Co-funded by the
Erasmus+ Programme
of the European Union

InMotion

Innovative teaching and learning strategies in open modeling and simulation environment for student-centered engineering education

Report for the working package WP 1.4

Comparative study of tools for Computer Modeling and Simulation

Novosibirsk State
Technical University

POLYTECH
Peter the Great
St.Petersburg Polytechnic
University

2017

Table of Contents

1 Comparison criteria

A goal of the work is to study general purpose modeling and simulation tools, such as Matlab/Simulink, DYMOLA, Wolfram SystemModeler, Rand Model Designer and ISMA from two distinct points of view – for engineers working and for students learning.

The aforementioned tools have been chosen because:

1. It is undeniable that Simulink and Modelica-based tools (such as Dymola and SystemModeler) dominate the markets of the US and the EU;

2. The Russian partner universities suggested two tools, which are strongly used in the educational process at these universities – Rand Model Designer and ISMA. They were introduced during the InMotion retrainings and were successfully accepted by the EU and Malaysian Partners. RMD generalizes the experience of Simulink and Modelica and strictly follows the UML standard, which is very important for educational purposes. ISMA specializes in numerical methods for a hybrid system simulation. These methods are recognized by the CMS specialists (see Eurosim papers, for example). For both software, there were published textbooks and manuals. Therefore RMD and ISMA have been listed for the analysis.

Let's start with comparison criteria that are important for engineers in their work.

Object-oriented approach. Object-oriented Modeling (OOM) is a modern technology of computer simulation (also referred to as computer modeling) widely used in scientific research, design of technical systems and analysis of business processes. OOM helps to create robust computer models in a faster and more cost-efficient manner. Object-oriented modeling encourages the creation of component libraries, which can be modified, reused and exchanged, thereby saving time in new project development.

Computer modeling practically emerged at the same time with an appearance of first computers. At that point, computer models were created manually using programming languages like Fortran or even Assembly. Conversely, the level of abstraction of any designed system is significantly higher than the level of abstraction of any programming language. Thus, at the time, a developer had to keep conformity between the model and its

program implementation in his own mind, which, evidently, had led to numerous errors. Oftentimes, programmers used own program implementations of numeral methods to reproduce behavior of continuous systems, though this approach led to questionable results. Partially this issue had been addressed by the use of professional libraries (1980-1983): LINPACK, EISPACK, LAPACK (computational methods in linear algebra), ODEPACK (numerical methods for solving ordinary differential equations).

From the 1950s, new high-level programming languages emerged, together with supporting software tools that created the executable computer models automatically. General Purpose Simulation System (GPSS) was one of the first modeling languages. It was meant to be used as a simulator for queueing systems. This language is notable as it relied on object-oriented approach; however, this terminology was not used at that time. Transactions in the simulated system input were created as copies of the standard "Transaction" class, with different values of their attributes, and were destroyed after the execution.

Simulation programming language Simula-67 was released in the late 1960s. Simula introduced classes, objects, inheritance, and polymorphism. Regrettably, this revolutionary project did not become widespread in the field of simulation practice, mainly due to limited computing power and high OOM use overheads of those days. At that time, an emphasis was put on single-component isolated mathematic models, transcribed as large and unique equation systems, while the need for increasing speed and accuracy of computation dominated over other problems, such as structuring and modification of models, for instance. These circumstances led to the significant delay, measured in decades, in full-scale application of object-oriented approach in modeling. Conversely, Simula-type objects are re-implemented in a range of object-oriented programming languages, such as C++, Java and Object Pascal that are still in use today.

Universality. A good simulation tool should be capable of implementing all the sorts of dynamical models. These are, first of all, discrete, continuous and discrete-continuous (hybrid) models. A queuing model, a controller for a plant, and an engine coolant system

accordingly may be considered as examples. Secondly, the set of models supported should include multi-component models: both with oriented (input/output components) and non-oriented connections (contact/flow components, physical). A control system model would be a multi-component model with oriented connections and a hydraulics model containing a few tanks connected would be a multi-component model with non-oriented connections. And last but not least a good simulation tool should allow building models with variable structure, i.e. structure varying over model time. E.g. an electronic circuit with a circuit breaker.

