Implementation of a Real-Time Simulator for Dynamic Systems

H. M. Abdull-Kader,
Menoufia University, Faculty of Computers and Information, Information Systems Dept., Egypt,
hatem6803@yahoo.com

A. B. El-Sisi,
Menoufia University, Faculty of Computers and Information, Computer Science Dept., Egypt

K. A. Mostafa,
Menoufia University, Faculty of Computers and Information, Information Technology Dept., Egypt

Imbaby I. Mahmoud
Engineering Dept., Atomic Energy Authority, Cairo, Egypt,
imbabyisma@frcu.eun.eg

Abstract

Real-time systems refer to systems that have real-time requirements for interacting with a human operator or other agents with similar timescales. An efficient simulation of real-time systems requires a model that is accurate enough to accomplish the simulation objective and is computationally efficient. In this paper a real time modeling system for dynamic systems will be studied. Normally, Real time modeling can be classified into hardware and software systems, but this work focuses on the software techniques and systems. Finally a demonstration example for real time simulator has been simulated for complex dynamic system, namely a small nuclear fusion device (Egyptor Tokamak). The obtained results agree well with published work. Such simulator can be considered an imperative requirement for predicative control tasks.

1. Introduction

Real-time simulation has been used for many years for operators training and design or testing of hardware and software in situations in which it is inconvenient to use the real system [1-4]. Examples include flight simulators for pilot training, plant simulators for operator training, and a wide range of applications in which the simulator is used to test hardware and embedded software in the loop. One of the key parameters of a real-time simulator is the frame rate. Many real time simulators, are used for operator training, perform satisfactorily with frame times in the range 10 to 100ms [5]. All the calculations needed to advance the simulation by one frame must be completed, along with all necessary data transfers within one frame. In some applications real-time simulation is used as a test environment for real hardware or embedded software referred to as the system under test (SUT) [6]. In some cases much shorter frame times (<10μs) are required because of the high-frequency dynamics of both the simulated system and (SUT). Such applications are found, for example, in aerospace, automotive and power electronic systems [7], [8]. The techniques described here focus on power electronic systems, but they are equally relevant to other applications requiring similar frame times. In the literatures, many dynamic systems applications can be found such as robotics, industrial production systems, and nuclear power plants [9-12].

In this paper we will focus on the nuclear power plant application. This type of application can be classified in two main categories namely nuclear fusion experiments and fission reactors. The structure of this paper is as follows. Section 2 gives an overview about real-time modeling. In section 3 the details of case study is introduced. Section 4 describes the proposed simulator for our case study. The result of the proposed simulator is shown in section 5. Finally, some conclusion are put forward in Section 6.

2. Modeling Techniques of Real-Time Systems

Real-time systems differ from traditional data processing systems in that they are constrained by certain nonfunctional requirements (e.g., dependability and timing). Although real-time systems can be modeled using the standard structured design methods, these methods lack explicit support for expressing the real-time constraints. Standard structured design methods incorporate a life cycle model in which the following activities are recognized:

(1) Requirements definition. An authoritative specification of the system's required functional and nonfunctional behavior is produced.
(2) Architectural design. A top-level description of the proposed system is developed.
(3) Detailed design. The complete system design is specified.
(4) Coding. The system is implemented.
(5) Testing. The efficacy of the system is tested.

Real time simulation systems can be implemented via software or hardware. For hard real-time systems, this has the significant disadvantage that timing problems will be recognized only during testing, or worse, after deployment. Researchers have pointed out that the time requirements should be addressed in the architectural design phase [13]. The architectural design activities are defined as the logical architecture design activity, and the physical architecture design activity. The logical architecture embodies commitments that can be made independently of the constraints imposed by the execution environment, and is primarily aimed at satisfying the functional requirements. The physical architecture takes these functional requirements and other constraints into account, and embraces the non-functional requirements. The physical architecture forms the basis for asserting that the application's nonfunctional requirements will be met once the detailed design and implementation have taken place [14]. The physical architecture design activity addresses timing (e.g., responsiveness, orderliness, temporal predictability and temporal controllability) and dependability requirements (e.g., reliability, safety and security), and the necessary schedulability analysis that will ensure that the system once built will function correctly in both the value and time domains. Appropriate scheduling paradigms are often integrated to handle nonfunctional requirements.

3. Methods for Modeling dynamic systems

Methods for modeling dynamic systems can be classified into different categories. The first one is mathematical models and the second is heuristics models. Also there are some methods which use a combination between the previous two methods called hyperid methods.

