Preprints of the 2013 IFAC Conference on Manufacturing Modelling, Management, and Control, Saint Petersburg State University and Saint Petersburg National Research University of Information Technologies, Mechanics, and Optics, Saint Petersburg, Russia, June 19-21, 2013

Using OpenMVLShell in Research and Education

A. A. Isakov.

Distributed Computing and Networking Department, St. Petersburg State Polytechnical University, St. Petersburg, Russia (e-mail: isak239@mail.ru).

Abstract: *OpenMVL Project* is considered by author as the starting point of discussion and the first executed step in the direction of the new standard for tools for modeling and simulation of complex dynamical systems (CDS). *OpenMVLShell* is the open source software for comparison different approaches to the problem of designing modern environments for CDS modeling and simulation first of all. However it may be used in Education too as its analog OpenModelica. *OpenMVLShell* is based on Object-Oriented Modeling (OOM) approach and uses Model Vision Language (MVL) suggested by MvStudiumGroup. MVL is object-oriented language for modeling and simulation of hierarchical multi-component systems with event-driven behavior based on differential-algebraic equations (hybrid systems). Possibilities of *OpenMVLShell* are illustrated by two examples. The first one demonstrates how to use *OpenMVLShell* for comparison different DAE solves. The second one illustrates using *OpenMVLShell* in «Modeling and Simulation» course as instrument of active learning.

Keywords: simulators, standards, package design, simulation languages, object modeling techniques, problem solvers, iterative methods, factorization methods.

1. INTRODUCTION

Visual environments for modeling and simulation of complex dynamical systems are widely used in industry, scientific research, and education. Simulink, StateFlow, Simmechanics, SimPowerSystems (MathWorks), MvStudium, Rand Model Designer (MvStudium Group), Dymola, Ptolemy are the only examples from the long list of tools used in practice.

In industry these tools are used for traditional investigation models of object for study and collective designing new end products by teams of developers using «executable specification» technology for example. Industry is interested in new forms of graphical modeling languages, design technologies, applied libraries, instruments for debugging and visualizing.

Researchers are more interested in manifold types of mathematical models, numerical libraries, technologies for computational experiments and processing means.

In education modeling and simulation are the base methods for active learning. Tools for active learning should have intuitively clear modeling languages, be carefully documented, and should be usability.

A problem of choosing between tools for practical work is very important for all groups of users. For industry this problem is particularly significant because good choice has an effect on product cost and quality. It is nice if the choice is not adventitious and based on knowledge and comparison of tool's properties [Breitenecker, Proper, 2009].

Comparison of tools from the point of view of their practicality and effectiveness for manufacturing and education is difficult because it depends on human factors. But we can speak about used algorithms, numerical methods and their measurable characteristics.

A visual environment for modeling and simulation of complex dynamical systems may be considered as the special type of numerical software. From this point of view a tool builds and solves numericaly a set of differential-algebraic equations described behavior of complex device in different modes. We want to be sure that for any computer model (at least for test problems) the result of modeling and simulation for different tools will be approximate. A process of modeling and simulation of complex dynamical system may be divided into following stages:

1) Building a current final system using current topological and component equations;
2) Reducing or approximation of a current final system;
3) Solving a current system with the help of program realization of numerical method;
4) Visualizing of behavior (current trajectory).

A developer of commercial, research or education tool may choose any algorithms and methods for solving concrete mathematical problem and don't have to describe details (trade secret). But the results of any stage have to be available for a user in standard (approved) form. If program text will be available for all comers it will be even better.

OpenMVL Project is considered by author as the starting point of discussion and the first executed step in the direction of the new standard for tools for modeling and simulation of complex dynamical systems. *OpenMVLShell* is the open source software for comparison different approaches to the problem of designing modern environments for CDS modelling and simulation first of all.

2. OPENMVL PROJECT

2.1 Prototype of a standard

History of numerical software development gives a few «standards» for systematized collections (LINPACK), libraries (NAG, IMSL). All of them are based on:

- problem classification;
- methods classification;
- demands to numerical software;
- numerical software classification.

