

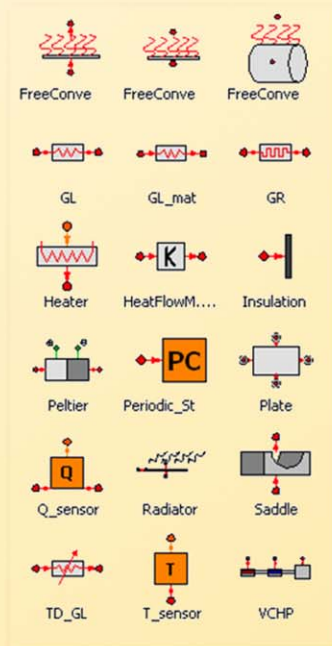
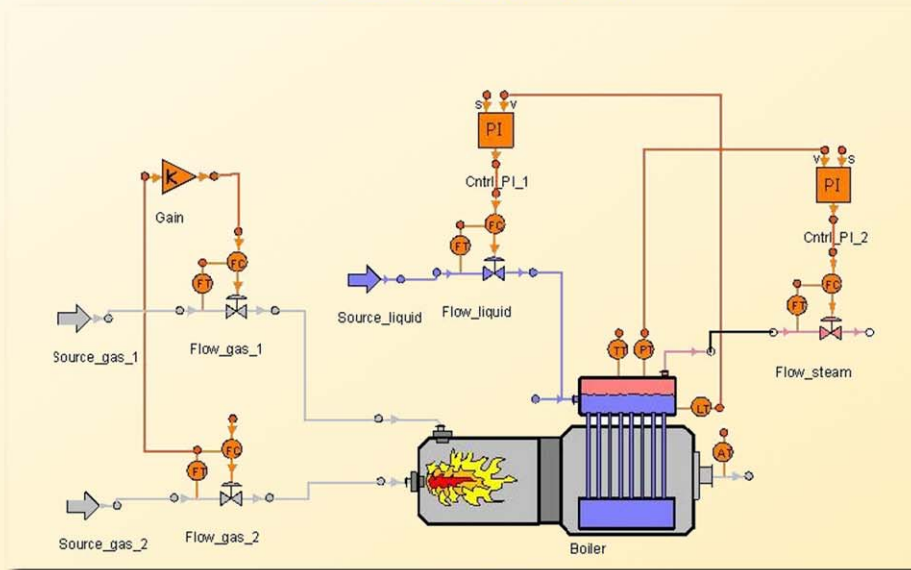


EcosimPro

Modelling and Simulation Software

Installation & Getting Started Guide

Version 4.8



EMPRESARIOS AGRUPADOS

This page intentionally left blank



CONTENTS

1. INTRODUCTION.....	5
1.1. SUMMARY	5
1.1.1. Potential users.....	5
2. GENERAL INFORMATION.....	7
2.1. ECOSIMPRO WEB SITE	7
2.2. MANUALS	7
2.3. SUPPORT AND CONSULTING SERVICES	7
2.4. REPORTING A PROBLEM	8
2.5. USER GROUPS OF ECOSIMPRO	8
3. INSTALLATION PROCEDURE	9
3.1. MINIMUM HARDWARE REQUIREMENTS	9
3.2. SOFTWARE REQUIREMENTS	9
3.3. STEPS TO INSTALL THE PROGRAM	9
4. INSTALLING THE C++ COMPILER	11
4.1. INSTALLING MICROSOFT VISUAL C++ 6.0.....	11
4.2. INSTALLING MICROSOFT VISUAL STUDIO 2003.....	11
4.3. INSTALLING MICROSOFT VISUAL STUDIO 2008.....	11
4.4. ENVIRONMENT VARIABLES	11
4.4.1. Microsoft Visual C++ 6.0.....	12
4.4.2. Microsoft Visual Studio 2003.....	12
4.4.3. Microsoft Visual Studio 2008.....	13
4.4.4. Gcc 4.4 on Windows.....	13
5. INSTALLING THE PROGRAM.....	15
6. INSTALLING THE PROTECTION BASED ON HARDWARE KEYS.....	17
6.1. WARNING	17
6.2. INSTALLATION STEPS	17
6.2.1. STEP 1: Check that the program is installed.....	17
6.2.2. STEP 2: Install the hardware key software (USB driver and network server).....	17
6.2.3. STEP 3: Insert the hardware key and run it.....	17
7. OVERVIEW OF THE PROGRAM.....	19
7.1. SHORT HISTORY	19
7.2. CURRENT APPLICATIONS.....	20
7.3. KEY CONCEPTS	21
7.4. WORKING IN ECOSIMPRO	22
7.5. THE SIMULATION LANGUAGE.....	23
7.6. LOOK-AND-FEEL OF AN COMPONENT	23
7.7. LOOK-AND-FEEL OF AN PORT TYPE	23
7.8. LIBRARIES	24
7.9. GRAPHICAL USER INTERFACE	24
7.9.1. Main window.....	24
7.9.2. Schematics.....	27
7.9.3. Object Editors	27
8. EXAMPLE OF MODELLING AND SIMULATION.....	29
8.1. YOUR FIRST COMPONENT USING SOURCE CODE.....	29
8.2. REUSING A COMPONENT CREATED WITH SCHEMATIC	37
9. MORE SIMULATIONS IN THE INSTALLATION	41

9.1.	EXAMPLES IN THE DEFAULT_LIB LIBRARY	41
9.1.1.	<i>WHEN example</i>	41
9.1.2.	<i>Copying graphs to Word</i>	42
9.1.3.	<i>Analyzing results in other Office applications</i>	42
9.1.4.	<i>The freezer</i>	43
9.1.5.	<i>Lorentz equations</i>	45
9.2.	EXAMPLES IN THE ELECTRICAL LIBRARY	47
9.2.1.	<i>Diode bridge example</i>	47
9.3.	EXAMPLES IN THE CONTROL LIBRARY	48
9.4.	EXAMPLES IN THE THERMAL LIBRARY	49
9.5.	EXAMPLES IN THE TURBOJET LIBRARY	50



1. Introduction

1.1. Summary

EcosimPro is a very powerful continuous-discrete simulation tool. It offers extensive capabilities for modelling and simulation, based on more than twenty years' engineering experience simulating complex systems.

