PowerDEVS

Version 2.2

User's Guide

Ernesto Kofman and Federico Bergero

CIFASIS – CONICET
FCEIA, UNR, Argentina

2012

Contents

Chapter 1

Introduction to PowerDEVS

1.1 Features of PowerDEVS

PowerDEVS is an integrated tool modeling and simulation based on the Discrete Event System Specification (DEVS) formalism.

In spite of being a general purpose DEVS simulation tool, PowerDEVS is well suited for modeling and simulation of continuous and hybrid systems through the usage of Quantized State System (QSS) algorithms.

The main features of PowerDEVS are described below:

- Models can be built as hierarchical Block Diagrams using the *Model Editor* GUI.

- Atomic blocks can be taken from existing libraries and models or can be created using PowerDEVS *Atomic Editor* tool or any text editor.

- PowerDEVS libraries contain blocks for modeling continuous, hybrid and discrete time systems. For modeling classic discrete event systems, a Petri Net library is also provided.

- PowerDEVS can use Scilab as Workspace for defining model parameters, performing result analysis, etc.

- PowerDEVS can run under Linux–RTAI Operating System, allowing precise time synchronization, hardware interrupts access and different features suitable for PC based automatic control systems.

- PowerDEVS engine and models are programmed in C++.

- PowerDEVS is an Open Source multiplatform tool under General Public License (GPL).

- Source code as well as Windows and Linux binaries are available at SourceForge: `http://sourceforge.net/projects/powerdevs/`

1.2 PowerDEVS Architecture

PowerDEVS is composed by several independent programs:

- The *Model Editor*, which contains the main graphical user interface allowing the hierarchical block-diagram construction, library managing, parameter selection and other high level definitions as well as providing the linking with the other programs.
- The *Atomic Editor*, which permits editing PowerDEVS atomic models for elementary blocks by defining the corresponding DEVS transition, output and time advance functions using C++ language.
- The *Preprocessor*, which automatically translates the model editor files into C++ code to build up the simulation code. It also links the C++ code of the different atomic models and compiles it together with the C++ PowerDEVS engine to produce a stand alone executable file which simulates the system.
- The *Simulation Interface*, which runs the stand alone executable and permits to change different simulation parameters (experimental setup) such as final time, number of simulations to perform, and the simulation mode (normal simulation, timed simulation, step-by-step simulation, etc).
- An instance of Scilab, which acts as a workspace, where simulation parameters can be read, and results can be exported to.

1.3 Basic Usage

Simulating an existing model

Figure 1.1 shows the PowerDEVS Model Editor main window. A model can be opened from the `File` menu or using the `Ctrl+o` shortcut.

The model shown in Figure 1.1 corresponds to the control of a DC motor. It can be found at `powerdevs/examples/hybrid/dc_drive/dc_drive_buck.qsm`. PowerDEVS Model Editor saves the models with extension `.qsm`.

After opening the model, it can be simulated. Clicking on the blue *play* icon or going to `Simulation->Simulate` at the menu (or just pressing the