In our case the standard may include:

- model's type classification (nonlinear algebraic equation, ordinarily differential equations, algebraic-differential equations);
- methods of decomposition and aggregation of complex models (oriented and non-oriented components (bonds));
- allowable methods of transformation, reducing, and approximation of models;
- demands to numerical software;
- list of required solvers;
- demands to results of modeling and simulation (solved system, numerical solution, dates for visualizing).

The tool named *OpenMVLShell* based on prototype of declared standard was developed by the author. *OpenMVLShell* project is «open source» project and may be used and modified by any interested user.

The detailed description of *OpenMVLShell* may be found in [Isakov, 2010, 2011, 2012].

2.2 OpenModelicaShell and OpenMVLShell

OpenMVLShell environment based on and extends OpenModelica approach.

OpenModelica Project [Fritzson, 2006]:

- is free;
- first versions of OpenModelica preserved all features needed for object-oriented modeling but its modeling language was available only in text (not graphical) form;
- OpenModelica environment allows simulations and visualises results with the help of plots. It is enough for Education.

OpenMVL Project:

- is open source project;
- uses Model Vision Language instead of Modelica. MVL based on object-oriented approach as Modelica, allows to model hybrid systems with no limitation on form of local behavior, and follows UML.

OpenMVL Project supports all stages of computational experiment and has following components:

- Model Editor;
- Interactive main component with model engine for getting user's parameters and running application;
- Test-bench for numerical experiments and visualizing of result with the help of time and phase diagrams.

2.3 OpenMVLShell Current status

1 The Environment Structure and Components are shown in Fig. 1.

2 The list of main mathematical problems should be solved is formulated.

3 The site of Distributed Computing and Networking Department is used as temporary project's site. The web-page contains project documentation, description of mini-projects and is used for dialog with users.

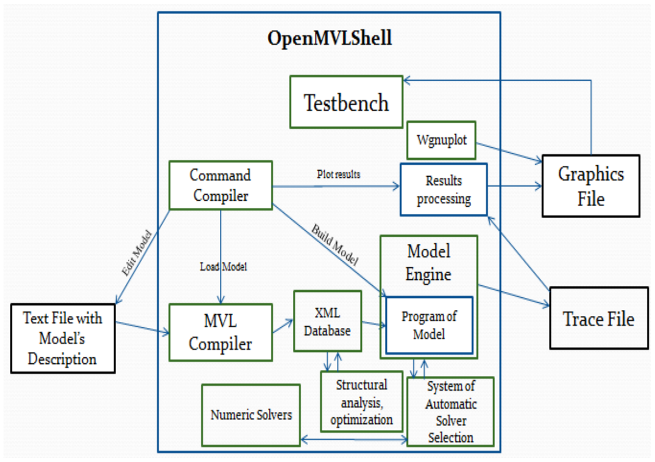

Fig 1. OpenMVLShell*'s* structure and components

2.3 Using OpenMVLShell in education. Possible items for students and post-graduate students.

OpenMVLShell project is realized now by students of Distributed Computing and Networking Department of State Polytechnical University. The following is the list of discussed problems:

1 Building final system

a. Structural Analysis – structurally degenerate system;

b. Structural Analysis - algorithms for analyzing and transforming the structure of final system;

2 Reducing and approximation of model;

a. Symbolic differentiation;

b. Minimalizing dimension of solved system;

3 Automation of computational experiments

a. Visual languages for computational experiments;

b. Visual debuggers

c. Generators of Tests problems.

4 Numerical methods and numerical software

a. Switching point problem for hybrid systems;

b. preconditioners for iterative methods;

c. converting of models build by different tools.

