Instrumental Analysis of Hybrid Systems with PDE

Yu.V. Shornikov, A.V. Bessonov, M.S. Myssak, D.N. Dostovalov

***Abstract*—** A class of hybrid systems (HS) with modal behavior specified on the class of systems with partial differential equations (PDE) is considered. Architecture of instrumental environment for simulation of complex dynamic and hybrid systems is given. Syntax-oriented facilities for the considered problems specification are proposed. Algorithms of finite difference method for the transition from PDE to ordinary differential equations system are given. Universal data structure for storing HS models has been designed and proved. Algorithms of variable step with accuracy and stability control of numerical scheme for solving high-dimensional Cauchy problems are proposed. The algorithms are based on explicit methods of Runge-Kutta type. Sequential and parallel implementation of numerical methods is presented. The example of specification and analysis of reaction-diffusion problem is given.

***Keywords*—** Autogenerated parsers, grammar, hybrid system, Runge-Kutta methods, sequential and parallel implementation, software architecture, system of PDE.

I. Introduction

MANY engineering problems are characterized by points of discontinuity in the first derivative of the phase variables. Such combined discrete-continuous problems called hybrid or switched systems [1], [2]. In this paper we propose a new system architecture of ISMA (that in Russian means Instrumental Facilities of Machine Analysis) based on the methodology of hybrid systems (HS). This article examines a new application of ISMA – systems of partial differential equations (PDE) with constraints. PDEs are used to describe processes in the chemical-technological systems, elasticity problems, etc. To achieve this goal a series of consecutive problems is solved: the textual specification of ISMA environment models expanded by constructions describing PDEs; a special data structure for storing the model in memory is developed; approximation algorithms are implemented for transition from a system of PDE to ODE system. In many cases, the problem is compounded by high dimensionality and stiffness of the considered system. To solve moderately stiff problems integration algorithms based on the explicit methods to control accuracy and stability of the numerical scheme can be applied [3], [4]. Furthermore when problem dimension reaches several thousands of equations and more its calculation by sequential methods becomes ineffective and requires the use of multiprocessor computer systems. In this situation parallel computation of local behaviors using cluster technologies can significantly improve the quality and efficiency of calculations. This paper discusses sequential and parallel implementation of algorithms of variable step based on two-stage and three-stage schemes of Runge-Kutta type of respectively second and third order of accuracy. These integration algorithms are well suited for solving hybrid problems including moderately stiff problems. As an example for specification and comparative analysis of the considered algorithms the reaction-diffusion problem [5, 6] is examined.

This work was supported by grant 14-01-00047-a from the Russian Foundation for Basic Research, RAS Presidium project № 15.4 "Mathematical modeling, analysis and optimization of hybrid systems".

Yu.V. Shornikov is with the Design Technological Institute of Digital Techniques Siberian Branch of Russian Academy of Science, Novosibirsk, Russia (e-mail: shornikov@inbox.ru).

A.V. Bessonov, M.S. Myssak, D.N. Dostovalov is with the Department of Automated Control Systems, Novosibirsk State Technical University, Novosibirsk, Russia (e-mails: abv.poste@gmail.com, maria_myssak@mail.ru, dostovalov.dmitr@mail.ru).

II. Class of Systems

There are many systems (mechanical, electrical, chemical, biological, etc.), the behavior of which can be conveniently described as a sequential change of continuous modes [1]. Each mode is given by a set of differential-algebraic equations with the following constraints:

$$\begin{aligned} &y' = f(x, y, t), x = \varphi(x, y, t), \\ &pr : g(x, y, t) < 0, \\ &t \in [t_0, t_k], x(t_0) = x_0, y(t_0) = y_0, \\ &x \in R^{N_x}, y \in R^{N_y}, t \in R, \\ &f : R^{N_x} \times R^{N_y} \times R \to R^{N_y}, \\ &\varphi : R^{N_x} \times R^{N_y} \times R \to R^{Nx}, \\ &g : R^{N_x} \times R^{N_y} \times R \to R^{S}. \end{aligned} \quad (1)$$

The vector-function $g(x,y,t)$ is referred to as event function or guard [2]. A predicate pr determines the conditions of existence in the corresponding mode or state. Inequality $g(x,y,t) < 0$ means that the phase trajectory in the current mode should not cross the border $g(x,y,t) = 0$. Events occurring in violation of this condition and leading to transition into another mode without crossing the border are

referred to as one-sided.

This class of systems is expanding by the addition of boundary conditions for PDEs. Continuous behavior of HS is determined by the systems differential-algebraic equations and PDEs. In the proposed implementation the equations with the maximal order not higher than second are considered. Applied algorithms do not impose a restriction on number of variables – i.e. their number is theoretically unlimited. Nevertheless should take into account that the introduction of each new variable leads to a tremendous increase in the number of equations generated as a result of the finite differences method. Therefore the real limit on the number of variables is imposed by computing resources: computer software as well as computer itself. The considered equations are nonlinear type. The linear equation are also supported and regarded as a narrow equations type.

In this paper we consider the type of nonhomogeneous PDE. The coefficients used in partial differential equations considered in this paper can be either constant or variable. This paper discusses the parabolic type equations. Boundary conditions of considered problems must be rectangular area Ω – rectangle, parallelepiped, etc.

