*Int. J. Engng Ed.* Vol. 21, No. 1, pp. 41–48, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00


# Learning Faults Detection and Diagnostics Using a Virtual System*

GLORIA MOUSALLI-KAYAT, JESÚS CALDERÓN-VIELMA, ADDISON RÍOS-BOLÍVAR and FRANCKLIN RIVAS-ECHEVERRÍA
*Universidad de Los Andes, Mérida, Venezuela 5101. E-mail: mousalli@ula.ve*

*The present work focuses on teaching fault detection and diagnostics in industrial processes, through a computational tool that integrates a virtual instrument developed in LabVIEW™ 6.0 and a computer application in MatLab® 6.1 for simulating an industrial process. For integrating the applications, the dynamic data exchange (DDE) protocol has been used. The objective of this work has been to use the integration of the LabVIEW™ interface with the MatLab computational tool in order to allow students to interact with a process for detecting, by inspection or fault-detecting filters, the possible faults that can occur.*

## INTRODUCTION

THE INCLUSION OF computers in the educational environment has its roots in the mid-1950s, based on the Skinner behaviorist theory [1]. The paradigm used for developing the teaching of applied technology was called 'programmed instruction', based on the design of instructional elements composed of a series of small 'steps' which require the active response of students, giving them instantaneous feedback.

At the beginning of the 1970s, with the rising popularity of artificial intelligence [2], it was discovered that these techniques could be used for computer-aided learning. For instance, Carbonell [3] developed an intelligent tutorial system for teaching South-American geography. This was the first step towards intelligent tutorial systems.

Another type of program that has expanded rapidly over the last decades is the simulator, which allows a similar environment to that found in industrial control rooms to be created so that it becomes possible to accumulate knowledge and experiences that can be used in real situations. These simulators try to support learning, by emulating real-world situations [4]. They have been widely used for applications in the engineering field, because they can be used for simulating chemical, physical, mechanical and other processes using personal computers (PC).

The present work uses the capabilities of LabVIEW™ [5] to develop human–machine interfaces, combining it with a processes simulation program using MatLab® [6] to create a tutorial-type computational tool whose main objective is to teach people fault detection and diagnosis methods. Learning is achieved through a group of virtual tests whereby students can observe the behavior of the system under different fault conditions.

The paper is structured in the following way. First, the DDE protocol for process communication and the application program that interfaces LabVIEW™ MatLab will be introduced. Next, a brief introduction to fault detection will be presented. Then the computational tool that has been developed for learning fault detection and diagnostics using a virtual system will be described. Finally, the results and conclusions are presented.

## PROCESSES COMMUNICATION: DDE PROTOCOL

The dynamic data exchange (DDE) protocol is one of the communication methods most commonly used for exchanging data between Microsoft Windows® applications. The DDE protocol is based on the messaging system developed by Windows® [7], hence two Windows programs, as shown in Fig. 1, carry out a 'DDE conversation' by exchanging messages between them. These two programs are known as the server and the client. A DDE server is a program which has access to data that can be useful for other programs. The DDE client is the program that obtains these data from the server.

A DDE conversation begins when the program that acts as the client transfers a message to all the Windows® programs that are being utilized. This message indicates which general data category the client needs. The DDE server that possesses the data can respond to this message, and it is at that moment that the conversation begins. Only one program can be client for another program and the server for others, but this requires two different DDE conversations. A server can give data to multiple clients and a client can obtain data from

* Accepted 4 August 2004.

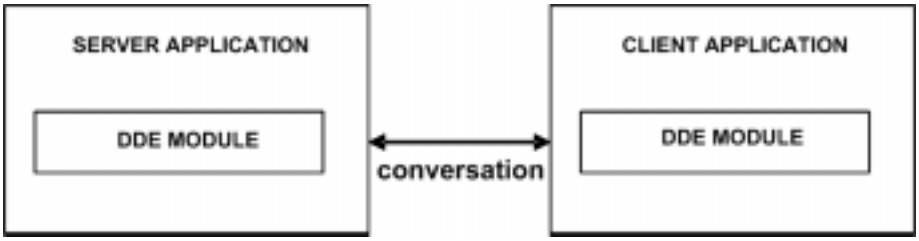


Fig. 1. Diagram of a DDE conversation.

multiple servers, but this requires multiple DDE conversations.