Stochastic models. Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results and are widely used in physics, economics, management problems and so on. So stochastic models are rather useful for modern general purpose simulation tool.

Compliance with a standard.

If a simulation tool represents models according to a well-known standard, it eases reading and understanding these models for engineers and modelers, helps them work together and communicate with each other. General mathematical language is such standard for textual models. From the other hand, for visual models there is no any general standard, although UML might be used as one.

In late 1990s OOM, essentially, has had a major breakthrough in modeling practice, due to the introduction of the Unified Modeling Language (UML) and the language for physical systems modeling Modelica reinforced with corresponding modeling software tools.

UML language is used to design specifications of complex software and hardware systems at the early stages of development. Its importance for discrete-event simulation lays foremost in canonization of concepts of object-oriented approach in software development, as it became de-facto standard of object-oriented approach. Moreover, these concepts were combined with exceptionally convenient and descriptive state diagrams (machines), invented in 1980's by D. Harel. UML language is easily extendable to simulate

continuous-discrete (hybrid) systems. One indication of UML popularity could be seen in introduction of the State Flow subsystem, which supports state diagrams, to the Matlab suite.

Symbolic solve and reducing of equation systems. Nowdays systems of differential-algebraic equations almost always are solved by numerical methods implemented as software in contrast to analytical solving by mathematicians and engineers of past. Although the majority of problems are not solvable analytically, there are cases when symbolic computations may work or at least reduce order of ODE system which comes handy speeding up the computation, so decreasing simulation time.

Model exchange. As should be expected, there is no 'perfect' simulation tool to model and simulate any kind of problem of any applied field. Many different tools and libraries are usually used together to solve a single complex problem. Therefore tools used should be able to use results produced by each other and, even better, to import models from other tools in order to simulate them with native models simultaneously within the tool.

Hardware in the loop (HIL). For the purpose of semi-physical modeling and simulation, a simulation tool should have abilities to connect to hardware, which is essentially useful and used in case of designing of control systems.

Standalone executable model. For some models, there is a requirement to be runnable on different platforms, outside of the native simulation tool. For this purpose a good tool should be able to generate an executable file, a web-page even would be better.

Animation. In general, a model should be animated in order an user to be able to see simulation progress and results in a convenient way.

As for educational purposes, we should use other criteria to assess aforementioned modeling and simulation tools. They are to be chosen in regard with the goal of teaching of students. If you teach them basic mathematic modeling techniques, you likely do not need an advanced simulation suit. Rather you need a simple mathematical computational package. If you perform specialized course on, for example, electrical power system, you,

of course, use a specialized tool, Simulink + SimPowerSystems, choosing from the analyzed ones. There are still some important requirements for simulation tools used in the education process.

Operating system. Obviously, the most used operating system family in the world is Microsoft Windows. Although, there are many enough users of the other ones – GNU\Linux and macOS. Since there is a fee for use of Microsoft Windows, some universities might consider installing GNU\Linux in order to save money. So might their students. Therefore a good simulation tool for educational purposes should run on both Windows and GNU\Linux, run on macOS is desirable as well; i.e. it should be cross-platform.

Costs. Matlab+Simulink, for example, costs over than 1,000$ per each computer on which the software to be installed for educational purposes. Even for a big university buying it would be quite expensive. If there is another software which meets your requirements and is less expensive, one rather chooses the latter.

To be used on a student's computer. There are students who can not work as efficiently as they are able to during classes. For them, it is easier and faster to make all the work at home, where none disrupts them. Taking such persons into account, we have the following.

Descriptions of the tools are given in the next section and the overall comparison table, including all the criteria and tool analyzed, is given in the annex.

2 Simulation tools

2.1 Matlab/Simulink

MATLAB suite, probably one of the best-known universal simulation environments, was developed in the late 1970s. It provides an opportunity to describe an isolated single-component dynamic system in a vector-matrix form, to call built-in solver for ordinary differential equations, as well as other solvers from professional systematic collections, and to visualize solutions in form of a time and phase diagrams. Despite the fact that the models in MATLAB suite are described in algorithmic language, similar to Fortran, not in mathematical language, the suite development was a big step forward in the field, thus it justifiably gained the merited recognition.

