JMCAD (JMCADRTS, JMCADRTC)

1 Description

The software package is designed to ***JMCAD*** dynamic analysis and design of a wide variety of systems and devices. In terms of features it is an alternative to similar software products ***LabView, Simulink, VisSim, Bauman*** et al convenient editor of block diagrams, extensive library of standard blocks and built-in programming language can implement a model of almost any degree of complexity, while providing clarity of presentation. The software package ***JMCAD*** successfully used to design control systems, servo drives and robotic manipulators, thermal power plants, as well as for solutions of nonstationary boundary value problems (heat conduction, fluid dynamics, etc.).

Widely used in the learning process, allowing to simulate various phenomena in physics, electrical engineering, in the dynamics of machinery, etc. Can operate in clusters, including those in remote access to technology and information resources.
For users with ease of use is also due to the localization of ***JMCAD*** interface into different languages and the availability of extensive documentation.
Versions are available ***JMCAD*** kernel source, libraries, and is an open system with full documentation and a set of demos. Also, the complex includes modules for maximum performance and real-time control (***JMCADRTS, JMCADRTC***).

The software package designed using ***JMCAD*** language ***Java*** (http://java.sun.com) and can be used in various operating systems (***Windows, Linux, Solaris, Unix,*** etc.).

2 Application

The software package implements ***JMCAD*** following modes:

- SIMULATION, which provides:
  - modeling of continuous, discrete and hybrid dynamic systems, including the presence of data exchange with external programs and devices ;
  - edit the parameters of the model in the mode of «on-line»;
  - calculated in real time, or zooming in modeling time;
  - restart and playback of simulation results;
  - dynamic signal processing.
- OPTIMIZATION, allows one to solve the problem:
  - minimize (maximize) defined quality indicators;
  - find the optimal parameters of the designed system in multiobjective formulation with constraints on the quality and optimized parameters.
- ANALYSIS, which provides:
  - calculation and construction characteristics of static and dynamic systems;
  - calculation of transfer functions;
  - visualization of the results of analysis of statically and dynamically.

- SYNTHESIS, allowing regulators to design:
  - to set the desired frequency response;
  - for a given location of the dominant poles.
- MONITORING AND CONTROL, allowing you to create virtual prototiry:
  - remote control instrumentation and control devices;
  - mimics the multimedia and animation effects.

The advantages of ***JMCAD*** include:

- openness by using the ***Java*** language and the implementation of several mechanisms for sharing data with external programs;
- the possibility to use different operating systems (***Windows, Linux, Solaris, Unix,*** etc.);
- easy to build complex models through the use of nested structures, and algorithms for signal tracing of typical units, convenient way to set the parameters and equations;
- efficient numerical methods;
- a large number of tutorials and demos with detailed comments.

3 Structure

The software package has three separate, independent unit ***JMCAD, JMCADRTS, JMCADRTC***. Each unit can operate independently, as well as the creation of distributed systems, you can use them together. Sharing allows you to create complex distributed systems with the ability to quickly and easily develop the system. This functional feature allows you to change a running system without stopping it. Simply create a fork and backup unit, which develop and produce, and after the test and this block is included in the system as default. Replacing the old block without stopping the entire system.

JMCAD - the main block to create and edit models (Fig. 1). You might also consider using it to run the model in operation (Fig. 2). Running the model in the mode of operation is performed through the command line using the parameter ***-single***.

JMCADRTS - block to run the model in operation (Fig. 2). Running the model in the mode of operation is performed through the command line.

JMCADRTC - block for the interface and control model (Fig. 3). Starting the interface and control model is produced via the command line.

