in real-time. With the ability to contain analog I/O, the PLC is also ideal for interface to real-time simulation hardware, such as the real-time digital simulator (RTDS) for power systems [2].

The RTDS is a custom parallel processing hardware platform that allows power systems to be simulated and its accessories (controllers, transformers, relays) to be tested in real-time [3]. Through the use of analog I/O, power control devices can be seamlessly tested as if they were part of the physical power system running on the simulator. This allows for the testing of any such control device containing low voltage I/O and allows the gauging of this control scheme as a legitimate real-world application. The ability of the RTDS for control and protection system testing has been further explored in [2]. This makes the PLC-RTDS platform, an ideal platform for testing the viability of the PLC as a real-world control platform for computational intelligence techniques. In this paper, a case study of implementing a controller based on neural networks for damping speed oscillations in generators is explored [4]. The PLC platform implements the neural networks required to realize the adaptive control and these neural networks are trained using particle swarm optimization (PSO) algorithm [5]. To the knowledge of the authors, computational intelligence techniques have not been implemented on PLCs which are known to be robust platforms for industrial controls.

Power system stabilizers (PSSs) are used as an auxiliary control system to a generator's excitation system. The purpose of the PSS is for power system oscillation damping during small and large system disturbances by providing supplementary control signals to the generator's automatic voltage regulator (AVR) [6]. The speed oscillations can take the form of intra-area and inter-area modes in a multi-machine power system: intra area modes form where two or more synchronous machines swing together against a comparatively large power system or load center and inter-area modes involve combinations of many machines on one part of a power system swinging against machines on another part of the system [7].

Advanced power system modeling and control techniques have been explored on a wide variety of platforms, including digital signal processors (DSPs) and field programmable logic arrays (FPGAs), in great detail. However, research has neglected the staple of industrial control, the programmable logic controller. The PLC platform is used extensively in industry due to its very high reliability and expandability. This expandability includes a wide variety of digital and analog I/O modules along with many different communication modules. The PLC is also designed with a powerful processor with the ability to do real-time control of a wide variety of control application [1].

This paper demonstrates the potential of PLCs for implementing computational intelligence paradigms including neural networks and particle swarm optimization in real time for modeling and control of synchronous generators in a multimachine power system environment.

This work is supported by the NSF CAREER Grant ECCS # 0348221 and US Dept. of Education GAANN funding awarded to Dr. Venayagamoorthy.

978-1-4244-2279-1/08/$25.00 © 2008 IEEE

II. Multi-Machine Two-Area Power System

The multimachine power system studied to demonstrate the PLC implementation of a conventional PSS and a neural network based damping controller (intelligent PSS) is the standard two-area four machine power system in Fig. 1 [7]. This power system consists of two fully symmetrical areas linked together by two transmission lines. Each area is equipped with two synchronous generators rated at 20 kV/900 MVA. All the generators are equipped with identical speed governors and turbines, AVRs and exciters. Generators G1 and G3 are both also equipped with conventional PSSs (Fig. 2). The loads for each area are represented as constant impedances and are split between the two areas such that Area 1 transmits approximately 413 MW of power to Area 2. Three electromechanical modes of oscillation are present in this system: two inter-plant/intra-area modes, one in each area, and one inter-area low-frequency mode [8]. The parameters of this system are given in [8]. Fig. 3 shows the individual controllers on generators G1 and G3.

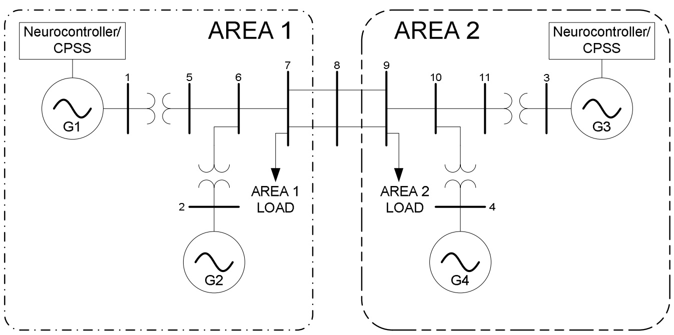

Fig. 1. Multi-Machine Two-area power system.
Figure 1.