The EcosimPro development team is drawn from a diverse range of engineering disciplines: simulation, object-oriented programming and modern user interface design.

Normally engineers doing simulations spend 80% of the time programming and 20% modelling, dealing with a range of fields outside their own specialisation. EcosimPro's main design objective is to reverse this, to allow the engineer to spend 80% of the time on physical modelling and only 20% programming. EcosimPro handles internally the complexity of sorting equations, optimising the numeric model, and solving systems of linear and non-linear equations. EcosimPro has its own modelling language, EL. EL is intuitive for the engineer, and equations are written practically the same way as in algebra. It incorporates the latest advances in programming technology such as encapsulation, enumeration data types, multiple inheritance, and multidimensional arrays, to enable an engineer familiar with the language to easily reuse existing, tested components to build new models.

EcosimPro's graphical user interface will be familiar to any user of development environments under Microsoft® Windows®, especially Microsoft Visual C++®. It is an intuitive, visual development environment in which the modeller can quickly and easily create libraries, components, simulation files, etc.

EcosimPro is an open tool. Unlike other simulation systems, EcosimPro is not domain-specific: libraries have already been developed for several different domains. Moreover, it is easy to extend by adding libraries of custom components written in the EcosimPro modelling language. It is straightforward to reuse additional subroutines in FORTRAN, C or C++ which can be called from the language.

EcosimPro can be used to study:

- transient behaviour
- steady state of models
- parametric studies
- complex experiments using FORTRAN, C or C++ functions, etc

More than 50 000 man-hours have been invested in the development of EcosimPro, financed mainly by the company itself and by the European Space Agency (ESA).

This version of the application has been tried and tested in different fields under the most critical and demanding conditions.

1.1.1. Potential users

EcosimPro can be of great help to engineers, physicists, chemists, mathematicians, biologists, etc, for modelling simple and complex processes.

As stated previously, EcosimPro can be applied easily to multidisciplinary fields: chemistry, fluids, control, mechanics, etc. In general any problem modelled with differential-algebraic equations.

EcosimPro is useful for university students to learn how to model physical problems, and for writing end-of-course dissertations/projects and theses for their doctorates.

There are three levels of users:

- **LEVEL 1:** Users who develop libraries of components. They need to have a profound knowledge of the physics and mathematics of the problem to model and simulate it. They need to create new components using EcosimPro's modelling language EL. For example, a creator of a basic ELECTRIC library with capacitors, resistors, inductances, etc.
- **LEVEL 2:** Users who create models (also known in EcosimPro as components) based on existing libraries. They do not need to have the extent of knowledge LEVEL 1 users have, but they do of course need to know what is under the components. In EcosimPro they are the typical users who generate and simulate the schematics that represent the different physical systems. For example a user creating an electric circuit based on the ELECTRIC library.
- **LEVEL 3.** Users who just run simulations from existing models. They need only be capable of changing the input data and running the simulations to obtain the results. They do not require any special math knowledge; they just need background knowledge of the final application. For example, an operator in a plant needs to know what happens in a scenario if some of the plant parameters change. He can run the simulation with the new parameters to obtain the results.



2. General information

2.1. Ecosimpro web site

The EcosimPro web site is <http://www.ecosimpro.com>.

2.2. Manuals

All the manuals are usually located in the **manuals** directory of the installation in PDF format.

Manuals available are:

- **Installation & Getting Started** (this one).

- **User Manual.**

It describes the different options in the man-machine interface for using the program, for instance, how to compile, run an experiment, use the schematics, etc.

- **EL Language.**

It is a reference manual for the simulation language used in the software.

- **Mathematical Algorithms and Simulation Guide.**

It describes the mathematical operations done by the program to sort the equations, and also the examples of the libraries coming with the installation

- **External Connections Manual.**

It describes the different ways to connect the models generated with the tool from other programs.

It is recommended that user first follow the guide to Installation & Getting Started and create the examples, and then take a look at the User Manual.

2.3. Support and Consulting Services

EA Internacional offers several other services in addition to this product:

- **Modelling support**

This service has been set up to help users to optimise their models, to use the appropriate techniques in each case, to study non-convergence errors in models, etc. EA engineers have many years of experience in modelling complex systems, enabling them to assist users with less experience and thus reduce costs and optimise their resources. Contact us for more information at info@ecosimpro.com.

- **Courses**

The standard course lasts for three days, but they can be adjusted to the specific needs of each company. They can also be directed towards a particular discipline. Many of the courses offered (mainly those in universities) are free and anybody can attend. Take a look at our web site for additional information.

- **System modelling and simulation service**

EA will undertake to carry out simulation projects that are tailor-made to the needs of our clients. In such cases, EA will create the system to be simulated. Contact us for more information at info@ecosimpro.com.

2.4. Reporting a problem

If you encounter an error in the software, please contact by e-mail with bugs@ecosimpro.com asking for an account for reporting errors or enhancements

The URL of the Software problem report system is:

<http://sprs.ecosimpro.com>

2.5. User groups of Ecosimpro

There are two mailing lists for EcosimPro users to share their experiences using EcosimPro. The goals of this group are to:

- ✦ Share experiences using EcosimPro.
- ✦ Solve problems.
- ✦ Get advice about modelling techniques in EcosimPro Language (EL).
- ✦ Get the latest news about releases, bugs found, etc.

Both can be found in <http://groups.yahoo.com>

- **ecosim-group**. The language of this list is English.

Group Web: <http://groups.yahoo.com/group/ecosim-group/>

Group Email Addresses:

Post message:	ecosim-group@yahoogroups.com
Subscribe:	ecosim-group-subscribe@yahoogroups.com
Unsubscribe:	ecosim-group-unsubscribe@yahoogroups.com
List owner:	ecosim-group-owner@yahoogroups.com

- **ecosim-group-esp**. The language of this list is Spanish.

