Open Source Software for Scientific Computation

SCiLAB

Research, Development and Applications

Shi Li, Long-Hua Ma, Claude Gomez (Eds)

Tsinghua University Press

Open Source Software for Scientific Computation

SCILAB

Research, Development and Applications

Edited by
Shi Li, Long-Hua Ma, Claude Gomez

Selected Papers from 2006 International Workshop on
Open Source Software SCILAB and its Engineering Applications
(OSSS-EA'06)
Sept. 28-29, 2006, Hangzhou, China

Tsinghua University Press
Beijing

图书在版编目（CIP）数据

科学计算自由软件 SCILAB 研究、开发与应用 = Open Source Software for Scientific Computation SCILAB Research, Development and Applications /李实等编. —北京：清华大学出版社，2006.10

ISBN 7-302-13876-1

Ⅰ. 科… Ⅱ. 李… Ⅲ. 数值计算—应用软件，SCILAB—国际学术会议—文集—英文 Ⅳ. O245-53

中国版本图书馆 CIP 数据核字（2006）第 113206 号

出 版 者：清华大学出版社　　地　址：北京清华大学学研大厦
http://www.tup.com.cn　　邮　编：100084
社总机：010-6277 0175　　客户服务：010-6277 6969

责任编辑：陈国新
印 刷 者：北京嘉实印刷有限公司
装 订 者：三河市春园印刷有限公司
发 行 者：新华书店总店北京发行所
开　本：165×235　印张：22.75
版　次：2006 年 10 月第 1 版　2006 年 10 月第 1 次印刷
书　号：ISBN 7-302-13876-1/TP・8342
印　数：1~500
定　价：68.00 元

Preface

With the development of computer and its popularization, scientific computation has evolved into of the three magic weapons of scientific research with its own right. In addition, industrials in various domains have considered scientific computation as a key technology. In the light of this ongoing development, it is expected that scientific computation software will play an important role for enlarging wide applications. Scilab, as a free open-source software package for scientific computation, provides a powerful and interactive environment with extensive mathematical capabilities, sophisticated graphics, and high-level programming language for rapid prototyping. Due to its distinguished features of "open source", Scilab has been widely used by more and more engineers and scientists in both academia and industries.

As a matter of fact, since it was developed in 1990s by INRIA (Institut National de Recherche en Informatique et en Automatique) and ENPC (Ecole Nationale des Ponts et Chaussees), France, Scilab has been distributed freely along with the source code via the Internet and used in educational and industrial environments around the world. Meanwhile, Scilab is also being disseminated in all over the world by various introductory documentations in several languages such as French, English, Spanish, Portuguese, Chinese and so on.

Since 2002, an annual event of software development contest based on Scilab platform and environment has been carrying on in China, which is no doubt to promote and advance a worthy cause for the development of open source software engineering, especially for the research, development and applications of Scilab. In order to promote an in-depth research and development in this field and enrich scientific achievements, LIAMA and ZheJiang University, China jointly organize 2006 International Workshop on Open Source Software Scilab and its Engineering Application (OSSS-EA'06) on September 28-29, in Hangzhou, China under the sponsorship of Zhejiang University, INRIA/Scilab Consortium-France, LIAMA in China and so on. During the workshop the Award Ceremony of 2006 SCILAB Contest is celebrated so that some young and promising prize winners can take a good opportunity to exchange knowledge and expcrience with participants in the workshop. It is indispensable and very important to boost up the strength of R&D and further popularize the applications of scientific achievements in this field.

In this book, 30 papers are included, which are selected from papers submitted to 2006 International Workshop on Open Source Software Scilab and its Engineering Application (OSSS-EA'06). According to their corresponding topics the papers are classified as 3 different parts which may be introduced as follows.

Part 1 is about some researches on Scilab and Open Source Software Engineering, which contains 7 papers and the major contents the Integrating modelica in scicos; Evaluating Interpolation Kernels for InSAR Using Scilab; Scilab Grid Toolkit with ProActive; Some practical experiences with Scilab Scicos and RTAI-Lab at the SUPSI laboratory; Targeting the Scicos Code Generator HDL Model Example; The Present and Future of Scilab's Engineering Application; and Tools for building Scilab toolboxes.

Part 2 focuses on the design and development of toolbox for Scilab which contains 9 papers and the major contents A Complex Network Toolbox; A Testing Tool for Character Recognition; An ITS Dynamics Model Simulator; Audio Editor Toolbox, Campus Emergency Management System; Scilab scicos toolboxes; SciSpeech Toolbox; Telecommunication Toolbox; The realization of SCADA based on Scilab .