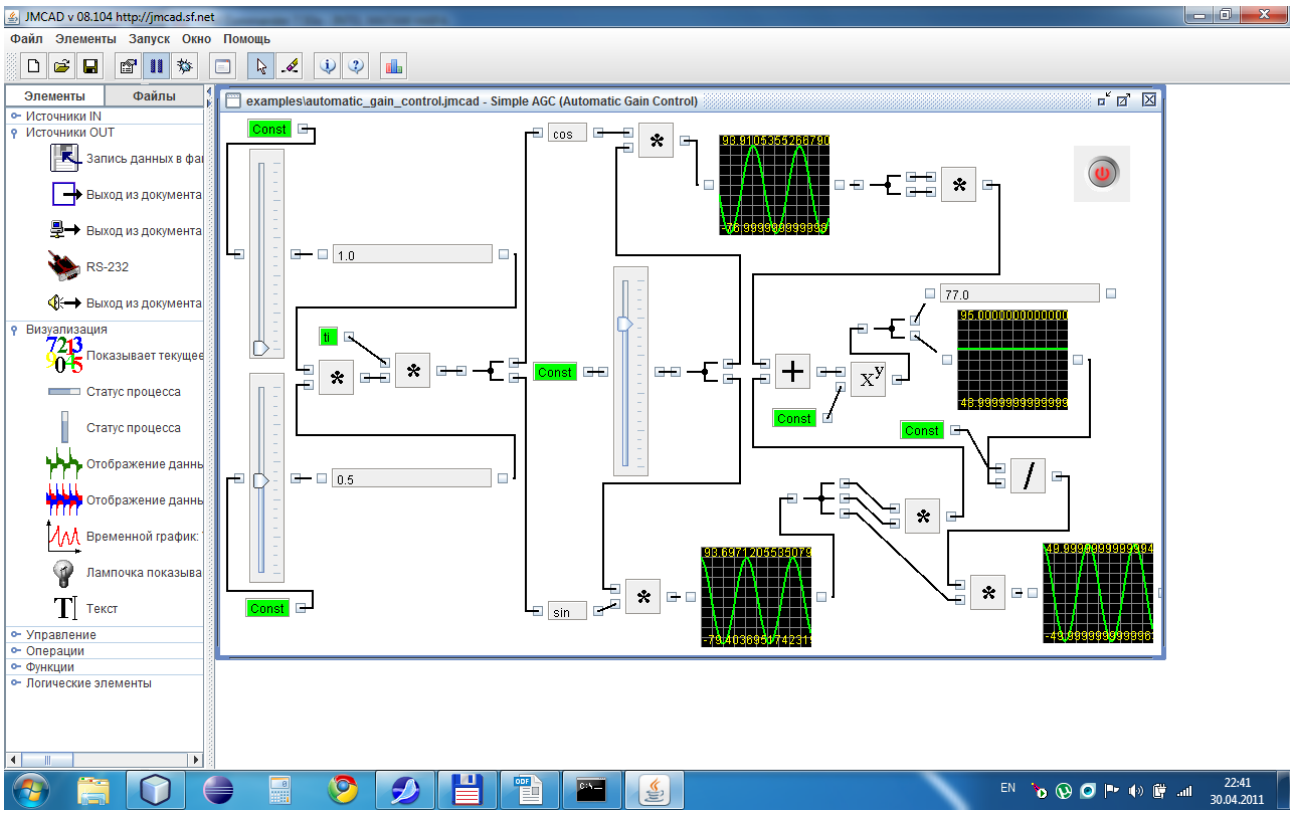
JMCAD v 08.104 http://jmcad.sf.net
Файл Элементы Запуск Окно Помощь
Элементы
Файлы
Источники IN
Источники OUT
Визуализация
Управление
Операции
Функции
Логические элементы
examples\automatic_gain_control.jmcad - Simple AGC (Automatic Gain Control)
Const
cos
sin
1.0
0.5
77.0
22:41
30.04.2011