Although MATLAB suite was primarily oriented toward creation of traditional mathematical models, an additional package added the functionality of component-based modeling. In 1992, this package acquired its present name - Simulink. Graphical language of Simulink package has been used already at the time of analog machines and is familiar to developers of control systems – it allows building model of standard blocks, which have real physical equivalents in technical devices. Equations – often disliked by engineers – give way when graphical language of blocks becomes the main tool of model description. Simulink subsystem uses block language to describe models and exempts engineers from the time-consuming process of compiling system of equations corresponding with the whole model. The system of equations is then passed on to MATLAB suite solvers and the result of numeric solution is passed to different “visualizer”.

Visual simulation technology of the Simulink suite comes down to an assembly of models from readily available “blocks” - elements of standard libraries with adjustable parameters. In essence, we are talking about OMM, as those “blocks” are principally objects of library classes. Well-designed library of basic classes allows creation of more complex classes by using available proven designs as elements. Structural complexity of new classes is disguised under multi-level hierarchical structure, thus complex classes on higher levels appear as basic elements. However, MATLAB’s internal algorithmic language

has to be used for any development or modification or of the abovementioned set of basic classes.

A Simulink model, from one hand, can be connected to hardware for rapid prototyping, hardware-in the-loop (HIL) simulation, and deployment on an embedded system and, from the other hand, can be configured and made ready for C, C++, HDL, PLC code generation.

One of the well-known Simulink applications is design of control systems, for which there included even specialized libraries such as Aerospace, Robotics and Powertrain blocksets in Simulink. Simscape provides, withing the Simulink environment, modeling of the following physical systems: rotational and translational mechanical systems, including multibody ones, electronic systems, fluid dynamics, and electrical power systems.

For hybrid systems Simulink's add-on – StateFlow – is used. It includes state transition diagrams, flow charts, state transition tables, and truth tables to model how a modeled system reacts to events, time-based conditions, and external input signals.

As one of the key disadvantages of Matlab/Simulink, visualization capabilities are said. They are limited to variable depending on time or other variable plots.

Matlab/Simulink runs on all three platforms. The price of Matlab + Simulink bundle is 1100$, excluding extra addons. Matlab + Simulink costs 55$ for students, additional addons – 16$. There is still trial version, which can be used for 30 days.

2.2 DYMOLA

Modelica language solved the problem of component simulation automation in the components with non-directional connections use. External variables of Simulink model components are referred to as "entrance" and "exit", which underlines the principal implication of the information transmission direction. However, in many application areas, there is no direction of external variables, for instance, the direction of currents and voltage symbols in electric circuits are rather conventional. Connection of external variables of different blocks in electrical circuits simply means the equality of the voltages

and the equality of the sum of currents to zero. This fact fundamentally changes the method of constructing the final equations, which, in turn, leads to substantial problems. Modelica authors overcame these difficulties through introduction of components with non-directional connections, which in effect, broadened the range of application areas for computer simulation. Previously complex models were built using a limited set of basic components that could not be created in input simulating language, however, today Modelica allows creating component libraries for electrical, hydraulic, mechanical, and other physical systems utilizing language's own capabilities. Nowadays, user could build any complex structure, where final equations for the structure are automatically generated, similarly to block models of Simulink. Moreover, Modelica classes could be inherited, defined and redefined.

Dymola is based on Modelica. The Dymola environment is completely open in contrast to many modeling tools that have a fixed set of component models and proprietary methods for introducing new components. Users of Dymola can easily introduce components that match the user's own and unique needs. This can be done either from scratch or by using existing components as templates. The open and flexible structure makes Dymola an excellent tool to simulate new or alternative designs and technologies.

Dymola's models can be heterogeneous, consisting of components from different engineering areas. Libraries, including standard Modelica set for many different engineering domains, are available and contain components for mechanical, electrical, control, thermal, pneumatic, hydraulic, power train, thermodynamics, vehicle dynamics, aerospace, defense, flight dynamics, air-conditioning, etc.

In Dymola symbolic manipulations are used to solve differential algebraic equations (DAE), handle algebraic loop and reduce degrees-of-freedom caused by constraints. These techniques together with special numerical solvers enable real-time Hardware-in-the-Loop Simulations (HILS).

Dymola has real-time 3D animation facilities and allows importing CAD files.

