The Macrotheme Review
A multidisciplinary journal of global macro trends

"pse" – universal framework for visual programming

Stefan Sedivy*, Ingrid Zuziakova**, Peter Fabo*, Lubomir Pepucha*
**Research Centre University of Zilina, Slovakia*
***Faculty of Civil Engineering, University of Zilina, Slovakia*

Abstract

The subject of this paper is an overview of the characteristics of a universal interactive framework for creating simulation models based on the programming language Python and a library SciPy - Scientific Python. The basic structure of the framework, programming model, the procedure for creating custom components, dissemination and use with external simulation tools is described and examples are presented.

Keywords: python, programming, simulation, framework, block editor, open source

I. INTRODUCTION

Python programming language has in recent years built up a strong position in the rapid-prototyping and became dominant programming tool in many scientific and technical areas. Draft framework for the creation of simulation models based on the fact that the Python language [1] and its extensive infrastructure currently lacks suitable block editor, which could be easily tied with the visual form of the model with its program implementation.

Any available universal simulation environments such as. Matlab-Simulink, Scilab-Xcos, Modelica and others are built on the top of existing infrastructure and expansion or modification required has to be done using non-trivial knowledge of the internal structure of the programming environment, moreover, sometimes associated with additional costs for the purchase of specific development tools. Fundamental axiom of our proposed framework is therefore the exclusive use of a single programming language Python for all parts of the framework including its modification, creation of custom components and its extensions. Technological methods directly contained in the structure of Python itself, such as usage of libraries created in other languages are not affected.

II. FRAMEWORK REQUIREMENTS

Framework requirements were defined as follows

- Publicly available open-source project released under the GNU-GPL
- Exclusive use of a homogeneous multi-platform infrastructure of Python programming language, it is simple and widely accepted by technical and scientific community, cross-platform, open source with plenty of libraries a.e. PyQt [2], SciPy [3], Matplotlib [4], in

the field of natural and technical sciences, with a broad community support and with available literature

- Simple and intuitive internal structure of the framework allowing its easy extension and modification. Output structures of the framework should easily allow building simulation tools and specialized generators for specific simulators (eg. SPICE, Modelica)
- Framework environment itself must be suitable for the development of simulation models from different areas, with maximum ease of use, conceptually should be built as a separate plugin easy to implement in specialized applications
- Single bond to the host operating system through standard Python library system allowing the framework to interact with its environment (eg. integration of measuring and laboratory equipment) and communication via the Internet
- The possibility of using scripts in the actual framework, creating dynamic models and the dynamic changes in the parameters of components using the adaptive simulation tools.
- The possibility of real-time simulation with interactive components in the diagram for use the program as a demonstration of interactive teaching tool.

III. Framework Structure

Basic entities with which the user operates are the Component and the Net. The basic concepts are illustrated in Figure 1.

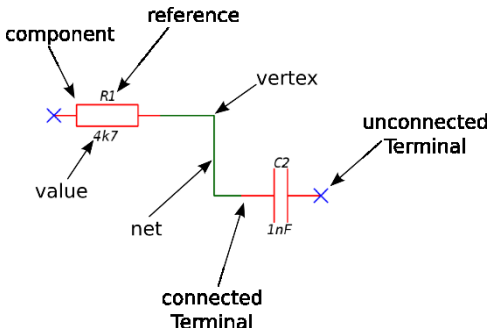

Fig. 1 Basic concepts, Component and Net

Standard entity *Component* presents a graphical representation of the selected algorithm that transforms the input information to the output information. In general, Component has *m* arranged inputs and *n* arranged outputs - terminals. Net represents the link between a component, and its task is to transfer information from one component output terminal to the input terminals of other components.

Net contains a reference to the start and end component which it links together, the number of the terminal of the component and the list of vertex of which it is formed. Each component and interconnection is clearly identified in a diagram by its unique number, optionally a name.

Arranged set of components and nets connecting them forms a *Diagram*. Each component can be parameterized using optional parameters, which can be static - their values are defined by the user at the beginning of the simulation, or dynamic - their values can be changed during the simulation based on its condition.

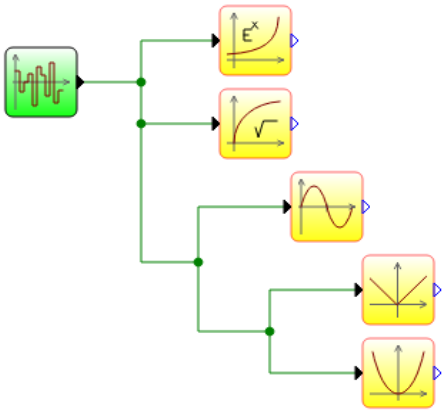

Fig. 2 The link among multiple Nets.

In addition to standard entities, there are connecting entities specifically established into the framework - *Connection* node component enabling connection among multiple independent Nets in Figure 2, and *Port* that allows connection between the parts of the diagram or multiple separate diagrams in the case of larger models in Figure 3.