Fig. 1

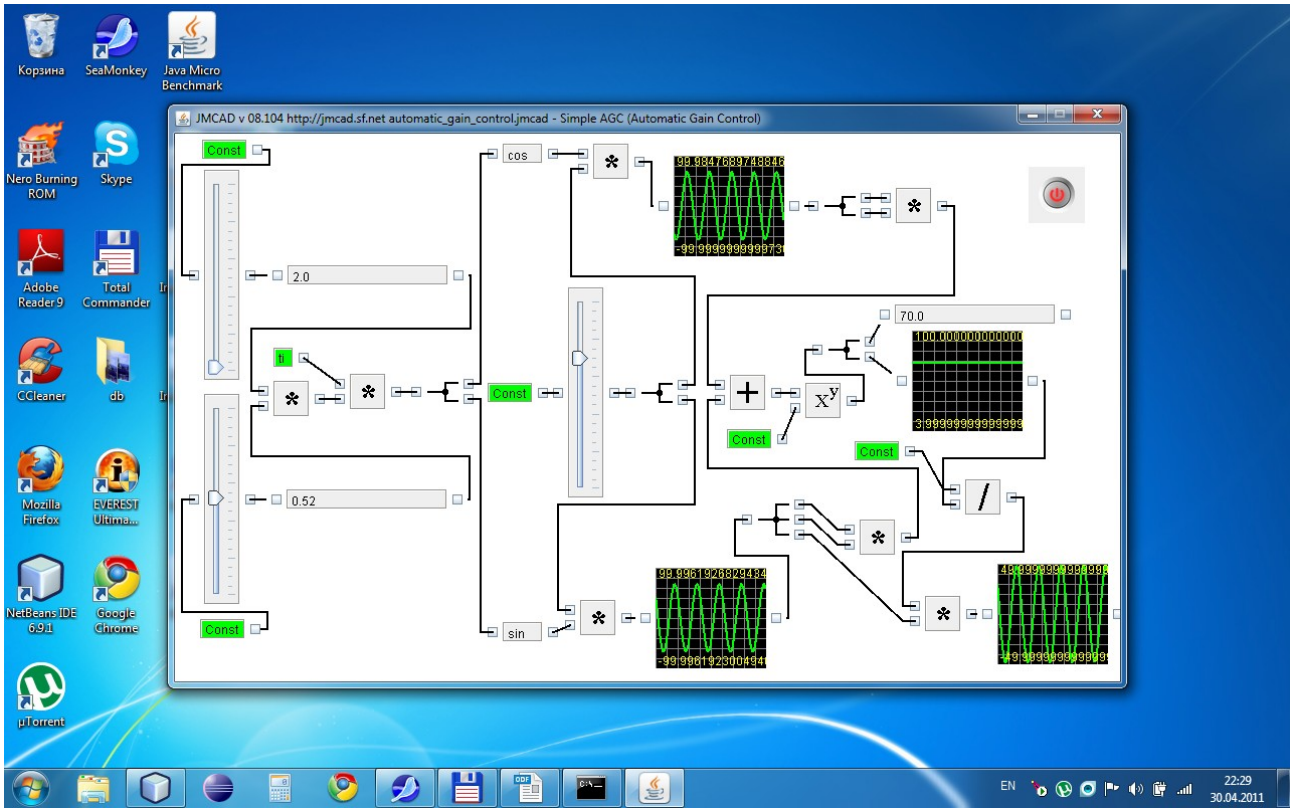
Корзина
SeaMonkey
Java Micro Benchmark
Nero Burning ROM
Skype
Adobe Reader 9
Total Commander
CCleaner
db
Mozilla Firefox
EVEREST Ultima...
NetBeans IDE 6.9.1
Google Chrome
µTorrent
JMCAD v 08.104 http://jmcad.sf.net automatic_gain_control.jmcad - Simple AGC (Automatic Gain Control)
Const
cos
sin
2.0
0.52
70.0
22:29
30.04.2011