In Dymola a user can perform model calibration, tune its parameters using design optimization in order to better fit model against data collected.

Model Management includes support for encryption of models, version control from Dymola (CVS, Subversion and GIT) and utilities for checking, testing and comparing models.

Dymola supports import and export of functional mockup units (FMI) in all formats and full compliance with the FMI specification. In addition to FMI, Dymola also supports export of S-function blocks for direct integration into the Simulink environment and C code generation. The tool chain is fully compatible with HILS platforms such as dSPACE and Concurrent.

Dymola works on Windows and Linux. Dymola prices are: Simulation – 1500$, Standard – 5900$, Light – 2900$. All the prices are given as they were in 2002, for current prices you are to contact sales managers of the developer or resellers.

2.3 Rand model designer

RMD is a high-performance environment for the development of component models of complex dynamical systems. RMD uses an object-oriented high-level modeling language, based on the object paradigm of UML, allowing quick and efficient creation of complex models. Input language makes no demands for knowledge in programming: it uses an intuitive common form to describe mathematical dependences and visual diagrams to describe the structure and qualitative changes in the behavior of the simulated system.

Continuous behavior may be represented in the form of implicit time dependencies, nonlinear algebraic equations (NAE), ordinary differential equations (ODE), and differential-algebraic equations (DAE). Equations are inputted by the user though the equation editor, similar to one in "mathematical" suites. RMD supports stochastic models.

Usage of both scalar and vector-matrix forms is possible. Alternatively, equations may be inputted in a block form, as it is done in the Simulink environment. SysLib

database, which contains Simulink blocks, helps those users who are accustomed to a graphic description the continuous behavior.

Models with discrete and hybrid behavior are usually described using hybrid automata. Hybrid machines within RMD are UML state machines without parallel (orthogonal) activities.

RMD includes an option for keeping the specifications of any complex model as a class, and consequently consider abovementioned class's behavior as "elementary" piecewise continuous state machine activity.

Hierarchical multi-component models with components with the "input-output" and internal hybrid automata (with attributed continuous activities in form of equations or class instance activities, corresponding to complex subsystems models) demonstrate fairly complex behavior, which is best described by a component hybrid machine composition, with an extremely large number of states. It is not considered necessary to create this composition in an explicit form, as an appropriate state behavior could swiftly be created for the implementation of a specific event.

Inheriting Modelica's technology for creating the multi-component model with non-oriented components (external variables such as "contact-flow") RMD introduces two significant additions:

- component hybrid automata may have continuous behavior, expressed as equation systems of any size and any type (NAE, ODE, DAE);
- equations, corresponding to component hybrid automata are created "on the fly" during the implementation, rather than at the compilation stage.

Construction of the variable structure models or models with "dynamic" components became possible owing to the introduced ability to create the equation for the whole model composition of component hybrid automata. Thus it is possible to create "agent-based models", as the component type (oriented or non-oriented) becomes irrelevant.

The program code of executable model is automatically generated based on a mathematical model and then compiled, which leads to high performance in conducting computational experiments. The automatic building of the aggregate system of equations takes into account its structure, reduces the dimension and part of the equations is solved symbolically, which together with applying the special numerical methods makes it possible to work with large systems of equations (thousands of differential-algebraic equations) including in real-time.

RMD provides powerful tools for debugging and demonstrating the results of experiments, two-dimensional and three-dimensional animation. The typical computational experiments are supported (receiving parametric dependencies, calculating the probability of an event, calculation of the value expectation of the variable, a global sensitivity analysis). Input language supports the possibility of an "internal" computational experiment during the functioning of the model.
Visual model can be used independently of the development environment and be utilized by the external application using a special API.

Typical applications for *Rand Model Designer*:

- mechanical systems;
- electric circuits;
- liquid hydraulic systems;
- gas systems;
- control systems;
- macroeconomics.

RMD requires Windows to run. There is the standard version of RMD which costs 150€ and the professional one for 1100€. The same prices are for students, although there is a trial version. For educational organizations special prices are applied according to personal inquiry.

2.4 Wolfram SystemModeler

Wolfram SystemModeler is developed by Wolfram MathCore and based on the Modelica language.