3. RESEARCH. THE ITERATIVE DASPK's METHODS

OpenMVLShell has its own numerical library in which were used LSODI, DASSL, and RADAU as DAE's solvers till now. It was interesting to compare these solvers with DASPK [Ascher, Petzold, 1998] because last one allows more general form of differential-algebraic equations and uses Krylov subspace projection methods for solving linear systems of algebraic equations [Brown et al., 1993]. DASPK was embedded in *OpenMVL* numerical library. Using special class of test it is shown that DASPK's iterative methods with different preconditioners may be recommended for simulation large scale models.

The following is an example of made investigation with the help of *OpenMVLShell* . *OpenMVLShell's* numerical library was filled up by DASPK. In DASPK is acceptable general form of differential–algebraic equations [Li, Petzold, 1999]:

$$G(x', x, time) = 0, x(0) = x_0\ x \in \Re^n \qquad (1)$$

Additionally, it is possible side by side with direct method for solving linear systems of algebraic equations using iterative methods, that are Krylov subspace projection methods with different types of preconditioners**.** Till now into *OpenMVLShell's* numerical library were embedded LSODI, DASSL, and RADAU as DAE's solvers. It was interesting to compare these solvers with DASPK for large scale systems that usually have sparse Jacobi matrix. Instead of original variant of LSODI from *ODEPACK* [Hindmarsh, 1983] was used modified variant with *MA28* [DUFF, 1977] as the solver for sparse linear systems of algebraic equations.

DASPK offers four types of preconditioners based on partial LU-decomposition [Saad, 1994]: ILU zero level $ILU(0)$; ILU first level – $ILU(1)$; $ILUT(\tau)$, partial LU-decomposition with threshold technics; $ILUTP(\tau, p)$, modified partial LU-decomposition with threshold technics $ILUT(\tau)$, taking account of size of diagonal elements [Saad, 1994].

3.1 Test problem

Generator of matrix with given eigenvalues. In [Senichenkov, 1983], it was suggested to generate matrix with given eigenvalues with the help of:

$$A = S \cdot \Lambda \cdot S^{-1},$$

$$S = E - u^T \cdot v, S^{-1} = E + u^T \cdot v,$$

where E – identity matrix,

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & . & . & . & . & . & . & . \\ 0 & 0 & . & . & . & . & . & . & . & . \\ 0 & . & . & \lambda_k & . & . & . & . & . & . \\ 0 & . & . & . & \eta & 1 & 0 & . & . & . \\ 0 & . & . & . & 0 & \eta & 1 & \Big\}\text{m lines} & & . \\ 0 & . & . & . & 0 & 0 & \eta & . & . & . \\ 0 & . & . & . & . & . & . & . & & \\ 0 & . & . & . & . & . & . & . & \mu & \nu \\ 0 & . & . & . & . & . & . & . & -\nu & \mu \end{pmatrix}$$

u , v vector-columns and $(u, v) = 0$.

Hereby generated matrix A will have: $\lambda_1, \lambda_2, ..., \lambda_k$ – k different real eigenvalues, m – multiply real eigenvalues (m is a dimension of diagonal block), $\mu \pm i\nu$ –complex eigenvalues. It was shown [Senichenkov, 1983], that if vectors u and v have integer components and eigenvalues are limited by value Lmax(n), generated matrix A will have exactly given eigenvalues.

Generator of DAE. Instead of general DAE form (1) let us consider form used in LSODI:

$$\begin{cases} G_d(x'_d, x_d, x_a, t) = 0 \\ G_a(x_d, x_a, t) = 0 \end{cases}, \qquad x_d(0) = x_{d0}, \qquad (2)$$

$$x = \begin{bmatrix} x_d \\ x_a \end{bmatrix} \in \Re^n, t - время$$

where x_d – differential unknowns, x_a – algebraic unknowns. It is enough specify initial conditions for differential unknowns x_{d0}, then vector x_{a0} may be found as a solution of algebraic equations G_a. This operation is called coincidence of initial conditions. LSODI and DASPK can do it. So it possible to compare not only task time for DAE but for coincidence of initial conditions too.

It was generated two test problems. Both problems are used hybrid automation (Fig. 2).