Fig. 2

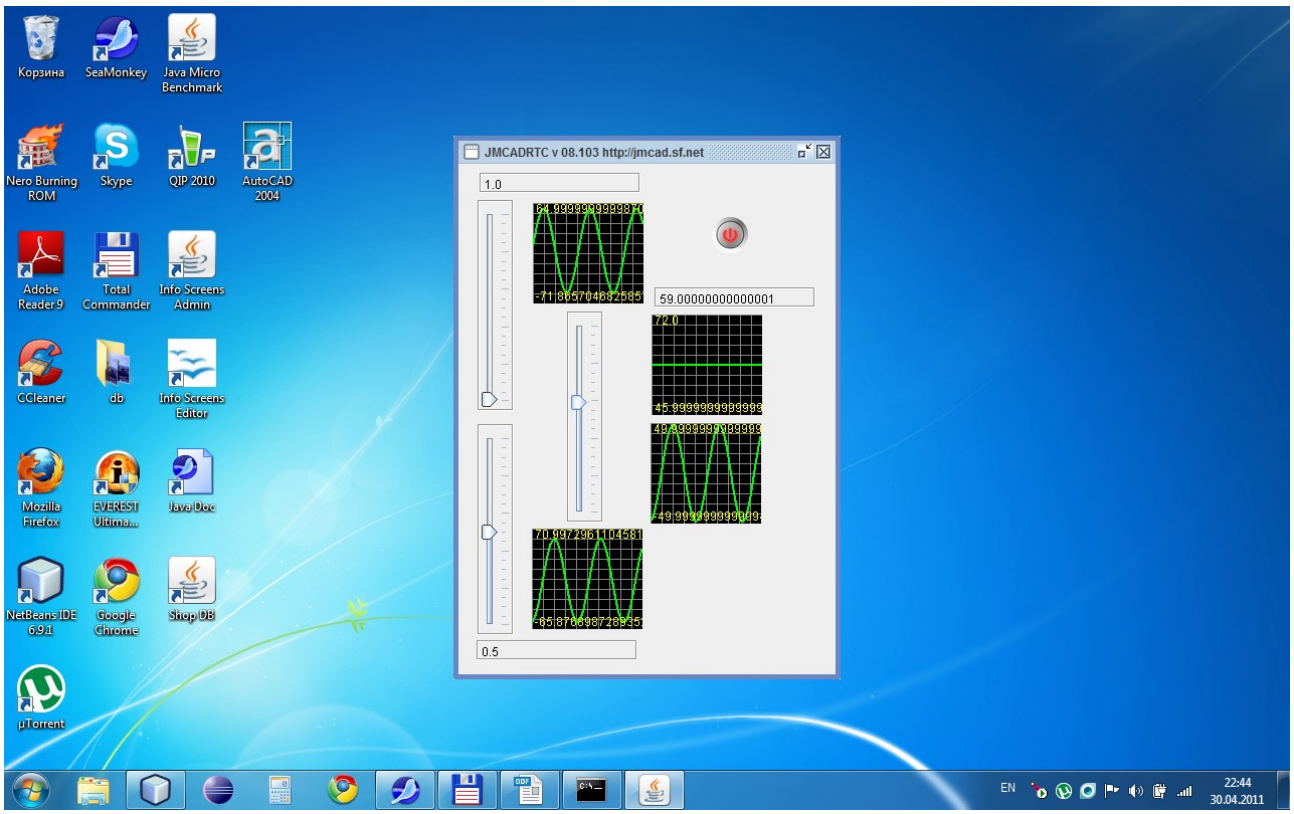

Fig. 3

4 Installing the system

The software package ***JMCAD*** can be installed on any computer both locally and with ability to work across the network.

To install the system must be installed ***JMCAD*** Java (http://java.sun.com/javase/downloads/). It is recommended to install the version of ***Java SE Runtime Environment 7 (JRE)*** or higher to run the complex, and for the development using a version of ***Java SE Development Kit 7 (JDK)*** or higher.

The software package can be downloaded from ***JMCAD*** server http://jmcad.sf.net. On the server, there are two variants of the distribution:

• ***JMCAD-XX.XXX-bin.zip*** - distribution software package ready for work;

• ***JMCAD-XX.XXX-all.zip*** - distribution software system for business and developers, which contains the source code;

To install ***JMCAD*** unzip the file to ***JMCAD-XX.XXX-bin.zip*** or ***JMCAD-XX.XXX-all.zip*** on your hard drive. When unzipped distribution directory will be created with the same name as that of the archive file. The directory will contain the relevant files intended distribution.

5 Starting and use

To run the software package used by the script files ****. bat*** (****. sh***). Sample script files to run ***JMCAD***, ***JMCADRTS***, ***JMCADRTC*** found in the directory ***examples***.

5.1 JMCAD

To run the software package ***JMCAD*** used by the scripts ***__JMCAD .*. bat*** (***__JMCAD. *. sh***). Instead of a file is selected * corresponding to the operating system will be used where the software system.

You might also consider using it to run the model in operation (Fig. 2). Running the model in the mode of operation is performed through the command line using the parameter ***-single***. Command line for starting the model in the mode of operation is as follows:

java [parameters] <CLASSPATH> JMCAD -single <MODEL>

where:

[parameters] — parameters for the virtual machine ***Java***. For large models with a shortage of RAM is necessary to specify the value ***-Xmx1000m***, which determines the amount of available memory for the virtual machine ***Java***. You can also use the ***-server*** option to increase the speed of calculation. More details about the values and their application may check the documentation for the virtual machine, ***Java***. These parameters are optional and may not be indicated.