Models are built using Drag&Drop technology. An engineer can use additional paid and free libraries from a wide and expanding range of different areas available in the SystemModeler Library Store. Libraries contain reusable components with viewable source code, documentation, and detailed examples. The software includes Modelica Standard Libraries as well. A user can create and distribute its own created libraries of components.

Real-world machines and systems are rarely confined to a single physical domain such as mechanical, electrical or thermal. As well as in DYMOLA and Matlab/Simulink, one may build multi-domain physical models. Real problems are almost always hierarchical. While SystemModeler supports this kind of models, individual submodels are separately testable and reusable, allowing you to quickly explore alternative designs and scenarios.

Discrete-continuous systems are modeled by combining discrete signals and the built-in StateGraph library with continuous physical components. SystemModeler's numerical solvers detect and handle discontinuities in hybrid systems, so models with sudden events such as switches, collisions or state transitions are correctly simulated.

Besides regular, multi-variable and parametric plots, SystemModeler provides advanced visualization facilities. One can attach visualization geometries from CAD software to components, and automatically create live 3D animations for models with 3D mechanical components. If you need, you can connect to Mathematica for programmable custom visualization. Interactive simulation is a feature. A model parameter can be adjusted without recompiling.

Created models can be imported and exported as an FMU, via the Functional Mock-up Interface (FMI) standard for model exchange.

A model can be compiled for further use, for example, in desktop applications. The compiled executable reads parameter values and initial conditions from an XML file and results are returned in a standard format so one may easily use them.

SystemModeler simulations can be completely controlled over from within Mathematica's interactive notebook environment. A user can fit model parameters against to real-world data using Mathematica's optimizers. Sensitivity analysis is also included. As model equations and properties are accessible from Mathematica, one may use all the Mathematica capabilities to analyze it, such as symbolic analysis, finding close-form solutions, investigating approximate solutions and so forth.

A full suite of control systems features is built into Mathematica, including stability and frequency analysis, visualization and controller design. SystemModeler models can be automatically linearized into Mathematica's standard state-space representation for linear time-invariant (LTI) systems.

Wolfram SystemModeler runs on all three platforms. Wolfram SystemModeler's price is 365$ (1170$ for bundle with Mathematica). There is also a trial 30-days version and some options for students to buy the tool, which are more expensive in comparison to Matlab suite. Only SystemModeler would cost 100$ for a student, and SystemModeler+Mathematica – 250$. Although you can buy the tool only for a year, that is less expensive.

2.5 ISMA

ISMA is the acronym for Computer-aided analysis tools in Russian and is developed by NSTU's Department of automated control systems. The very core of a model in ISMA is a hybrid system, as a hybrid system is a generalization of a classic dynamical system.

A user of the tool can build models with distributed parameters, described by elliptic or parabolic partial differential equation systems with Neumann and Dirichlet boundary conditions.

In ISMA there are several input languages: textual language to solve a problem of any class supported, textual blocks can be combined with common control systems blocks forming block-textual models. Modes of hybrid systems can be described with state diagram in a graphical way. There are also graphical languages for several fields of science, such as electrical power systems and chemical kinetics. Regular applications of ISMA are mechanics, electromechanics, biological systems, electrical power systems, chemical kinetics, solid mechanics.

To deal with stiffness it is common for world-known mathematical packages to use implicit numerical methods, while in ISMA there developed original explicit methods with stability control.

Fully implicit methods cannot be used because they require the calculation of right hand part of an ODE system at a potentially dangerous area, where the model is not defined. Therefore in ISMA explicit methods are commonly used. Taking into account that explicit methods are known by poor stability, integration methods with accuracy and stability control were developed. Generally, accuracy and stability control are used to limit the size of the integration step. It 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.

Peculiarities of numerical analysis are defined by the configuration and implementation of the solver in the scheme interpreter. The solver is configured to numerical analysis not only of smooth dynamical systems but also systems with ordinary discontinuity and stiff systems. For the analysis of the stiff modes new original m-phasic methods of p-order were developed and included in the solver library. There shown key numerical methods included in ISMA in the table below.

DISPF(5, 6), DP78ST(8, 13), RKF78ST(7, 13), RK2ST(2, 2), RK3ST(2, 3), and DISPS1 are integration algorithms based on explicit methods of the Runge-Kutta type. All of these algorithms have accuracy and stability control and are applied to solving multi-mode problems.