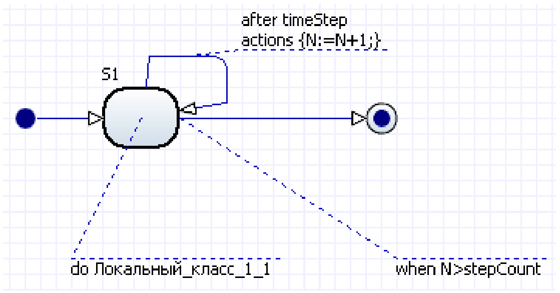

Fig. 2. Hybrid automation for test problems.

Here - *N* counter of runs, *stepCount* – number of runs in experiment, *timeStep* – task time of a run. Do-activity of the automation is described as test DAE in general form (3).

The first experiment compares task time of coincidence of initial conditions for LSODI and DASPK. The value of *timeStep* is small enough (0.01) and both packages are repeatedly consistent initial conditions.

Test DAE is written in the form:

$$\begin{cases} \dfrac{dx}{dt} = Ax + Cy + d \\ By = f \\ x(0) = x_0 \end{cases}, \qquad (3)$$

$$x, d \in \Re^{n_d}; y, f \in \Re^{n_a}; A \in \Re^{n_d \times n_d};$$
$$C \in \Re^{n_d \times n_a}; B \in \Re^{n_a \times n_a}; t - time$$

where n_d – number of differential unknowns, n_a – number of algebraic unknowns. Number of differential unknowns in experiments is small (2), number of algebraic unknowns is modified from 10 till 500. Eigenvalues of a matrix B were driven parameters.

In the second experiment it was compared task time of solving DAE for LSODI and DASPK. Value of *timeStep* was equal to 10 s (model time). Test DAE has form:

$$\begin{cases} G_d(x'_d, x_d, x_a, t) = 0 \\ G_a(x_d, x_a, t) = 0 \end{cases}, \qquad x_d(0) = x_{d0}, \qquad (2)$$

$$x = \begin{bmatrix} x_d \\ x_a \end{bmatrix} \in \Re^n, t - время$$

Eigenvalues of a matrix A were driven parameters. LSODI and DASPK consider test problem (4) as DAE in general form in this experiment.

3.2 The results of experiments

It was interesting in what way task time of solving systems of linear equations in DASPK depends on:

1 eigenvalues of A and B (condition number for B, stiffness for A);

2 values of absolute and relative tolerances for DAE's solvers;

3 sparseness of A and B;

4 type of preconditioner.

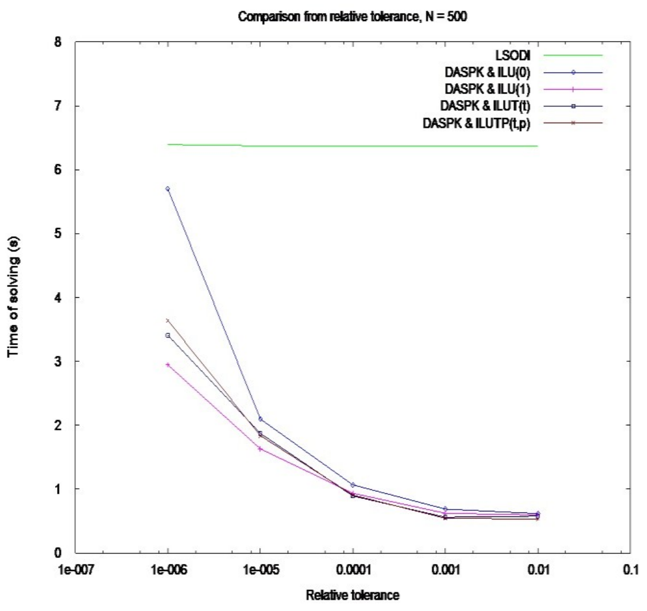

Fig. 3. Task time – tolerance diagram.

