Rand Model Designer's Numerical Library

A. A. Isakov Yu. B. Senichenkov
Institute of Computer Science and Technology, Peter the Great St. Petersburg Polytechnic University, Russia
senyb@dcn.icc.spbstu.ru

Abstract

Numerical Libraries of visual simulation environments of complex dynamic systems differ from those of specialized collections of software implementations of numerical methods by means of the heuristic control program designed for automatic selection of numerical methods, accuracy control, and identification the specific features of systems of algebraic, ordinary differential and differential-algebraic equations on a local trajectories of the event-driven dynamic systems.

Keywords: dynamical and hybrid systems, component «physical», «causal», and «agent»-based models, behavior of event-driven complex dynamical systems, differential-algebraic equations, numerical methods for NAE, ODE, DAE

1 Introduction

Universal environments of visual simulation for complex dynamic systems (Simulink + Toolboxes, Modelica – based environments, Rand Model Designer, et cetera) are employed for construction, reproduction and behavior visualization of event-driven systems. Those come equipped with particular numerical libraries, which differ from general collections of professional numerical programs by means of heuristic control programs (also known as calling programs) for automatic selection of numerical methods.

Users of specialized collections, which are creating control programs on their own, usually deal with solution of a single system, or multiple systems, where system properties are pre-determined, or are determined in the modeling process. This allows users to pick out appropriate methods experimentally.

However visual environment users face exponentially large number of tasks generated by the state machine model, even when working with a simulation of a relatively small event-driven system. Systems of equations correspond to a particular state of a multi-component model and are built automatically; while a numerical method is routinely selected for those systems in the course of the process.

Reference article (Shampine et al., 2003) describes the software package MATLAB ODE Suite (The MathWorks) for numerical integration of event-driven dynamical systems. While referenced research works (Shampine et al., 2000; Shampine al., 1999) describe methods used for event location and for solving systems of differential-algebraic equations of high index.

Discussion on numerical problems in software environments which use Modelica language, are described in reference material (Fritzson, 2011; Sanfelice et al., 2010; Navarro-López, 2011; Senichenkov et al., 2013).

Rand Model Designer environment (Kolesov et al., 2013; Kolesov et al., 2014; Martin-Villalba et al., 2015; Kolesov et al., 2015; Isakov, 2015) contains a traditional library of software implementation of numerical methods (Fortran programming language), and a control module (C ++ programming language), which allows selection of specific solution methods for systems of equations on a given trajectory. Numerical Library and control module – hereinafter referred to as numerical library – are expressed in a form of dynamic link library (dll). This library provides the utmost rapid numerical solution for problems arising in the "release" mode, as well as debugging and tracing in the "debug" mode.

Open text software implementations of numerical methods (ODEPACK, RADAU, DDASPK ... - http://www.netlib.org) as well as solution algorithms of NAE, ODE, DAE presented in reference materials (Forsythe et al., 1977; Dongarra et al., 1979; Rice, 1981; George et al., 1981; Pissanetzky, 1984; Ascher et al., 1998; Shampine et al., 2003; Shampine et al., 2000; Shampine et al., 1999; Hairer et al., 1987; Hairer et al., 1991, 1996; Hindmarsh, 1983; Shampine et al., 1997), were used to create a RMD library. However, selected methods had to be modified and supplemented with original algorithms.

In addition to solution of equations on trajectories, it is also necessary to analyze the structure of the equations, to simplify the construction of the system, eliminating the variable "links", to determine state "switch-points" of event-driven systems, to ensure consistent initial conditions and to reduce index for differential-algebraic equations, as well as to determine the initial approximations for iterative methods.

Today visual simulation environments are preferred (Isakov, 2014):

- to support object-oriented modeling (OMM) technology
- to utilize modeling languages that allow user to simply describe most required tasks
- to comply with, not institutionalized, yet generally established, standards, such as, for instance Unified Modeling Language (UML) for discrete event-driven systems (Rumbauth et al., 2005).

Unification of the input language level only, as done for discrete event-driven systems (UML – http://www.uml.org), does not guarantee the same behavior in various environments of the same hybrid event-driven dynamic system. Unification of the numerical libraries is required for optimal performance. It is desirable to achieve the same result as for the classified collections – when a given system, solved by numerical methods from different collections, has the same numerical solution for a predetermined comparison criterion.