Both applications should be running and both of them give a statement concerning their calling functions to Windows before the DDE communication begins. The calling function accepts any DDE message that Windows sends to the application.

A DDE client begins a conversation with another application (a DDE server) by sending a connection message. After establishing a connection, the client can send orders or data to the server and it can request the data values that the server manages. When the DDE communication is completed, the client sends a message to the server closing the conversation.

For developing a DDE communication or conversation, it is necessary to identify the data type to be exchanged. This operation is carried out by means of three strings with information concerning the server: application, data topic and data item.

More precisely, it needs to know: the application or 'service name' that specifies the name of the application server that the client is connected to; the data topic (this is often the file name, but the definition can vary) for opening the connection with the application server; and, finally, the data item (often the name of a variable). Data and commands are transferred as text format. The 'topic' is the second level in the three strings and defines the object of a DDE conversation, and this is usually an important point for the server application and for the client. The 'element' identifies the data or value that is being sent during the DDE conversation between the server and the client.

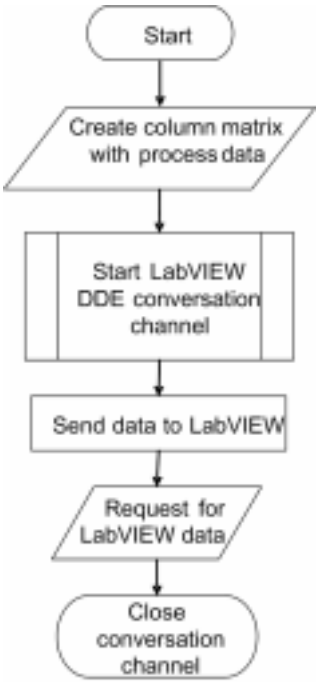


Fig. 2. DDE MatLab client: function flow diagram.

## THE MATLAB APPLICATION INTERFACE PROGRAM (MATLAB-API)

MatLab® is a mathematical environment whose fundamental operating elements are matrices [6]; this allows its immediate application in the solution of lineal algebra problems and includes graphic capabilities and basic programming structures with similar syntax to programming languages such as C, Fortran and Basic. MatLab® possesses a graphic programming environment called 'Simulink', which simulates linear and non-linear systems by means of programming based on block diagrams.

The DDE protocol can be used for interacting with Simulink through another application [8], for which MatLab® can be either a client or a server. If MatLab® is a client, it is necessary to build a dialog box in Simulink with suitable functions for initiating a DDE conversation for exchanging data [9]. A function flow diagram is shown in Fig. 2 for integrating the virtual process in MatLab® with LabVIEW™.

The functions used by MatLab for DDE client applications are: 'ddeinit', 'ddepoke', 'ddereq', 'ddeadv', 'ddeexec', 'ddeunadv' and 'ddeterm'. These functions are used for creating the processes, as described in Fig. 2.

## THE LABVIEW APPLICATION INTERFACE PROGRAM (LABVIEW-API)

LabVIEW™ is a graphic programming environment which has a direct interface with a personal computer for developing applications such as data acquisition and data analysis. LabVIEW™ uses graphic symbols (icons) to describe the action sets. The data flow is through the connections in a block diagram.

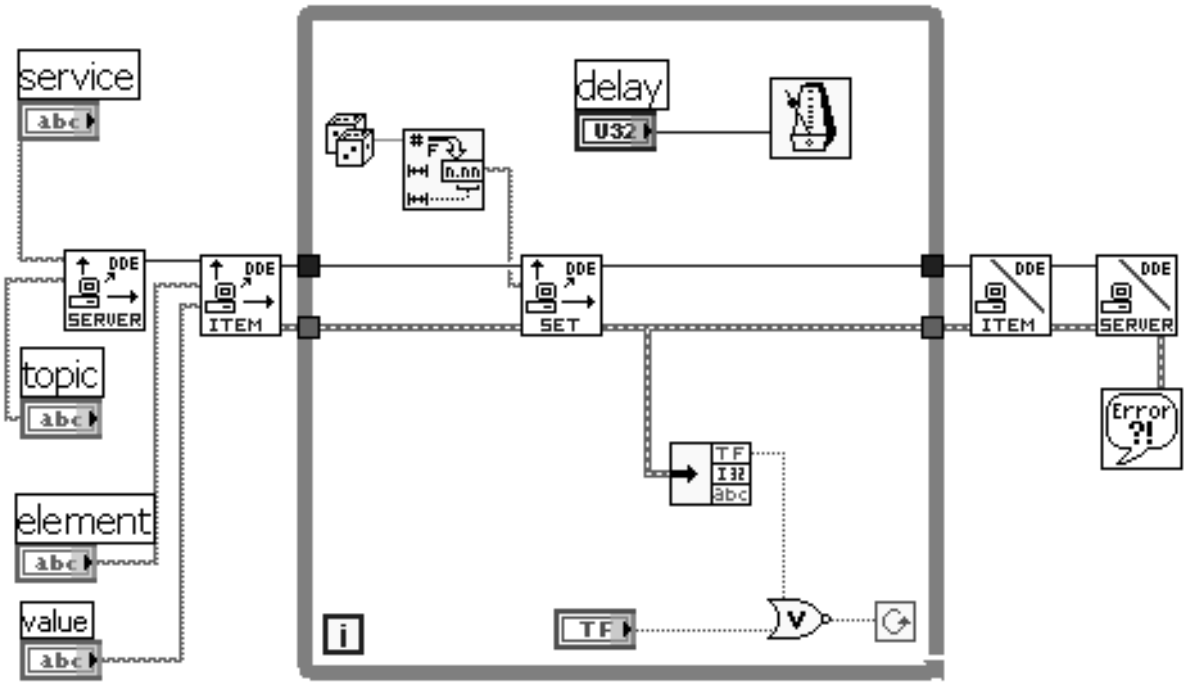


Fig. 3. Block diagram for a LABVIEW DDE server.

LabVIEW$^{TM}$ provides a group of VIs (virtual instruments) or program DDEs; these facilitate the creation of VIs that act as DDE clients for other applications (these VIs request or send data to other applications) [5]. Furthermore, VIs can also be created that act as information servers which can be used through other applications. As a server, LabVIEW$^{TM}$ is not able to use communication based on connection.

The DDE protocol used by LabVIEW$^{TM}$ is based on ASCII, and the transmission ends when a null byte appears.

A number has to be transformed into a string format before it can be sent from LabVIEW$^{TM}$ to another application. Also, the data received as a result of a request should be in numbers.

A LabVIEW$^{TM}$ VI application can be created to act as a server for the data elements. In general, a VI indicates that it can provide information regarding a specific service and topics. LabVIEW$^{TM}$ can use any name for the service and the topic.

Figure 3 shows the block diagram of a VI DDE server that provides data to other client applications. In this case, the data comes from random numbers. The random number generator can be easily replaced by real-world data from a data acquisition system or from series connections.

## FAULT DETECTION

It is well known that operational reliability should be ensured by correct operation of the processes, appropriate control systems and coordination. The whole infrastructure is held by diverse support systems inside an integral automation structure, where the quality of the information and its exchange are guaranteed in terms of reliability, security and productivity. At any stage in the production chain, the information should be managed to ensure high efficiency and operational productivity.

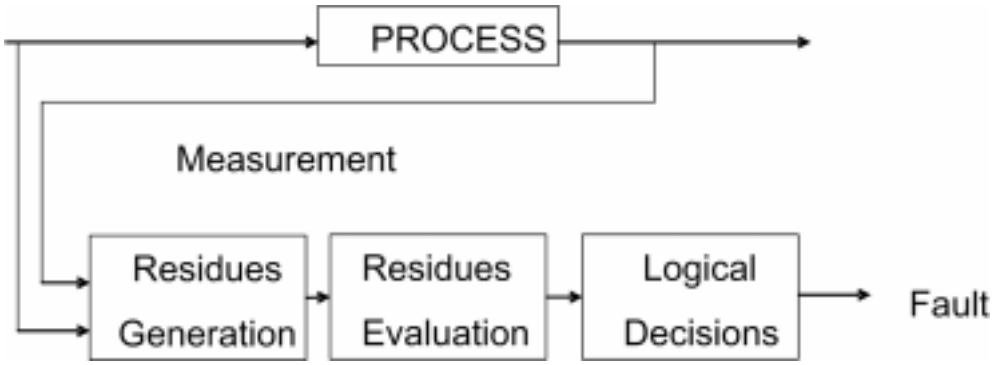


Fig. 4. An SDD system.

Inside a reliable and safe operational context, the systems that allow events recognition should be presented, which should guide decision-making when the behavior of the productive process is affected by any adverse eventuality. Since reliability is very close to the concept of security, it is fundamental to provide industrial processes with strict security mechanisms whose basic elements are supervision, diagnostic and detection (SDD). These, using the indicators and measured variables of the processes, maintain continuous supervision of the evolutionary process of the product and report anything abnormal.

SDD systems have the capacity to respond to unexpected situations during processing, so their main task is fault diagnosis and detection (FDD). An FDD system, such as the one shown in Fig. 4, uses the measurements of the process in order to produce some residuals, which, by means of evaluation functions and decision logics, identify and isolate the fault. Any system that, starting with measured variables of the process, allows residuals to be generated and evaluates these in depth with regard to fault recognition is known as a 'faults detection and diagnosis filter'.

From the point of view of generating comparison-based residuals, the filter design techniques for FDD can be classified as 'methods based on physical redundancy and methods based in models' [10]. We use model-based methods, as this is a precise analytic redundancy approach.

In analytic techniques, all the information coming from the process measurement devices is used to obtain a mathematical model for diagnostic purposes. A detailed description of the procedure used to generate these residuals can be found in [10–12]. The following four steps briefly describe the procedure:

1. Use direct substitution in the model equations.
2. Use the model together with the real process, so that in both the same inputs are applied.
3. Use a state observer. This is an extension of the parallel-method model. The main aim is residual generation, with precise directional properties, by means of an appropriate selection of the observer's gain.
4. Create an inverse model in order to reconstruct the faults.

*Observer-based filter design*

State observers are analytic techniques based on the design of the fault detection and diagnostic (FDD) filters. The FDD filter design can be divided into two stages: the first phase is generating the residuals (detection). The second stage is evaluating the residuals in order to determine the origin of the fault (faults separation). In this way, the residuals are vectorial signals that contain information about the time and locality of the faults. In principle, the residuals should be zero in the absence of faults and, obviously, other than zero when a fault appears.

With these premises, the state observers can be used for residual generation. The main goal is to build a classical observer for the system described by equation (1), in order to produce a vector of the estimated states. The residuals are obtained by comparing the estimated output with the measured output of the physical plant.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad x(0) = x_0 \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

Where a gain matrix $D \in R^{nxq}$ exists in such a way that the estimate $\hat{x}(t)$ of the state vector x(t) will be the solution for the equation:

$$\begin{aligned} \dot{\hat{x}}(t) &= A\hat{x}(t) + Bu(t) + D(y(t) - C\hat{x}(t)), \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{2}$$

The system outputs (2) are the estimated outputs and the observer gain matrix D should be appropiately selected. If we define an error signal as:

$$e(t) = x(t) - \hat{x}(t), \tag{3}$$

which produces an innovation in the output defined by

$$\eta(t) = y(t) - \hat{y}(t), \tag{4}$$

then the error dynamics and the corresponding output error will be given as

$$\begin{aligned} e(t) &= (A - DC)e(t) + A_f f_p + B_f f_a - DC_f f_a, \\ \eta(t) &= Ce(t) + C_f f_a \end{aligned} \tag{5}$$

If D is selected so that (A-DC) is stable, all the eigenvalues will have a negative real part. When a t goes to infinite, the estimation error will become null (i.e. $e(t) = 0$). In this case, it is said that the observer is exponential or asymptotic. Since, for $t < t_0$, the process has a normal operation (i.e. faults do not exist) at that moment, the residuals or the innovation of the output is approximately zero. When any fault occurs, for $t \geq t_0$, the residual is different from zero and is favorable for fault detection [13].

*Knowledge-based methods*

Knowledge-based methods are very useful in cases where it is difficult to find an analytic model to obtain the residuals. These methods are based on existing knowledge concerning the behavior of the system outputs. This knowledge will allow us to infer the process operational conditions. One of the techniques that can be used for this is artificial neural networks [14]. The most common approach for fault detection consists of a neural network that possesses, as inputs, the variables of the process used for detection and, as outputs, a group of signals according to the operational condition. In this approach, as in most neural network applications, it is recommended that all the input signals should have been previously normalized at between 0 and 1 in order to be able to guarantee that the training

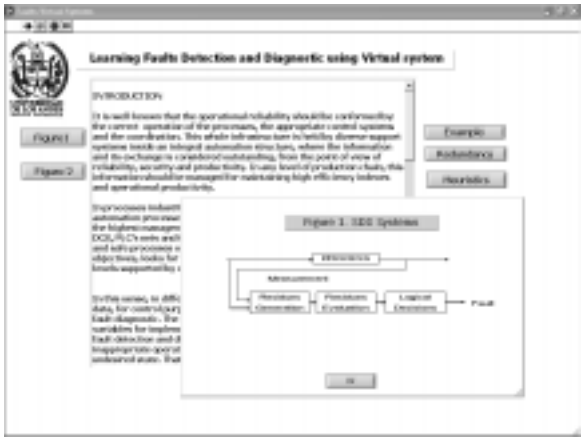

Fig. 5. Computational tool: main screen.

will not depend on the range of variability of each signal. On the other hand, each desired output should be chosen in a way that represents a particular condition of the process. For instance, 1, if the neural network output is 1, can indicate the presence of a fault condition; otherwise, the neural network output takes on the value of –1.

## DESCRIPTION OF THE COMPUTATIONAL TOOL

A tutorial-type software was developed to help in the teaching–learning process of fault detection and diagnostic methods. The computational tool developed here allows the user to interact with the process through a human–machine interface (HMI) that has been designed using LabVIEW$^{TM}$. The process is simulated in MatLab and this virtual instrument tries to emulate a control room where the operator observes the HMI of the process in order to detect the operating conditions of the system.

Table 1. Tank dimensions

| Tank | Area ($cm^2$) | Height (cm) | Initial Conditions (cm) |
|---|---|---|---|
| 1 | 2500 | 100 | 0.5 |
| 2 | 2000 | 80 | 4.5 |
| 3 | 3000 | 80 | 0.2 |

The main screen of the computational tool is shown in Fig. 5. This screen presents an introduction to fault detection and diagnostic methods. The user will be able to learn the different techniques using the connections to the redundancy and heuristic methods information. From this main screen the user will have access to the process that was chosen for reinforcing the understanding of the fault systems. All the information in the computational tool has been made in English and Spanish, in order to increase the number of student that can use it.

The application described in this work consists of a system with three interconnected tanks, as depicted in Fig. 6.

The system presented in Fig. 6 consists of three tanks, where tank 1 receives a constant flow $u(t) = 5000$ $cm^3/s$ and feeds tanks 2 and 3. It is assumed that the tanks have the dimensions shown in Table 1.

Starting with these values and considering that tank 1's flow is equal to 60% of the input flow u(t), and the flow of tank 2 is equal to 40% of the input flow u(t), the resistance values and flows of each of the four valves was obtained (see Table 2).