The expected outcomes of experiment for used computer are:

1 Time task of iterative solution of linear systems (3) (Fig. 3, problem dimension is equal 500) strongly depends on driven tolerance as expected.

2 It is possible as always to build test problem with large condition number for which DASPK failures.

3 It is no sense to use iterative methods with preconditioners for small dimensions. Overhead expenses on building preconditioner are too expensive (Fig. 4).

For used test problems ILUTP preconditioner came out on top but ILU(0) is preferable for small tolerances.

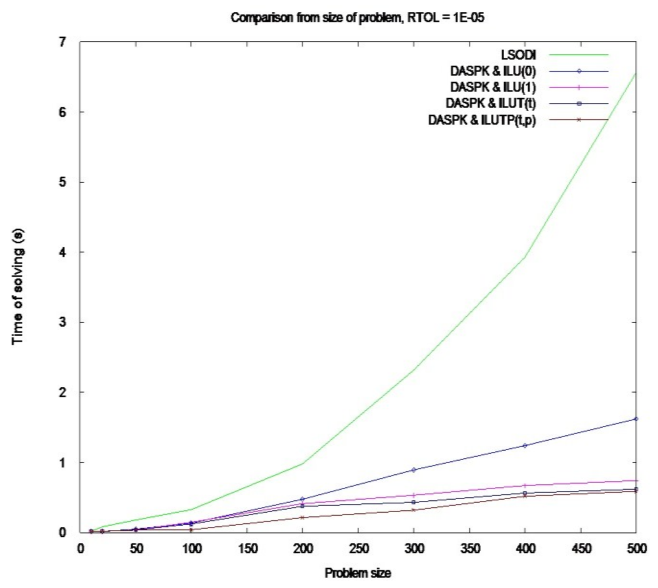

Fig 4. Time task-dimension diagram.

It is important to underline that these experiments are only example of using *OpenMVLShell* for research and comparison of DAE's solvers. In our case all information about the environment is available to researcher and he can plan his experiments as he wants taking in account all details.

All newcomers can reproduce and modifier results.

4. EDUCATION. PRACTICAL WORKS FOR «SYSTEM MODELING» COURSE

OpenMVLShell is used at Distributed Computing and Networking Department of state Saint Petersburg Polytechnical University (Russia) in courses «Modeling of systems», «Component modeling», «Modeling and simulation of complex dynamical systems» for practical works. Also it used by post graduate students for research.

OpenMVLShell for practical works is used as a free tool for modeling and simulation and as a collection of numerical methods.

Future developers of software for modeling and simulation of complex dynamical systems may use *OpenMVLShell* as a kit. They can choose an existing component and modify it or develop and add a new component to current variant of *OpenMVLShell*. This way needs modifying, testing and comparison a new version with existing one.

Students work as a team. Each team member has his own task, plan and operating schedule. The team starts with seminars. The goal of seminars is to present existing components of *OpenMVLShell* and argue their role in future project. The best way to understand tool's facility is to build a models using modern object-oriented modeling language, in our case it is MVL. During seminars students discus the language properties and illustrate them by examples. After that they begin discussing the Project, form the list of tasks and allocate duties. From this moment the head of the team organizes and controls the workflow using project management system Redmine. Redmine system has features used in OpenMVL project:

- Multiple projects
- Flexible role-based access control
- Flexible issue tracking system
- Gantt chart and calendar
- News, documents & files management
- Feeds & e-mail notifications.
- Project's wiki
- Project's forum
- Simple time tracking functionality
- Custom fields for issues, time-entries, projects and users
- SVN integration
- Multiple LDAP authentication
- User self-registration
- Multiple languages
- Multiple databases
- REST API

OpenMVLShell and suggested by author variant of active learning process was presented at «Innovation technique in Education» [Isakov, 2012]. «Innovation technique in Education» is All-Russian annual student competition. The presented paper was one of the best in the nomination.

There is experience using *OpenMVLShell* in the course «Methods of optimization». A task was to modify *OpenMVLShell* (modeling language and methods) for solving parametric optimization problems [Isakov et al., 2012].