In this article, we have tried to describe thoroughly the process of building, conversion, and numerical solution of systems of equations generated by the state machine of the hierarchical multicomponent event-driven model with variable structure within the visual modeling environment of the Rand Model Designer (RMD – www.mvstudium.com).

The Rand Model Designer (Kolesov et al., 2015) allows the model to be built in a form of an isolated dynamic or hybrid system, a hierarchical multi-component system of a permanent structure with components with the "input-output" external variables or components with "contact-flow", and model with variable (dynamical) structure.

2 Description of the complex dynamical systems behavior. Classical dynamical systems

Local continuous activity (continuous activity of hybrid automaton) could be expressed in form of:

- implicit dependencies on time
- linear and nonlinear algebraic equations (NAE)
- systems of ordinary differential equations in normal form or in form of equations not resolved with respect to derivatives (ODE)
- systems of differential-algebraic equations (DAE)

Each type of equations - NAE, ODE or DAE – has its own set of methods with mandatory program AUTO. Automatic mode is set by default, however user could replace it by any other method from the corresponding group.

Vector-matrix form of equations is preferred. It is "standard" form of equations for specific numerical method, which does not require further change when referring to a particular software implementation of the method.

However, especially in the early stages of design, user rarely thinks about the choice of method, and rather writes the equation in "free" form, which subsequently has to be converted into a "standard" form.

For example, even the simplest form of behavior description by way of scalar equations sequence with substitutions requires Equations Analyzer to regularize custom unknown variables and equations, to validate procedure and substitutions accuracy, as well as to determine “algebraic cycles”, i.e. hidden equations in those.

3 Consideration of solving systems structure

Solving of large-scale systems of nonlinear algebraic equations serves as the key task in the numerical simulation of complex dynamic systems. This task is important in itself; however, it becomes even more relevant in the context of ensuring consistent initial conditions, solving differential-algebraic equations, and implementing of the implicit methods for solving ordinary differential equation systems.

Within the RMD environment, Newton's method is utilized as a main method for solving systems of nonlinear algebraic equations. However, this method requires impeccable initial approximation. In the event of Newton's method failure, the task of finding solution is reduced to the task of minimizing a quadratic functional (Powell method).

The basic operation of the Newton’s method is solving systems of linear algebraic equations at each iteration. When using direct methods, it is important to pre-determine the matrix structure and call a corresponding modification of the method. Notably, RMD environment mostly relies on various modifications of the Gauss method for fully filled, band and sparse matrices

The implementation of the Gauss method for large sparse systems MA28 (Pissanetzky, 1984) provides the possibility of bringing the original system to block triangular form by means of rows and columns rearrangements. Whereas, bringing the original system to block-diagonal form, makes it possible to solve systems in parallel.

In case of block-triangular systems, only systems corresponding to diagonal blocks could have a natural solution. Consequently, the solution rate increases, but the question on optimal size of the diagonal block remains. Oftentimes, in case of dealing with small blocks, system solution overheads are inadequately high.

4 Standard description forms of continuous activity

In RMD equation and substitution differ syntactically. By default entry $x = f(c,k,t,y)$, where f - real-valued function (also known as real function), is considered as substitution, if variables $\{c,k\}$ on the right side are declared constant or parameters and value of variable y is determined at the time of calculation of the substitution. Here $t = \{Time, time\}$,

where ($Time$) – global, and ($time$) – local time.

Equations are written in a scalar form (1):

$$\begin{cases} z - q(c,k,t,z) = 0 \\ f(x,y,c,k,t) = 0 \\ g(x,y,c,k,t) = 0 \end{cases} , \qquad (1)$$

or vector-matrix form (2):

$$F(x,c,k,t) = 0, x, F \in \Re^n , \qquad (2)$$

and could be complemented by substitutions.

RMD gives a possibility for the user to "see" the system, which is being solved, in a designated window during the model execution. This is not only imperative for multi-component systems, with automatically built equations (based on component equations with consideration of their links), but also for the single-component systems, where final equation may differ from the original one.