Group Web: <http://groups.yahoo.com/group/ecosim-group-esp/>

Group Email Addresses:

Post message:	ecosim-group-esp@yahoogroups.com
Subscribe:	ecosim-group-esp-subscribe@yahoogroups.com
Unsubscribe:	ecosim-group-esp-unsubscribe@yahoogroups.com
List owner:	ecosim-group-esp-owner@yahoogroups.com



3. Installation Procedure

3.1. Minimum hardware requirements

- PC running Windows 2000, Windows XP or Windows Vista.
- Pentium II 133 MHz
- 128 MB of RAM memory
- 310 MB free disk space

3.2. Software requirements

This program needs to invoke a **C++ programming language compiler** at run time. For this version, the compatible versions are the following:

- Microsoft Visual C++ 6.0
- Microsoft Visual Studio 2003
- Microsoft Visual Studio 2008
- Gcc 4.4 on Windows

Additionally, and depending on the type of license purchased, it may be necessary to install the **software necessary to manage the licenses**.

3.3. Steps to install the program

To install the program correctly, follow these steps:

- Install one C++ programming language compiler. **(CHAPTER 4)**
- Install the program. **(CHAPTER 5)**

Optionally, depending on the type of license purchased:

- Install the Software for licensing based on Hardware Keys. **(CHAPTER 6)**
- Install the Software for licensing based on Software License. **(CHAPTER 7)**

This page intentionally left blank



4. Installing the C++ compiler

4.1. Installing Microsoft Visual C++ 6.0

The required version is the minimum configuration (Standard Edition). Simply follow these steps:

- Put the MS Visual C++ CD-ROM in the drive bay
- Run setup.exe from the CD
- Enter the CD key
- On some PCs Internet Explorer 4.01 will be installed. This is not necessary for this software but Microsoft installs it for Visual C++
- Select the DCOM option (only in some installations)
- Select Typical Installation when prompted
- It is not necessary to install the MSDN software or to register as a Microsoft user
- **If the system asks if you want to install the environment variables to access the compiler, click on YES**, otherwise the compiler will not be accessible.
- Reboot the system

4.2. Installing Microsoft Visual Studio 2003

Follow these steps:

- Put the Microsoft Visual 2003 CD-ROM in the drive bay.
- Run setup.exe from the CD
- Enter the CD key and install the compiler.
- Reboot the system
- Test that it works. Open a DOS window and enter the command `cl` (MS C++ compiler). If the Operating System displays "unknown command" the path is wrong. If you get "Microsoft ..." it is correct and you can continue.

4.3. Installing Microsoft Visual Studio 2008

Follow these steps:

- Put the Microsoft Visual 2008CD-ROM in the drive bay.
- Run setup.exe from the CD
- Enter the CD key and install the compiler.
- Reboot the system
- Test that it works. Open a DOS window and enter the command `cl` (MS C++ compiler). If the Operating System displays "unknown command" the path is wrong. If you get "Microsoft ..." it is correct and you can continue.

4.4. Environment variables

The C++ programming language compiler must be installed BEFORE installing the program.

Additionally, when someone uses a compiler it is necessary to set previously a set of values related with some locations required by the compiler. This is usually done by modifying the Operating System (OS) environment variables named PATH, LIB and INCLUDE.

The environment variables PATH, LIB and INCLUDE are edited in the menu "Edit > Options...", so it is not necessary anymore to set any environment variable externally. The variables for each compiler are listed in the following section. For more information see User Manual.

4.4.1. Microsoft Visual C++ 6.0

INCLUDE

C:\Program Files\Microsoft Visual Studio\VC98\atl\include

C:\Program Files\Microsoft Visual Studio\VC98\mfc\include

C:\Program Files\Microsoft Visual Studio\VC98\include

LIB

C:\Program Files\Microsoft Visual Studio\VC98\mfc\lib

C:\Program Files\Microsoft Visual Studio\VC98\lib

PATH

C:\Program Files\Microsoft Visual Studio\Common\Tools\WinNT

C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin

C:\Program Files\Microsoft Visual Studio\Common\Tools

C:\Program Files\Microsoft Visual Studio\VC98\bin

4.4.2. Microsoft Visual Studio 2003

PATH

C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\bin

C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\bin

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\bin

INCLUDE

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include

C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\include

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Include

LIB

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\lib

C:\Program Files\Microsoft Visual Studio .NET 2003\SDK\v1.1\Lib

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\PlatformSDK\Lib



4.4.3. Microsoft Visual Studio 2008

PATH

C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE

C:\Program Files\Microsoft Visual Studio 9.0\VC\bin

INCLUDE

C:\Program Files\Microsoft Visual Studio 9.0\VC\include

LIB

C:\Program Files\Microsoft Visual Studio 9.0\VC\lib

C:\Program Files\Microsoft SDKs\Windows\v6.0A\lib

4.4.4. Gcc 4.4 on Windows

PATH

C:\Program Files\gcc\mingw_gcc_4_4\MinGW\bin

C:\Program Files\gcc\mingw_gcc_4_4\msys\1.0\bin

This page intentionally left blank



5. Installing the program

When the installer is executed, the program is installed into a given folder in the hard disk of the computer. The installer suggests a default folder for installing the program.

Follow these steps:

- Put the CD in the drive bay (only if you have a CD-based installation).
- Run the program executable file (.exe)
- Type the serial number in **serial** entry field.
- Install the program in the directory of your choice (a default directory is suggested)
- Restart the system.
- The main executable is located in the directory installation/**bin**. Create a shortcut to this program on your desktop. You can then start the program by double clicking on the icon.

This page intentionally left blank



6. Installing the protection based on Hardware Keys

In this chapter is described how to install both USB driver and network server for the hardware keys. In addition, it is provided **SecureUpdate**, a program that is used for updating license information.

6.1. Warning

The hardware key software (USB driver and network server) works in Microsoft Windows Networks.

Hardware Key must be inserted in USB port **after** installing hardware key software and installing the program.

6.2. Installation steps

6.2.1. STEP 1: Check that the program is installed

Install the program on each machine in which you want to use it following the instructions in this manual.

6.2.2. STEP 2: Install the hardware key software (USB driver and network server)

If you have purchased a standalone license, install the following program (that can be found in the installation files) with the **complete** option in each machine in which you have installed the program.

!SentinelHK1.0.2!Sentinel Keys Protection Installer\English!SETUP.EXE

If you have purchased a floating license edition, install the program below also in the machine in which you want to plug the network USB hardware key. This machine may be the same in which you installed the simulation tool.

6.2.3. STEP 3: Insert the hardware key and run it.

If you are using a standalone license, insert USB key in your machine USB port and run the simulation tool.

If you are using a floating license, insert hardware key in the USB port of the machine that you want to act as network server. Then run the simulation tool in each machine.