5. OPENMVLSHELL AS DEBUGGING TOOL FOR AN ENGINEER.

Designing a new applied library is a hard formalizable process demanding special low level methods for debugging. Errors may arise while modeling and simulation simultaneously. Errors of model developer might be accompanied by tool's errors. Tool's built-in visual debuggers are helpless in this case. The source of the tool is inaccessible. Tool' developers assistance of is not easily accessible.

OpenMVLShell has open source for algorithms and numerical methods used in RMD. It is possible to use it for traditional low level debugging.

To do this developer of a model may:

1 Design a new model using RMD.

2 Convert its graphical specification in MVL.

3 Download MVL text representation of the model in OpenMVLShell, and build executable model.

4 Run model and start debugging under standard debugger.

6. CONCLUSIONS

Scientific, educational, and manufacturing problems solving with the help of modern tools become more complex and manifold. It is time to discuss and adopt next standard for universal tools based on modern technologies for modeling and simulation of complex dynamical systems.

OpenMVL is open source project which is now already used for Research and Education.

We hope that step by step it will transforms in full-fledged open laboratory for students and developers of tools for modeling and simulation of complex dynamical systems.

REFERENCES

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

Breitenecker F., Proper N. (2009). Classification and evaluation of features in advanced simulators. Proceedings MATHMOD 09 Vienna, Full papers CD Volume.

Brown P.N., Hindmarsh A.G., Petzold L.R. (1993). Using Krylov methods in solution of large-scale differential-algebraic systems. Technical report. Numerical Mathematics Group, UCRL-JC-113507.

DUFF I.S. (1977). MA28—a set of FORTRAN subroutines for sparse unsymmetric linear equations.

Fritzson P. (2006). Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press.

Hindmarsh A.C. (1983). ODEPACK. A Systematized Collection of ODE Solvers in Scientific Computing, R. S. Stepleman, editor.

Isakov A.A., Senichenkov Y.B.. (2010). OpenMVL – visual modelling package. Works of international scientific and technical conference “XXXIX week of science”, Saint-Petersburg Polytechnical State University – pp. 151-153.

Isakov A.A., Senichenkov Y.B.. (2011). Program Complex OpenMVL for Modelling Complex Dynamic Systems. Electronic Journal «Differential Equations and Control Processes», Mathematics and Mechanics Faculty of Saint-Petersburg State University.

Isakov A.A. (2012). OpenMVL – visual modelling scientific package. All-Russian conference «Innovation Technologies in Educational process», Belgorod State National Research University – 2012 – pp. 11-15.

Isakov A.A., Gorbunov A. Y., Markov A. O., Mishina A. C., Pchelko P. V., Toptygin A. V., Chugreev D. A. (2012). Component of numeric optimization for computer modeling system OpenMVLShell // All-Russian competition research in computer science and information technology, Belgorod State National Research University – pp. 622-625.

Isakov A.A. (2012). Open project OpenMVL – designer of visual simulators // Materials of international seminar ―COMOD 2012", SPb: SPbSTU publishers. – pp. 21-25.

Isakov A.A. (2012). Experimental shell in OpenMVL project // Computer tools in Education, №5 – cc. 33-41.

Li S., Petzold L. (1999) Design of new DASPK for sensitivity analysis. S. Technical Report, Department of Computer Science, University of California Santa Barbara.

Saad Y. (1994). ILUT: A dual threshold incomplete LU factorization, Numer. Linear Algebra Appl. 1, 387.

Senichenkov Y.B. (1983). Test liner systems of algebraic and differential equations / LPI Transactions, Leningrad., № 391, pp. 84-87.

http://www.mvstudium.com, http://www.rand-service.com – official site of package “Rand Model Designer”.

OpenMVL project http://dcn.ftk.spbstu.ru/index.php?id=276.

http://www.redmine.org/projects/redmine/wiki. Official site of Redmine project.