3.1.1. Mathematical Methods. In this method the system can be governed by a set of equations called system equations. Before discussing dynamic models, let us recall that time-invariant dynamic systems are in general modeled by static functions, by using the concept of the system's state. Given the state of a system and given its input, we can determine what the next state will be. In the discrete-time setting we can write

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

where x(k) and u(k) are the state and the input at time k, respectively, and f is a static function, called the *state-transition function*.

Dynamic models of different types can be used to approximate the state-transition function. As the state of a process is often not measured, *input-output* modeling is usually applied. The most common is the NARX (Nonlinear Auto Regessive with exogenous input) model:

$$y(k+1) = f(y(k),\ y(k-1)...,y(k-ny+1),\ u(k),\ u(k-1),...,u(k-nu+1)) \tag{2}$$

Here $y(k), \&, y(k - ny + 1)$ *and* $u(k), \&, u(k - nu + 1)$ denote the past model outputs and inputs respectively and n_y, n_u are integers related to the model order (usually selected by the user).

In this sense, we can say that the dynamic behavior is taken care of by external dynamic filters added to the dynamic system. In equation (2), the input dynamic filter is a simple generator of the lagged inputs and outputs, and no output filter is used. Since the dynamic models can approximate any smooth function to any degree of accuracy, models of type, given by equation (2), can approximate any observable and controllable modes of a large class of linear or discrete-time nonlinear systems.

3.1.2. Heuristics Methods. In this method we can not describe the behavior of the system by a known set of equations. But we need fitting methods to get approximation. So we need information about the system from the field of this system. Two common sources of information for building dynamic models are the prior knowledge and data process measurements. The prior knowledge can be of a rather approximate nature (qualitative knowledge, heuristics), which usually originates from "experts", i.e., process designers, operators, etc. In this sense, dynamic models can be regarded as simple dynamic expert systems. For many processes, the data that are available as records of the process operation or special identification experiments can be designed to obtain the relevant data. Building dynamic models from data involves methods based on dynamic logic and approximate reasoning, but also ideas originating from the field of neural networks, data analysis and conventional systems identification. The acquisition or tuning of dynamic models by means of data is usually termed dynamic identification.

In the literature there are two main approaches to the integration of knowledge and data in a dynamic model. These approaches can be distinguished as:

(1) The expert knowledge expressed in a verbal form is translated into a collection of If–Then rules. In this way, a certain model structure is created. Parameters in this structure (membership functions, consequent singletons or parameters) can be fine-tuned using input output data [15].

(2) No prior knowledge about the system under study is initially used to formulate the rules, and a dynamic model is constructed from data. It is expected that the extracted rules and membership functions can provide a posteriori interpretation of the system's behavior. Related to which one of the previous an expert can confront this information with his own knowledge, can modify the rules, or supply new ones, and can design additional experiments in order to obtain more informative data. These techniques, of course, can be combined, depending on the particular application

4. Selection of model Structure and parameters

With regard to the design of dynamic models, two basic items are distinguished: the structure and the parameters of the model. The structure determines the flexibility of the model in approximation (unknown) mappings. The parameters are then tuned (estimated) to fit the data at hand. A model with a rich structure is able to approximate more complicated functions, but, at the same time, has worse *generalization* properties. Good generalization means that a model fitted to one data set will also perform well on another data set from the same process. In dynamic models, structure selection involves the following choices:

1. *Input and output variables.* With complex systems, it is not always clear which variables should be used as inputs to the model. In the case of dynamic systems, one also must estimate the order of the system. For example the input-output NARX model, this means to define the number of input and output lags n_y and n_u, respectively. Prior knowledge, insight in the process behavior and the purpose of modeling are the typical sources of information for this choice. Sometimes, automatic data-driven selection can be used to compare different choices in terms of some performance criteria.

2. *Structure of the rules*. This choice involves the model type and the antecedent form . Important aspects are the purpose of modeling and the type of available knowledge.

3. *Number and type of membership functions for each variable.* This choice determines the level of detail (granularity) of the model. Again, the purpose of modeling and the detail of available knowledge, will influence this choice [16].

4. *Type of the inference mechanism, connective operators.* These choices are restricted by the type of dynamic model [17].

To facilitate data-driven optimization of dynamic models (learning), differentiable operators (product, sum) are often preferred to the standard min and max operators. After the structure is fixed, the performance of a dynamic model can be fine-tuned by adjusting its parameters. Tunable parameters models are the parameters of antecedent and consequent membership functions (determine their shape and position) and the rules (determine the mapping between the antecedent and consequent dynamic regions).