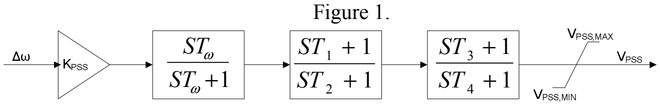

Fig. 2. Conventional power system stabilizer.

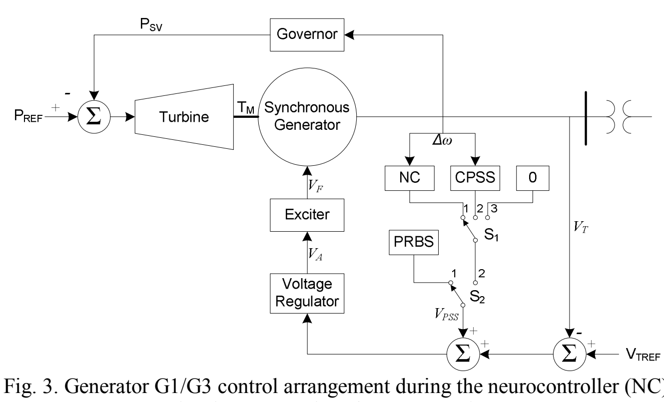

Fig. 3. Generator G1/G3 control arrangement during the neurocontroller (NC) development and NC/CPSS operation.

This power system is a test system commonly used to show the effectiveness of controllers in damping slow-mode oscillations [7, 8]. This system is implemented in RTDS such that the practical implementation of an intelligent PSS on the PLC platform can be demonstrated in real-time system. Although the system is interfaced to the RTDS simulator, the simulations are run in real-time and very closely approximate real-world implementations. This allows the PLC platform and designed intelligent PSS to be evaluated as a practical, real world, control system as compared to a pure non-real time simulation study.

The conventional power system stabilizer (CPSS) is widely used in power systems. The CPSS is designed using the theory of phase compensation in the frequency domain and are introduced as a lead-lag compensator. The parameters and time-constants of the CPSS are designed against a linearized model of the power system to be controlled. To have the CPSS respond well and damp both intra-area mode and inter-area mode oscillations, its parameters must be fine tuned for a given operating point. Do to the non-linearity of the power systems and being designed against a linear model, the designed parameters cannot be guaranteed to work well in a practical system as operating conditions change [6].

III. Design of a Neurocontroller

The design of the neurocontroller (NC), the intelligent PSS, is based on the form used in implementing indirect adaptive control. This layout consists of two separate neural networks: a neurocontroller (NC) and a neuroidentifier (model). The diagram for development of the NC is shown in Fig. 4. The training of these neural networks is carried out using the PSO algorithm, which is discussed later in this section. The dashed lines in Fig. 4 represent this update to the respected neural network.

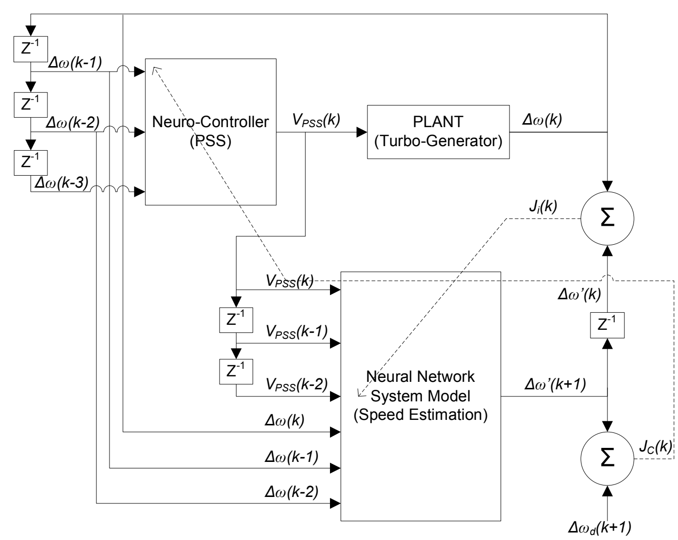

Fig. 4. Indirect adaptive control structure for implementing the NC.