<CLASSPATH> - used to connect the libraries to the class. All libraries are required for the model should be available to the virtual machine ***Java***. There are several options for specifying the ***Java*** virtual machine where to find the class libraries. Described in detail in the documentation for the virtual machine ***Java***. For this case the line description must be present libraries are needed to start and run the model. The main classes of all libraries have a library located in the ***jar*** archive containing class and ***JMCAD.jar*** to run ***JMCAD***.

<MODEL> - command line parameter that specifies the model file (****. jmcad***) to run.

5.2 JMCADRTS

Running the model in the mode of operation is performed through the command line. Command line for starting the model in the mode of operation is as follows:

java [parameters] <CLASSPATH> JMCADRTS <MODEL>

where:

[Parameters] - parameters for the virtual machine ***Java***. For large models with a shortage of RAM is necessary to specify the value ***-Xmx1000m***, which determines the amount of available memory for the virtual machine ***Java***. You can also use the ***-server*** option to increase the speed of calculation. More details about the values and their application may check the documentation for the virtual machine, ***Java***. These parameters are optional and may not be indicated.

<CLASSPATH> - used to connect the libraries to the class. All libraries are required for the model should be available to the virtual machine ***Java***. There are several options for specifying the ***Java*** virtual machine where to find the class libraries. Described in detail in the documentation for the virtual machine ***Java***. For this case the line description must be present libraries are needed to start and run the model. The main classes of all libraries have a library located in the ***jar*** archive containing class and ***JMCADRTS.jar*** to run ***JMCADRTS***.

<MODEL> - command line parameter that specifies the model file (****. jmcad***) to run.

5.3 JMCADRTC

Starting the interface and control model is produced via the command line that looks like this:

java [parameters] <CLASSPATH> JMCADRTC <MODEL>

where:

[Parameters] - parameters for the virtual machine ***Java***. For large models with a shortage of RAM is necessary to specify the value ***-Xmx1000m***, which determines the amount of available memory for the virtual machine ***Java***. You can also use the ***-server*** option to increase the speed of calculation. More details about the values and their application may check the documentation for the virtual machine, ***Java***. These parameters are optional and may not be indicated.

<CLASSPATH> - used to connect the libraries to the class. All libraries are required for the model should be available to the virtual machine ***Java***. There are several options for specifying the ***Java*** virtual machine where to find the class libraries. Described in detail in the documentation for the virtual machine ***Java***. For this case the line description must be present libraries are needed to start and run the model. The main classes of all libraries have a library located in the ***jar*** archive containing class and ***JMCADRTC.jar*** to run ***JMCADRTC***.

<MODEL> - command line parameter that specifies the model file (****. jmcad***) to run.

6 Development

The software package is an open system ***JMCAD*** with the possibility of developing new modules. The system was developed using the language ***Java*** (http://java.sun.com), which allows its use on different operating systems (***Windows, Linux, Solaris, Unix,*** etc.).

The system is the kernel that controls the system. The kernel is responsible for starting, stopping work and model. In the software package used three kernels - ***JMCAD***, ***JMCADRTS***, ***JMCADRTC***. The principle of operation is that the nucleus controls the data transfer between the elements, and all operations are performed in a cell autonomous. The kernel synchronizes data transmission between the elements and operates in three stages (start-up, work, stop) :

- ***start*** - engine runs method ***calc_pre();*** for all elements of the model. Method ***calc_pre();*** is used to prepare the item for work;
- ***operation*** - the kernel starts the method ***calc();*** in the loop for all elements of the model with the specified delay. For items that are a source calls the ***start(long t0, long ti, long dt);***. If 0 is specified then the system operates in real time;
- ***stop*** - start kernel method ***calc_post();*** for all elements of the model. Method ***calc_post();*** is used to prepare the item to a halt.

6.1 The class structure

The system is based modular laid. This approach makes it easy to develop new items. Appointment and functions of the system elements are classified into different files:

- ***JMCAD*.java*** — core systems and graphical user interface;
- ***JMCAD_Internationalize_xx_XX.properties*** — localization of the interface. In place of the symbols ***xx*** and ***XX*** are indicated symbols defining country and language;
- ***JMCAD.menu*** — menu of visual elements;
- ***_*.java*** — visual elements. Also for the convenience of the files that refer to this visual element have a similar name.