In this last case you can speed up the connection with the network server using the following file:

!SentinelHK1.0.2!sntlconfig.xml

This file must be copied in the program installation directory, inside **/bin** folder, and it has to be modified with the IP direction of the machine that is wanted to be network server.

The semantics of the file is as follows: if the **sntlconfig.xml** file is not present in **/bin** folder, the hardware key driver will perform a broadcast over the local network, but if it is present, the IP inside the file will be used to connect with network server directly. Please, take into account that if the file is present but is bad configured, the program will not run.

This page intentionally left blank



7. Overview of the program

7.1. Short history

The name EcosimPro name has been changed in Version 3.0; before it was known as Ecosim. Due to some commercial issues, EA Internacional decided to change the name to EcosimPro.

Here is a brief history of EcosimPro's evolution.

Ecosim 1.0 (1994)

The development of this version began in 1989 within the framework of a contract with the European Space Agency (ESA) under the Spanish Space Programme for the development of a European tool to simulate environmental control and life support systems on manned spacecraft. EcosimPro 1.0 ran under UNIX in 1993.

Ecosim 2.0 (1996)

The setback in the European manned space programmes and the general approach adopted in the development of Ecosim gave rise to the improvement and subsequent extension of the tool to other types of applications up to the time EcosimPro 2.0 emerged in 1996, running under UNIX. This version has been used in important companies, not only in the space industry: DORNIER and DASA of Germany, SNECMA/SEP and CEA of France, ALENIA of Italy, TECHSPACE-AERO of Belgium and VOLVO of Sweden, but also in the development of numerous applications in other sectors: energy (IBERDROLA) and airports (AENA).

EcosimPro version 3.0 (2000)

Based on experience gained during the development and application of the tool, EA Internacional was able to assess its feasibility and make the decision to make it completely commercial and generally applicable to industrial simulation.

A complete redesign was carried out using an object-oriented approach and C++ to produce a robust modelling software package.

The design targeted a PC with Windows because of the wide market and current power of INTEL based machines.

Its characteristics were compared in detail with other reputable tools and a goal was set to carry out a series of improvements that basically focused on:

- A language for simulation and experiments that should be as simple and robust as possible
- The algorithms for arranging and resolving systems of differential-algebraic equations, handling discontinuities and encompassing state-of-the-art technology
- The user graphics interface, that should be constructed as far as possible with commercial tools (eg, Microsoft) to cover a very wide environment (Windows NT, 95 and 98) and that should also be powerful for solving very complex modelling problems involving hundreds of equations

This version was also subjected to a comprehensive test campaign in different areas of application that culminated in the first commercial version at the end of 1999. The test campaign included, but was not limited to, the following:

- European ESPRIT project, with the application of EcosimPro 3.0 in the fields of Waste Processing and Water Treatment
- Tests in university groups with wide experience in simulation, especially in the fields of Control, Process Simulation, Energy, Propulsion and Chemistry
- Environmental Control and Life Support Systems (ECLSS) for the European Space Agency
- Project to develop a commercial library for the Energy sector with a Spanish architect-engineering company

EcosimPro version 3.3 (2004)

A new tool for creating schematics emerges: EcoDiagram. This new tool replaced SmartSketch to create graphical models.

EcosimPro version 3.4 (2005)

New capabilities and improvements have been added both in EL language and graphical interface to supply the user with a more usable tool and a greater modelling quality.

EcosimPro version 4.0 (2006)

The adaptation of EcosimPro to a multiplatform environment has created the needs of making deep changes in this version. EcoStudio and EcoDiagram have been integrated into a single application. Also, EcosimPro shows three different views: Code View, from where components can be created using EL language, Schematic View, where symbols for the components schematics can be created, and Simulation View, where experiments are managed and executed.

EcosimPro version 4.4 (2007)

There are new capabilities in this version: libraries versioning, units, new functions, upgrades in the monitor, new calculation wizard, etc.

7.2. Current applications

The following table shows the main applications of EcosimPro to the present time:

Main Applications with EcosimPro			
AREA	SYSTEM (Client)	Typical Applications/Results	Technologies Involved
SPACE	Environmental Control and Life Support Systems	System definition and design Mission preparation Training Assessment of the mission	Fluids Mass and heat transfer Chemical reactors Control
	Thermal Control of Satellites	Passive insulation (thermal load) Cooling loops	Heat transfer Thermodynamic cycles Control
PROPULSION	Liquid-fuel rocket engine	Mass, hydraulic and heat balances	Fluids
	Gas-fuelling of ionic engines	Adjustment and optimisation of controls	Mass and heat transfer
	Feeding fuel and combustibles to satellite engines	Dynamic simulation (operating transients)	Chemical reactors Control Cryogenics
ENERGY	IBERDROLA	Energy Balances	Fluids
		Dynamic simulation (operating transients)	Mass and heat transfer
		Adjustment and optimisation of controls	Chemical reactors Control Cryogenics
DEFENCE	SEA-SKIMMER Anti-missile Defence System	Trajectory	Kinematics and Dynamics
		Configuration of control	Warheads
		Success criteria	Servo control systems



WASTE PROCESSING	Técnicas Reunidas	Mineral processing Aqueous metallurgy Recovery of heavy metals from liquid effluents	Leaching Cementation Filters Heat-Exchangers
WATER TREATMENT	Bristol Water	Abstraction and treatment of water to potable standards	Chemical reactors Coagulation and precipitation Disinfecting and oxidation Sand filters
TRANSPORT AND LOGISTICS	Air terminal passenger traffic (AENA)	Flow of passengers in space and in time	Queue theory
		Average process and connection times for different kinds of passengers	Optimisation by assignor/supervisor of resources
		Evolution in time of necessary process elements: check-in counters, controls, trains, baggage reclaim halls, etc	
TRANSPORT AND LOGISTICS (Cont.)	Automatic Baggage Handling (AENA)	Average flows and times of baggage handling Different control schemas Resources (available space and man-hours) for operation Defective operation (with failures) Sensitivity studies (capacity and design margins)	Queue theory Optimisation
TRANSPORT AND LOGISTICS	Storage and Distribution of Merchandise (EXEL-LOGISTIC)	Warehouse occupation levels Resources necessary (available space, machines, staff) for operation Defective operation (with failures)	Queue theory Optimisation

7.3. Key concepts

The fundamental concepts are:

- Component:** This represents a model of the system simulated by means of variables, differential-algebraic equations, topology and event-based behaviour. The component is the equivalent of the “class” concept in object-oriented programming.
- Port connection type:** This defines a set of variables to be interchanged in connections and the behaviour and restrictions when there are connections between more than two ports. For instance, an electric connection type uses voltage and current as variables to be used in connections. The connection port avoids connecting individual variables; instead, sets of variables are managed together.
- Partition:** To simulate a component, the user first has to define its associated mathematical model; this is called a partition. A component may have more than one partition. For example, if a component has several different boundary conditions, depending on the set of variables selected, each set of variables produces a different mathematical model, or partition. The next step is to generate experiments for each partition. The partition defines the causality of the final model.