Part 3 consists of engineering applications based on Scilab which contains 14 papers and the major contents A Stereo Matching Method for Color Image; Dynamic Analysis and Simulation of Water Management; Electrostatic precipitator modeling and simulation;

Function versus geometry; Secondary Clarifier and A2O Process Modeling; Modeling of the Activated Sludge Process; Modeling and Simulation of Adaptive Optics; Modeling and Simulation for Transpiration Cooling Control System with Distributed Parameters Based on Scilab; Identification and simulation of mechanical structures; Simulation of cars waiting at a toll booth using Scilab; Sustainable Cooktop Design; The Counting Algorithm of Transistor Based On Image Detect Technology; Two efficient numerical methods for finding global minimizers of continuous functions of one variable; Using Scilab to Perform a Cogeneration System Simulation and Optimization.

We are grateful to all contributors for making the workshop a success. We would like to express our sincerely thanks to France Telecom, R & D Beijing; Thomson Broadhand R & D Beijing; Bull Information Systems Beijing; Hangzhou Zheda Feiteng Technology Co., Ltd; Jinhua Hansheng Mechanical & Electric Control Engineering Co., Ltd; Zhejiang JiaHuan Electronic Co., Ltd. for their support. We also thank Mr. Zhe Peng for his elaborately composing and design in the publication of this book; and Ms. Li-fang Zhou, Mr. Sheng-ming Chen and Mr. Ke-jie Fang for their beneficial helps in organizing the workshop.

Long-Hua Ma, Shi Li

Claude Gomez

Hangzhou, Aug.29, 2006

CONTENTS

Part1. Researches on Open Source Software Scilab

Part2. Design and Development of Toolbox for Scilab

Part3. Engineering Applications Based on Scilab

Part 1

Researches on Open Source Software Scilab

Challenges in Integrating Modelica in the Hybrid System Formalism Scicos

Ramine Nikoukhah

INRIA-Rocquencourt, Domaine de Voluceau, 78153 Le Chesnay Cedex, France
ramine.nikoukhah@inria.fr,
WWW home page: http://www.scicos.org

Abstract. Both Modelica and Scicos propose formalisms for modeling hybrid dynamical systems. Despite many differences, their integration within a single tool is considered. It is shown in particular that even though the two methods of constructing models are very different, there implementations of hybrid dynamics are so similar that an integration at a very low level can be considered.

1 Introduction

Many different formalisms have been used for modeling systems with both continuous-time and discrete-time dynamics. Such systems, referred to as hybrid dynamical systems, are encountered in many systems and control applications. Scicos and Modelica propose two such formalisms that allow easy model construction, and can be used efficiently for numerical simulation and real-time code generation. They consider very general dynamic behaviors and are suited for general purpose modeling and simulation tools.

Modelica is a language based formalism in which the goal is to be able to express all the dynamics of the system without the use of any other programming language. Scicos on the other hand is based on a hierarchical graphical editor in which the model is constructed by connecting elementary blocks, the dynamics of which can be expressed in standard programming languages such as C. Despite these fundamental differences, the treatment of the hybrid dynamics in both formalisms are so similar that it is only natural to consider the integration of the two within a single tool. It is of course of no surprise that the two formalisms are similar, in particular as far as the discrete dynamics is considered, since both are largely inspired by synchronous languages such as the Signal language [1].

Even though it is possible to consider extensions to the Modelica language to allow the use of Scicos models, in this paper we focus on the integration of Modelica within the Scicos formalism. This integration is already started; in the current version of Scicos, one can define sub-diagrams using Modelica components (for example an electrical circuit). This however is only a first step since only continuous-time dynamics are allowed to be defined in Modelica. The use of discrete dynamics within the Modelica subpart is not allowed since it poses synchronization problems with the rest of the diagram defined in Scicos formalism.

In this paper, we examine the way hybrid dynamics modeling is implemented in Scicos and Modelica, and examine the similarities and the differences. We then propose solutions for a harmonious integration of Modelica in the Scicos tool. We assume the reader is familiar with Modelica and Scicos; information about Modelica can be obtained from www.modelica.org. For information about Scicos, visit www.scicos.org; see also [2].

2 Why use Modelica in Scicos?

Standard Scicos is not well suited for physical component level modeling. For example, when modeling an electrical circuit, it is not possible to construct a Scicos diagram with a one to one correspondence between the electrical components (resistor, diode, capacitor,. . .) and the blocks in the Scicos diagram. In fact, the Scicos diagram does not resemble the original electrical circuit .

Consider the electrical circuit depicted in Figure 1. This circuit contains a voltage source, a resistor and a capacitor. To model and simulate this diagram in Scicos, we have to express the dynamics explicitly.