One of the most parameter to select the model is *model accuracy* which define the ability of a model to capture the system at the right level of detail and to achieve the simulation objective within an allowable error bound. Computational efficiency involves the satisfaction of the real-time requirements to simulate the system, in addition to the efficiency of model computation. In existing applications, it is a user's responsibility to construct the model appropriate for the simulation task.

5. Case Study: Small Nuclear Fusion Device Construction

Many fusion-related plasma experiments use magnetic field confinement to contain the charged plasma. A toroidal device called a Tokamak, which is first developed in Russia, is the most successful device yet found for magnetic confinement of plasma [18], [19]. In this device a combination of two magnetic fields is used to confine and stabilize the plasma, a strong toroidal field (TF), is produced by the current in the windings, and a weaker "poloidal" field, is produced by the toroidal current (current in the plasma). In addition to confining the plasma, the toroidal current is used to heat it. The resultant helical field lines spiral around the plasma and keep it from touching the walls of the vacuum chamber. There are many small fusion experiments used as hardware simulator for large fusion projects like (TEXT, TEXTOR, ASDEX) [20]. The Egyptor tokamak is one from these experiments [21-23].

The Egyptor is a small tokamak device of noncircular cross section (R=30 cm, a=10 cm) intended primarily to study of plasma-wall interaction. Figure (1) shows a block diagram for Egyptor tokamak. Studies can be carried out in this small machines , the results of these studies are key in making predictions for larger devices [24]. The detailed balance of (particle production and loss), and of (power input and loss) determine the state of the plasma in a tokamak. The particle balance equation [25] is basically single-ion-species plasma, setting $n_e = n_i = n$, is given by

$$\frac{\partial n}{\partial t} = nn_n \langle s\,v \rangle_i + \frac{1}{r}\frac{\partial}{\partial r}\left[rD_A \frac{\partial n}{\partial r} \right] \quad (\mathrm{m^{-3}.s^{-1}}) \tag{3}$$

where $< s v>_i$ is ionization rate of the neutral particles that fuel the plasma, n_e is the electron density, n_i is the ion density, n is the particle density, n_n is the neutral density, r is the minor radius, and D_A is the diffusion coefficient. At steady state the above equation is equal to 0, which can be stated in the following form:

$$\frac{d}{dr}[(\frac{r}{n})(\frac{dn}{dr})]+[\frac{e^2V_e^2}{K(T_i+T_e)}]nr = 0 \tag{4}$$

where $\boldsymbol{T_i}$ and $\boldsymbol{T_e}$ are ion and electron temperatures, **e** is electron charge, **K** Boltzman's constant, and $\boldsymbol{V_e}$ is electron velocity. Substituting $[(e^2V_e^2)/K(T_i+T_e)] = b$, the equation becomes

$$\frac{d}{dr}[(\frac{r}{n})(\frac{dn}{dr})]+bnr = 0 \tag{5}$$

The power balance may be written separately for each species [25] as following.

i- For the electrons, the local power balance is given by

$$\frac{\partial}{\partial t}\left[\frac{3}{2}n_e eT_i\right] = \frac{1}{r}\frac{\partial}{\partial r}r\left[n_e ec_e\frac{\partial T_e}{\partial r}+\frac{3}{2}T_e eD_A\frac{\partial n_e}{\partial r}\right]+P_{ie}-P_b-P_R-P_c+P_\Omega+P_{ae}+P_{ae}(\mathrm{w.m^{-3}}) \tag{6}$$

ii- For the ions, the local power balance is given by

$$\frac{\partial}{\partial t}\left[\frac{3}{2}n_i eT_i\right] = \frac{1}{r}\frac{\partial}{\partial r}r\left[n_i ec_i\frac{\partial T_i}{\partial r}+\frac{3}{2}T_i eD_A\frac{\partial n_i}{\partial r}\right]+P_{ei}-P_{cx}+P_{ai}+P_{ai}(\mathrm{w.m^{-3}}) \tag{7}$$

The parameters c_e and c_i are the electron and ion thermal diffusivities, respectively, and $\boldsymbol{P}$ (W.m^{-3}) is the power density of the various mechanisms indicated by the subscripts:

ie ion-electron collision, *W* ohmic heating, ***b*** bremsstrahlung radiation, *a* alpha power, ***R*** radiation, ***a*** auxiliary power, ***c*** cyclotron radiation, ***cx*** charge exchange, and ***ei*** electron-ion collision.