6.2 Compilation

To compile and package all the classes in the ***jar*** file using a batch file ***__make_jar.bat*** (***__make_jar.sh***). These batch files are for each block in a directory with source files for this unit:

- ***src – JMCAD*** (***JMCAD.jar***);
- ***src_rts – JMCADRTS*** (***JMCADRTS.jar***);
- ***src_rtc – JMCADRTC*** (***JMCADRTC.jar***).

6.3 Creating the visual elements

All models are ***JMCAD*** of the visual elements that determine the effectiveness of this model. In the system there is a large set of predefined standard elements. But there is always an objective necessity to increase the standard set of standard elements, adding new elements or modifying existing ones. To do this in software system ***JMCAD*** provides easy way to create new elements.

To create the basic structure of a visual element used inheritance base class for all visual elements ***JMCADObject***. This class contains variables and methods needed to create a new visual elements.

The main variables:

- ***in*** — the array of input values. By default, the array size is 0. The size of the array determines the number of entries in the element. Arriving at the input value stored in cell array corresponding to the index entry;
- ***out*** — the array of output values. By default, the array size is 0. The size of the array determines the number of outputs from an element. Once all the inputs of the element ***in*** the values start method ***calc();*** who performs the action it and then all the values in the array are passed ***out*** to the other elements;
- ***in_text*** — array of names of input values. The array size must match the array of input values ***in***. The names describe the Latin alphabet and numbers. Must begin with a letter and contain no special characters;
- ***out_text*** — array of names of output values. The array size must match the array of output values ***out***. The names describe the Latin alphabet and numbers. Must begin with a letter and contain no special characters;
- ***ToolTipText*** — text prompt appears when you hover over an item;
- ***w, h*** — width and height of the element. The values are given in the constructor;
- ***isGeneranor*** — variable determines the type of standard or source. By default, the variable is ***false***, that defines the element as a standard method and the kernel runs the ***calc();***. If you specify a variable to ***true***, the element has a type of the source and the kernel starts the method ***start(long t0, long ti, long dt);***
- ***isVisual*** — variable defines the appearance of the item when it is used in the interface and control model (Fig. 3). The default value is ***false***, which makes the element visible in the interface and control model;
- ***panel_c*** — central panel element;
- ***edit_panel*** — panel to create graphical interfaces that can be used to edit the properties of the element. Double clicking on the item in edit mode model, a dialog box containing this panel.

The main methods:

- ***calc_pre()*** — method used to prepare the item for work;
- ***calc()*** — method will be invoked at each step of the model with a specified delay. For items that are a source calls the ***start(long t0, long ti, long dt);***. If 0 is specified then the system operates in real time;
- ***calc_post()*** — method used to prepare the item to a halt;
- ***start(long t0, long ti, long dt)*** — a method for items that are the sources and will be called at each step of the model with a specified delay. If 0 is specified then the system operates in real time;
- ***edit_pre()*** — method used to create graphical user interface for editing the properties of the element. Graphical interface is provided on the panel ***edit_panel***, which is then displayed when double clicking an item in the dialog box;
- ***edit_post()*** — if the method is called when the element properties dialog box, clicked accept the changes;
- ***paint_info(Graphics g)*** — method provides access to graphical methods of the element;
- ***parse(String pst)*** — provides a method for reading data from the model. The method body must begin with the string ***super.parse(pst)***;
- ***write(RandomAccessFile fout)*** — method ensures that the parameters of the element in the model. The method body must begin with the string ***super.write(fout)***.

6.4 Creating and localization of documents

The software package ***JMCAD*** contains algorithm automatically display the documentation for each element and creates a common documentation.

Double clicking on the item in edit mode model, a dialog box containing a button to pop the documentation for this item.

To display the documentation in the directory should be ***help*** file documentation for this item. The file has the format of the ***HTML*** data and should have the same name as the class of the element. To localize the documentation to the file name prefix ***_xx_XX***. Where ***xx*** specifies the country, and ***XX*** specifies the language.

6.5 Localizing the user interface

Localization of the interface produced by standard methods of language ***Java***. Strings for localization found in the files to localize the user interface ***JMCAD_Internationalize_xx_XX.properties***. Where to place the symbols ***xx*** and ***XX*** indicates the characters defining the country and language. More detailed information about the localization techniques in ***Java*** can be found in documentation for programming in ***Java***.