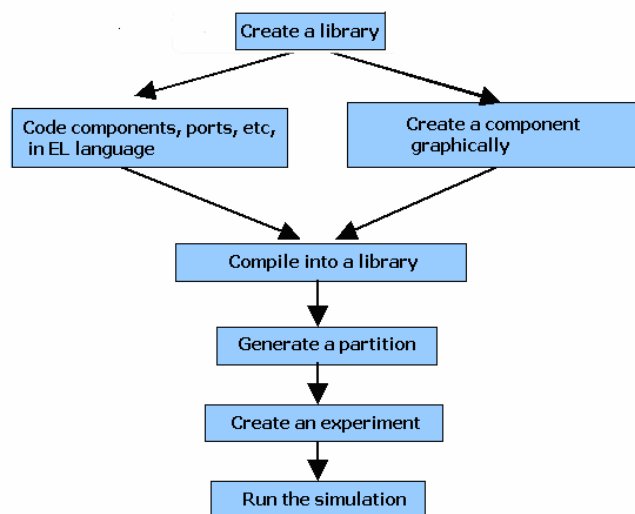
- **Experiment:** The experiments performed for each partition of the component are the different simulation cases. They may be trivial for calculating a steady state or very complex with many steady and transient states changing multiple variables in the model.
- **Library of components:** All the components are classified by disciplines into libraries.

7.4. Working in EcosimPro

A EcosimPro component can be built either from scratch or formed from basic predefined components. The components are taken either from a system library or are defined by the user. Each component contains a mathematical description of the real-world component it represents. Components have interfaces with each other by means of connection ports. Components can be joined together by their ports to form a new component. EcosimPro is hierarchical in the sense that models may also have ports and so can be joined to other components to create more complex components. Components can inherit behaviour from other components, allowing reuse of tried and tested code from parent components.

An experiment defines the initial conditions and the boundary conditions of the mathematical model, and the desired solutions (transient or steady state). Experiments can range from trivial to very complex using sequential language with **for**, **while** and **if** statements, function calls, etc, and some special functions for calculating steady states, integration of the model, etc.

The process of building components in EcosimPro consists of the following steps:



The first step is to create (or reuse) a library, then to code the new component(s) and port type(s), either using the EcosimPro Language (EL) or the graphical interface tool. The next step, simulating a component, involves generating an associated mathematical model or partition, then creating an experiment for that partition. Finally the simulation is run, either in batch mode or visually in the EcosimPro Experiments Monitor.

The EcosimPro tool consists of the following software modules:

- The EcosimPro Simulation Language (EL) for expressing both the simulation components and the experiment as text
- EcosimPro libraries of components classified by discipline (Gas Turbine, Energy, Propulsion, Control, etc)
- The EcosimPro main window providing a user-friendly environment for creating new components using the EcosimPro Simulation Language (EL)



- The schematic tool. This is a tool designed by EA International for drawing a schematic simulation model which can then be compiled as an EcosimPro component
- A graphical experiment monitor to run simulations and plot graphs of variables

7.5. The Simulation Language

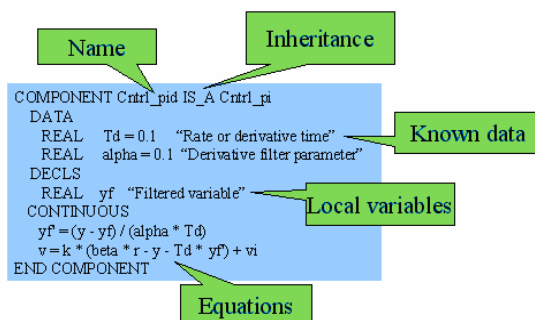
The simulation language (EL language) has a rich set of language features for expressing both models and experiments. The language provides sequential statements similar to FORTRAN, C, etc, as well as continuous and discrete statements for modelling physical behaviour.

The program compiler checks the lexical, syntactical and semantic correctness of any new component. If the compiler detects an error it reports the problem with the line number and gives an explanation for the error.

EL provides capabilities for defining components, connection ports, functions, enumeration types, etc. The final simulation model is translated into native C++ code for use in other software modules to run the simulations programmatically.

7.6. Look-and-feel of an component

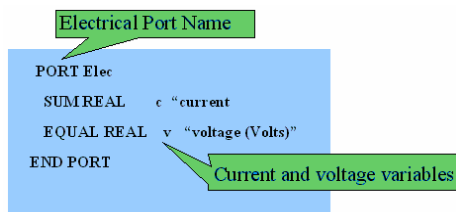
The figure below illustrates the look and feel of an component modelled in EL language.



This component models a typical PID controller. It shows how inheritance allows much of the code to be reused from the parent components. Each new component defines its own data, variables and equations.

7.7. Look-and-feel of an Port Type

EL allows the modeller to create very powerful connection port types by including intelligent behaviour. For instance, that the behaviour of a variable in a multiple connection must follow the rule of equality in all the connections (EQUAL flag) or that the addition of all the variables must be equal to zero (SUM flag). The modeller can then insert equations manually without worrying about multiple connections, as the tool handles it automatically. The example below shows an electrical port with two variables: current and voltage. The first is SUM and the second EQUAL behaviour.



7.8. Libraries

Everything will be inserted into a library. Usually the libraries are classified by discipline such as Electrical, Control, Fluid, etc. The program is supplied with a set of standard libraries and others can be purchased separately. The figure below shows a typical configuration of libraries. There is a workspace containing several libraries such as CONTROL, ELECTRICAL, etc. Each library contains different items, such as components, port types and functions.