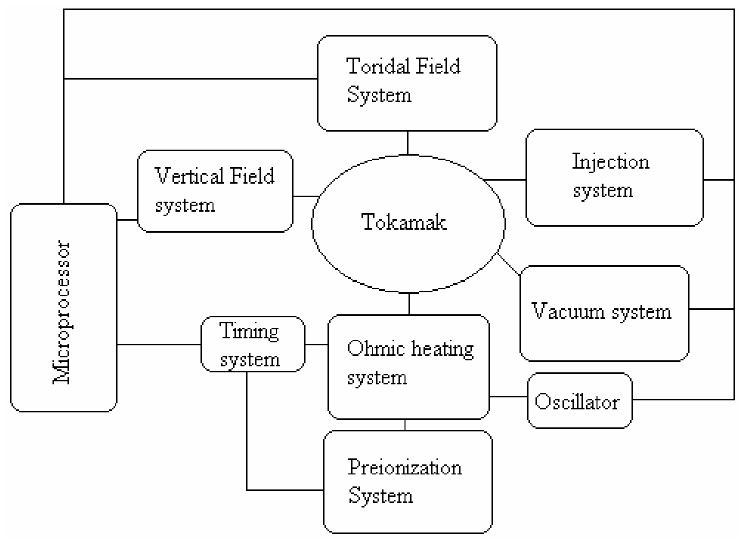

Figure 1 Block Diagram for Egyptor Tokamak

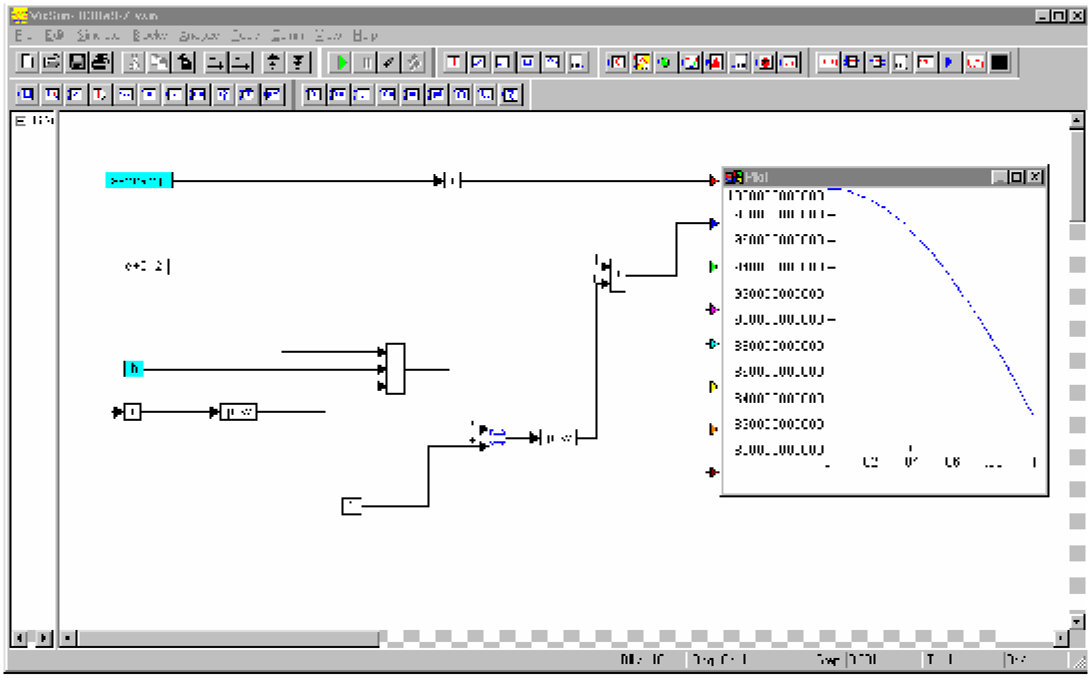

Figure 2-a. Implemented Simulator Program for Bennett Distribution.

6. Proposed Simulator for Tokamak Device

A proposed program for modeling and simulation of Tokamak is designed and implemented in **VisSim** environment, where **VisSim** is a visual block diagram language for nonlinear dynamic simulation. A block API allows users to create their own blocks in C/C++. Add-ons allow real-time analog and digital I/O for real-time simulation, embedded system C code generation, optimization, neural nets, OPC, frequency domain analysis, scaled fixed point, IIR and FIR filter design. In this environment, the system is modeled by the graphical interconnection of function blocks [26]. For flexibility, variables are used to denote system parameters and then are assigned values in a separate compound block. Data analysis is also included in the program. The program can be distributed with **VisSim** viewer or through generated C code from **VisSim** block diagram, which means it does not depend on the **VisSim** environment. Differentiation and Integration Blocks are adjusted to suit the nature of the Tokamak equations, which have two variables radius (r) and time (t) as described in the pervious section. Equation (5) is modeled using VisSim environment at first. Secondly, the diffusion rate is taken into account in the model. For comparison purposes, the Bennett distribution of the form $n = n_0/(1+n_0*b*r^2)^2$ [27], which is applied as an approximated solution of the particle balance equation is also modeled for verification purposes of the simulator. Figure (2-a) shows the implemented simulator program for Bennett distribution and figure (2-b) gives Bennett distribution.