A. Neural Network System Model

A neural network based model is used in the NC design to estimate the speed deviations of a generator in the next sample time step. This model is developed using the series-parallel nonlinear auto regressive moving average model [9]. The model output at the time step *k+1* depends on both past *n* values of its output as well as past m values of its input. The inputs and outputs of the model are speed deviation of the plant (generator G1 or G3) and the output of the neurocontroller, and the estimated speed deviations respectively. Here, both n and m are chosen to be 2. The main reason for choosing three time step values is because a third order system is sufficient for the modeling the generator dynamics for this study. The model is a multi-layered feedforward neural network (Fig. 5) trained using the PSO algorithm. The input vector to the model network is $\bar{X}$ and the estimated speed deviation at instant ($k+1$) is $\Delta\omega'(k+1)$.

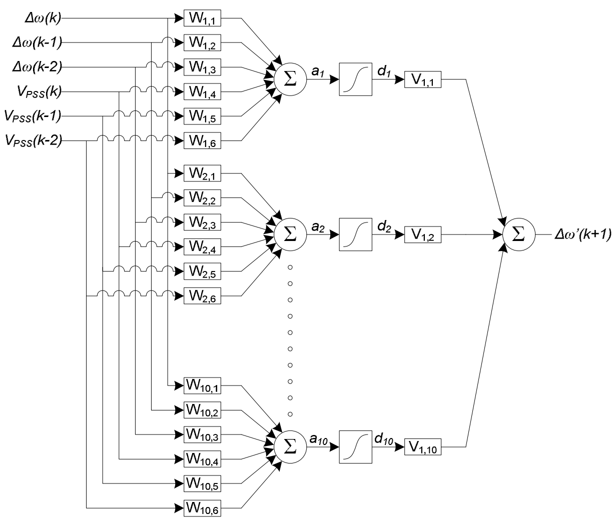

Fig. 5. Neural network system model structure.

$$\overline{X} = \begin{bmatrix} \Delta\omega(k), \Delta\omega(k-1), \Delta\omega(k-2), \\ V_{PSS}(k), V_{PSS}(k-1), V_{PSS}(k-2) \end{bmatrix} \tag{1}$$

$$a_i = \sum_{j=1}^{10} W_{i,j} \cdot X_j \tag{2}$$

$$d_i = \frac{1}{1+e^{-1 \cdot a_i}} \tag{3}$$

$$\Delta\omega'(k+1) = \sum_{i=1}^{10} V_i \cdot d_i \tag{4}$$

B. Neurocontroller

The neurocontroller is also a multi-layer feedforward network trained with PSO algorithm. The inputs to this system are the actual speed deviation and the two previous values of a generator and the output of the neurocontroller is the supplementary control signal to the AVR, V_{PSS} as shown in Fig. 3. The training of the neurocontroller is similar to that described in [4]; however PSO is used in lieu of backpropagation algorithm.

C. PSO Algorithm

PSO is a type of evolutionary computing technique. The algorithm is based on the simulation of the social interaction of birds within a flock and school of fish. Being a population based search algorithm, a swarm consists of particles which are potential solutions to the problem solved or optimized. The changes in the particles position in the search space is influenced by the past knowledge of the swarm as well as the particles own past knowledge of the search space.

At initialization, each particle is randomly assigned to a point in the search space, as well as given a random starting velocity. The particle is then flown through the search space with the initial velocity. The particle is then evaluated as to how well it solves the problem at hand; this evaluation is called the particle's fitness. This is then compared to the particle's memory of its best solution of the problem, the *pbest* position. If the newest solution is better than the current *pbest* (the current fitness lower than the *pbest* fitness), the *pbest* position is updated to the current position. Once all the particles have been evaluated the *pbest* with the lowest fitness is compared to the *gbest* position fitness. If this *pbest*'s value is lower than the current *gbest* fitness then the *gbest* position is update to this *pbest*'s location. This *gbest* represents the social aspect of the algorithm. After these updates have been done the PSO equations are again evaluated, and take the form seen in (5) and (6). Also, an example of a single particle update can be seen in Fig. 6 where: $x_{id}(k)$ is the i^{th} particle's d^{th} dimension current position; $x(k+1)$ is the i^{th} particle's d^{th} dimension position after the PSO update, at the next time step; $v_{id}(k)$ is the i^{th} particle's d^{th} dimension current velocity; $v_{id}(k+1)$ is the i^{th} particle's d^{th} dimension velocity at the next time step; p_{id} is the *pbest* position of i^{th} particle's; p_{gd} is the groups best position or *gbest* for the d^{th} dimension; w is the inertia weight constant; c_1 and c_2 are the cognitive and social acceleration constants respectively [5].

$$\begin{aligned} v_{id}(k+1) = w \cdot v_{id}(k) + c_1 \cdot rand_1 \cdot (p_{id}(k) - x_{id}(k)) \\ + c_2 \cdot rand_2 \cdot (p_{gd}(k) - x_{id}(k)) \end{aligned} \tag{5}$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \tag{6}$$

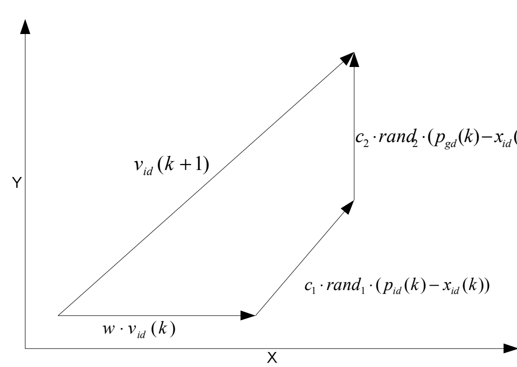

Fig. 6. PSO particle update process for two dimensional case [5].

IV. Programmable Logic Controller and Real-Time Digital Simulation Platforms

The PLC platform is a tried and true platform for control and automation. Due to its design, it has many advantages over general purpose computer, DSP, and FPGA array based control systems. The PLC is designed to be in an industrial environment and is built to withstand this environment which can contain electrical noise, electromechanical interference, mechanical vibrations, extreme temperatures above 140 degrees Fahrenheit and non-condensing humidity of 95% [10]. Other platforms mentioned above would require modification to withstand these kind of environmental conditions. PLCs are also highly modular and only require a simple module change to add extra features while a complete system redesign would be needed with a computer, DSP or FPGA based design. Since the PLC executes a single program in a sequential fashion it can recover from power failure quickly since there is no boot-up procedure, and thus have a larger edge against the computer systems [10].

The PLC platform used in this study is the Allen-Bradley ControlLogix 5561 processor along with component rack, power supply and analog IO cards. This line of PLC processors and hardware provide the needed processing power to execute the control algorithms in question.

As mentioned before, the PLC control system is interfaced to the RTDS. This simulator allows for the simulation of power systems in real-time while connecting auxiliary control components to the simulation via analog I/O. The RTDS and PLC hardware test setup for this study is shown in Fig. 7.

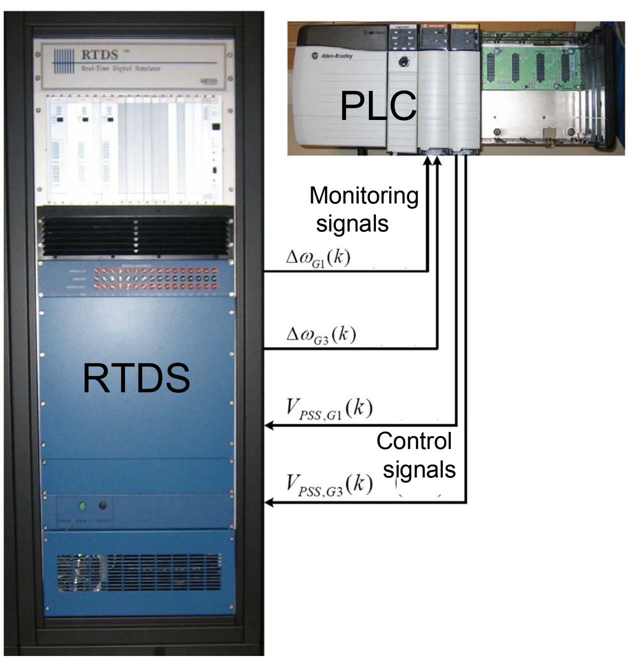

Fig. 7. RTDS and PLC platform laboratory setup showing monitoring and control signals.