RADAU5(3, 3), MK22(2, 2), MK21 (2, 2) are based on fully-implicit schemes which are aimed at solving single-mode systems. The first one is the implicit fifth order three stage Runge-Kutta method, the second and third schemes belong the (m, k)-class of numerical methods.

Method (p, m)	Description
DISPF (5, 6)	Stability control, systems of medium and low stiffness
RADAU5 (3,3)	Stiff systems
DISPF1_RADAU	Adaptive method DISPF in combination with RADAU5 with stiffness control, essentially stiff systems
DP78ST (8, 13)	Stability control, variable order and step, systems of medium stiffness and high precision
RKF78ST (7, 13)	Stability control, variable order and step, systems of medium stiffness and high precision, based on Runge-Kutta-Feldberg method
RK2ST (2,2), RK3ST (2, 3)	Explicit methods with stability control for analysis of non-stiff systems
DISPS1	Algorithm of variable order with adaptive stability region
MK22 (2, 2), MK21 (2, 2)	Freezing of Jacobean matrix, stiff systems
MK11F	Algorithm of analysis of implicit problems

DISPF1_RADAU is the variable structure algorithm of alternating order and step based on explicit Runge-Kutta numerical formulas of the first, second, and fifth order and the implicit Runge-Kutta type method of the fifth order. This algorithm has accuracy and stability control and can be applied to single-mode as well as multi-mode systems.

Finally, MK11F is the algorithm based on the L-stable Rosenbrock method aimed at solving implicit problems.

The correct analysis of hybrid models significantly depends on the accuracy of detection of the change of the local states of the HS. Therefore, the numerical analysis is necessary to control not only the accuracy and stability of the calculation, but also the dynamics of the event-function. The degree of approximation by the time the event

occurred is defined by the behavior of event driven function. In ISMA the original event detection algorithm providing the event-dynamics behavior as a stable linear system the solution of which is asymptotically approaching to the border surface of a mode is used, which is based on work of an American researcher, Joel Esposito.

ISMA runs on all three platforms since it is written in Java.

Conclusions

All the simulation tools analyzed are capable of modeling and simulation of complex systems of discrete, continuous, and hybrid types, as well as of component models (input-output). For physical (contact-flow) and/or stochastic models any of the tools but ISMA can be used. Only RMD provides full support for models with variable structure. Dymola, Matlab/Simulink and SystemModeler implement FMI for model exchange and can be used for HIL-simulations..

Building a model of a big complex system, use of the tools providing the object-oriented approach, such as Modelica-based Dymola and SystemModeler, or Rand Model Designer (which includes the standard library of Modelica), is preferred.

To create an animated model, you choose either SystemModeler, or Dymola, or RMD. All the analyzed tools but ISMA can export a model as an executable file.

For control systems design most of the specialists use Matlab+Simulink bundle. SystemModeler+Mathematica and Dymola may be easily used although. They provide approximately the same opportunities and let you use the OO-approach.

Being a user of Wolfram Mathematica, you should consider using another Wolfram tool – SystemModeler – which is a great general purpose modeling tool for many engineering problems.

If you obtain incorrect result simulating a hybrid system, you should give ISMA a chance, since the software uses original explicit methods with accuracy and stability control, which are a better choice than implicit ones in some cases. Moreover, ISMA includes original event detection algorithm. And if you model a system with distributed parameters, which is described by partial differential equations, you don't use any of the mentioned tools, but ISMA and Matlab.

For education purposes choice of a tool is not easier at all. As differential equations and state machines are almost standard for the input language of the most of simulation tools, the theory of dynamical systems, basics of modeling and simulation should be taught using the language of mathematics, but any specific simulation tool language. For a basic

mathematic modeling course, a teacher can use almost any mathematical package, performing ODE and DAE systems solving. For a more advanced course on component modeling you have to use a visual tool allowing visual hierarchical modeling; any of the analyzed tools can be used, it depends on licenses you already have and their costs. Almost all the programs have trial 30 days version, so your students can download them and work at home for that period of time. Most of the tool are cross-platform: ISMA, SystemModeler and Matlab/Simulink run on all three platforms (Windows, Linux and Mac), Dymola runs of Windows and Linux, and RMD runs only on Windows.