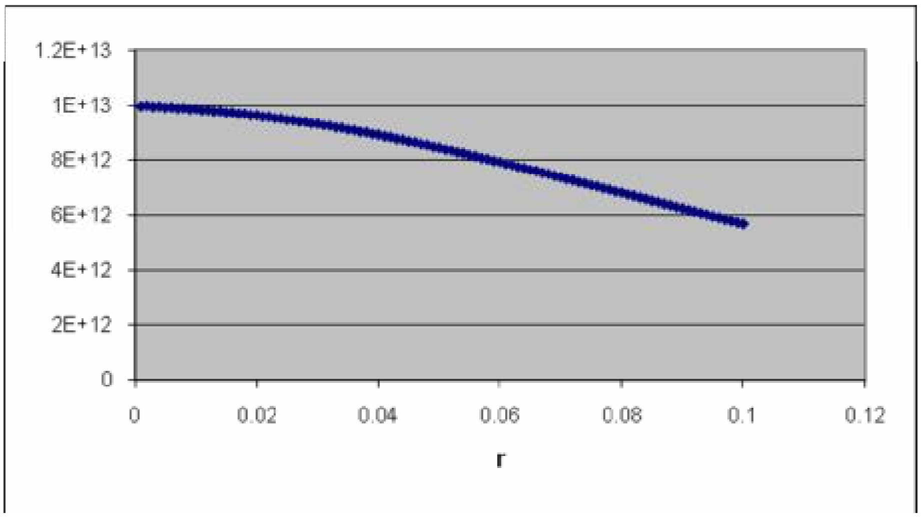

Figure 2-b. Bennett Distribution

7. Simulation Results

In this section the simulation results of our case study Small Nuclear Fusion Device (Egyptor Tokamak) using ViSim is presented. Figure (3-a) shows the implemented simulator program for power balance equation (ion temperature), and figure (3-b) shows radial ion temperature distribution.

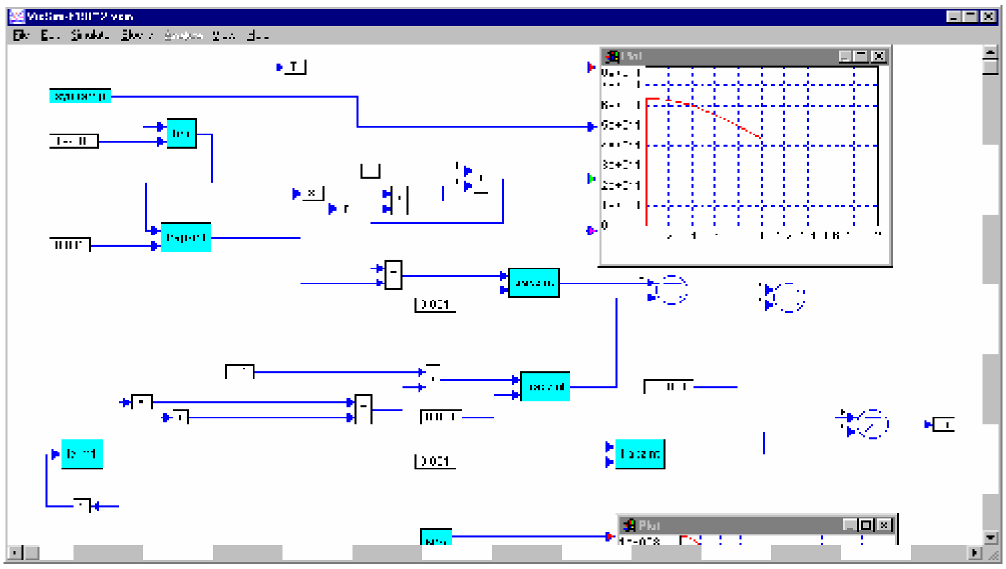

Figure 3-a. Implemented Simulator Program for Ion Temperature.