V. Neurocontroller and PSO Implementation

Neurocontroller development and implementation is accomplished in two steps: first the neural network model is trained for generator speed prediction and then the neurocontroller is trained using the neural network model of the system. Both of these neural networks are trained using offline using the PSO algorithm. Offline training is only possible due to the computational complexity of the fitness function evaluations. All of the control components, including the model, neurocontroller and PSO are implemented in structured text PLC programming language using Allen-Bradley's RSLogix 5000 programming software.

In order to train the networks, the speed deviation of each generator is communicated to the PLC via the analog channels of the RTDS; these channels transmit a voltage from -10 volts to + 10 volts. This speed deviation is also up scaled in the RTDS hardware to take full advantage of the 16 bits of resolution available in the analog channels and then later downscaled back to the original value when received by the PLC. This is done in order to minimize the quantization error of the analog channels to maximize the resolution of the transmitted signal. The pseudorandom binary signal (PRBS) forced training signal is also transmitted to the PLC in a similar fashion. Both of these values are used to train the model and control neural networks.

The training of the model neural network is accomplished by implementing the PSO algorithm in the PLC. First the PLC would capture 25 seconds of speed deviation and PRBS signal at 40 Hz sampling frequency. Then the PSO particles are initialized randomly between [-0.1, 0.1]. Next each of the PSO particles' fitness is evaluated. This is done by applying the testing data points captured earlier to the neural network and calculating the mean-squared-error (MSE) between the identifier output and the speed deviation at the next time step. The neural network model's fitness equation takes the form given in (7) and (8). This MSE is the fitness for each particle and is used to update the *pbest* and *gbest* values. Once a satisfactory solution is attained the controller is trained.

$$J_i(k) = \Delta\hat{\omega}_{G1}(k) - \Delta\omega_{G1}(k) \tag{7}$$

$$fitness_i = \frac{1}{1000}\left[\sum_{k=0}^{999} J_i(k)^2\right] \tag{8}$$

At the point a suitable model is attained and the training of the neurocontroller is started. Another set of 1000 data points are again captured. For the controller to damp oscillations in the system the target speed deviation is set to zero. The error from zero, which takes the form of (9), is then back-propagated through the model network to obtain a ΔV_{pss} signal for all data points. In order to find the value of ΔV_{pss}, the decision error vector of the model networ is found and takes the form given in (10). Next the decision error vector is used to find the activation error vector and takes the form given in (11). Finally this activation error is used to find the change inf the inputs of the model network, which contains ΔV_{pss} as given in (12) and (13). This value is then used in the fitness evaluation of the neurocontroller. The neurocontroller fitness equation takes the form given in (14). This is used to evaluate each controller (particle) evolved by PSO and is used to update the *pbest* and *gbest* values. Two neurocontrollers are independently trained for generators G1 and G3.

$$J_c(k) = \Delta\hat{\omega}(k+1) - \Delta\omega_d(k+1) \tag{9}$$

$$e_d(k) = J_c(k) \cdot \overline{V}^T \tag{10}$$

$$e_{ai}(k) = d_i \cdot (1 - d_i) \cdot e_{di} \tag{11}$$

$$\Delta\overline{X}(k) = \overline{e}_a(k) \cdot \overline{W} \tag{12}$$

$$\Delta V_{pss}(k) = \Delta X_4(k) \tag{13}$$

$$fitness_c = \frac{1}{1000}\left[\sum_{k=0}^{999}\left(\hat{V}_{PSS}(k) - V_{PSS}(k) - \Delta V_{pss}(k)\right)^2\right] \tag{14}$$

VI. Implementation Results

Experiments have been carried out with the RTDS to demonstrate the PLC's capability as a platform for implementing the conventional power system stabilizer (Fig. 2), the neural network system models and the NCs. These experiments where conducted with 5 different setups: no control, CPSS on generators G1 and G3 (parameters given in Table I), NC on generator G1 and CPSS on generator G3, CPSS on generator G1 and NC on generator G3 and finally, NCs on both generators G1 and G3.