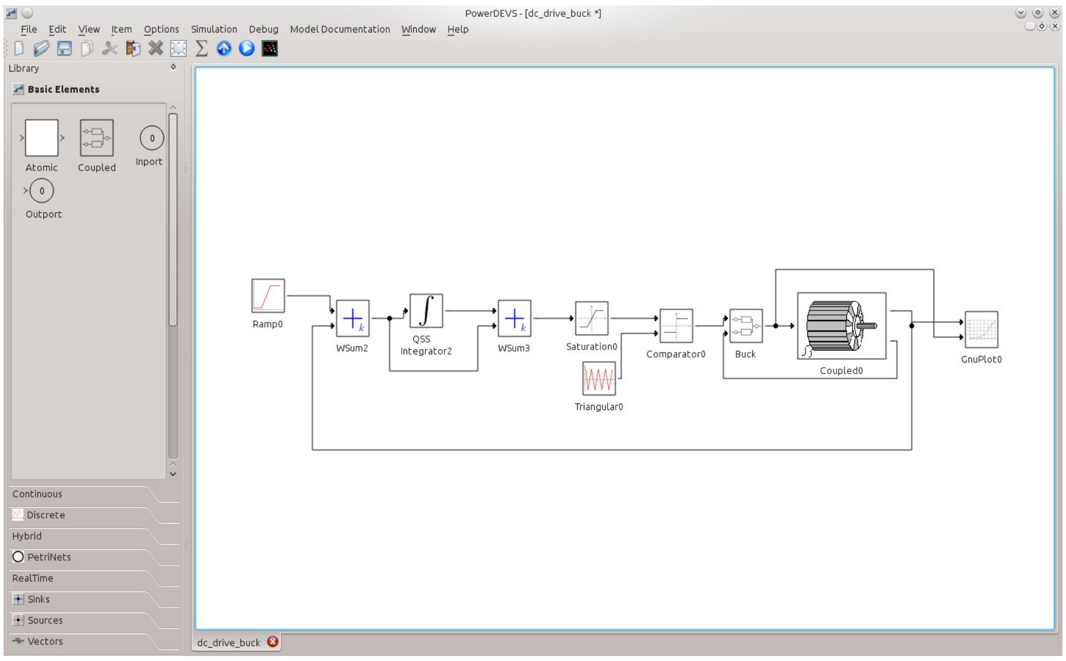
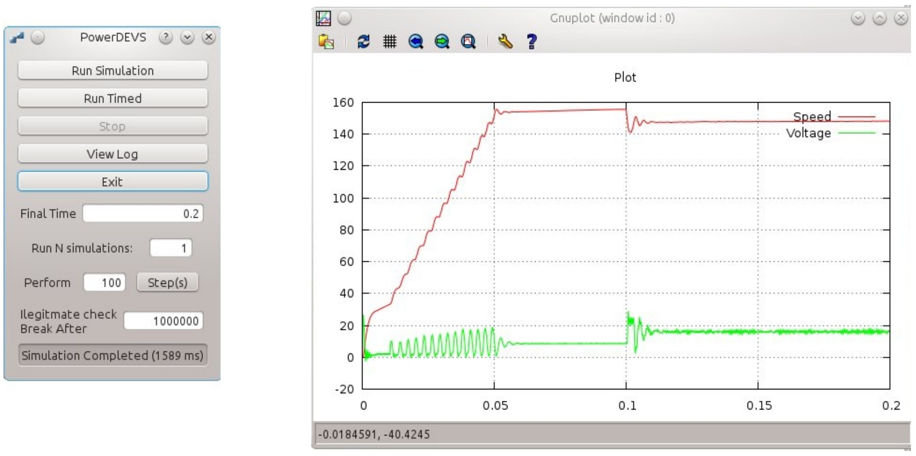


Figure 1.1: PowerDEVS Model Editor

shorcut key `F5`) PowerDEVS invokes the Preprocessor that automatically generates the C++ code and compiles it. Then, it opens the Simulation Interface, which can be seen at the left of Figure 1.2



Final Time 0.2
Run N simulations: 1
Perform 100 Step(s)
Ilegitmate check Break After 1000000
Simulation Completed (1589 ms)
-0.0184591, -40.4245

Figure 1.2: PowerDEVS Simulation Interface and Results

Clicking on the `Run Simulation` button the simulation is executed. The model in Figure 1.1 contains a `GNUPlot` block. This block opens an instance of Gnuplot to plot the signals it receives as it is shown at the right of Figure 1.2.