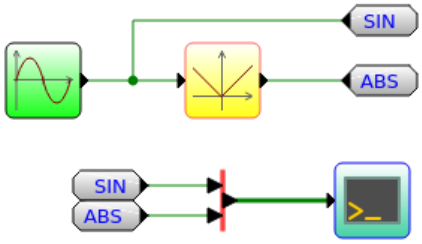

Fig. 3 Usage of component Port.

In general, in the framework can be defined a link between any terminals of components in the form of a multigraph – eg. the case of electrical circuits. For the purposes of simulation models this can be limited to the shape of a oriented tree, where the root of the tree is the output terminal and the leaves of the tree are the input terminals.

Carrier of the information in the framework are the terminals of components. Information is stored in the terminal in the form of one-dimensional array. If the array contains more than one value, in the graphical representation is Net connected to the terminal with a thicker line than a bus. The arrays of the values of multiple terminals can be aggregated by means of special components *BusCompressor* to any depth and extend back through *BusExtractor* components, as shown in Figure 3 and Figure 4.

Graphical representation of components and their interconnections itself is separated from the logical structure of the diagram, virtual links are therefore not shown in the diagram. At the same time the separation of logical and graphical representation allows the framework implementation to support various graphic libraries used by Python infrastructure.

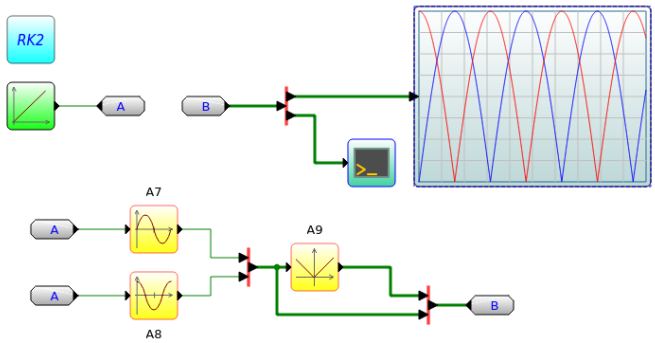

Fig. 4 The link among multiple Nets.

From the point of user view the components are grouped according to their functional or logical meaning

• Sources - Resources - generators, load datas from a file, load informations from connected devices and the Internet
• Sinks - Appliances of information - write to a file, console output, graphs, sending information to an internet connection
• Control - Control Components Communications
• Linear - components for linear transformation of information
• Nonlinear - components for the nonlinear transformation of information
• Signal - Components for connectingediting , aggregation links scalar to vector and its decomposition
• Discrete - discrete and logical components
• Interactive - components for interactive management chart during the simulation
Components from all groups in the diagram can be combined.

In the simulation of more extensive diagrams can create separate diagrams - blocks and use them in the simulation as separate components. Block diagram are expanded as macros with a separate namespace, Figure 5 and 6.

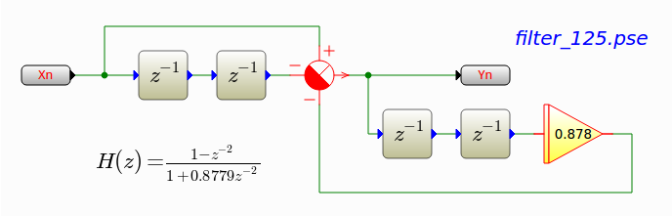

Fig. 5 Block representing discrete IIR filter second order, Xn is an input port and Yn is an output port of block.

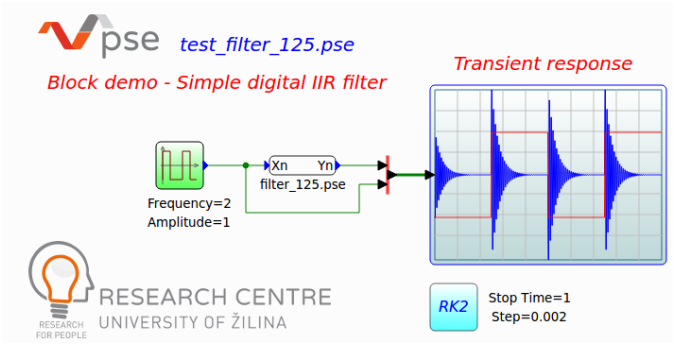

Fig. 6 Simulation of IIR filter block, transient response

The internal structure of the framework is designed so that it can be easily edited and modified according to the specific requests of users. To create a new simulation components need to inherit from class Component basic structure of the component. Then define its input and output terminals, the relationship between values of terminals and the graphic display component.

IV. EXAMPLE OF USE

For basic demonstration and a test, the framework has been extended by a simple simulation tool based on the one-step integration method Runge-Kutta 2nd order. The part of simulation tools are methods for validating connections among components and a recursive algorithm to find and check of the occurrence of loops in the tree. For the transformation of the input vector component on the output are used precompiled methods of Numpy libraries and the simulation itself runs in a separate thread outside the graphic system.