7.9. Graphical user interface

7.9.1. Main window

The tool provides a very powerful Man-Machine interface for building new components. This window is shown below:

There are three types of views that display suitable library components that the user may want to create or view (Code View), generate schematics (Schematic View) or simulation experiments (Simulation View).

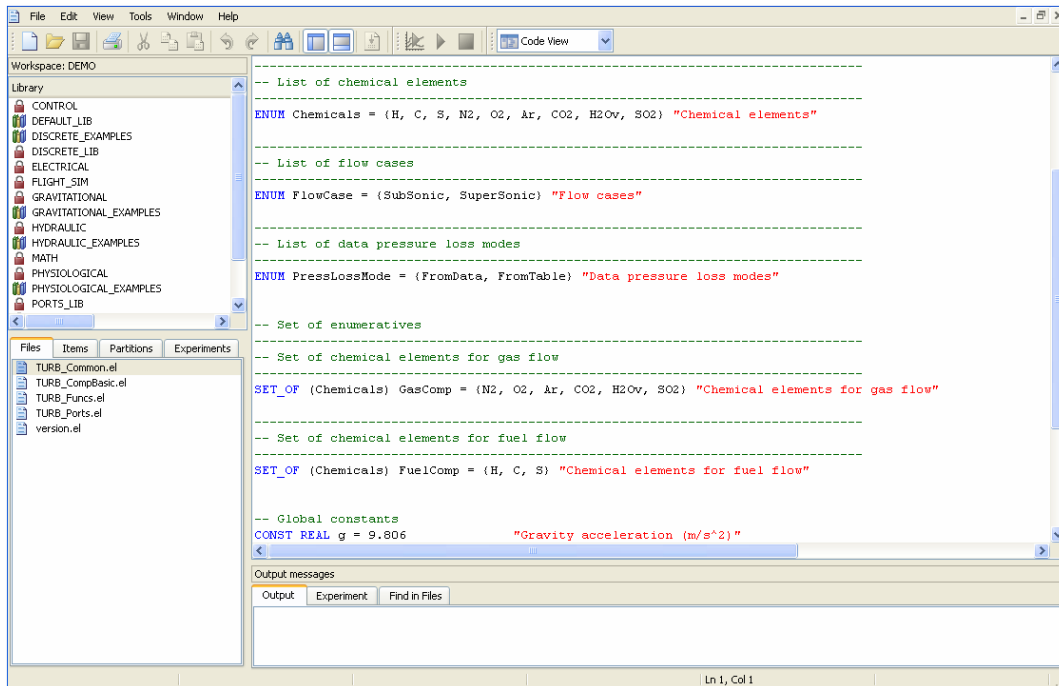
Each view has common areas, namely:

- Editing area: the right-hand area contains the editing windows. Each window is associated with a file that can contain one or more components, ports, functions, etc
- Workspace area: the left-hand side provides tabs for different views of the library
- Message windows: the bottom area provides three different windows that display messages from the system, general messages (output tab), messages associated to the simulation and the experiment (experiment tab) and search messages (find in files tab)

Selection of the view required will activate the components necessary for the work.

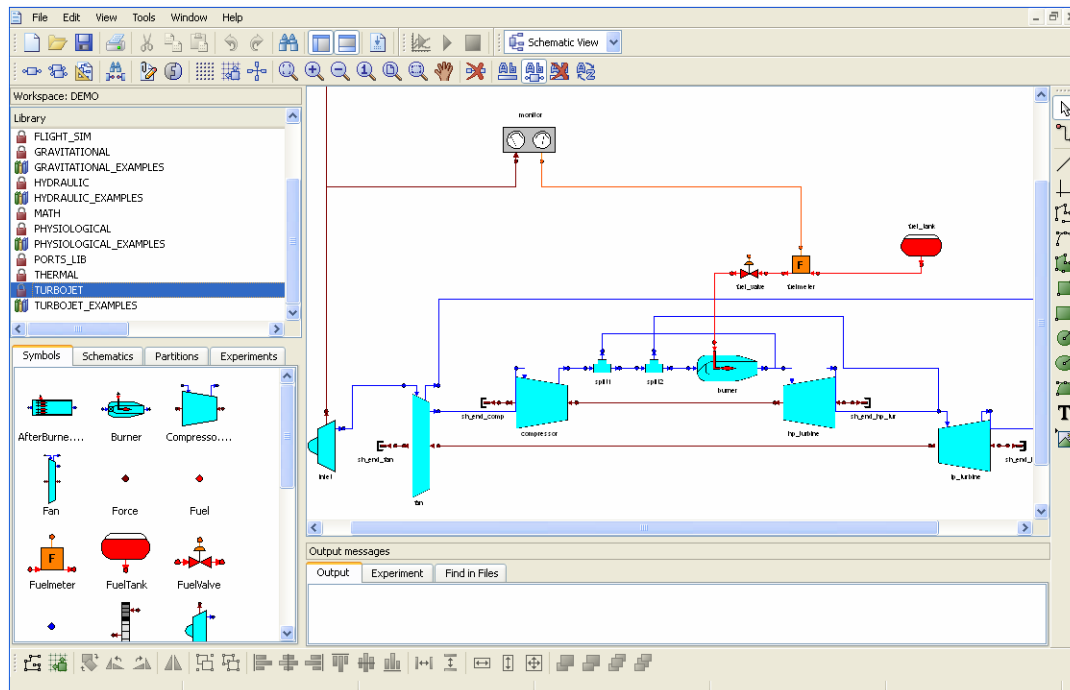
The following will be displayed in the Code View of the Workspace area:

- Files, items, partitions and experiment: for EL files, items, partitions and experiments associated with the library