User has to be notified about the Equation Analyzer operations: namely about allocation and regularization of required variables and equations. The process of allocation and regularization of variables and equations, as well as the one of determination of the equations structure is called structural analysis in RMD. It uses a bit-block system matrix, which indicates the occurrence of the unknown variables in the equation. The structural matrix for large systems takes a lot of memory space, while the structural analysis of the system takes a lot of time. This is especially true if system of equations is underdetermined and if conditions require finding a non-degenerate structural subsystem of maximum size. Oftentimes, at the early stages of building a new model, user needs Customer Support to eliminate underdetermined information.

Linear equations brought to the vector-matrix form

$$A(t)\cdot x(t)=b(t); \qquad (3)$$

are divided into the equations with fully filled, band and sparse matrix systems.

For systems with a fully-filled matrices, software implementation of the Gauss method with an assessment of the condition number of LINPACK package (Dongarra et al., 1979) (DGECO, DGESL) is used, for band matrices – subprograms (DGBCO, DGBSL). For systems with sparse matrices (Isakov, 2014) variation of the Gauss method (Pissanetzky, 1984) was realized, which provided the possibility of reducing a matrix to a block triangular form. It has been concluded that preliminary matrix reduction to block triangular form is justified for numerous technical tasks across various fields. Moreover oftentimes the final system turns out to be in a block-diagonal form, which makes it possible to solve systems in parallel. Taking into consideration specific properties of the diagonal blocks particular numerical methods may be employed when dealing with those. In particular, blocks where unit size equals one could be regarded as equations with a single variable, thus employing appropriate methods (Forsythe et al., 1977).

Nonlinear equations

$$F(x,c,k,t)=0 \qquad (4)$$

are solved with the Newton's method. In the RMD environment, in Newton's method, frequency of recalculation of the Jacobian matrix may vary, depending on the speed of convergence. Moreover method failure results in call in for the Powell method.

Ordinary differential equations with relation to the first derivative are recorded in scalar or vector form (5):

$$\frac{dx}{dt}=F(x,c,k,t), \quad x(0)=x_0; x, x_0, F\in\Re^n . \qquad (5)$$

Normal form, with relation to the second derivatives, is permitted (6)

$$\begin{aligned}&\frac{d^2x}{dt^2}=F(x,c,k,t), \quad x(0)=x_0; \frac{dx}{dt}\Big|_{t=0}=V_0;\\&x, x_0, F\in\Re^n\end{aligned} \qquad (6)$$

for the implementation of numerical methods that use it as the initial form (Hairer et al., 1987).

Differential-algebraic equations (7)

$$F(x,\frac{dx}{dt},c,k,t), \quad x(0)=x_0 \qquad (7)$$

are reduced to a semi-explicit form (8)

$$\begin{cases}\dfrac{dx}{dt}=x', x'(0)=x'_0\\ F(x,x',c,k,t)=0, x(0)=x_0\end{cases}, \qquad (8)$$

if those are not recorded in the following form (9)

$$\begin{cases}\dfrac{dx}{dt}=F(x,y,c,k,t), x(0)=x_0\\ G(x,y,c,k,t)=0; x,x_0,y,G,F\in\Re^n\end{cases}. \qquad (9)$$

It is possible to use explicit methods for the semi-explicit form (Hairer et al., 1987), resolving a nonlinear algebraic equations systems as required by a suitable method of NAE group.

Additional (in comparison with the solution of differential equations) operation is required when solving differential-algebraic equations – ensuring consistent initial conditions (utilizing suitable methods of NAE group).

The ability to successfully use the semi-explicit form depends on the properties of the Jacobi matrix. It is possible to use certain methods from the DAE group, for high index equation systems, if the system does not require conversion (Hairer et al., 1996). Alternatively it is possible to differentiate the equation to reduce the index.

Differentiation could be done symbolically, as in math suites, designated for symbolic computation (also known as algebraic computation). Or alternatively, differentiation could be done numerically, as in visual modeling environments. Character differentiation block is planned to be implemented in Rand Model Designer in the near future. Meanwhile, differentiation is carried out by means of "linear differentiator".