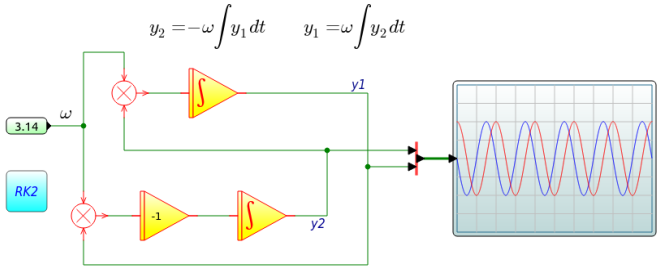

Fig. 7 Simulation system of linear differential equations.

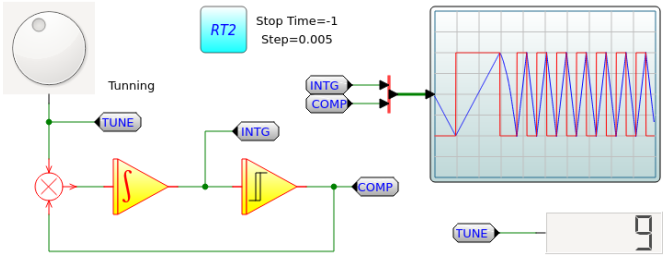

Fig. 8 The use of interactive components in the diagram. Interactive components Dial is for real-time the simulation is used to frequency setting generated by the oscillation of the oscillator.

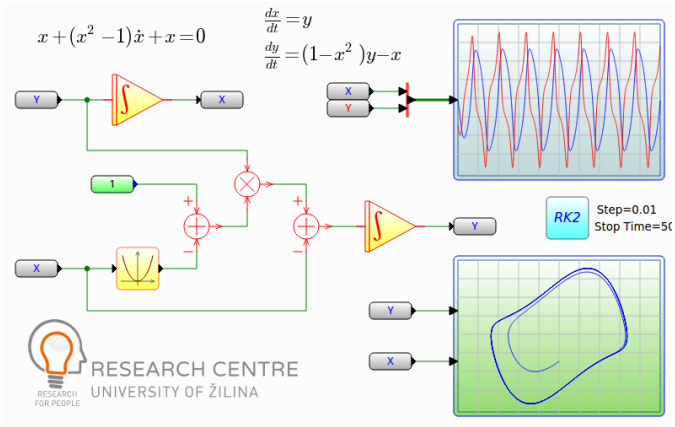

Fig. 9 Simulation of Van der Pol oscillator with non-linear damping.

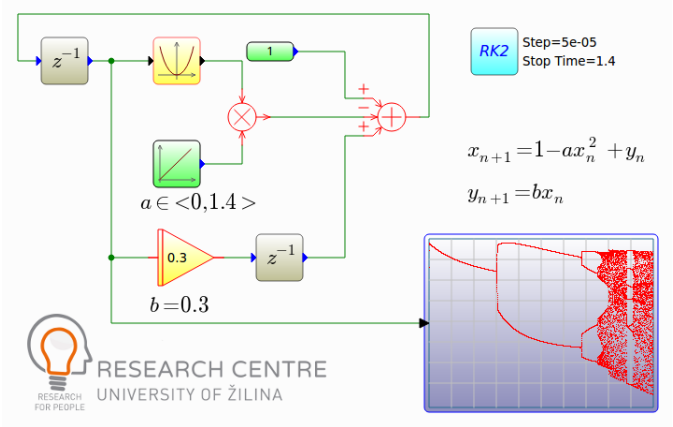

Fig. 10 Simulation of nonstationary parametric solutions of differential equations - Hennon map and bifurcations in one-dimensional discrete dynamical system

V. CONCLUSIONS

Framework was developed primarily for the creation of simulation models for predictive simulation parameters of road infrastructure within the project at Research Centre of the University of Žilina.

Description of the application framework includes a number of other options that the Article does not discussed, such as interactive management of the experiment and collect data from laboratory instruments, cooperation and data exchange between the Framework in the Internet environment, the possibility of creating interactive educational text in an environment Ipython and many others. Sources framework of the "pse" are free available on the web site of the Research Center of University of Žilina (http://www.researchcentre.sk/)

ACKNOWLEDGMENT

The research is supported by the European Regional Development Fund and the Slovak state budget by the projects "Research Centre of the University of Žilina" - ITMS 26220220183 and "Support and development of Center of transport research – CVD PLUS" - ITMS 26220220160.

EUROPEAN REGIONAL DEVELOPMENT FUND

REFERENCES

[1] www.python.org
[2] http://www.riverbankcomputing.co.uk/software/pyqt/intro
[3] Travis E. Oliphant. **Python for Scientific Computing**, Computing in Science & Engineering, **9**, 10-20 (2007), DOI:10.1109/MCSE.2007.58
[4] Hunter, J. D., Matplotlib: A 2D graphics environment, Computing In Science & Engineering, vol. 9., number 3, pages 90-95 (2007)