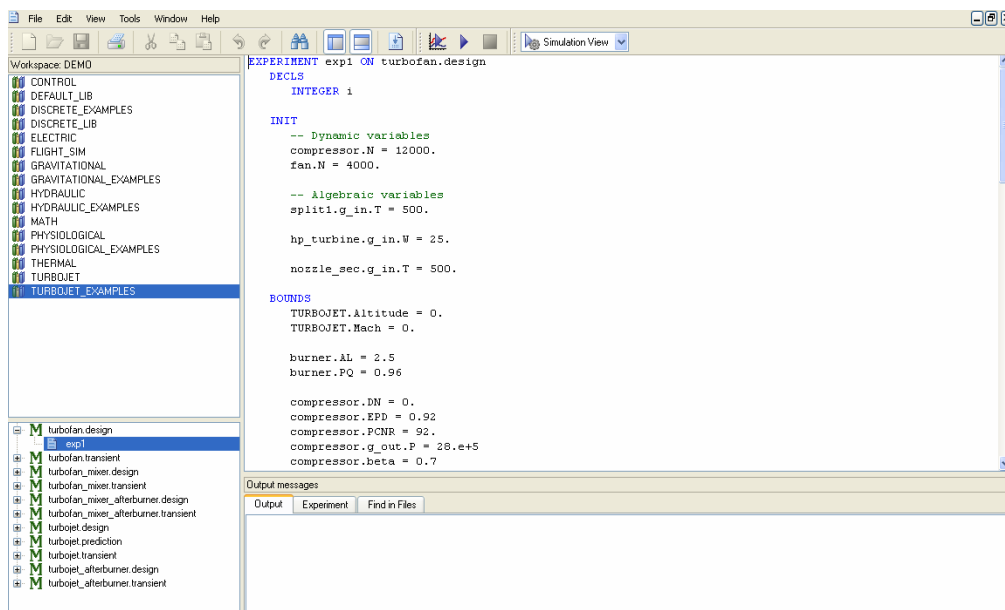
The following will be displayed in the Schematics View of the Workspace area:

- Symbols (associated with the library components), schematics, partitions and experiments



The following will be displayed in the Simulation View of the Workspace area:

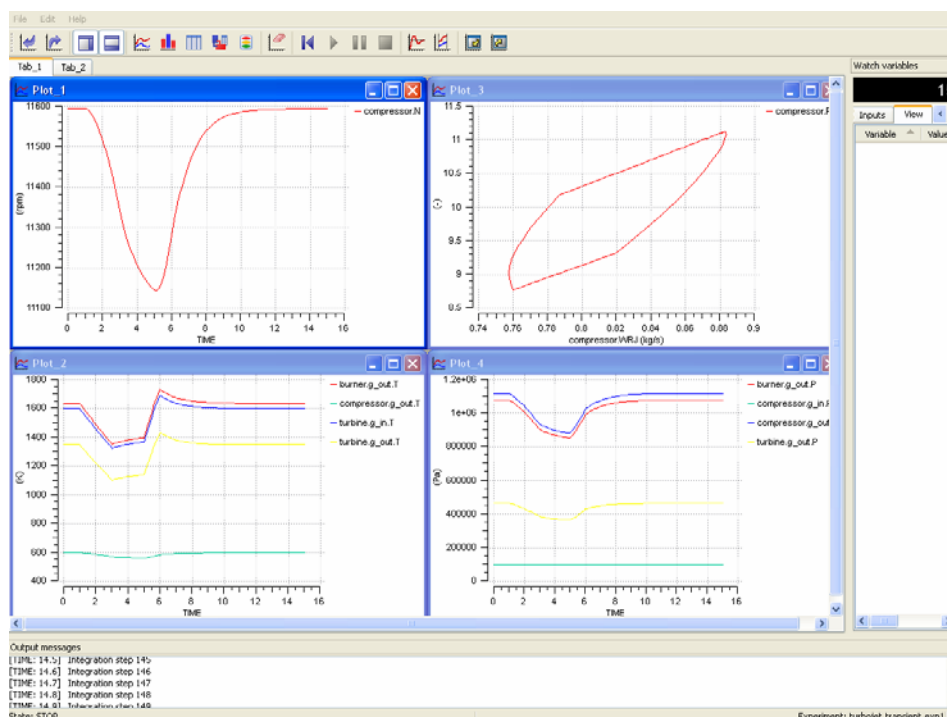
- Experiments: experiments associated with the models generated



The user can create components, libraries, mathematical models, experiments, etc, and also run simulations without graphical plots. Simulations with graphical output are run in the Experiment Monitor (see the next section).

7.9.1.1. Experiments Monitor

This is used to run simulations with graphical output. It is launched from the EcosimPro Main Window in "Simulation View". After creating an experiment, right click on the experiment name in the explorer area and select the option "Simulate in Monitor", then the Experiments Monitor is displayed. It looks like this:



The window is divided into several areas:

- Menu area (top). This provides menus for creating plots, tracing variables, calculating steady and transient states, etc. Below the main menu there are three video recorder type buttons: Start, Pause, Stop and Quit the Simulation