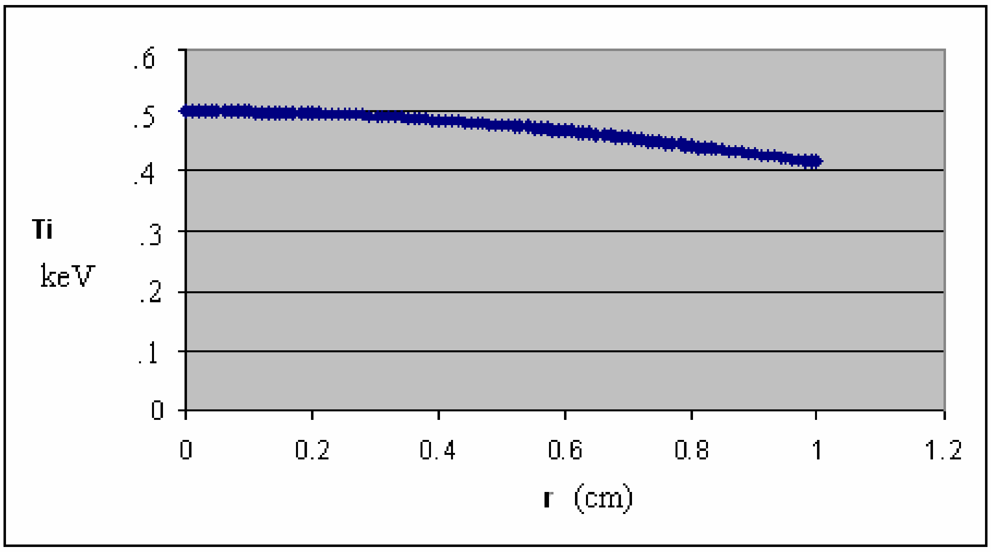

Figure 3-b. Radial Ion Temperature Distribution.

Finally figure (4-a) the implemented simulator program for particle balance equation is given, and figure (4-b) shows the radial density distribution inside plasma. The height and width of curves are controlled by Tokamak parameters (b and internal parameters in figure (2-a)).

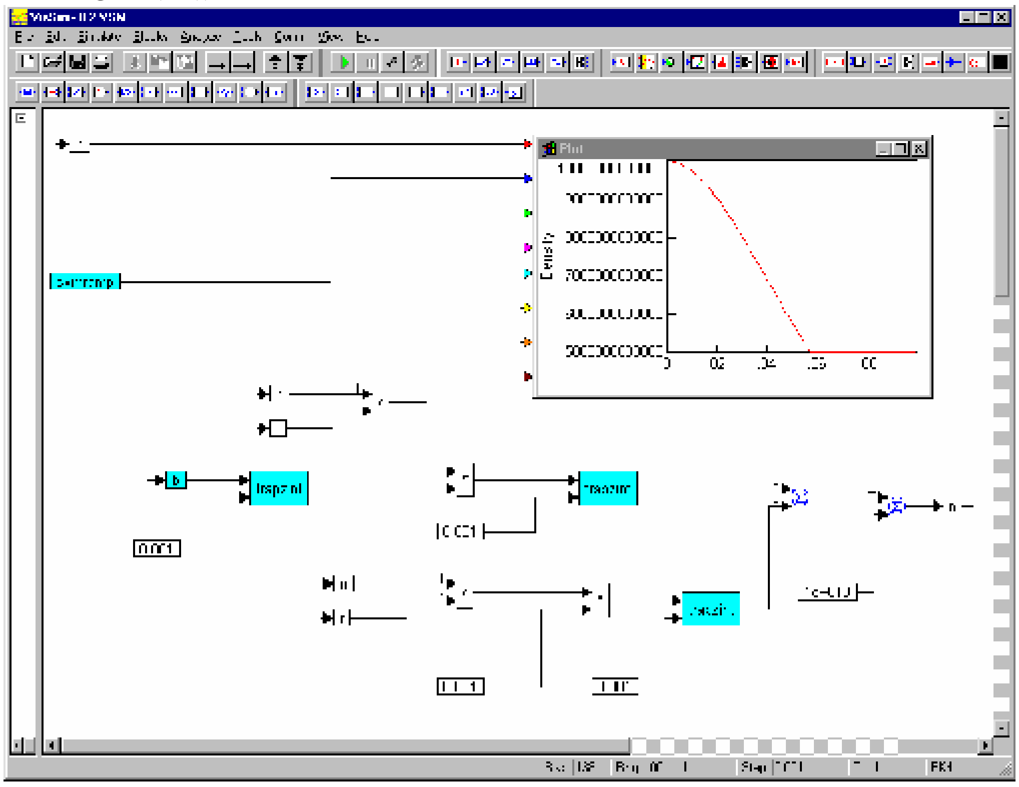

Figure 4-a. Implemented Simulator Program for Particle Balance.

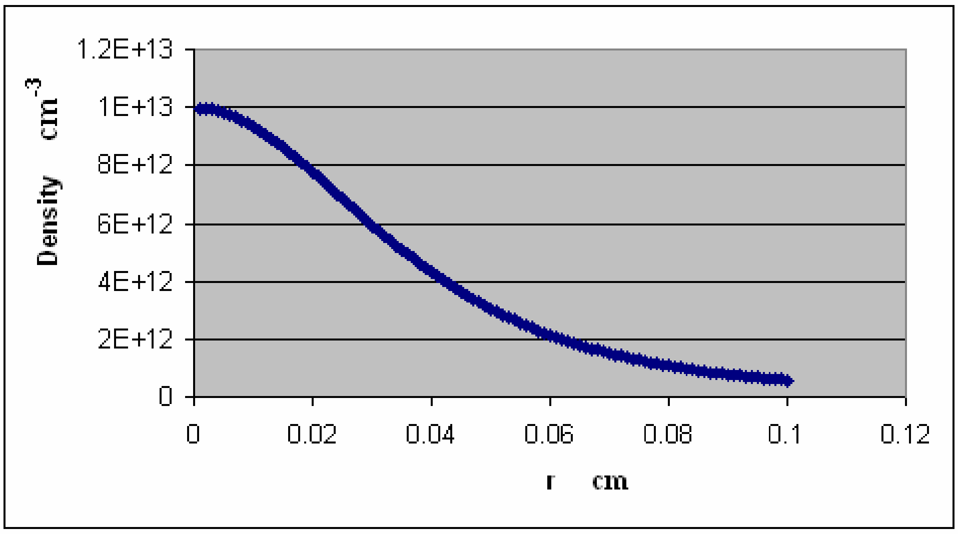

Figure 4-a. Implemented Simulator Program for Particle Balance.

Comparing with published experimental and analytical results in [28] as shown in figure (5), we obtain good agreement. Moreover, any enhancements can be easily added to the simulator in a block diagram form. Simulation of the Tokamak system can be used in conjunction with control system of Tokamak [22] using the same software. Hence, Model Predictive Control (MPC), a method which continuously updates the controller and is able to predict and act during power supply saturation can be applied. In this case, we used the real-time mode for hardware-in-the-loop control part of the whole system. Simulating in real-time mode has the effect of retarding a simulation so that one simulation second equals one second in real time.

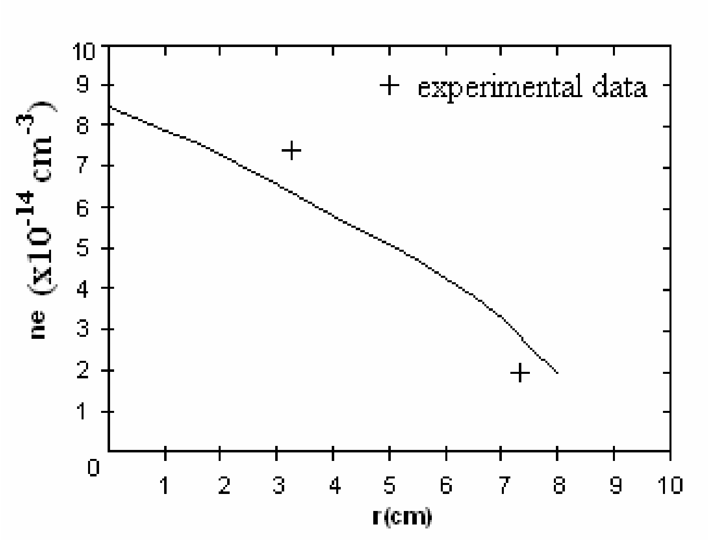

Figure 5. The electron density profile in [28].

8. Conclusions

In this paper studied a real time modeling system for dynamic systems has been focused for the software real time modeling techniques and systems. A simulator for Tokamak Egyptor is designed and implemented. It allows the introduction of the techniques of PC interfacing with industrial and scientific systems using easy to manipulate environment (e.g. block diagram method) to engineers involved in the nuclear field. The obtained results of comparison with published work is good. Using the same environment, Model Predictive Control (MPC) methods can be used. Moreover, any enhancement to the model can be added easily due to the nature of block diagram programming.

9. References

[1] Kangsun Lee, Paul A. Fishwick, "OOPM/RT: A Multimodeling Methodology for Real-Time Simulation", ACM Transactions on Modeling and Computer Simulation, Vol. 9, No. 2, April 1999, pp. 141–170.

[2] J. M. Giron-Sierra, J. A. Gomez-Pulido, B. Andres-Toro, "X-Windows Simulation of Steam Power Plants Based on Physics Principles" Proceedings of the Winter Simulation Conference, 1993.

[3] T. G. Kirner, C. Kirner, "Simulation of Real-Time Systems: An Object-Oriented Approach Supported by a Virtual Reality-Based Tool", IEEE computer society, Proceedings of the 38th Annual Simulation Symposium 2005.

[4] W. B. Rouse, "Human-Computer Interaction in the Control of Dynamic Systems", Computing Surveys, Vol. 13, No. 1, March 1981

[5] R. Crosbiea, J. Zenora, R. Bednara, and D.Worda, "High-Speed, Scalable, Real-Time Simulation Using DSP Arrays", IEEE computer society,Proceedings of the 18th Workshop on Parallel and Distributed Simulation, 2004, pp. 52-59.

[6] R. Vincent, B. Horling, V. Lesser, and T. Wagner, "Implementing Soft Real-Time Agent Control", *AGENTS'01,* Montr´eal, Quebec, Canada, May 28-June 1, 2001.

[7] H. Kopetz, "Software *Engineering* for Real-Time:A Roadmap", ACM 2000, pp. 203-211.

[8] S. Abourida, C. Dufour, J. Bélanger, G. Murere, N. Léchevin, and B. Yu "Real-Time PC-Based Simulator of Electric Systems and Drives", APEC 2002, pp. 433-438.

[9] B.G. Penaflor, et al, "Real-time control of DIII-D plasma discharges using a Linux alpha computing cluster", Fusion Engineering and Design, 56-57, 2001, pp. 739-742.

[10] J. Sousa, et al, "A distributed real-time system for event-driven control and dynamic data acquisition on a fusion plasma experiment", Fusion Engineering and Design, 48, 2000, pp. 31-36,.

[11] J.A. Romero, et al., "Real time profile control at JET", Fusion Engineering and Design, 43, 1998, pp. 37-58,.

[12] K. Nishimura, et al, "Real-time plasma control system for LHD", Fusion Engineering and Design,39-40, 1998, pp. 169-172,.

[13] Lecture Notes, "Seventh College on Microprocessor-Based Real-Time Systems in Physics", Abdus Salam ICTP, Trieste, Italy, 2002.

[14] M. Dinkel, U. Baumgarten, "Modeling Nonfunctional Requirements: A Basis for dynamic Systems Management", ACM, 2005.

[15] A.Abreu and LuisCustodio Carlos Pinto-Ferreira, "Fuzzy Modeling: a Rule Based Approach", Proceedings of the fifth IEEE International Conference on Fuzzy Systems, 1996, pp. 162-168.

[16] C.-W. Xu, Y.-Z. Lu, "Fuzzy model Identification and self learning for dynamic systems", IEEE Transactions on Systems, Man and Cybernetics, pp. 683-689, 1987.

[17] C.-C. Wong, N.-S. Lin, "Rule extraction for fuzzy modeling", Fuzzy Sets and Systems, no. 88, 1997, pp. 23-30.

[18] Artsimovitch L. A., Nuclear Fusion, 12, 215 (1972).

[19] Kadomtseve B. B. "Tokamak Plasma: A Complex Physical System" Translation editor P.Ewlaing, Institute of Physics Publishing Bristol and Philadelphia (1992).

[20] Wesson J. with contributions " TOKAMAK" Clarendon Press Oxford (1987).

[21] A. El-Sisi and M. Masoud, " Egyptor Tokamak Reconstruction, Development and Operation" *Internal Report, SIDC*, EAEA, 2001.

[22] R. Flohr t. al., "The Tokamak Experiment " UNITOR"-A Device for Plasma-Wall Interaction Research" Physica 104C, 423-433, 1981.

[23] A. B. El Sisi; H. Hegazy "EGYPTOR Tokamak: Modification of the Original Design Using Permanent Compensation Coils and First Results of the Breakdown Discharge, Journal of Fusion Energy, Volume 22, Number 3, September 2003, pp. 191-194(4).

[24] Annual Report of the EURATOM/UKAEA Fusion Programme 1997/98.

[25] J. Sheffield , "Tokamak Start-Up," Edited by H. Knoepfel, Plenum, New York 1986.

[26] Imbaby I. Mahmoud and S. A. Kamel, " Using a Simulation Technique for Switched-mode High Voltage Power Supplies Performance Study", *IEEE Trans. IAS*, Vol. 34, No. 5, Sept.-Oct. 1998, pp.945-952.

[27] Masoud M. M., Ph. D Thesis, Faculty of Science, Cairo Univeresty, 1971.

[28] M. H. Hughes, "Numerical Calculations of Transport in ALCATOR" Princeton Plasma Physics Lab. PPPL-1411, Jan., 1978.