Let $x = x(t)$ be the function, derivative of which is to be determined. Consider the following equation

$$\frac{d\tilde{x}'}{dt} = K \cdot (x - \tilde{x}'), \tilde{x}'(0) = \tilde{x}'_0. \qquad (10)$$

For sufficiently large positive K solution to this additional equation tends to the specified function, while solution derivative tends to function derivative (if the derivative is "stable") (Emeljanov et al., 1997). Regrettably, even for the stable problems, error in the boundary layer may be excessive. However, experience has shown that for many models, even considerable errors in the boundary layer (arising from the unreliable initial conditions) are acceptable. It is important to keep in mind this source of potential errors.

5 Discrete dynamic and hybrid systems

RMD uses hybrid simulated time – continuous periods alternating with short time slots, where events are put in order (Kolesov et al., 2013; Kolesov et al., 2014). Discrete dynamic and hybrid systems are described by means of hybrid machines, equipped with a hybrid time.

To solve difference equations (11) a hybrid machine with an "empty" continuous activity and discrete actions is created

$$\begin{cases} Z_{k+1} := F(Z_k); k = 0,1,...; Z_0 = Z(0) \\ Z_k := Z_{k+1}; F, Z \in \Re^n \end{cases}. \qquad (11)$$

The hybrid machine with continuous activities in the form of equations is essentially a generalization of classical dynamical systems, with all the inherent difficulties of numerical behavior reproduction - Zeno effect and sliding modes.

Hybrid machines within RMD are UML state machines without parallel (orthogonal) activities (Rumbauth et al., 2005), which relate to additional numerical tasks, namely determination of the "switch points", or event location, set by the operator "When", resulting in a state change.
Operator "When" may contain variables of any type, thus the main operational method for event localization is the method of bisection interval.

In case the event is associated with real variables and equality to a predetermined value, it is possible to formulate and solve the problem of finding a root on a given trajectory. For example, LSODAR program of ODEPACK – a collection of solvers for the initial value problem for ordinary differential equation systems (Hindmarsh, 1983). includes a root finding capability. However, it is difficult to determine the significance of this problem, due to the lack of data on the use frequency of operators "after", "signal", "when" complex system models.

6 Component systems with «inputs-outputs»

While components with "input-output" that have internal state machines produce exponentially large number of possible final systems, there are no major obstacles in creating those. Whatever way the components are connected, the structure of the component equations essentially does not change, if only left side variables of the substitutions that are defined as exit variables may become unknown variables of the new algebraic equations.
When using components with "input-output" the problem of differentiation of input variables is simplified, if they satisfy the differential equations within the components where they are formed. It is enough to pass the calculated value of the derivative to the right component.

7 Component systems with «contacts-Flows»

Components with "contact-flow" are utilized for "physical" simulation, where they generate an additional problem related to internal hybrid automata.

When using components with "input-output", if component equations for unknown variables are differential equations, then they remain differential equations in relation to the same variable also in resulting system for corresponding state in composition of component hybrid automata.

When using components with "contact-flow", if component equations for unknown variables are differential equations (differential variables) then in the resulting system they may turn into algebraic equations for composition of component hybrid automata. In result user's initial differential variables become algebraic. That is due to the concept of index of system of algebraic-differential equations (Hairer et al., 1996). Equation Analysers of visual simulation environments heave to be able to recognize the systems with high index and transform those in order to lower the index. This, in turn, requires differential equations. Some programs (Hairer et al., 1996) could solve systems of low index without reducing it, but recognition of such system remains a problem.

Resulting hybrid automaton of the whole model or the composition of component hybrid automata has a very large number of states even for relatively small models and requires significant efforts to form all the possible equations at the stage of compilation, in case of components with "contact-flow".

There are no restrictions on state equations of component automata in RMD, but the equations for realized composition states of component automata are being built at the time of model execution. It is an attempt to avoid building large number of resulting systems and practically to take into account the frequency of their appearance at a certain trajectory, while at the stage of execution only of system of realized states.

8 Component systems with variable structure

Processes of birth and death, queuing analysis, agent-based models all deal with dynamically changing number of objects (components), that are appearing and being destroyed leading to connections between them appearing and being destroyed. In this case the number of states of hybrid automaton of the whole model and the number of corresponding behaviours (systems of equations) becomes variable. Components may have any type of external variables e.g. “contact-flow”. For instance electric stations that have dynamic number of consumers, systems with reservation.

In such a case, creation of equations “on the go” becomes the only available means of creating equations. Fast algorithms of final system composition according to component equations and algorithms for connection equations for such models become one of the most important tasks.

9 Debugging

RMD has a debugger for conventional tracing debugging at the level of input language. The user cannot control the work of numerical methods on continuous sections by using debugger, but the user can use file “Tracing” that if required will hold fairly sufficient details about functioning of numeric algorithms, collected by using “print” operators provided by developers. It should be noted that the source texts of numerical libraries of visual simulation environments are not available to the users of visual simulation environments even if those are inevitable modifications of open specialized collections. This makes «trace.txt» file problematic to understand and to work with.

The more useful instruments are dynamic windows that are connected with the built and modified system of equations being solved at the moment: Final system, Structure Matrix, Jacobi’s Matrix, Eigenvalues of the Jacobi’s Matrix.

It is difficult to use all available information mainly because the user is creating own or using ready components with local equations, and receives information about automatically created resulting system, moreover large size systems are difficult to analyse.

Almost all the visual simulation environments face the problem of transformation of collected information into intuitive tips that help comprehending the behaviour properties of the whole model, indicate possible ways of model transformation in order to improve performance provide required precision and satisfy requirements for memory.

10 Timeline

The duration of continuous sections of trajectories is determined by the frequency of occurrence of events leading to a change in behaviour. Change of behaviour is usually accompanied by overhead costs for initialization of software implementation of numerical method. Selection of time progress step at each continuous section is determined by external circumstances and features of the problem being solved in combination with features of the selected method.

For example in case of solving nonlinear algebraic equations by iterative methods, internal step is determined only by external circumstances while taking in account the limitations associated with the choice of the initial approximation in form of solution from the previous step.

In case of differential equations, the internal step may considerably depend on the choice of the method. The rigidity of differential equations system can become an insurmountable obstacle for explicit methods. The problems with rapidly oscillating solutions are challenging for almost all methods if output information is needed with a large enough time step. Trying to increase step when solving rigid problems beyond minor boundary level contradicts with the need to save time for event localization.

In RMD the information about problem features on trajectory and costs of numerical solution is available in the “Timeline” tool. The user selects a desired constant advance step of model time and time interval to observe the model behaviour, and receives integral characteristics: time spent on execution of discrete actions, time spent on integration at continuous sectors, time spent on matching initial conditions when solving differential-algebraic equations.

11 Testing

Testing numerical libraries of the visual simulation environments also has specific characteristics.

Generally the libraries are sets of software implementations of numerical methods from open collections, modified according to the requirements of the environment: agreements on the acceptable forms of user equations, necessary information about behaviour visualization, information collected for processing the results of computational experiments with the model. The modifications lead to both errors and overhead costs. Both require testing that can be done by using existing collections of testing examples.

However, the information about overhead costs that occur as a result of model decomposition into components followed by aggregation of into single hybrid system of the whole model by the environment is considered to be of the most importance. At this point hybrid models of large size that can be built by hands and whose characteristics are known in advance are considered to be of the highest value. If decomposition of such models is possible by using components that are available in the environment, then it is possible to assess the overhead costs that occur when resulting hybrid automata are built by the environment. Besides the assessment of overhead costs it also becomes possible to compare the accuracy of solutions on trajectories. The authors are not aware of any test methods allowing verification of hybrid systems, besides a few minor attempts that manage to provide symbolic solution (http://www.maplesoft.com/products/maplesim).

References

U.M Ascher and L.R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations.* SIAM. Philadelphia, 1998.

J.J. Dongarra, C.B. Moler, J.R. Bunch, and G.W..Stewart. *LINPACK Users' Guide*, Philadelphia, 1979.

S.V. Emeljanov and S. K. Korovin. *New types of feedbacks. M.: Science*, 1997.