Thus, the proposed expansion of the instrumental environment designed for the analysis of PDE type equation (2): heterogeneous, non-linear, second order, with constant and variable coefficients, with an unlimited number of variables N and limited by N-dimensional rectangular grid.

$$
\begin{aligned}
&\frac{\partial z}{\partial t} = \psi\left(x, z, t, p, \frac{\partial z}{\partial p}, \frac{\partial^2 z}{\partial p^2} \right), \\
&x = \varphi(x, t), \\
&pr : g(x, t) < 0, \\
&x(t_0, p) = x_0, z(t_0, p) = z_0, \\
&t \in [t_0, t_k], p \in [p_0, p_m], \\
&\left.\frac{\partial z}{\partial n}\right|_p = \tilde{z}(p), \\
&x \in R^{N_x}, z \in R^{N_z}, t \in R, p \in R^{N_p}, \\
&\varphi : R^{N_x} \times R \to R^{N_x}, \\
&\psi : R^{N_x} \times R^{N_z} \times R \times R^{N_p} \times R^{2N_p} \to R^{N_z}, \\
&g : R^{N_x} \times R \to R^{S},
\end{aligned}
\tag{2}
$$

where n denotes the normal to the boundary and $\tilde{z}(p)$ is a given function to the boundary.

III. ISMA Simulation Environment

Simulation environment of complex dynamical and hybrid systems called ISMA is developed at the department of Automated control systems of Novosibirsk state technical university (Russia) [7].

Specification of hybrid systems is carried out using graphical and symbolic languages that are the system content of instrumental environment. Analytical content is provided by numerical methods and algorithms for computer analysis corresponding to the chosen class of systems and methods for solving these models. ISMA environment is developed subject to simplicity of description of dynamical and hybrid models in the language that is maximally close to the object language. Main features of ISMA are the following:

1) Composition of hybrid models is carried out in visual structural-textual form;
2) Structural form of model description corresponds to the classical description of systems by block diagrams and includes all necessary components such as integrators, accumulators, amplifiers, signal sources, nonlinear elements and others;
3) Language of symbolic specification is approached maximal to the language of mathematical formulas;
4) Special module for specification of problems of chemical kinetics in the language of chemical reactions which automatically translates them into a system of differential equations;
5) A variety of traditional and original numerical methods included methods that are intended for the analysis of ODE systems of medium and high stiffness;
6) Computer simulation in real time;
7) Graphic interpreter called GRIN provides a wide range of tools for analysis and visualization of simulation results such as scaling, tracing, optimization, displaying in the logarithmic scale and phase plane;
8) Extension of system functionality by adding new typical components and numerical methods.

Architecture of ISMA software package (Fig. 1) is designed to unify existing mathematical program software for analysis of problems in various object domains: chemical kinetics, automation, electricity, etc. Blocks that belong to this paper marked with a dark color.

IV. A New Description Language of HS Models with PDEs

One of the many approaches used is ISMA to describe HS models is textual representation. For this purpose a special language LISMA (Language of ISMA) is developed [8].

Grammar of LISMA corresponds to the second type according to the Chomsky classification and is context-free. This means that the left side of all productions are represented by single nonterminals which are objects that represents an entity of language (eg: a formula, an arithmetic expression, a command) and do not have a specific character value.

All nonterminals of LISMA grammar can be divided into the following groups:

- main group is the main unit that is the body of programm model and includes all the remaining units;
- expression is a number of nonterminals used to describe algebraic expressions, logical expressions, operators and operands;

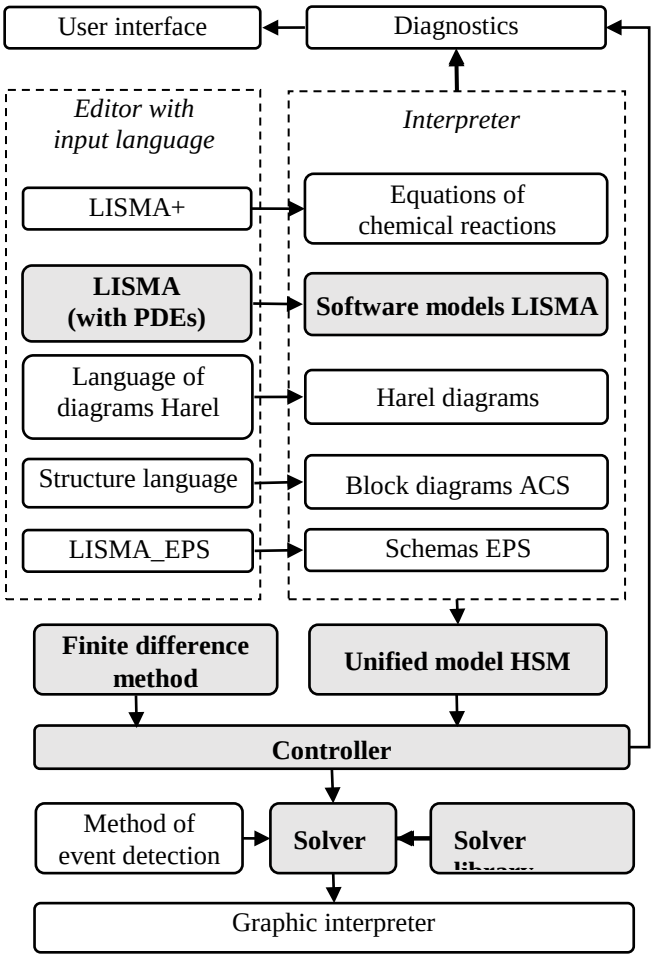

Fig. 1. Architecture of ISMA

• function is a call of embedded function which are an integral part of any mathematical expression(eg: tg, sin, pow, swrt, etc.);

• constant is constants. Values can be defined by expression that is evaluated at the stage of semantic analysis;

• algebraic equations are algebraic equations whose solution of is performed at each step of the model calculation;

• differential equations and initial condition – these non-terminals are used for the description of the Cauchy problem;

• state is a unit describing the discrete behavior of HS. Includes state and transition condition to this state;

• setter are instant actions performed at the moment of HS state change;

• macros is a macrounit of mathematical model, which substitution is performed at the preprocessing stage;

• partial differential equations are facilities to work with PDE which are described in more details hereinafter.