The possible faults that have been considered in

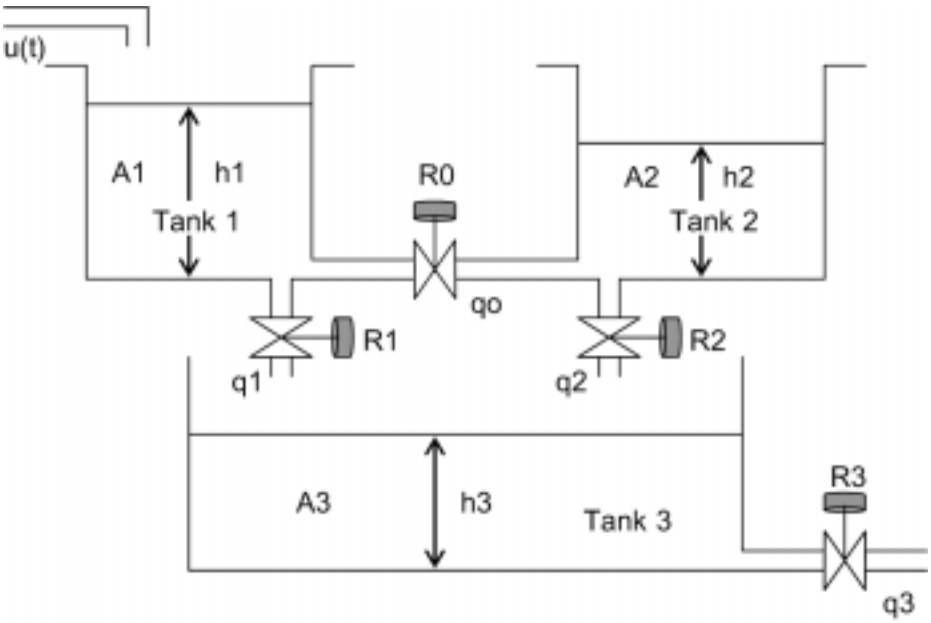


Fig. 6. System with three interconnected tanks.

Table 2. Valve resistance

| Valve | Resistance |
|---|---|
| R0 | 100 |
| R1 | 30 |
| R2 | 25 |
| R3 | 60 |

this system are blockage of any of the four valves, either individually or at the same time. The system can be represented as follows:

$$\dot{h}_1(t) = -\left(\frac{1}{r_0}+\frac{1}{r_1}\right)\left(\frac{1}{a_1}\right)h_1(t) + \left(\frac{1}{r_0 a_1}\right)h_2(t) + \left(\frac{1}{a_1}\right)u(t)$$

$$\dot{h}_2(t) = \left(\frac{1}{r_0 a_2}\right)h_1(t) - \left(\frac{1}{r_0}+\frac{1}{r_2}\right)\left(\frac{1}{a_2}\right)h_2(t)$$

$$\dot{h}_3(t) = \left(\frac{1}{r_1 a_3}\right)h_1(t) + \left(\frac{1}{r_2 a_3}\right)h_2(t) - \left(\frac{1}{r_3 a_3}\right)h_3(t)$$

where $r_i$ is the inverse of the valve's resistance (Ri) and $a_i$ is the area of tank i.

Under normal operational conditions, the four valves that comprise the system are open and it is assumed that the liquid supply is constant. Over time, the levels in the three tanks are increased, and the supply of liquid from tank 1 to tank 2 and from tanks 1 and 2 to tank 3 is not interrupted.

Figure 7 illustrates the HMI designed for this process using LabVIEW™. The user will be able to generate any of the four possible faults and to observe the behavior of the tank levels. In addition to this interface, the user will have a help file that includes the system equations and the tank level behaviors under different fault conditions for the system.

## RESULT AND DISCUSSION

The computational tool implements the designed detecting filters using the two methods introduced previously. One of the filters is based on an analytic redundancy method that calculates the residuals using a state observer. The other filter is based on a heuristic method and implemented through a neural network that was previously developed using a different operation. Both filters are simulated in MatLab and when they detect a fault they send a signal to LabVIEW™, which immediately turns on a light signal indicating that the fault has occurred. Using this system, the students can evaluate the filter operation and efficiency based on the results obtained.

In the example, LABVIEW is a DDE server and MatLab is a DDE client. For integrating MatLab with LABVIEW we used MatLab's function ddeula.m.

We have exploited the LABVIEW capabilities

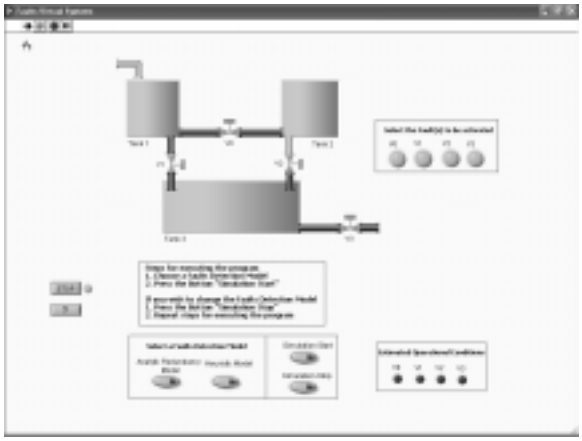

Fig. 7. Virtual instrument developed using LabVIEW™.

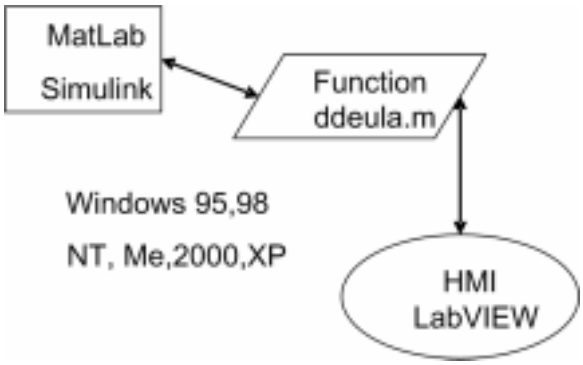


Fig. 8. Block diagram for integrating MatLab and LABVIEW.

for developing virtual instruments and creating human–machine interfaces. The simulation capabilities of MatLab together with the benefits of using the DDE protocol allowed us to create a computational tool for fault detection and diagnostics. Figure 8 shows the block diagram that implements the MatLab–LABVIEW integration.

Preliminary results show that students who have been developing topics on fault detection have used this computational tool. It has been clearly demonstrated that students have found this an effective methodology for understanding the topic of fault detection, as in just a few hours they are able to acquire the basic knowledge needed in this field and to investigate the possibility of acting on the processes that have faults. This qualifies the tool as a facilitating mechanism in the teaching–learning process.

## CONCLUSIONS

This approach to fault detection and diagnostics can be used effectively in the teaching process or for individual learning in the area, allowing on-line interaction with MatLab.

This kind of computational tool helps students to interact with industrial processes, to modify

operational conditions and to verify faults and their consequences. This helps to obtain supervision-based learning of real operational conditions. Futhermore, the use of a computer-based tutorial offers an interactive way to study fault detection and diagnostic methods based on state observers and artificial neural networks.

The use of simulator-type educational software in combination with traditional educational techniques allows the lecturer to involve students in their future work environment. Thus, the future engineer will have the opportunity, from the beginning of his/her training, to develop his/her skills in a real-world environment.

## REFERENCES

1. B. F. Skinner, *Teaching Machines* (1958), pp. 969–977.
2. J. Aguilar and F. Rivas, *Introducción a las Técnicas de Computación Inteligente*, Meritec, Venezuela (2001).
3. J. R. Carbonell, AI in CAI: An artificial intelligence approach to computer-assisted instruction, *IEEE Transactions on Man-Machine Systems* (1970), pp. 190–202.
4. A. Galvis, *Ingeniería de Software Educativo*, Ediciones Uniandes, Colombia (1992), pp. 22–23.
5. National Instruments Corporation, *LABVIEW User Manual*, USA (2000).
6. MathWorks Inc., *Matlab Language of Technical Computing*, USA (1999).
7. C. Petzold, *Programación en Windows® 95*, McGraw-Hill Interamericana de España, S.A, España (1996).
8. J. Calderón-Vielma et al., Integración de herramientas de programación para la enseñanza de procesos, XVIII Interamerican Congress of Chemical Engineering, Puerto Rico (1998).
9. J. Calderón-Vielma, Laboratorio virtual para la enseñanza de automatización e instrumentación industrial. Coloquio de Automatización y Control, Venezuela (1999).
10. A. Rios, Sur la synthèse de filtres de détection de défaillances, Ph.D. thesis, Université Paul Sabatier, Toulouse, France (2001).
11. F. Szigeti, J. Bokor, A. Edelmayer and R. Tarantino, Fault detection filter design for linear time varying systems: Algebraic-geometric, 4th European Control Conference, Belgium (1997).
12. L. Keviczky, J. Bokor, A. Edelmeyer and F. Szigeti, Detection filter design in system perturbation application to robust change detection and identification, *Proceedings of the 12th IFAC World Congress*, vol 7, Australia (1992), pp. 517–520.
13. M. A. Massoumnia, A geometric approach to the synthesis of failure detection filters, IEEE Trans. Aut. Control, AC-31.9 (1986), pp. 839–846.
14. A. Rios, G. Mousalli and F. Rivas, Invertibility and neural networks based FDI filter, IASTED-ISC, Japan (2002).

**Gloria Mousalli-Kayat** is a Professor in the Measurement and Evaluation Department at the Universidad de Los Andes, Venezuela. She teaches Statistics and Computer Science in the Faculty of Education undergraduate and postgraduate school. Her M.Sc. (2002) is in control and automation engineering. She has developed intelligent tools for the Venezuelan Oil Company and medical applications. She received the prize for very young researchers awarded by the Venezuelan Scientific and Technological Research Academy (PPI, 2003).

**Jesús Calderón-Vielma** is Professor of the Circuits and Measurement Department at the Universidad de Los Andes –Venezuela. He teaches Control Applications and Virtual Instrumentation in the Automation and Instrumentation Engineering undergraduate and postgraduate school. His M.Sc. (2002) was in automation approaches. He was head of the Automation and Instrumentation Laboratory of the Universidad de Los Andes. He has developed virtual intrument applications for education. He is the ISA Faculty Advisor in the Universidad de Los Andes. He was awarded the prize given by the Venezuelan Scientific and Technological Research Academy (PPI).

**Addison Rios-Bolivar** is Associate Professor of the Control Systems Department at the Universidad de Los Andes –Venezuela. He teaches Robust Control and Process Control in the Control Systems Engineering undergraduate and postgraduate school. His M.Sc. (1994, Universidad de Los Andes) and Ph.D. (2001, Universitè Paul Sabatier, France) were both in the field of fault detection and diagnosis in dynamic systems using robust approaches. He is head of the Control System Department of the Universidad de Los Andes. He has developed fault detection tools for the Venezuelan Oil Company. He has written more than 40 publications in the area of process control and fault detection. He has received a number of prizes given by the Venezuelan Scientific and Technological Research Academy (PPI), the Universidad de Los Andes (PEI) and the Venezuelan Council for Higher Education Development (CONADES).

**Francklin Rivas-Echeverría** is Professor of the Control Systems Department at the Universidad de Los Andes, Venezuela, and is the Control and Automation Engineering postgraduate school head and Director of the Intelligent Systems Laboratory. He teaches Intelligent Control in the Control Systems Engineering undergraduate and postgraduate school. His M.Sc. and Ph.D. thesis were both in the field of intelligent systems. He has developed intelligent tools for several Venezuelan companies and has written more than 70 publications in the area of intelligent systems. He has received a number of prizes awarded by the Venezuelan Scientific and Technological Research Academy (PPI), the Universidad de Los Andes (PEI) and the Venezuelan Council for High Education Development (CONADES).