Creating a new model using existing blocks

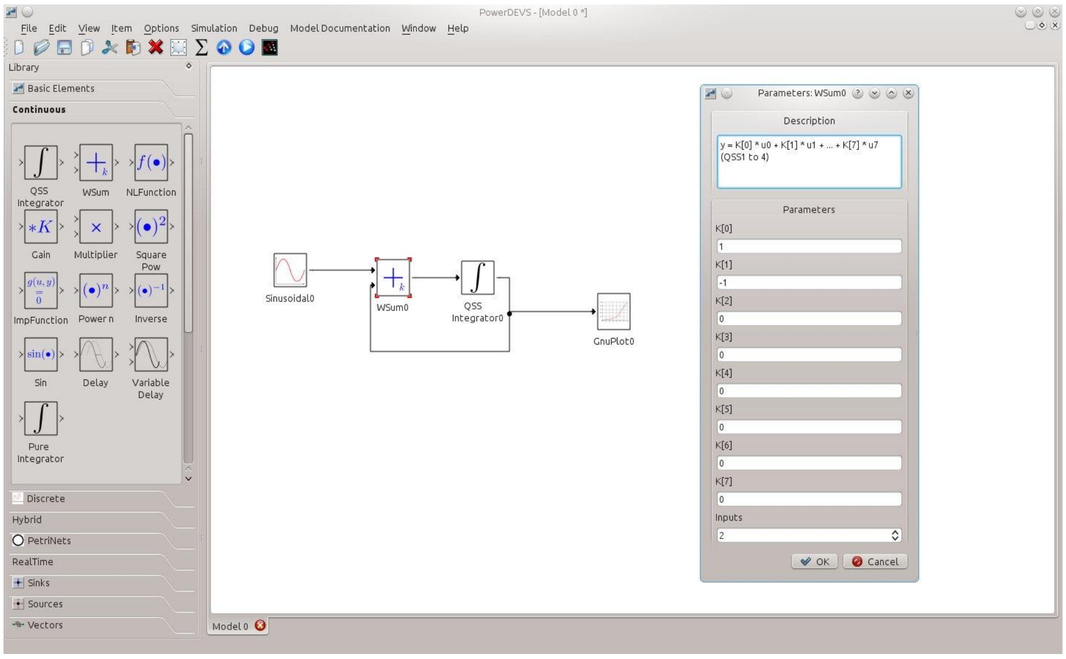
Using the PowerDEVS Model Editor, a new model can be created from the `File` menu or using the `Ctrl+N` shortcut.

Suppose that you want to build a model that simulates the first order continuous time system

$$\dot{x}(t) = \sin(2 \cdot \pi \cdot t) - x(t)$$

A block diagram for the continuous time system consists in an *Integrator* (computing $x(t)$ from $\dot{x}(t)$), a *Sum* block that calculates $\sin(t) - x(t)$ and a source block that provides the input signal $\sin(t)$.

Figure 1.3 shows this block diagram in the PowerDEVS Model Editor. The blocks are drag and dropped from the libraries at the left. The `QSS_Integrator` and the `WSum` blocks were taken from the `Continuous` library. The `Sinusoidal` source block was taken from the `Sources` library and the `GNUPlot` block was taken from the `Sinks` library. Then, the blocks are connected clicking on an input port and dragging until reaching the corresponding output port. Connections can be also performed from output ports to input ports or from input ports to previously existing lines.



Parameters: WSum0

Description

y = K[0] * u0 + K[1] * u1 + ... + K[7] * u7
(QSS1 to 4)

Parameters

K[0]: 1
K[1]: -1
K[2]: 0
K[3]: 0
K[4]: 0
K[5]: 0
K[6]: 0
K[7]: 0
Inputs: 2

Figure 1.3: PowerDEVS Model Editor

Double–clicking on a block opens the corresponding parameter dialog window, where the block parameters can be modified. At the right of Figure 1.3

the parameters of the `Wsum` are shown. There, the second parameter (which corresponds to the gain of the second input port) was modified so it is -1 instead of 1.

After modifying the parameters, the model must be saved before it can be simulated.

1.4 Features of DEVS and QSS Algorithms

PowerDEVS represents and simulates its models based on the *Classic DEVS Formalism.* This formalism allows to represent any system performing a finite number of changes in a finite interval of time.

Thus, in theory, any discrete time or discrete event model can be represented by DEVS and so by PowerDEVS.

In the libraries of PowerDEVS we have included blocks to model arbitrary discrete time systems and also to model *Timed Petri Nets.* However, blocks for different discrete formalisms can be eventually included.

However, the main feature of PowerDEVS is that it has a complete library to simulate continuous and hybrid systems using *Quantized State System* (QSS) methods.

QSS methods perform numerical approximations of continuous time systems. These approximations can be expressed as DEVS models.

These algorithms, besides having some nice stability and error bound propeties, show noticeable advantages in the simulation of systems with frequent discontinuities.

In PowerDEVS, QSS methods are implemented in the `QSS_Integrator` block. This block receives and provokes events representing changes in piecewise polynomial trajectories. The remaining blocks of PowerDEVS continuous and hybrid libraries perform different math operations on these trajectories in order to build arbitrary block diagrams. Source and Sink blocks were also programmed in order to work with QSS methods.

Chapter 2

Getting started

Chapter 3

DEVS formalism

Chapter 4

Basic modeling with PowerDEVS

Chapter 5

Simulation with PowerDEVS

Chapter 6

Creating and editing blocks

Chapter 7

PoweDEVS and Scilab

Chapter 8

Realtime simulation

Appendix A

Engine functions

Appendix B

PowerDEVS libraries