However existing description tools are not suitable enough to model boundary value problems of PDE systems. Therefore new language structures are introduced to describe specific elements. Before the development of grammar of language elements a comparative analysis of multiple peers was made. In these grammars the following characteristics were emphasized: flexibility and extensibility, usability, ease of perception, the corresponding mathematical description. Using these criteria multiple languages were evaluated with the ability to describe models with PDE equations. The main ones are the following: FlexPDE, Wolfram Mathematica, gPROMS, EMSO, ASCEND. In many languages to describe the partial derivative functional style is used, in which the derivative is a function of several arguments. Based on review and analysis of mentioned simulation environments LISMA has been extended by description of boundary value problems with PDEs. This extended grammar was developed in the ideology of inheritance.

To describe a system of differential equations, boundary conditions and initial values a new LISMA language features are introduced. Explicit declaration of variables that should be subjected to discretization is introduced. For this purpose a special structure is used with grammar written in the Backus-Naur form as follows:

apx _var → **'var'** *var_ident* **' ['**
DecimalLiteral **',' ***DecimalLiteral* **']'**
apx_var_tail **';'**

apx_var_tail → **'apx'** *DecimalLiteral*
| **'step'** (*FloatingPointLiteral* | *DecimalLiteral*)

For example, the following expression corresponds to the given grammar:

var x[0, 20] **apx** 30;

var y[0, 30] **step** 0.5;

In this structure boundaries of the variable are defined in square brackets. The following constructions are the keyword **apx** (short for approximation) or the keyword **step**. Keyword **apx** used if we want to break the considered domain of definition for a fixed number of segments, thus realizing the priority execution speed. We use the keyword **step** if step size is important – an accuracy priority. Number of segments and step size are written following the keyword.

Several elements are introduced in the description of the equations. First, it is an explicit indication of variables that affect the equation on the left side of the equation. This record type is optional and can be omitted as before on the left side to specify variables in parentheses. Partial derivatives are described in a functional style. Letter **D** is used as function name – the most concise version, which is not lost in code, mostly lowercase. The arguments used name of a differentiable function, the variable on which the derivative is taken and the order of the derivative. If the derivative is first-order, the latter argument is omitted suggesting that it is taken equals to 1 by default. This approach to the description of the derivative satisfies all the previously entered criteria: it allows you to use all sorts of variables which should be differentiated; it does not contain descriptive information duplication and laconic, and thus it is practical and easy to perception; and finally it is easy to relate to mathematical description. As a result, the description of the PDE as follows:

partial_operand → **'D' '(**' *Identifier* **','** *Identifier*
(**','** *DecimalLiteral*)? **')'**

Below is an example of this language construct:

```
c1' = Kh*D(c1,x,2) + D(Kv*D(c1,z), z);
c2' = D(c1,x,2)*pow(x,2);
```

For numerical solution of PDEs by FDM is necessary to determine the boundary conditions – values of the derivative at the edges of the grid under consideration. It looks as follows:

edge → **'edge'** *edge_eq* **'on'** *Identifier edge_side* **';'**
edge_eq → *Identifier* **'='**
(*FloatingPointLiteral* | *DecimalLiteral*)
edge_side → **'left'** | **'right'** | **'both'**

Construction contains a partial derivative equation with a certain value on the right side. This allows you to specify a description for a particular variable value and the type of border: left, right or both. Below is an example of all three types of boundary conditions:

```
edge D(c1, x) = 0 on left;
edge D(c1, x) = 20 on right;
edge D(c1, y) =0 on both;
```

LISMA grammar is designed in such way that the choice of the inference rule requires only the two foolowing symbols of analyzed chain. Therefore the grammar of LISMA corresponds to the LL(2) type.

Lexical and syntactic analyzers for modified language are developed using the library Antlr4. This library allows to build parser and lexer in JAVA on the description of LL(*)-grammaer in language close to EBNF. This library is ideal for problems of fast and efficient automatic construction of parsers. Under the ISMA project it is decided to use antlr4, because it is quite applicable to the existing grammar and own work to create parsers require significant investment in the development and documentation. In addition, experience in using Antlr4 can be successfully applied in other projects.

To work with ANTLR LISMA grammar is described in a special file of .g4 format containing elements of lexer (keywords, identifiers, numers, etc.) and elements of parser. ANTLR generates the classes required to java: LismaLexer is a lexical analyser, LismaParser is a syntax analyser, LismaBaseVisitor is the instrument for convenient bypassing of syntax tree. MacrosPreparser is the instrument preprocessing the input model and generating macro. It is used before parsing the model in ISMA. Then the model is partitioned to tokens by lexical analyzer and the tree obtained as the result of parsing is bypassed. The last stage is the semantic analysis in which the check, the interpretation of model parts and the fill of data structure in memory are made. For this stage a number of tools is developed that reflects different aspects of modeling in ISMA. Let us consider them in the order in which them are used in semantic analysis:

- StateVisitor fills information about hybrid system states;
- VariableTableAggregatorVisitor searches variables and fills tables of model variables in whole and for each state in particular;
- PopulateExpressionVisitor generates right sides of all equations based on variables registered on the previous step;
- PopulateStateInfoVisitor build the map of state changes (transition conditions between certain states);
- BoundInfoVisitor forms the boundary conditions of model with PDE;
- CalcConstVisitor calculates constants and initial conditions of Cauchy problem.