G. Forsythe., M. Malcolm, and C. Moler. *Computer methods for mathematical computations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

P. Fritzson. *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*, Wiley-IEEE Press, 2011.

A. George and W-H J. Liu. *Computer solution of large sparse positive define systems*. Prentice-Hall, Englewood Cliffs, Inc, New Jersey, 1981.

E Hairer, S. P. Norsett, and G. Wanner. *Solving ordinary differential equations I. Nonstiff Problems.* Springer-Verlag, 1987.

E. Hairer and G. Wanner. *Solving ordinary differential equations II. Stiff and Differential-algebraic Problems.* Springer-Verlag, 1996.

A. C. Hindmarsh. ODEPACK, A Systematized Collection of ODE Solvers, in *Scientific Computing*, R. S. Stepleman et al. (eds.), North-Holland, Amsterdam, IMACS Transactions on Scientific Computation, 1: 55-64, 1983.

A. A. Isakov, Yuri B. Kolesov, and Yuri B. Senichenkov. A new tool for visual modelling – Rand Model Designer 7. *IFAC-PapersOnLine* 48(1): 661-662, 2015.

A.A. Isakov OpenMVLShell visual environment. In Yu. B. Kolesov, and Yu. B. Senichenkov. *Mathematical modeling of hybrid dynamical systems.* St. Petersburg, Peter the Great Polytechnic University, 2014.

Yu. B. Kolesov and Yu. B. Senichenkov. Object-oriented modeling with Rand Model Designer. *The papers of annual scientific conference.* St. Petersburg, Peter the Great Polytechnic University, 6-12, 2015.

Yu. B. Kolesov and Yu. B. Senichenkov. *Mathematical modeling. Component technologies.* St. Petersburg, Peter the Great Polytechnic University, 2013

Yu. B. Kolesov and Yu. B. Senichenkov. *Mathematical modeling of hybrid dynamical systems.* St. Petersburg, Peter the Great Polytechnic University, 2014.

J. Lygeros, C. Tomlin, and S. Sastry. *Hybrid systems: Modeling, analysis and control.* http://inst.cs.berkeley.edu/~ee291e/sp09/handouts/book.pdf, 2008.

C. Martin-Villalba, A. Urquia, Y. Senichenkov, and Y. Kolesov. Two approaches to facilitate virtual lab implementation. *Computing in Science and Engineering* 16 (1): 78 – 86, 2014.

Eva M Navarro-López,and Rebekah Carter. Hybrid automata: an insight into the discrete abstraction of discontinuous systems. *Int. J. Syst. Sci.* 42(11): 1883-1898 2011.

S. Pissanetzky. *Sparse matrix technology.* Academic Press Inc. London, 1984.

J. R. Rice. *Matrix computations and mathematical software.* McGraw-Hill Book Company, New York, 1981.

J. Rumbauth, I. Jacobson and G. Booch. *The unified modeling language. Reference manual.* Second edition. Addison-Wesley, 2005.

Ricardo G. Sanfelice and Andrew R. Teel. Dynamical properties of hybrid systems simulators. *Automatica* 46(2): 239-248, 2010.

Y. Senichenkov. *Numerical modeling of hybrid systems. St. Petersburg,* Peter the Great Polytechnic University, 2004.

Y. Senichenkov, Y. Kolesov, and D. Inikhov. Rand Model Designer in Manufacturing Applications , *IFAC-PapersOnLine: Manufacturing Modelling, Management, and Control,* 7(1): 1572-1577, 2013.

L.F. Shampine, I. Gladwell, and S. Thompson. *Solving ODEs with Matlab.* Cambridge University Press, 2003.

L.F. Shampine and S. Thompson. Event location for ordinary differential equations. *Comput. Math. Appl*., 39:43-54, 2000.

L.F. Shampine, M.W. Reichelt, and J.A. Kierzenka. Solving index-1 DAEs in Matlab and Simulink. *SIAM review*, 41: 538-552, 1999.

L.F. Shampine and Mark W. Reichelt. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, 18(1): 1-22, 1997.

Lena Wunderlich. *Analysis and numerical solution of structured and switched differential-algebraic systems.* Berlin: TU Berlin, Fakultät II, Mathematik und Naturwissenschaften (Diss.), 2008.