For the courses which are to be developed under the InMotion project, recommendations are listed in ANNEX B.

ANNEX A OVERALL COMPARISON TABLE

Criterion/Software	Matlab/Simulink	Rand model designer	DYMOLA	Wolfram system modeler	ISMA
For general engineering					
Object-oriented language	partially	yes	yes	yes	partially
Discrete models	yes	yes	partially	partially	partially
Continuous models	yes	yes	yes	yes	yes
Discrete-continuous (hybrid) models	yes	yes	yes	yes	yes
Multicomponent ‘input/output’ models	yes	yes	yes	yes	yes
Multicomponent ‘contact/flow’ models	yes	yes	yes	yes	no
Variable structure models	partially	yes	partially	partially	partially
Stochastic models	yes	yes	libraries	libraries	no
Compliance with a standard	no	yes, UML	partially UML	partially UML	no
Explicit numerical methods with stability control	no	no	no	no	yes
Symbolic computations	yes	yes	yes	yes	no
Model exchange	FMI	no	FMI, export as S-functions	FMI	no
Hardware in loop (HIL)	yes	no	yes	yes	no
Standalone executable models	yes	yes	yes	yes	no
Animation, 3D	no	yes	yes	yes	no
For education purposes					
Cost	1,100.00$+ per computer	Standard version - 150.00€, Professional - 1 100.0€	Simulation – 1, 500.00$, Standard – 5, 900.00$, Light – 2, 900.00$ (2002)	Standard – 365.00$, Full + Mathematica – 1,170.00$	personal inquiry needed
Operating systems	Windows, Linux, Mac	Windows	Windows, Linux	Windows, Linux, Mac	Windows, Linux, Mac
To be used on a	Matlab Suite –	Trial version	Standard	30 days trial	personal inquiry

Criterion/Software	Matlab/Simulink	Rand model designer	DYMOLA	Wolfram system modeler	ISMA
student's computer	55.00$, Additional addons – 16.00$, 30-days full trial		DYMOLA with limited capacity	version	needed

ANNEX B RECOMMENDATIONS ON USE OF SIMULATION TOOLS FOR COURSES TO BE DEVELOPED

University	Course name	Level	Simulation tools
SPbPU	The basis of mathematical modeling	Bachelor	Any of the mentioned tools, the most of the course should be taught without solid connection to a particular tool
SPbPU	Technologies of computer modeling	Master	RMD or a Modelica-based tool, as the course will be focused on component modeling
SPbPU	Computer modeling of complex dynamical systems	Master	RMD or a Modelica-based tool; RMD contains Modelica Library
SMTU	Modeling and Simulation of Dynamic Systems	Bachelor	Any of the mentioned tools, the most of the course should be taught without solid connection to a particular tool
SMTU	Modeling and Simulation in Engineering using Modelica	Master	A Modelica-based tool, Wolfram SystemModeler is expected to be chosen
SMTU	Modeling and Simulation in Engineering using RMD	Bachelor	RMD
NSTU	Simulation fundamentals	Bachelor	Any of the mentioned tools, the most of the course should be taught without solid connection to a particular tool. Examples are expected to be given in ISMA.
NSTU	Modeling and Simulation in Engineering using Modelica	Bachelor	A Modelica-based tool
NSTU	Modeling and Simulation of hybrid systems	Master	ISMA
UniKL	Mathematical Modeling of Complex Dynamical Systems	Bachelor	Any of the mentioned tools, the most of the course should be taught without solid connection to a particular tool.
UniKL	Applied Marine Hydrodynamics	Bachelor	If the course is supposed to discuss problems, specified as PDEs, Matlab or SystemModeler/Mathematica should be used
UTP	Rotating Machine Stability	Bachelor	Any of the mentioned tools
UTM	Robotics	Bachelor	Any of the mentioned tools
UTM	Ship and Offshore Structure Modeling and Simulation	Master	Matlab or any tools suggested by the authors of the course
SPbPU	Visual environments for modelling and simulation	Doctoral	Any of the mentioned tools
SMTU	Computer modeling for marine engineering applications	Doctoral	Any of the mentioned tools