V. Unified Description Section in ISMA Environment

After the language grammar is defined, a data structure for storing models in memory after parsing should be developed. This data structure must be universal for all types of model specification. With several ways to describe HS – graphics, text or block diagrams, we should be able to lead each of them to a single universal form, which subsequently may be transmitted to solver input. This approach simplifies the task of unification of ISMA software. For a more specific application of the simulation environment ISMA it is necessary to develop a graphical part responsible only for model specification. In this description module the modules of calculation and graphical interpretation remain unchanged.

When designing such a data structure many factors should be taken into account. This is necessary to ensure that upon presentation of new conditions the whole system of modeling was not inapplicable. Thus, the developed data structure should be easily expandable. Adding new elements to the model should not be a need to rewrite large parts of the system. Furthermore, the model should excludes redundancy. The appearance of redundancy in the description of such complex structures as the model of a hybrid system with a differential-algebraic equations and PDEs is a potential source of errors. In addition, the model must be easily divided into blocks and must be easy to use. In order to accommodate these properties subject-oriented approach is chosen in the design of the data structure for storing HS model. The implementation language is Java.

The data structure represents a set of closely related classes which can be divided into four types (Fig. 2):

1) equation description classes;
2) expression description classes;
3) discrete behavior (states and transition conditions [9]) description classes;
4) entire model description classes.

Expressions are a sequence of elements called tokens. Tokens are both operands and operators. Two types of sequences to write expressions are supported: infix and postfix (inverse polish notation). Postfix notation is useful for

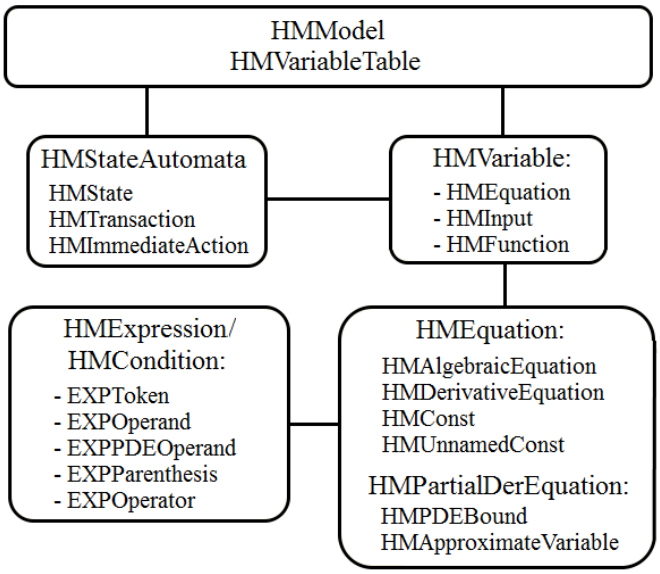

Fig. 2. Common HSM structure

computing values on the stack and is used for calculations in step semantic analysis to calculate values of the constants and initial conditions. Polish-inverted entry is also used in some versions of the solver ISMA to interpret the expression directly in the time of the model execution. If the expression contains only operators of algebraic is considered algebraic. If the expression contains logical operators it refers to a type of conditional expressions used in the description of the conditions of transitions between states. Algebraic expressions used in the description of the right sides of all considered types of equations: algebraic, differential, PDEs, and constants. All of them serve to describe continuous behavior of certain state of the hybrid system.

Within each state there exists the so-called variable table that stores equations and variables. For description of the states and conditions of the transitions between them responsible the appropriate type of classes. Collection of states, conditions of transitions between them, and a set of instantaneous action at the entrance to a condition is called hybrid automaton. Hybrid automaton has an initial state, which corresponds to the time t = 0. Complete set of all of the above classes called the HS model (HSM).

HSM also contains the table of variables which is the parent of the table of state variables. When calculating the right-hand side of the equation primarily the equations are searched in the table of current state variables and if the variable is not declared in the current state the search continues in parent table of variables of HSM. Therefore to change the mathematical model after state change it is enough to point the new current state.

Work with HSM is accompanied by a number of services:

• ConstValueCalculator is the service which calculates expressions in the right-hand side of constant at the stage of semantic analysis;

• HSM2TextDumpTranslator is the debugger of HSM representing the model in the computer memnory in textual form;

• Infix2PolizConverter/ Poliz2InfixConverter are converters of indix and postfix notation;

• Lisma2007Converter are translators of model to previous version of ISMA;

• FDMConverter, PDEInitialValueCalculator are services of PDE model analysis whose algorithms will be discussed below.

For solving PDE in the ISMA environment approximation of equations for difference grid by finite differences method [10] is used. This method is used to solve systems with rectangular boundaries (ex. hydrodynamics and electrodynamics). In such problems FDM has a sufficiently high accuracy. At the same time it wins in speed other methods. Consider the algorithm transition from of PDE to a system of ordinary differential equations.

Step 1. Construct a list of all variables for discretization. At this stage special structure for describing variables of storing type of discretization and accuracy (number of elements of grid) is analyzed.

Step 2. Divide equations into two groups: permanent and approximated. Here you need to select the equation that must be converted to the difference analogue – is analyzed right-hand sides and identifies those who are in the right part of the required variables.

Step 3. Construct N-dimensional grid. For all variables for sampling the number of elements for which they are divided is defined. The product of these values is the dimension of the grid.

The number of equations to be obtained by the algorithm corresponds to the equation:

$$N = n_p \prod_i n_i, \tag{3}$$

where N is the number of approximating equations, n_p is the number of PDEs, n_i, $i = 1,..,m$ is the number of grid points for each variable approximated.

Numerical approximation of derivatives is performed by the formulas:

$$\frac{\partial u_j}{\partial \zeta} = \frac{u_{j+1} - u_{j-1}}{2\Delta\zeta}, \frac{\partial^2 u_j}{\partial \zeta^2} = \frac{u_{j-1} - 2u_j + u_{j+1}}{(\Delta\zeta)^2}, 1 \le j \le N. \tag{4}$$

Step 4. Apply boundary conditions. For each variable boundary conditions under which the simulation and approximation takes place must be specified. At this stage the variables in the initial and final nodes of the grid (relative to the current variable) are replaced by the indicated boundary values. If not then the default is zero.

Step 5. Set initial conditions for all equations.

Step 6. Transition from the grid to the ODE system. Here you need to go over all grid points. Thus it is necessary to put a unique identifier for each new equation while maintaining a

connection that has been set by difference equations. As a result for each equation in each grid a copy of it in the system of ODE will be created and the initial condition (t = 0) will be specified.

VI. Scheme of Model Interpretation and Solving

Four basic levels of working with model can be destinguished: the interpretation of the input specification of the model, controller, solver, graphic interpreter. The Fig. 3 presents the first two levels in more detail.

Level of interpretation is responsible for converting the model described by the input specification of the universal representation HSM. After model is input to simulation environment in text specification before the numerical solution the model passes through a series of stages of analysis. The first two stages – the lexical and syntactic analysis. They are conducted by facilities of the parser generated library antlr4. If the model is correct, we have an abstract syntax tree (AST). Bypass AST and further retrieval of information allows you to fill a unified data structure a model description HS. This is done using a variety of services:

1) service of sequential approach syntax tree (design pattern "visitor");

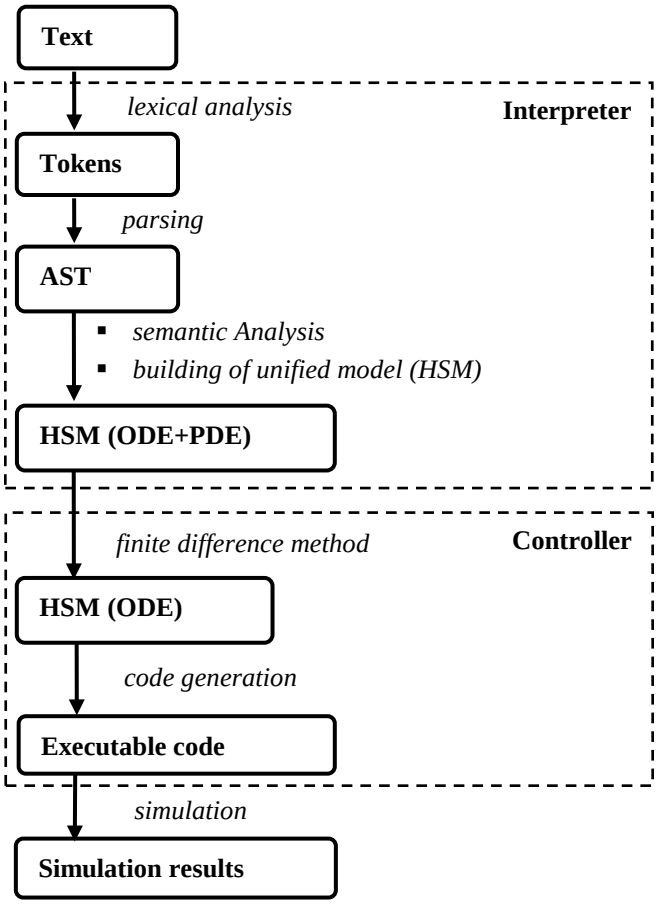

Fig. 3. Stages of interpreter and controller

2) conversion service from infix into postfix notation and vice versa;
3) value calculator of constants and initial conditions – a stack machine for the Polish-inverted notation;
4) service of model validation – semantic analysis.

In case the received data structure is correct in terms of semantics the interpretation is considered complete and can go to the level of controller.

Level of the controller is responsible for preparing the model HSM to the view which satisfies the conditions imposed by a solver. Also, the controller is responsible for loading of numerical methods, loading of library functions and other simulation settings. After the HSM model was obtained from the interpreter the finite difference method is performed according to the algorithm described earlier. As a result, the description of the PDE in the data structure is replaced by a system of ODE.

Generated data structure describing the internal representation of the hybrid system model based on the input specification of the model (text or graphics) should be counted and get simulation results. For this data structure is go to the input of the controller. The controller is a bonding layer, which produces all the necessary transformation and sets the simulation parameters. In addition, the controller is responsible for connection and control of library functions.

At this stage, the model is finally formed and fed to the input of the controller solver. Depending on the selected model parameters numerical method and the method of event detection HS is a calculation model. It is worth mentioning state changes in the calculation. ISMA environment has a unique method of detecting events [1] that increases the accuracy of solutions to rigid problems. Subsequent step in the simulation is the calculation. It is produced using a unique library of numerical methods ISMA. The simulation shows the resulting set of points on a plane integrated graphical visualization tools.

VII. Solvers

This section is devoted to the integration algorithms of variable step based on two-stage and three-stage explicit methods of Runge-Kutta type that implements schemes of second and third accuracy order respectively.

The algorithms are applied to numerical solving of Cauchy problem for ODE systems of the following form:

$$y' = f(y),\ y(t_0) = y_0,\ t_0 \le t \le t_k. \tag{5}$$

Consideration of autonomous problem does not reduce the generality because non-autonomous problem always can be cast to autonomous by introducing an additional variable.

Particular attention should be paid to the choice of the integration method. Fully implicit methods cannot be used because they require the calculation of $f(y)$ at a potentially dangerous area, where the model is not defined. Therefore here we will use explicit methods with solution: $y_{n+1} = y_n + h_{n+1}\varphi_n$, $n = 0, 1, 2, \ldots$ As a result we obtain the

dependence of the predicted integration step h_{n+1}.

Considering that explicit methods are known by poor stability this paper examines integration methods with accuracy and stability control. Generally accuracy and stability control are used to limit the size of the integration step. As a result projected step h_{n+1} is calculated as follows.

The choice of the next integration step size is based on the proved theorem [4] and can be written as follows:

$$h_{n+1} = \max\left[h_n, \min\left(h^{ac}, h^{st}\right)\right] \tag{6}$$

where h^{ac} and h^{st} are step sizes obtained as a result of accuracy control and stability control respectively. This formula allows to stabilize the step behavior in the area of solution establishing where stability plays a decisive role. Because the presence of this area severely limits the use of explicit methods for solving stiff problems.

A. Two-stage Runge-Kutta method

Suppose that for numerical solving of problem (5) the following implicit two-stage method of Runge-Kutta type is used:

$$\begin{aligned} &y_{n+1} = y_n + 0.5(k_1 + k_2),\\ &k_1 = h_n f(y_n),\\ &k_2 = h_n f(y_n + k_1). \end{aligned} \tag{7}$$

where y and f are real N-dimensional vector-functions, t is an argument, h is an integration step, k_1 and k_2 are method stages and 0.5 is a numerical coefficient defining accuracy and stability properties of (7).

Inequality for accuracy control has the following form:

$$0.5\|k_2 - k_1\| \le \varepsilon. \tag{8}$$

And inequality for stability control looks as follows:

$$v_n = 2 \max_{1\le i\le N}\left(\left|k_3^i - k_2^i\right| / \left|k_2^i - k_1^i\right|\right) \le 2, \tag{9}$$

where length of stability interval of the scheme is approximately equals to 2; k_1^i, k_2^i and k_3^i are the components of vectors k_1, k_2 and auxiliary vector k_3. Stage k_3 coincides with vector k_1 for next step and therefore does not lead to computational costs increasing.

B. Three-stage Runge-Kutta method

Consider implicit three-stage method of Runge-Kutta type for solving problem (5), which has the following form:

$$\begin{aligned} &y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{2}{3}k_2 + \frac{1}{6}k_3,\\ &k_1 = hf\left(y_n\right),\ k_2 = hf\left(y_n + 0.5k_1\right),\\ &k_3 = hf\left(y_n - k_1 + 2k_2\right). \end{aligned} \tag{10}$$

Inequality for accuracy control has the following form:

$$\left\|k_1 - 2k_2 + k_3\right\| \le 6\varepsilon. \tag{11}$$

And inequality for stability control looks as follows:

$$v_{n,3} = 0.5 \max_{1\le i\le N}\left(\left|k_1^i - 2k_2^i + k_3^i\right| \Big/ \left|k_2^i - k_1^i\right|\right) \le 2.5. \tag{12}$$

More detailed description of the designated methods can be found at [4].

VIII. Organization of Sequential Computations

Let the method (7) is used for numerical solving of problem (5) and let the approximate solution y_n is known at moment t_n with step h_n. Then to obtain the approximate solution y_{n+1} at moment t_{n+1} we have the following common algorithm:

Step 1. Calculate approximate solution y_{n+1} at moment t_n with step h_n according to performing method.

Step 2. Calculate approximate function value $f(y_{n+1})$.

Step 3. Obtain integration step accuracy characteristics.

Step 4. If solution is accurate then go to **Step 5**, else set integration step h_n equals to step h^{ac} corrected by accuracy according to performing method and go to **Step 1**.

Step 5. Obtain integration step stability characteristics.

Step 6. If solution is accurate then go to **Step 7**, else set integration step h_n equals to step h^{st} corrected by stability according to performing method and go to **Step 1**.

Step 7. Get size of next integration step using (6).

Step 8. Perform next integration step.

IX. Organization of Parallel Computations

Developed parallel algorithms are based on the presented above sequential algorithms with the following differences.

For definiteness we assume that computer system consists of p processors, $N > p$ and let k is a number of equations per rank. Then all of N equations should be evenly allocated between computing nodes. To achieve this goal classical Round-Robin algorithm is chosen. Also parallel algorithm was designed to reuse sequential algorithm.

Taking into consideration assumptions about beginning of sequential method base parallel algorithms can be defined in the following way:

Step 1. Allocate equations evenly between ranks using

Round-Robin algorithm.

Step 2. Calculate in each rank approximate solution y_{n+1}^j, $0 \le j \le k$ at moment t_n with step h_n according to performing method.

Step 3. Send obtained y_{n+1}^j from each rank to others.

Step 4. Calculate in each rank approximate function value $f^j\left(y_{n+1}\right)$, $0 \le j \le k$.

Step 5. Execute for each rank sequential algorithm from **Step 3** corresponding to accuracy control.

X. Reaction-Diffusion Problem Simulation

Let us consider specification features and solving of problem of the designated class. The model [5] is based on two partial differential equations in two dimensions describes the change in the concentration of ozone in the stratosphere during the day. Two variables c^i $(i=1,2)$ describes the atomic concentration of oxygen (O_1) and ozone (O_3) [moles/cm3]. They depend on three variables: the vertical position z, $30 \le z \le 50$ [km], horizontal x, $0 \le x \le 20$ [km], and from time to time t, $0 \le t \le 86400$ [sec.]. Equations take into account the horizontal diffusion, advection, and non-uniform vertical diffusion. A mathematical model has the following form

$$\frac{\partial c^i}{\partial t} = K_h \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial}{\partial z}\left(K_v(z) \frac{\partial c^i}{\partial z} \right) + R^i\left(c^1, c^2, t\right), \quad (13)$$

where $i=1,2$, $K_h = 4 \cdot 10^{-6}$, $K_v(z) = 10^{-8} \cdot e^{z/5}$,

$$R^1\left(c^1, c^2, t\right) = -k_1 c^1 - k_2 c^1 c^2 + 7.4 \cdot 10^{16} \cdot k_3(t) + k_4(t) c^2$$

$$R^2\left(c^1, c^2, t\right) = k_1 c^1 - k_2 c^1 c^2 - k_4(t) c^2, \qquad k_1 = 6.031,$$

$k_2 = 4.66 \cdot 10^{-16}$,

$$k_3 = \begin{cases} \exp\left(-22.62/\sin(\pi t/43200)\right), & npu\ t < 43200 \\ 0, & npu\ t \ge 43200 \end{cases},$$

$$k_4 = \begin{cases} \exp\left(-7.601/\sin(\pi t/43200)\right), & npu\ t < 43200 \\ 0, & npu\ t \ge 43200 \end{cases}.$$

This system is a hybrid and has two states with the transition condition $t \ge 43200$.

Boundary conditions $\partial c^i / \partial x = 0$ at $x = 0$, $x = 20$, $\partial c^i / \partial z = 0$ at $z = 30$, $z = 50$. Initial conditions are: $c^1(x,z,0) = 10^6 \alpha(x)\beta(z)$, $c^2(x,z,0) = 10^{12} \alpha(x)\beta(z)$, where $\alpha(x) = 1 - (0.1x-1)^2 + (0.1x-1)^4/2$,

$\beta(z) = 1 - (0.1z-4)^2 + (0.1z-4)^4/2$.

Computer model in the ISMA is:

```
// Constants
const k1 = 6.031;
const k2 = 4.66* pow(1, -16);
const pi = 3.1416;
const time = 0;

// Variables to be sampling
var z[30, 50] step 5;
var x[0, 20] step 5;

// Equations
k3 = exp(-22.62/sin(pi*time/43200));
k4 = exp(-7.601/sin(pi*time/43200));

Kh = 4* pow(10, -6);
Kv = pow(10, -8) * exp(z/5);

C1 '= Kh* D(C1, x, 2) + D(DKV1, z) + R1;
C2 '= Kh* D(C2, x, 2) + D(DKV2, z) + R2;

DKV1 = Kv * D(C1, z);
DKV2 = Kv * D(C2, z);
R1 = -k1*C1 - k2*C1*C2 + 7.4*pow(10, 16) * k3 + k4*C2;
R2 = k1*C1 - k2*C1*C2 - k4*C2;

// Boundary conditions
edge C1=0 on z both;
edge C1=0 on x both;
edge C2=0 on z both;
edge C2=0 on x both;

// Initial values
C1 (0) = pow(10, 6) * (1 - pow(0.1*x-1, 2) + pow(0.1*x-1, 4)/2) * (1- pow(0.1*z-4, 2) + pow(0.1*z-4, 4)/2 );

C2 (0) = pow(10, 12) * (1 - pow(0.1*x-1, 2) + pow(0.1*x-1, 4)/2) * (1- pow(0.1*z-4, 2) + pow(0.1*z-4, 4)/2 );

// Change of state
state st1 (time>43200) {
 k3=0; k4=0;
} from init;
```

In this example blocks of model description are marked by comments: constants, variables to be sampling, algebraic equations, the system of PDE, boundary conditions, the initial value (the Cauchy problem) description of the state st1.

Turning to the grid of size $J \times K$ by x and z respectively we obtain $\Delta x = 1/(J-1)$ and $\Delta z = 1/(K-1)$ are grid steps by x and z coordinates, c_{jk}^i is approximation of $c^i\left(x_j, z_k, t\right)$, where $x_j = (j-1)\Delta x$, $z_k = (k-1)\Delta z$, $1 \le j \le J$, $1 \le k \le K$. Thus we obtain differential equations system of $N = 2JK$ dimension. For example, variable x will be divided into 20 parts, and the variable divided into z 40. The result is a grid dimension of 800. The resulting chart dynamics of the concentration of the reactants are presented in Fig. 4.