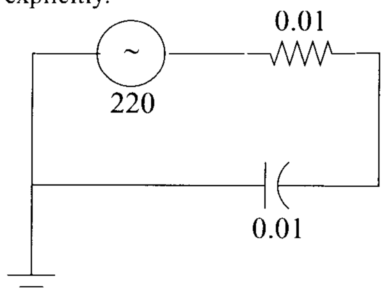

Fig. 1. A simple electrical circuit.

The resulting Scicos diagram is depicted in Figure 2. Note that this Scicos diagram does not look anything like the original electrical circuit in Figure 1.

Recently an extension of Scicos has been developed to allow modeling of physical components directly within the standard Scicos diagrams. This has been done, in particular, by lifting the causality constraint on Scicos' blocks and by introducing the possibility of describing block behaviors in the Modelica language. This extension allows us to model naturally not only systems containing electrical components but also mechanical, hydraulic, thermal, and other systems. For example, the electrical circuit in Figure 1 can be modeled and simulated by constructing the Scicos diagram in Figure 3. The electrical components come from the `Electrical` palette.

The electrical components in this case are realized as *implicit blocks*. Implicit blocks have implicit ports. An implicit port is different from an input or an output port in that connecting two such ports imposes a constraint on the values at these ports but does not imply the transfer of information in an a-priory known direction as it is the case when

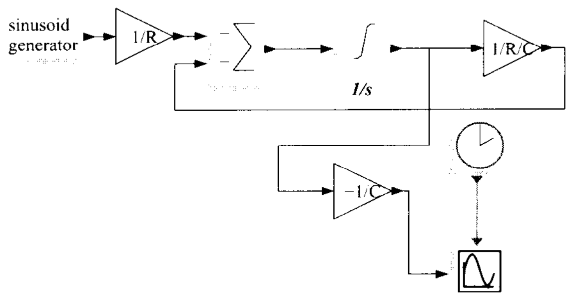

Fig. 2. Scicos diagram realizing the dynamics of the electrical circuit in Figure 1.

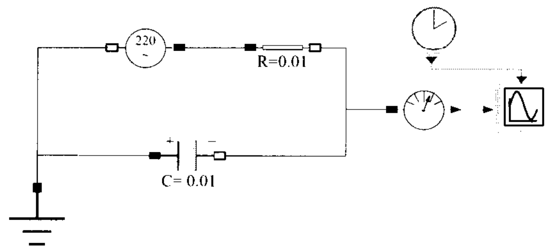

Fig. 3. Scicos diagram using Modelica component.

we connect an output port to an input port. For example, the implicit block `Capacitor` used in diagram of Figure 3 has current and voltage values on its ports but there is no way a-priori, without analyzing the full diagram, to designate any of them as input or output.

Implicit blocks and the construction of diagrams based on these blocks is fully in the spirit of the object oriented language Modelica. The description of block behavior is done in Modelica. Only a subset of the Modelica language can be used however since all the implicit blocks are grouped in a first phase of diagram compilation to obtain a single standard Scicos block with continuous-time dynamics. As long as the Modelica based components of the diagram contain only continuous-time dynamics, the convertion into a single Scicos block is straightforward, but Modelica is a rich language allowing the modeling of discrete and hybrid dynamics. These latter capabilities, if used within a Scicos diagram, poses many challenges. The main problem is how to assure synchronism among discrete-time dynamics of the Modelica and Scicos parts of the diagram.

3 Full integration of Modelica

We study here the possibility of a full integration of Modelica within the Scicos environment. The first step is to identify the similarities between the two and in particular to establish a sort of a dictionary for translating models from one formalism to the other.

3.1 Hybrid dynamics: Modelica and Scicos

We have presented an overview of some aspects of hybrid dynamics modeling in Scicos earlier in this paper; we now present the way Modelica implement similar dynamics.

Continuous-time dynamics As far as the continuous-time dynamics is concerned, both Scicos and Modelica work with systems of DAEs (Differential-Algebraic Equations). Modelica has the advantage of having the possibility to manipulate the equations symbolically and thus for example reducing the index if necessary, and optimizing the code. But at the end, the type of continuous-time dynamics obtained by Modelica is fully integrable in Scicos; in fact as we have previously seen, this integration is already realized in Scicos.

The zero crossing detection capabilities of Scicos are realized using the `when` statement in Modelica. For example

```
when x>0 then
```

where `x` is a continuous-time variable, corresponds to a zero crossing test in Scicos. This feature is also already integrated in the current Modelica integration in Scicos.

Events in Modelica are in general similar to events in Scicos. An event can cause a jump in a continuous-time variable. In Modelica this is done via the `reinit` operator. In Scicos, an always-active block having a continuous-time state can also have an input activation port. An event received on this input activation port could cause a jump in the continuous-time state. One such a block is the integrator block with re-initialization available in the `Linear` palette. The output of this block is the integral of its first input but when it is activated by an event through its activation input port, the value on the second input is copied into the state of the integrator.