The disturbance that is used to evaluate the performance of each of the control systems is a 3-phase fault on bus 7 (Fig. 1) lasting 10 cycles. The results of these tests can be seen in Figs. 8 to 13. In Figs. 8 and 9, the blue curve is the system response to the 3-phase fault with no CPSS installed. This shows the oscillatory nature of this system due to fault conditions. The second, green curve, illustrate the generator speed deviation with the CPSS implemented on the PLC. This shows the performance improvement by the CPSS as compared to the no CPSS case. Figs.10 and 11 shows the performance differences between the NC on generator G1 and CPSS on generator G3, NC on generator G3 and CPSS generator G1, and CPSS on both generators G1 and G3. In these two figures, the blue curve illustrates the speed deviation response of the system with CPSS controllers installed on both generators G1 and G3. The green curve illustrates the generator speed deviation with an NC on generator G1 and CPSS on generator G3. Finally, the red curve illustrates the generator speed deviation with the NC controller on generator 3 and CPSS on generator 1. Improved system responses as a result having NC over the CPSS is observed. In the final two figures, Figs. 12 and 13, the response the system speed deviation response is illustrated between NC configurations. The blue curve illustrates the generator speed deviation with an NC on generator G1 and CPSS on generator G3. The green curve illustrates the generator speed deviation with the NC controller on generator G3 and CPSS on generator G1. Finally, the red curve illustrates the system speed deviation response with NCs on both generators G1 and G3. This shows the improved system response with neurocontrollers over the CPSS. This yields the best system response for this disturbance. The maximum overshoot and settling time calculations are given in Table II.

TABLE I.
STANDARD CPSS PARAMETERS [11].

K	Tw s	T_1 s	T_2 s	T_3 s	T_4 s	V_{PSSMAX} PU	V_{PSSMIN} PU
20	10	0.05	0.02	3.0	5.4	1.2	0.2

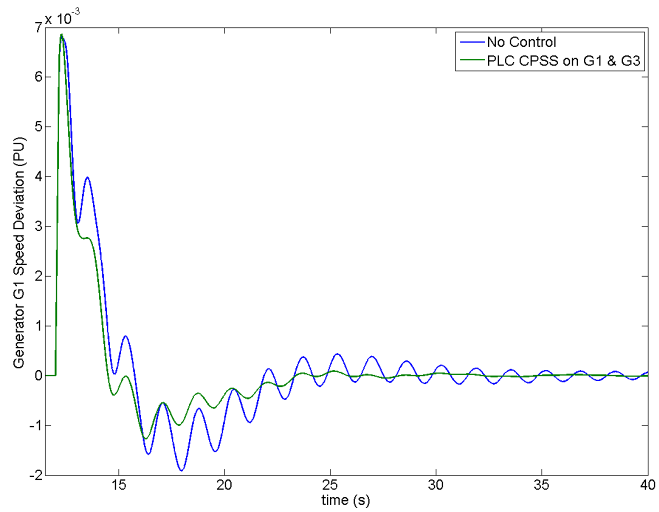

Fig. 8. Generator G1 speed oscillations after a 3-phase fault without (no control) and with CPSS on both generators G1 and G3.

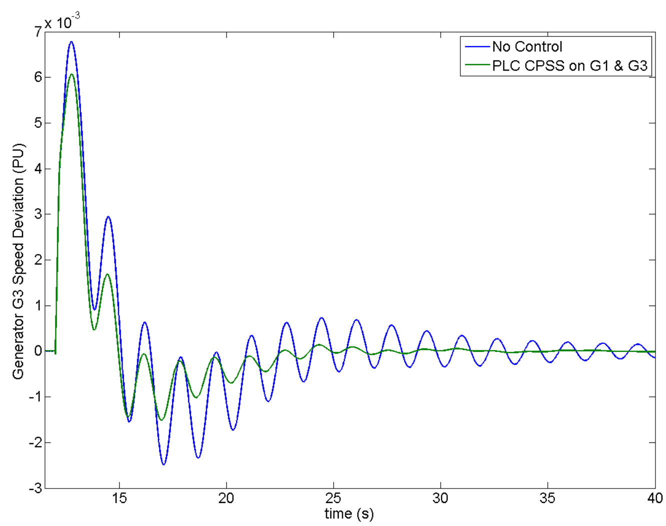

Fig. 9. Generator G3 speed oscillations after a 3-phase fault without (no control) and with CPSS on both generators G1 and G3.