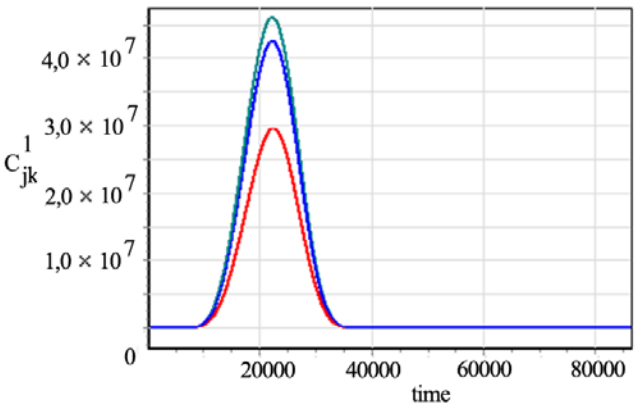

Fig. 4. Atomic nitrogen concentration diagram

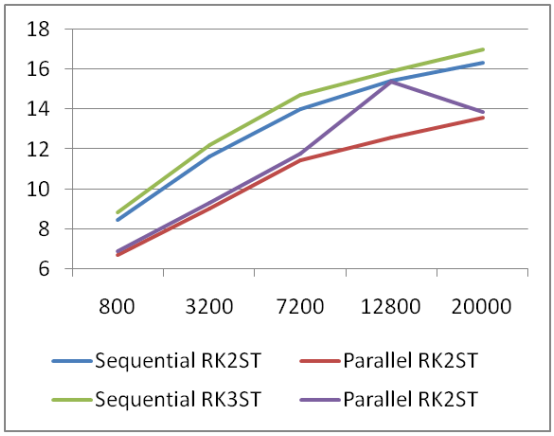

Fig. 5. Perfomances of implemented algortithms

In this example blocks of model description are marked by comments: constants, variables to be sampling, algebraic equations, the system of PDE, boundary conditions, the initial value.

The comparative analysis results of implemented algorithms in sequential and parallel version are given on the Table 1.

Table 1. Computational Costs of the Algorithms

Dimension	Algorithm			
	RK2ST		RK3ST	
	Sequential time (ms)	Parallel time (ms)	Sequential time (ms)	Parallel time (ms)
20x20 (N = 800)	4715	811	6839	982
40x40 (N=3200)	112573	8180	199685	11254
60x60 (N=7200)	1214869	93471	2431526	127165
80x80 (N=12800)	4988536	293569	8105641	4756598
100x100 (N=20000)	12098653	776501	24356423	1038794

Problem (13) was calculated for equations from 800 to 20000. MPI is chosen as paralleling technology because this approach is focused on cluster system and in future will allow calculating system of much more higher dimension if needed.

Dependence of computational time from system dimension is shown in Fig 5. Such a significant increase of computational costs (especially by sequential algorithms) is related to costs of construction and inversion of Jacoby matrix of increasing dimension. Also the higher system dimension the advantage of parallel algorithm becomes even more clearly.

XI. Conclusion

In this paper a new class of hybrid systems, continuous dynamics of which is defined by a system of DAE and PDE with boundary conditions, within ISMA instrumental environment is introduced. The architecture of ISMA is considered. New elements of LISMA language for description of PDE and boundary conditions language are presented. Grammar of new language inherited from the old language and is also context-free. Data structure for storing unified representation model is designed. Features of the considered hybrid systems such as the increased stiffness and high dimension make difficult to apply the commonly used sequential methods of numerical analysis. Steps of parsing and transition to the solver are discussed in detail. An example of reaction-diffusion is given, text model is obtained and numerical calculations are carried out. The experiments show that parallel algorithms based on explicit schemes with accuracy and stability control are more suitable for analysis of system of the designated class.

References

[1] E.A. Novikov, Yu.V. Shornikov, “Computer simulation of stiff hybrid systems: monograph”, Novosibirsk, Russia: Publishing house of NSTU, 2012.

[2] J. Esposito, V. Kumar, G.J. Pappas “Accurate event detection for simulating hybrid systems”, Hybrid Systems: Computation and Control (HSCC), Springer-Verlag, vol. LNCS 2034, 1998.

[3] E.A. Novikov, Yu.V. Shornikov “Numerical simulation of hybrid systems by Runge-Kutta method of second accuracy order”, Computing technologies, vol. 13 #2, pp. 98-104, 2008.

[4] E.A. Novikov “Explicit methods for stiff systems”, Novosibirsk: Nauka, 1997.

[5] Peter N. Brown, Alan C. Hindmarsh, Matrix Free Methods in the Solution of Stiff systems of ODEs, Lawrence Livermore National Laboratory, 1983. – 38p.

[6] Qinghua Feng, Bin Zheng, “Application of the Alternating Group Explicit Method for Convection-Diffusion Equations”, WSEAS Transactions on Mathematics, Issue 3, Volume 8, March 2009.

[7] Yu.V. Shornikov “Instrumental facilities of machine analysis” Yu.V. Shornikov, V.S. Druzhinin, N.A. Makarov, K.V. Omelchenko, I.N.Tomilov. Certificate of official program registration # 2005610126, Moscow: Rospatent, 2005.

[8] Yu.V. Shornikov, I.N. Tomilov, “The Program of Language Processor from Language LISMA”. Certificate of official program registration # 2007611024, Moscow: Rospatent, 2007.

[9] Benjamin De Leeuw, Albert Hoogewijs, “Statechart Normalizations”, WSEAS Transactions on Information Science & Applications, Issue 11, Volume 7, November 2010.

[10] Bin Zheng, Qinghua Feng, “Finite Difference Methods with Intrinsic Parallelism for Parabolic Equations”, WSEAS Transactions on Mathematics, Issue 4, Volume 8, April 2009.