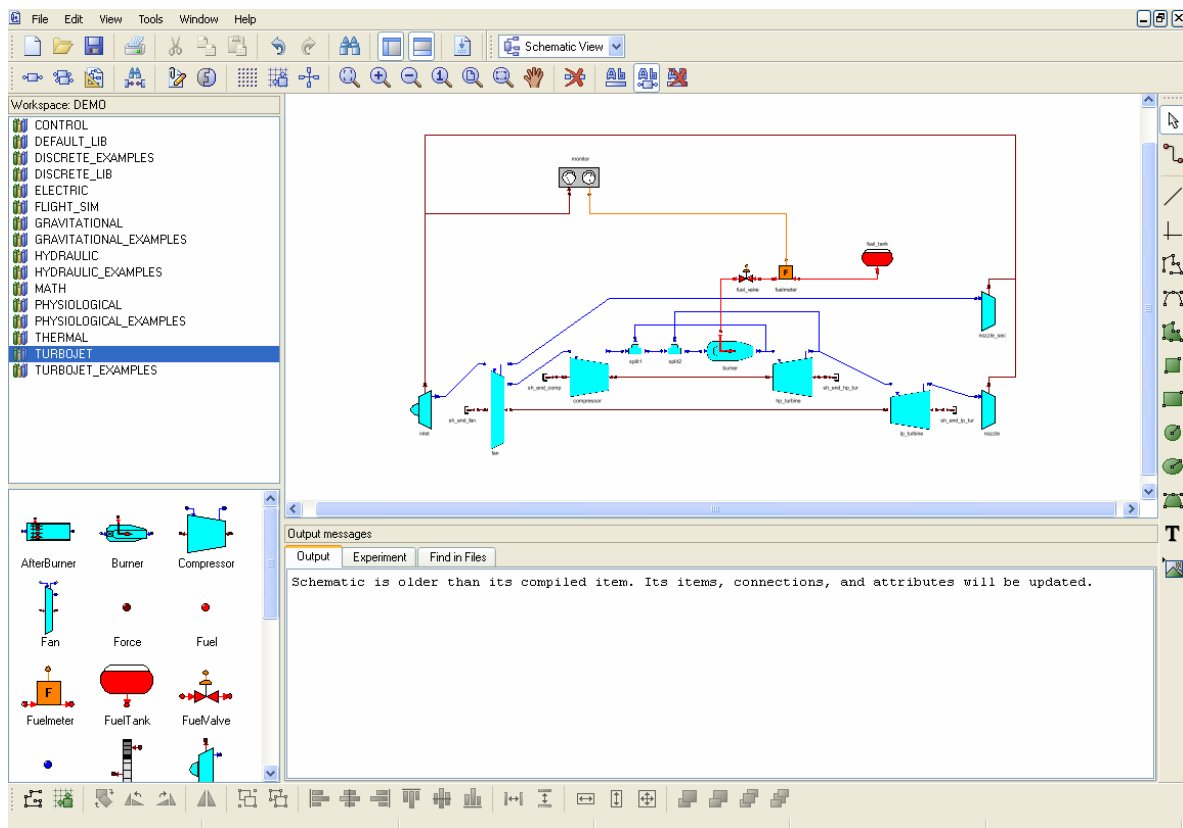
- Plotting area (large left-hand window). This contains all the graphs of the simulation. Multiple windows can be viewed with one or more plotted variables in each. Each of these windows can be copied and pasted into any Office application (Word, Excel, Power-Point, etc) for documentation purposes
- Watch area (right-hand window). This displays several variable values. Their values are updated during every communications interval. The current simulation time is shown at the top
- Message area (bottom). All simulation events are logged in this window. It enables you to follow up a simulation, trace all the integration steps and handle events

7.9.2. Schematics

In EcosimPro all components can be created using EL, even if they have very complex topology. But EcosimPro makes it easier to create complex (and simple) components graphically without having to learn the details of EL in "Schematic View".

The example of creating a component given in this manual shows you the most important concepts for any other application.

The following figure shows a component that has been created graphically. On the left-hand side of the window is the palette of available components (one icon for each component). The user drags and drops components from the palette onto the canvas, and then connects them and sets the attributes of every instance of the object. Finally, selecting the option *Compile* it creates a new component in EcosimPro. This component is then simulated in the normal way as if it had been created using the EcosimPro language (EL).



7.9.3. Object Editors

EcosimPro provides visual editors to enter the object properties. Double clicking an object displays a window for editing the object. The figure below shows an example:

Library : TURBOJET

Type : Compressor

Name : compressor

Show label

Name	Type	Value	Units	Description
DATA				
I	REAL	10	kg*m ²	Inertial moment (kg*m ²)
ND	REAL	10000	rpm	Design rotational speed (rpm)
CG1	REAL	1	-	Correction coefficient for corrected mass flow (-)
CG2	REAL	1	-	Correction coefficient for efficiency (-)
CG4	REAL	1	-	Correction coefficient for compression work (-)
F1	TABLE 2D			Compression work (J/kg*K) vs adime...
F2	TABLE 2D			Efficiency (-) vs adimensiona...
F3	TABLE 2D			Corrected mass flow (kg/s) vs adimensi...

The object editor accepts different data types: REAL, INTEGER, ENUMERATION, BOOLEAN and tables (1D, 2D and 3D). These editors allow the user to modify the attributes of an object instance in an intuitive way. The example in the figure below shows the 2D table editor for an attribute of a component:

File Edit Axis

Description : J/kg*K] vs adimensional speed (-)

	A	B	C	D	E	F	
1	X\Y	0.2	0.3	0.4	0.5	0.6	0.7
2	60	0.268984	0.274445	0.2821	0.29878	0.300624	0.306
3	70	0.359459	0.384362	0.419562	0.433043	0.444121	0.466
4	75	0.430983	0.432107	0.475888	0.496047	0.517361	0.531
5	80	0.53365	0.549752	0.577993	0.603177	0.613194	0.627

Rows : 100 , Cols : 100 , Cell : 1A

Number of rows : 1

Number of columns : 1

Cancel

The user can dynamically modify the values of the table, and the graph is updated automatically. The graph can be rotated in any direction.



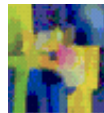
8. Example of modelling and simulation

In this section you will learn how to use EcosimPro to create a simple component, create an experiment for it, and run the experiment. All the examples given here can be found in the library DEFAULT_LIB.

First of all, open EcosimPro's Main Window by launching the EcosimPro.exe file.

It can also be executed from the Start menu in Programs>EcosimPro>EcosimPro.

It is a good idea to create a shortcut to this file on your Windows desktop, to avoid searching for it every time. Select the file in Windows Explorer and drag it onto the desktop. A new icon should appear like this:



Now you can start EcosimPro just by double clicking the shortcut.

8.1. Your first component using source code

To begin using EcosimPro we will first create a simple component to solve a differential equation. EcosimPro was designed to simulate complex systems, but it can also be used independently of a physical system as if it were a pure **equation solver**. The example in this section illustrates this use. It solves the following differential equation to introduce a delay to variable "x":

$$\frac{dy}{dt} = (x - y) / \tau$$

which is equivalent to

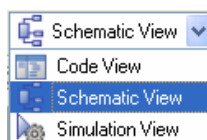
$$y' = (x - y) / \tau$$

where x and y have a time dependence that will be defined in the experiment. τ is data given by the user; we will use a value of 0.6 seconds. Basically this equation introduces a delay in the x variable with respect to y with value τ . To simulate this equation we will create an EcosimPro component with the equation inside. The main tasks are:

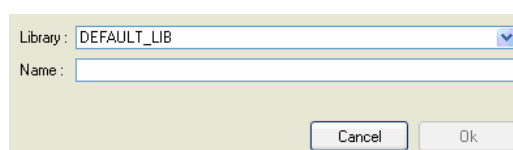
- Create a new source file associated with a library
- Compile the file
- Generate a default mathematical partition
- Create an experiment
- