.:PSIM:. – A LABVIEW-BASED SIMULATION SYSTEM AS A LEARNING AID

Petra Aradi
Department of Informatics, Faculty of Mechanical Engineering
Budapest University of Technology and Economics
Műegyetem rkp. 3., Budapest, H-1111, Hungary
e-mail: petra@rit.bme.hu

KEYWORDS
continuous simulation, education, tool, LabVIEW

ABSTRACT

.:PSim:. a LabVIEW-based simulation system suitable for both educational and industrial purposes is presented. Educational aspects of .:PSim:. are emphasized below, especially in areas of systems and control engineering as taught in various courses at Budapest University of Technology and Economics (BUTE). .:PSim:. is not just another simulation tool in LabVIEW, it is a set of building blocks like LEGO that can be combined into various constructions and can further be extended with new elements.

INTRODUCTION

.:PSim:. started as a collection of LabVIEW VIs developed by the author for simulating continuous time processes, according to the CSSL (Continuous System Simulation Language) recommendations. Later on the traditional time-domain simulation was extended with frequency domain methods. After implementing these "traditional" techniques, soft computing methods, specifically fuzzy systems and neural networks were added to handle complex non-linear models or models based on measured data. Bondgraph models, discrete-time systems, identification, stability analysis and compartment models were the latest enhancements in .:PSim:.
The main goal while developing .:PSim:. was to create an almost general-purpose simulation tool that is suitable both for educational and industrial purposes. .:PSim:. had to retain the characteristics of CSSL whilst including the emerging techniques like Soft Computing. As educational applications were the first concern an easy to use programming environment had to be used which is suitable both for illustration during lectures and for student projects as well.

WHY LABVIEW?

LabVIEW from National Instruments was chosen as the programming environment for .:PSim:. LabVIEW stands for Laboratory Virtual Instrumentation Engineering Workbench, and is available in various computer platforms, such as Linux, MacOS, Windows, Solaris, HP-Unix. LabVIEW applications are called virtual instruments or VIs for short. VIs have a user interface and a block diagram, where the actual program is built with LabVIEW's graphical components (Fig. 1). The ease of use and programming, together with LabVIEW's excellent connection to the outside world through data-acquisition, I/O and network protocols makes it the ideal tool for scientists and engineers.
LabVIEW has quite a number of front panel elements that facilitate the quick and easy fabrication of instrument-like user interfaces, naturally leaving the opportunity to create customized elements like buttons and displays. A similarly huge amount of functions – mathematical, I/O, etc. – are also readily available.

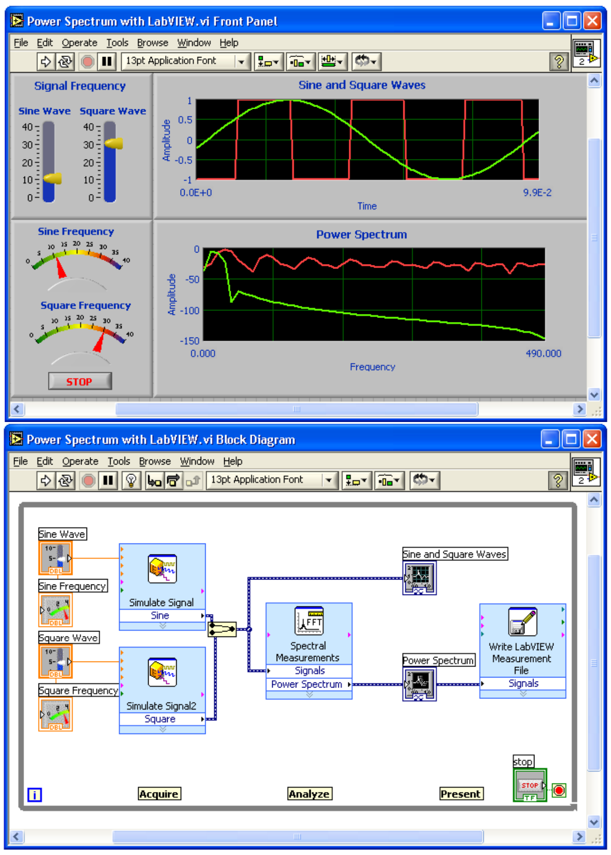


Fig. 1: Front Panel and Diagram Panel in LabVIEW

Years of experience in LabVIEW programming suggested its use as a quick and easy tool to produce spectacular and effective simulation programs to illustrate lectures even on the fly, during a lecture. Students also seem to be interested in just watching the birth of a simulation program with the rather advanced graphical programming capability, they normally do not learn to use. Even using LabVIEW's built-in functionality without any add-ons impresses them, especially when the computer is connected to a real world system, either just to measure it or even to control it.
According to the author's observations students with no previous experience in programming – not even text-oriented languages – could grasp the basics of algorithmic thinking and programming structures after four or five lectures followed by practice at a computer. Afterwards they are capable of solving simple programming and – what is more important – simulation problems with LabVIEW. They really enjoy the power of being able to tell the computer what to do, how to help them solve their tasks.
This explains why LabVIEW was chosen.

.:PSIM:. FOR LABVIEW

Numerical Integration

As mentioned above, .:PSim:. started as a CSSL implementation in LabVIEW with the characteristic blocks of the digital implementation of an analog computer. The most important of these blocks are integrators with four numerical integration methods implemented. It is of course possible to use other formulas as well, however it was so far unnecessary to use higher than second order integrators (e.g. Adams-Basforth) in educational applications. Numerical integration can be accomplished with fixed time-step or with variable time-step. To be in synch with the mathematical knowledge of undergraduate students fixed time-step methods are used in educational applications.

Beyond CSSL

It very soon became evident, that systems and control engineering courses are yearning for more powerful simulation methods than block oriented simulation with CSSL elements. That is why state space models and basic transfer blocks like proportional, integrator, lead-lag, PID, etc. were also implemented as sub-VIs. Adding these sub-VIs to the block diagram of a LabVIEW program rather complex systems may be built. It is possible to combine all the above mentioned system models within the same application.
As stability is an existential issue in the analysis of dynamic systems, various stability criteria (like Routh's table and Hurwitz's determinant) are also included in .:PSim:.

Furthermore, as frequency domain methods still have their significance in system analysis, such tools like Bode and Nyquist diagrams were also implemented. As the classical control theory states rather simple approximate schemes to connect frequency and time domain properties to establish the quality of control loops, methods like the Nyquist stability criterions were implemented accordingly.
Mostly, but not exclusively complex system models require the use of logical (Boolean) and non-linear functions (e.g. hysteresis, relay).
Digital controllers can also be modeled with the use of the discrete-time VIs.
Additional blocks are fuzzy rule-based systems that can be used as controllers, neural networks that can be trained to mimic a modeled system, bondgraph elements and a simple discrete event system.
Fig. 2 shows the previously mentioned .:PSim:. function libraries with additional VIs for file operations, conversion among mathematical models such as differential equation, state-space model, zero-pole-gain representation, as well as the series, parallel and feedback connections blocks.

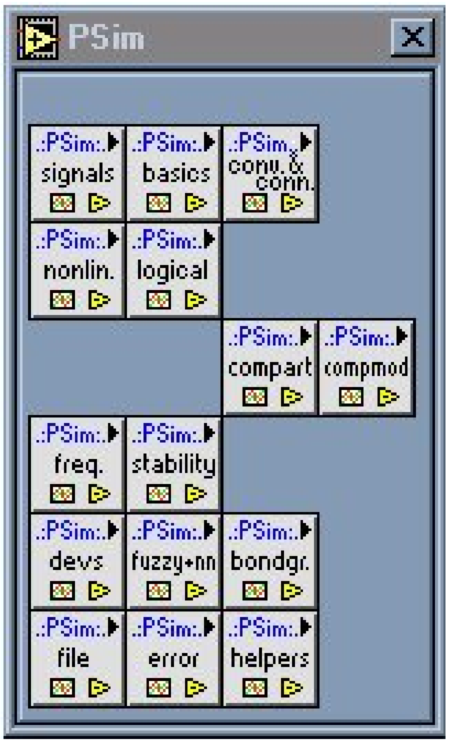


Fig. 2: .:PSim:. Function Libraries

.:PSim:. Compartment

The most recent additions to .:PSim:. are compartment models that are widely used in pharmacological and physiological modeling. Although compartment models are just a special kind of state-space models, it seemed important to further accentuate the versatility of LabVIEW to practitioners of other, non-engineering disciplines.
There are two compartment libraries available. Compartment model VIs (Fig. 3) are conventional

structures, that are widely used in pharmacokinetics (to investigate how medicines travel and transform in the body from intake to exit). These VIs are very easy to use with just a minimal LabVIEW expertise. Parameters according to the compartment structure represented in the given VI can be set, results both in graphical and numerical format can be studied and stored in files for further processing.

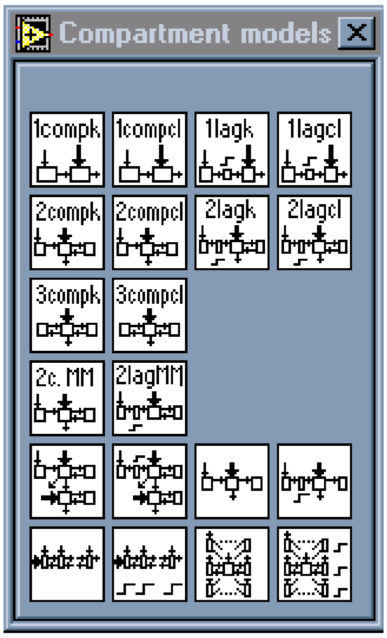


Fig. 3: .:PSim:.Compartment Model VIs

Fig. 4 illustrates the other library of compartment blocks. These VIs are general-purpose building blocks, aiming users with a more advanced LabVIEW knowledge.

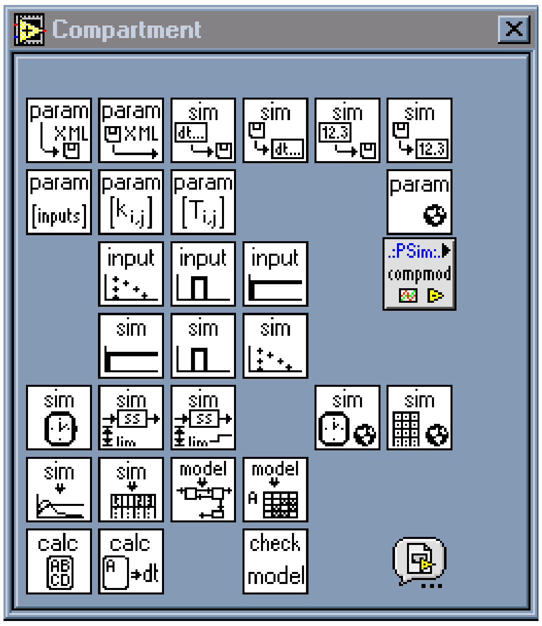


Fig. 4: .:PSim:.Compartment VIs for general-purpose models

XML model description language

To facilitate data manipulation, an XML-based compartment description language was developed. XML stands for Extended Markup Language, and is widely used to create custom markup languages for various purposes. XML data files are standard text files enhanced with the markup tags to group and identify parts of the data. XML document can be effortlessly read both by humans and computers, making it possible to modify the data directly with a simple text editor. Storing data with comments increase the efficiency of data processing and retrieval.
Compartment model and simulation parameters are stored in ComPSim-XML files that are based upon the XML structure definition (ComPSim-XSD the corresponding XML Schema Document is shown in Fig. 5).

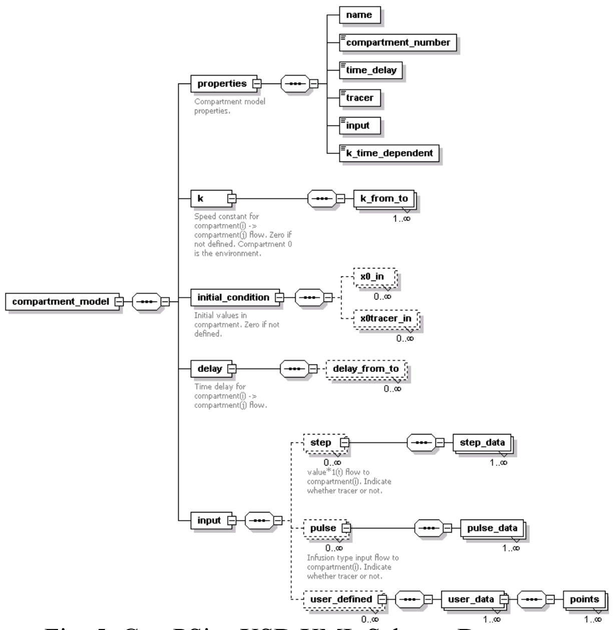

compartment_model
properties
Compartment model properties.
name
compartment_number
time_delay
tracer
input
k_time_dependent
k
Speed constant for compartment(i) -> compartment(j) flow. Zero if not defined. Compartment 0 is the environment.
k_from_to
1..∞
initial_condition
Initial values in compartment. Zero if not defined.
x0_in
0..∞
x0tracer_in
0..∞
delay
Time delay for compartment(i) -> compartment(j) flow.
delay_from_to
0..∞
input
step
0..∞
value*1(t) flow to compartment(i). Indicate whether tracer or not.
step_data
1..∞
pulse
0..∞
Infusion type input flow to compartment(i). Indicate whether tracer or not.
pulse_data
1..∞
user_defined
0..∞
user_data
1..∞
points
1..∞

Fig. 5: ComPSim-XSD XML Schema Document

There is an XML-based general purpose modeling language under development, based on the above mentioned compartment description language.

Source Code

So far .:PSim:. has been developed entirely in LabVIEW that means there are now platform dependent elements in it. There is not one DLL included, nor is there a single code interface node (CIN). (CIN calls code written in a text-based programming language, such as C, directly from a LabVIEW block diagram).

SIMULATIONS AS LECTURE AIDS

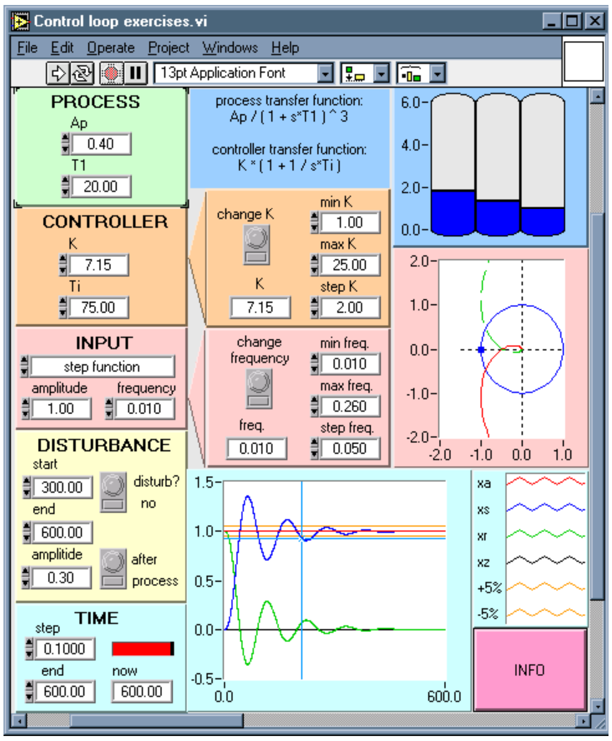
It is not just the author's experience that most engineering subjects are taught more straightforwardly when the lecturer utilizes the fruits of computer simulation and multimedia. The best possible way to show dynamical behavior of a technical system would be to have the system readily at hand in the classroom. However only a very small minority of real-world systems can fit in a classroom. That is where photos, schematics, and even more videos and simulation programs come up front.

As soon as the dynamic systems behave as in real life and give the appropriate response to stimuli on-line, then the style of a lecture is changed revolutionarily. Instead of presenting the theory in a rather dry fashion, the background and the inner workings of the system becomes alive.

Furthermore, simulations can be provided to aid the students' work at home, to help them deepen their understanding of the – for them sometimes really arcane – processes. In essence they can "play around" with the process, make experiments without the danger of making something irreversibly wrong. They harness the power of simulation to the effect of reducing risks, time and expenses by operating the system model.

.:PSIM:. APPLICATIONS

The main application area of .:PSim:. is in systems and control engineering courses at BUTE. One characteristic example of the very first programs is a simulated process with a controller (Fig. 6) that groups of 2-3 students have to use to prove their ability to tune controllers according to certain criteria (stability, speed, precision). There exist a real-world version of the illustrated three-tank water-level control loop, so the simulations' results could be compared to the process' actual responses.



Control loop exercises.vi
File Edit Operate Project Windows Help
13pt Application Font
xa
xs
xr
xz
+5%
-5%
INFO

PROCESS
Ap 0.40
T1 20.00

process transfer function:
Ap / (1 + s*T1) ^ 3

controller transfer function:
K * (1 + 1 / s*Ti)

CONTROLLER
K 7.15
Ti 75.00

change K
K 7.15
min K 1.00
max K 25.00
step K 2.00

INPUT
step function
amplitude 1.00
frequency 0.010

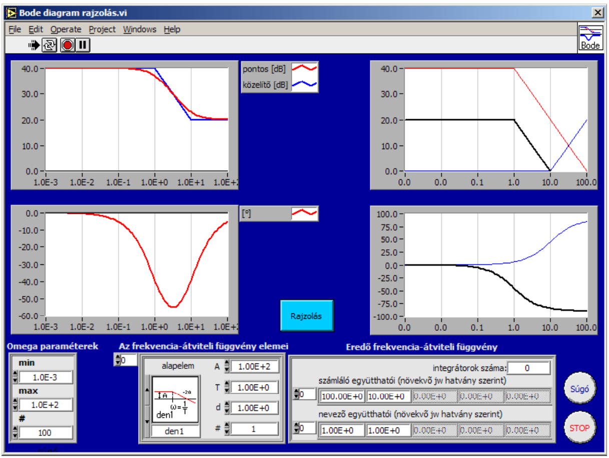
change frequency
freq. 0.010
min freq. 0.010
max freq. 0.260
step freq. 0.050

DISTURBANCE
start 300.00
disturb? no
end 600.00
amplitide 0.30
after process

TIME
step 0.1000
end 600.00
now 600.00

Fig. 6: .:PSim:. control engineering course example

Newer additions are .:PSim:. programs used to illustrate lectures, for example to show the connection between time and frequency domain, or to introduce and compare controller-tuning methods. Quite a large number of sample applications have been (and are continuously) developed to help students understand the theories. These samples are open to download from the department's web server with the necessary run-time application, so that they could be utilized without the LabVIEW development system. One such example aims to improve students' skill in sketching approximate Bode diagrams of transfer elements connected in series and comparing them to the exact diagram (Fig. 7).



Omega paraméterek
min 1.0E-3
max 1.0E+2
100

Az frekvencia-átviteli függvény elemei
0
alapelem den1
A 1.00E+2
T 1.00E+0
d 1.00E+0
1

Eredő frekvencia-átviteli függvény
integrátorok száma: 0
számláló együtthatói (növekvő jw hatvány szerint)
0 | 100.00E+0 | 10.00E+0 | 0.00E+0 | 0.00E+0 | 0.00E+0
nevező együtthatói (növekvő jw hatvány szerint)
0 | 1.00E+0 | 1.00E+0 | 0.00E+0 | 0.00E+0 | 0.00E+0

Fig. 7: Sketching an Approximate Bode Diagram

CONCLUSION

.:PSim:. proved many times to be a powerful tool in education. .:PSim:. simulations have found quite a number of applications in systems and control engineering courses at BUTE. For one they serve as standalone practice assignments to test and enhance students' knowledge in the laboratory. They are used as an illustration prepared to help visualize real world dynamic processes in lectures. As a hands on tool .:PSim:. is used to solve simulation tasks that show up during the lecture. Furthermore students can use .:PSim:. blocks to build LabVIEW simulation programs of their own.

It is the nature of every program that its development never ends, so .:PSim:. stands before further enhancements, such as the general-purpose XML-based language mentioned above.

REFERENCES

Bronzino, J.D. (Editor-in-Chief). 1995. *The Biomedical Engineering Handbook*. CRC Press

D'Argenio, D.Z., Schumitzky, A. 1997. *ADAPT II User's Guide: Pharmacokinetic/Pharmacodynamic Systems Analysis Software*. Biomedical Simulations Resource, Los Angeles

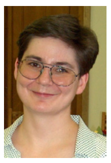
Dorf, R.C., Bishop, R.H. 1998. *Modern Control Systems*. Addison Wesley Longman

Kheir, N.A. (editor). 1995. *Systems Modeling and Computer Simulation.* Marcel Dekker, Inc.

Wells, L.K.; Travis J. 1997. *LabVIEW for EveryOne – Graphical Programming Made Even Easier.* Prentice Hall

Zeigler, B.P., Praehofer, H., Kim T.G. 2000. *Theory of Modeling and Simulation.* Academic Press

AUTHOR BIOGRAPHY



PETRA ARADI received her MSc and PhD in Mechanical Engineering at BUTE, in 1994 and 2000 respectively. She also obtained an MSc in Biomedical Engineering (BUTE, 2002). Since 1994 she works at the Faculty of Mechanical Engineering of BUTE, presently as associate professor in the Department of Informatics.

Her teaching areas are systems and control engineering, as well as microcontroller applications, PLCs and Internet programming.

Her research interests cover these teaching areas, with the recent addition of co-operative mobile robotics.