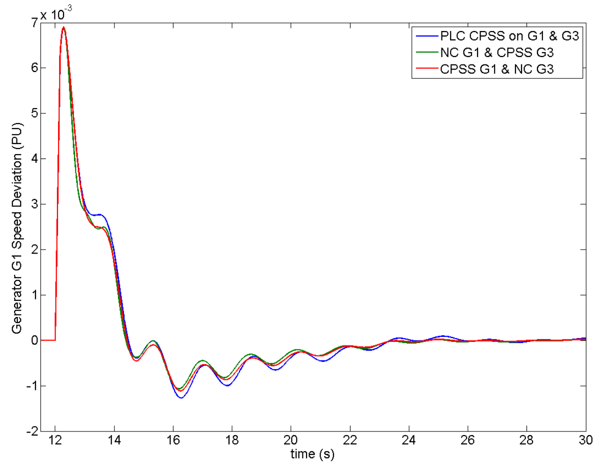

Fig. 10. Generator G1 speed oscillations after a 3-phase fault with a CPSS on both generators G1 and G3, NC on generator G1 and CPSS on generator G3, and CPSS on generator G1 and NC on generator G3.

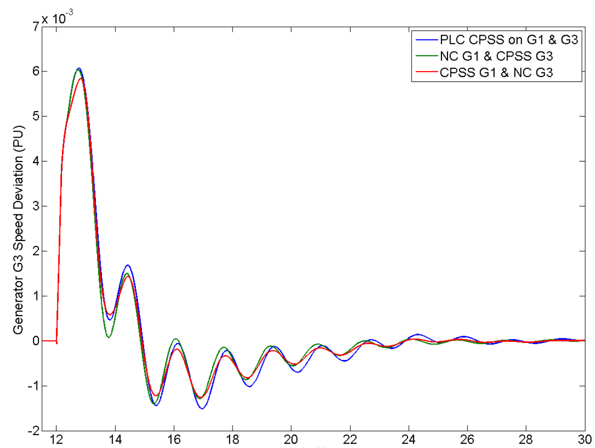

Fig. 11. Generator G3 speed oscillations after a 3-phase fault with a CPSS on both generators G1 and G3, NC on generator G1 and CPSS on generator G3, and CPSS on generator G1 and NC on generator G3.

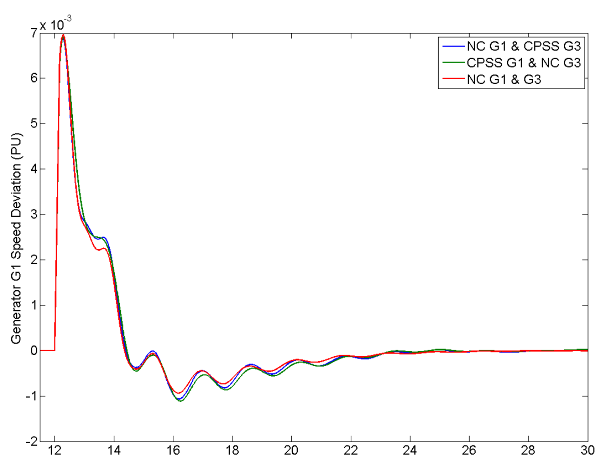

Fig. 12. Generator G1 speed oscillations after a 3-phase fault with an NC on generator G1 and CPSS on generator G3, CPSS on generator G1 and NC on generator G3 and NC on both generators G1 and G3.

