SIMULATING COMPLEX SYSTEMS IN LABVIEW

György Lipovszki
Department of Production Informatics Management and Control
Budapest University of Technology and Economics
H-1111. Budapest, Műegyetem rkp. 3-9 D. 428., HUNGARY
E-mail: lipovszki@rit.bme.hu

KEYWORDS

dynamical systems, mathematical modeling, other computational methods

ABSTRACT

Modeling and simulation of systems, especially in science and engineering can help to reduce risk and cost of design and testing processes. According to Cellier, the established mathematical models can be classified as follows: continuous time, discrete time, quantitative models and discrete event models.

A huge number of simulation software has been developed to support modeling and simulation efforts. All of these software tools support the use of one or more mathematical model classes. Despite all of these efforts, it is hard to find simulation software, which is capable of combining several model classes in a real industry standard environment. The paper presents a series of simulation software products, which have been developed using an industry standard programming environment widely applied to data acquisition, process control and data visualization: National Instruments' LabVIEW.

The first development was the TUBSIM, a continuous time simulation toolbox and it was followed by the discrete event extension called Discrete Event Simulator (DES). The elements of the toolboxes facilitate block oriented modeling using LabVIEW's high level graphical editor and calculating power of. Both LabVIEW-based simulation libraries are widely used in education and research. During the years several additional modules were and are still developed, e.g., fuzzy rule-based systems, optimization using genetic algorithm, compartment modeling systems for pharmacodynamical and pharmacokinetical applications, etc.

Applying these toolboxes one can model and simulate complex systems where continuous and discrete event driven parts are working together. The continuous part can insert events into the discrete event task list on the fulfillment of different continuous state variable conditions. The discrete event system can also change the value of state variables, together with input values in the continuous system.

INTRODUCTION

One of the most important applications of computers is imitating, or *simulating,* the operation of various kinds of real world facilities and processes. The facility or process of interest is usually called a *system*, and in order to study it scientifically we often have to make a set of assumptions about how it works. The assumptions, which usually take the form of mathematical or logical relationships, constitute a *model* that is used for trying to gain some understanding of how the corresponding system behaves.

If the relationships that compose the model are simple enough, it may be possible to use mathematical methods (such as algebra, calculus, or probability theory) to obtain *exact* information on the question of interest; this is called an *analytic* solution. However, most real-world systems are too complex to allow realistic models to be evaluated analytically, and these models must be studied by means of simulation. In a *simulation* we use a computer to evaluate a model *numerically*, and data are gathered in order to *estimate* the desired true characteristics of the model. Application areas for simulation are numerous and diverse. Below is a list of the most important types of simulation areas:

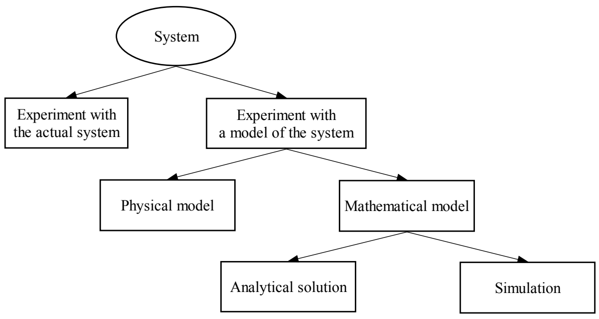
- Continuous systems simulation formulated by differential equations
- Stochastic discrete event driven systems
- Complex systems, mixture of previous types of systems

A *continuous system* is one in which the state variables change continuously with respect to time. A forklift-moving trough the workshop is an example of a continuous system, since state variables such as position and velocity can change continuously with respect to time.

A *discrete event driven system* is one in which the state variables change instantaneously at separated points in time. A post office is an example of a discrete system, since state variables – the number of people in the post office – change only when a new person arrives or when a person departs after being served. Few systems in practice are wholly discrete or wholly continuous, but one type of change predominates for most systems.

A *complex system* where the state variables can create events in the system when a given condition became valid and like a reverse action an event also can cause changing of system input value or a state variable in the continuous part of the model.

At some point in the lifecycle of most systems, there is a need to study them, to try to gain some insight into the relationships among various components, or to predict performance under some new condition. Figures 1 maps different ways in which a system might be studied.


System
Experiment with the actual system
Experiment with a model of the system
Physical model
Mathematical model
Analytical solution
Simulation

Figures 1 Ways to study a system

CONTINUOUS-TIME SIMULATION IN LABVIEW

Nowadays, computer-based materials are increasingly used in education. A number of interactive, computer-based course materials in various disciplines and subjects, from all over the world are available through the Internet. These courseware materials have diverse target audiences both in level and type of students, e.g., from primary school to university, from regular weekly lectures to distant-learning.

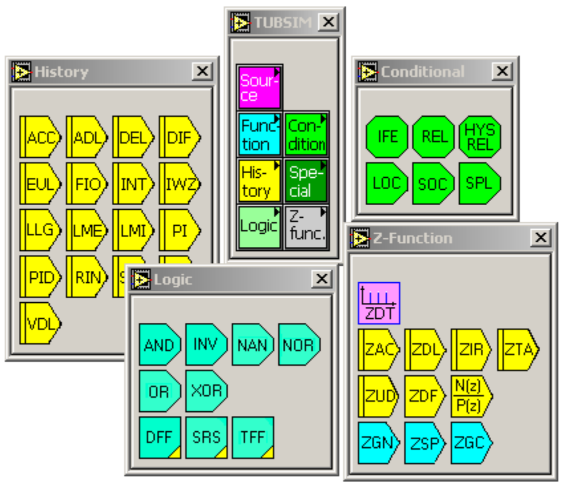
Systems engineering courses are taught prior to Control Engineering courses at BUTE, to introduce modeling and simulation theory, methods and applications, not only for control, but for other disciplines as well. Modeling and simulation theories, methods and techniques – parallel with and serving the needs of systems and control engineering – have undergone a significant development during the last century. Measurement technology achieved important milestones too. Up-to-date instruments and principles, accurate and fast measurements characterize today's measurement technology. Modern computing resources increase data management speed and capacity. Scientific results of the above-mentioned fields are joined together in interdisciplinary applications. The most recent information should be used in education, while at the same time the basics have to be taught, too.

Keeping the "old" knowledge and including the new improvements poses a problem for educators. They have to find the balance between the amount and quality of information relayed to students. In systems and control engineering frequency-domain techniques are still taught, however their importance have been reduced by the improvement in computing and time-domain simulation. They are however necessary, because of the insight and handy techniques they give to students. At the same time, modern modeling and control knowledge, such as soft computing, adaptive and optimal control has to be included in the curriculum. To satisfy both tasks a number of programs have been developed to help visualize the conventional information (text, diagrams, drawings, and equations) in textbooks and lecture notes. These programs could easily be considered as computer games. It is widely proven that learning-by-doing is a leader among the approaches of learning. Playing with these simulations radically improves the understanding and future application in real world situations.

The author in LabVIEW environment has developed the continuous simulation packages named TUBSIM. The package was initially developed according to the CSSL (Continuous System Simulation Language) recommendations.

TUBSIM for LabVIEW

TUBSIM is the interpretation of an analogue computer in the LabVIEW graphical programming environment. Analogue computers were widely used for simulation, but they had many disadvantages (e.g., the size of the computer grows with the size of the model).



Figures 2 TUBSIM VI icons in LabVIEW

When digital computers became universally available, analogue computers and analogue simulation were soon replaced with digital simulation. One group of the digital simulation systems uses the same principles as analogue simulation does. TUBSIM belongs to this group. The TUBSIM VI (icons) Library (Figures 2) contains the basic blocks typical to analogue computers (summers, integrators, potentiometers, signal generators). In addition, TUBSIM has different Boolean blocks, typical systems engineering elements (for example first order element, continuous time controllers – like PI, PID –, time delay block) and sampled time blocks. TUBSIM has successfully been used in various applications from teaching aids to large-scale industrial processes (e.g., the Secondary Side Water Chemistry model of the Nuclear Power Plant in Paks, Hungary).

Continuous Simulation Applications

A continuously increasing number of TUBSIM simulation applications are used in both introductory and advanced systems and control engineering courses at BUTE. There are other advanced studies (e.g. Computer Controlled Systems, MSc and PhD theses) too, which utilize similar applications.

State-space models are particularly hard to solve with conventional methods, especially when non-linearity and time-delays are involved.

An example of such a complex non-linear system from the area of biomedical engineering is the compartment model of entherohepatic circulation. Biomedical simulations are especially useful when "control systems" have to be designed to compensate the effects of illnesses. One such example is the development of medication regimes for patients with diabetes.

Simulation is a very appropriate way for presenting and comparing different types of controllers. The application of knowledge learnt in basic control theory courses are best tested with simulation programs. Currently the speed of LabVIEW applications, especially the ones with a large need of run-time calculations are considered. When for example a fuzzy controller has to be optimized with a genetic algorithm, DLLs (dynamic link libraries) are used to speed up the calculation.

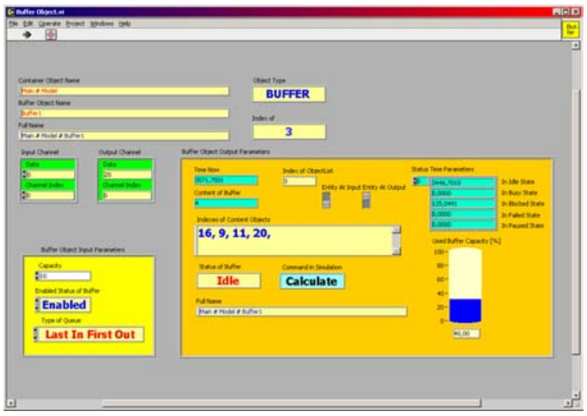
The advantage of developing native LabVIEW programs lies in the ease and speed of development; however the speed of calculation is best boosted with the use of external function calls from a DLL. DLLs open the world of object-oriented programming to LabVIEW, as well.

DISCRETE EVENT SIMULATION IN LABVIEW

The latest application library for LabVIEW is the Discrete Event Simulator (DES) package. This package contains approximately 140 elements, with LabVIEW terminology VIs (virtual instruments). Objects in the DES package for building complex logistic systems are: CONTAINER, ENTITY, SOURCE, BUFFER, MACHINE, SINK, JOIN, SELECTOR, PACK, UNPACK. Beside the main objects, there are so-called work procedures to create and destroy entities, to calculate different distribution functions, to set and get an object's attributes and to manage the task list – the heart – of the discrete event simulation system. The Discrete Event Simulator package is used to give an interactive experimental environment to study processes with uncertainties.

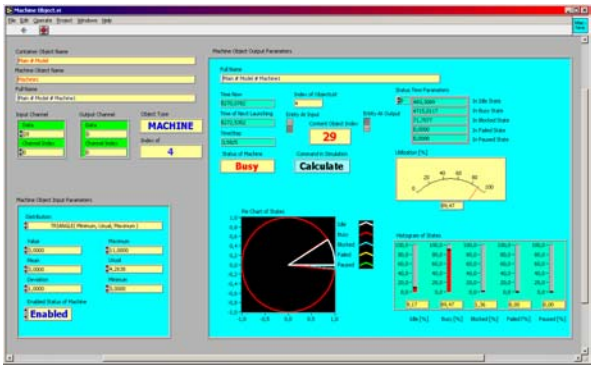
Attributes of existing objects in the DES package are the follows:

- CONTAINER: (parent object) this object allows the development of DES subroutines, which can be copied and renamed, so they can be applied with the same functionality and different inputs.
- ENTITY: this is the "working object" in the simulation system. It could represent information or material flow in the simulation model. It has any number of attributes represented by numerical or string values.
- SOURCE: this object creates new entities. The time duration between the produced entities is given by different types of distributions (Constant, Exponential, Normal, Triangular, Uniform and User Defined).



Figures 3 Front panel of Buffer Object

- BUFFER: this object temporarily stores entities, till the material or information flow of the object connected after the buffer becomes able to send or receive the outgoing entity (Figures 3).



Figures 4 Front panel of Machine Object

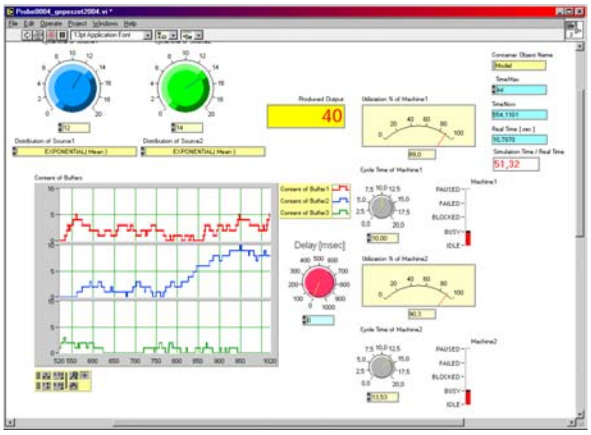
- MACHINE: this object delays the flow of entities with a given type of distribution (Constant, Exponential, Normal, Triangular, Uniform and User defined) (Figures 4).
- SINK: this object destroys "used" entity objects and frees the used part of memory.

- JOIN: this object has more than one input channels and has only one output channel. One of the input parameters the "Join Input Channel Index," is for selecting one of the input channels with a given strategy. In the next event the waiting entity from this channel is "read", and sends away into the information or material flow.
- SELECTOR: this object has one input channel and more than one output channels. With one of the input parameters the "Selector-Channel-Index," the output channel is selected with a given strategy and an entity (if exists) is sent out on this channel.
- PACK: with this object one or more entities can be packed into another entity. The Pack object has two inputs and one output. The first input channel receives the objects to be packed the second input channel receives a package entity. In a packaging process, the package arrives first and waits inside the Pack object till the required amount of entities arrive into the package object. When the process has finished only one entity is sent away, but it contains a given number of other entities.
- UNPACK: with this object we can unpack one or more entities from an entity (package entity). The Unpack object has one input and two outputs. The first output channel sends away the unpacked objects; the second output channel is used to send out the package object. The unpack process starts with the arrival of a package object (entity). After the package has arrived, the Unpack object unpacks its content into an inner buffer and sends them out over the first output channel as soon as possible.

Beside the main objects there are so called *work procedures* to create and destroy entities, to calculate different distribution functions, to set and get an object's attributes and to manage the *task list* – the heart – of the discrete event simulation system. Using this task list the simulator establishes a *next-event time advance* mechanism that always gets the most imminent future event and copies this event's time part into the simulation system clock. At this time (at the occurrence of the most imminent event), the state of the system is updated, and future event time(s) are determined. The process of advancing the simulation clock from one event time to another is continued and eventually finishes as some specified stopping condition is satisfied.

Discrete Event Simulation Application

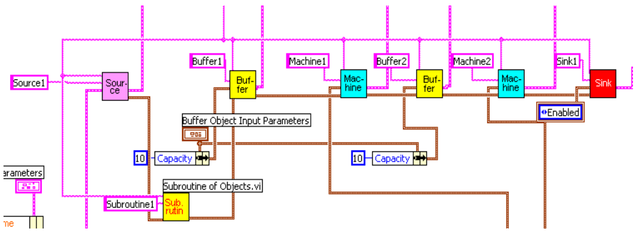
To introduce the possibilities of the DES package, a simple manufacturing problem is solved, where the material flow goes through buffers,



Figures 5 Control surface (Front panel) of DES simulation system

machines and a subroutine that also contains basic DES objects. The control surface (Figures 5) elements (fields, buttons, meters and displays) are changing the parameters and analyzing results. Graphical programming helps us in semantic control too (easy search of lost connecting lines) and in the reorganization of the logistic net.

The Discrete Event Simulator package is able to use any number of continuous simulation systems and to calculate them, because the continuous system clock could send events into task list.



Figures 6 Program surface (Diagram panel) of DES simulation system

Figures 6 shows a simple discrete event driven system architecture in the LabVIEW based graphical environment. Icons of the basic objects and commands can easily be connected together with the color identified "wires". This wires mean different data structures (record of different data types).

Complex Simulation Applications

Complex simulation – discrete event and continuous simulation together in the same system model – is becoming more and more important in the education and different industrial applications. The continuous system at every calculation (sampling) time inserts a new "calculation request" into the discrete event simulator's task list, that is how more than one continuous simulation subsystems can run together with different time steps inside the same simulation system. This is very important because each subsystem in the simulation model requires a different time step in order to optimize calculation time of the simulation. From view of discrete event simulation, every discrete event simulation system can contain special conditions, which need to be calculated by using continuous simulation sub-models. These complex systems cannot be solved analytically because the stochastic behavior of discrete events and the different nonlinearities of continuous subsystems.

Let us look at a special queuing system of airplanes waiting to land at an airport. It is a retrying system; airplanes retry to get landing permission until it is granted. The overall picture of the system behavior will be highly influenced by numerous small continuous subsystems, which are operating on the airplanes and/or strongly connected to their operation. All of these continuous subsystems can influence the new order of the waiting airplanes. The problem can be modeled as a complex (discrete-continuous) simulation system with dynamic priorities in the waiting lines.

Some modeling considerations are listed as follows:

- One of the most important conditions is the actual content of the fuel tank of the planes, which would make the waiting possible. The fuel consumption of a plane is a typical continuous simulation subsystem function. Under a minimal level of the fuel tank responsible landing is impossible and a queue reorganization request has to appear promptly in the task list of the simulation, otherwise the plane crashes.
- The number of the people on the planes is another important aspect, in other words how many lives are endangered because the landing queue calculation did not use this data. This data (number of people) is an important weighting factor in recalculating the landing queue.
- Required landing circumstances are an important condition for each type of planes. Huge jets can land without disaster only if they use the regular runway; however small airplanes are able to land on a highway or a field if necessary.
- The state of the health of the people on the plane is also a determinative aspect at the calculation of the landing queue. A special event, for example a heart attack, a pregnant woman going into labor, urgent need of a certain medical intervention or medicine could also reorganize the landing queue.

CONCLUDING REMARKS

This paper reviewed the use of continuous and discrete event simulation as teaching aid for systems and control engineering education. LabVIEW is the mostly used programming environment for these applications, with simulation packages – TUBSIM and DES – developed by the authors of this paper. Current research and development work includes the development and integration of special purpose DLLs for fuzzy logic, neural network and genetic algorithm applications. The Discrete Event Simulation package and any one of the continuous time packages can be combined to handle continuous-discrete hybrid systems.

REFERENCES

1. Åström, K.J., Wittenmark B. (1997) *Computer-Controlled Systems*, Prentice-Hall
2. Bequette, B.W. (1998) *Process Dynamics*, Prentice Hall
3. Dorf, R.C., Bishop, R.H. (1998) *Modern Control Systems*, Addison Wesley Longman
4. Gordon, G. (1969) *System Simulation*, Prentice Hall
5. Hartley, T.T., Beale, G.O., Chicatelli, S.P. (1994) *Digital Simulation of Dynamic Systems – A Control Theory Approach*, Prentice Hall
6. Kheir, N.A. (editor) (1995) *Systems Modeling and Computer Simulation*, Marcel Dekker, Inc.
7. Law, A.M., Kelton W.D. (1991) *Simulation Modeling & Analysis*, McGraw-Hill
8. Man, KF, Tang, KS and Kwong, S. (1999) *Genetic Algorithms*, Springer
9. Monsef, Y. (1997) *Modelling and Simulation of Complex Systems. Concepts, Methods and Tools*, Society for Computer Simulation International
10. Wells, L.K.; Travis J. (1995) *LabVIEW for EveryOne – Graphical Programming Made Even Easier*, Prentice Hall
11. Zeigler, B.P., Praehofer, H., Kim T.G. (2000) *Theory of Modeling and Simulation*, Academic Press
12. Pidd, M, 1992, *Computer Modeling for Discrete Simulation*, Chichester, England, John Wiley

AUTHOR BIOGRAPHY

GYÖRGY LIPOVSZKI was born in Miskolc, Hungary and went to the Budapest University of Technology and Economics, where he studied electronics and graduated in 1975. He is now Associate Professor at the Department of Department of Production Informatics Management and Control and his research field is development of simulation frame systems. in different programming languages.
His e-mail address is: `lipovszki@rit.bme.hu`
.