In Modelica, as it is the case in Scicos, events are used to help the numerical solver integrate the system more efficiently. This is done in particular by generating events at points in time where the continuous-time dynamics become non-smooth. For example the `abs` function (absolute value), is not differentiable at zero. So when the input value of this function crosses zero, a cold restart signal should be sent to the numerical solver. Both in Modelica and Scicos, these events are generated by default but can be turned off. In Modelica, this is done using the `noEvent` directive, and to some extent using the `smooth` directive. In Scicos, each block has direct control over the number of its zero-crossings.

Discrete dynamics In Modelica, the `when` statement is also used to implement the discrete dynamics. In particular a statement such as `when x>0 then` seen earlier applied to a continuous-time `x` can also be used with a discrete-time `x`. The event triggered in this case is synchronous with the discrete-time variable `x`. This usage of `when` can also be realized using the `if` and the `edge` statements. Conditional sub-sampling in particular is realized in this way.

In Scicos, discrete dynamics is realized using events and conditional sub-sampling is realized using the special blocks `IfThenElse` and `ESelect`. The main difference with Modelica is that in Modelica the primitive operator `sample` is used to implement an event clock. In Scicos, event clock is realized using a delay and a feedback. This type of realization is also possible in Modelica and the necessity of the primitive `sample` is not clear; it can very well be implemented as a model.

Another important language feature in Modelica is the possibility of performing an action when one, or more, of certain events is activated. For example the statement

```
when {x > 2, sample(0,2), x < 5} then
```

becomes true if any of the three conditions within the curly brackets becomes true. In Scicos, this union of events is realized using the `event sum` "block". See Fig. 4. Note that both in Scicos and Modelica, if two of these events become true synchronously, the corresponding operations are executed only once.

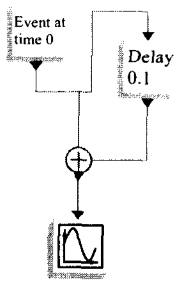

Fig. 4. Union of events realized with the sum operator.

An important conditional constructor in Modelica is `elsewhen` (or equivalently `elseif edge`). This constructor is used as follows:

```
when cond1 then
   ...
elsewhen cond2 then
   ...
end when;
```

In case `cond1` and `cond2` become synchronously true, this construction specifies unambiguously the operations that must be executed. In Scicos, the same construction can be realized in a more general form. In particular, a block can have more than one input activation port; this is the case for the `Select` block, which has two regular and two activation input ports. This block depending on the activation input by which it has been activated, copies one or the other of its inputs on its output (it selects one of the inputs). To be able to realize such operations, the Scicos block, when activated, receives the an integer specifying exactly the way by which it had been activated. In this case, there are three possibilities: activated through the first activation input port, through the second or both. The block can then execute different operations in each case. Note that this implementation is more flexible than the `elsewhen` construct in Modelica.

3.2 Scicos blocks expressed in Modelica

Considering the similarities between the two formalisms, it is natural to consider the construction of most, if not all, Scicos blocks in Modelica language. This task is a lot more ambitious than the integration already available in Scicos where only blocks with continuous-time dynamics can be modeled in Modelica.

To achieve the full integration, certain adaptations must be considered both in Scicos and in Modelica. We have already encountered the problem with the generation of events using the `sample` primitive and the dual usage of `when`. These however do not pose any real problem because they can always be implemented, in Modelica, using other primitives, for which we have counterparts in Scicos.

Another difference encountered previously is the counterpart of the `elsewhen` statement, which is more general in Scicos. But again this doesn't pose any problem because the Scicos implementation is strictly more general; it would have been a problem if a Scicos block were to be integrated in Modelica as an external function.

4 Conclusion

We have discussed a number key issues in integrating Modelica in the Scicos environment. We have shown that thanks to the similarity between the two formalisms, this task is realizable as long as some Modelica primitives are implemented as models.

The full integration of Modelica considered in this paper would be very valuable to Scicos since having access to formal description of blocks is essential for model reduction and optimal code generation. On the other hand, the Scicos environment provides a natural setting for Modelica language by providing a free Simulink like model editor where the user has the option of constructing new blocks either using the Modelica language or standard programming languages such as C based on Scicos formalism. This latter option is more efficient in some cases, but also can be easier for developers who are not familiar with the complex object oriented language Modelica.

References

1. A. Benveniste, and P. L. Guernic, "Hybrid dynamical systems theory and the signal language", *IEEE Transactions on Automatic Control*, AC-35, 1990.
2. S. L. Campbell, J. P. Chancelier and R. Nikoukhah, *Modeling and Simulation in Scilab/Scicos*, Springer-Verlag, 2005.