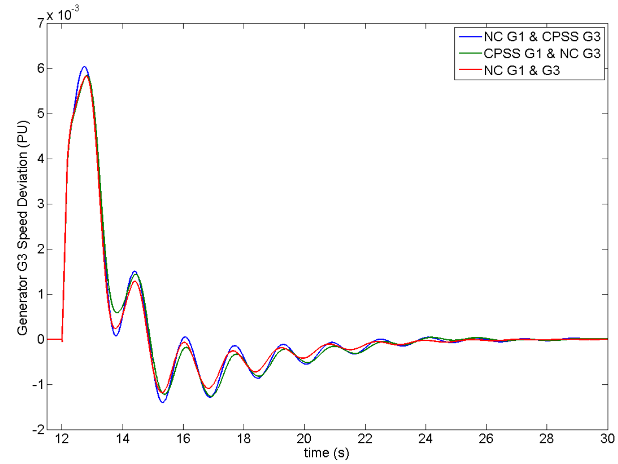

Fig. 13. Generator G3 speed oscillations after a 3-phase fault with an NC on generator G1 and CPSS on generator G3, CPSS on generator G1 and NC on generator G3 and NC on both generators G1 and G3.

TABLE II.
MAX OVERSHOOT AND SETTLING TIME CALCULATIONS

	Generator 1		Generator 3	
	Max Overshoot	Settling Time (s)	Max Overshoot	Settling Time (s)
No Control	0.6812%	31.3991	0.6782%	37.9489
CPSS G1 & G3	0.6872%	13.4979	0.6071%	18.8663
IDNC G1 & CPSS G3	0.6883%	12.2537	0.6036%	14.6109
CPSS G1 & IDNC G3	0.6903%	11.1544	0.5841%	11.6800

VII. CONCLUSION

The programmable logic controller is proposed as a research and industrial platform for implementing advanced modeling, control and optimization algorithms in real-time. The PLC platform is a better and more robust architecture for implementing advanced modeling and control techniques, with the ability to run on an existing hardware infrastructure in industry. In this paper, the development and implementation of an intelligent model and controller on a PLC for damping power system oscillations experienced by synchronous generator in a power system is illustrated. Computational intelligence paradigms - neural networks and particle swarm optimization, have been successfully implemented on a PLC in a real-time. The PSS case study presented in this paper demonstrates the PLC's ability to perform complex modeling, control and optimization. The PLC platform can be extended to implementations of intelligent techniques for many more applications including control of induction motors, permanent magnet synchronous motors, and their fault diagnosis and prognostics, and wide area power system monitoring and control.

REFERENCES

[1] K. T. Erickson, 'Programmable logic controllers', *IEEE Potentials*, 1996.

[2] P. Forsyth, T. Maguire, and R. Kuffel, 'Real time simulation for control and protection system testing', *IEEE Power Electronics Specialists Conference Proceedings*, 2004, pp. 329-335.

[3] M. Basler, and R. Schaefer, ‘Understanding power system stability’, *Pulp and Paper Industry Technical Conference,* 2007, pp. 37-47.

[4] W. Lui, G. K. Venayagamoorthy, and D. Wunsch, ‘Design of an adaptive neural network based power system stabilizer’, Neural Networks, 16, 2003, pp. 891-898.

[5] R. Kiran, S. R. Jetti and G. K. Venayagamoorthy, ‘Online training of a generalized neuron with particle swarm optimization’, *Proceedings of International Joint Conference on Neural Networks*, 2006.

[6] E. Larsen, and D. Swann, ‘Applying power system stabilizers part I, II, III’, *IEEE Transactions on Power Apparatus and Systems*, 1981, pp. 3017-3041.

[7] P. Kundur, *Power system stability and control*. New York: McGraw-Hill, 1994.

[8] M. Klein, G. Rogers, and P. Kundur, ‘A fundamental study of inter-area oscillations in power systems’, *IEEE Transactions on Power Systems*, 1991, pp. 914-921.

[9] K. Narendra, ‘Identification and control of dynamical system using neural networks’, *IEEE Transactions on Neural Networks*, 1990, 4-27.

[10] K. Erickson, *Programmable Logic Controllers: An Emphasis on Design and Applications*, Rolla: Dogwood Valley Press, 2005, pp. 1-20.

[11] S. Ray, and G. K. Venayagamoorthy, ‘Real-time implementation of a measurement-based Adaptive wide-area control system considering communication delay’, *IET Proceedings on Generation, Transmission and Distribution*, 2.1, 2007, pp. 62-70.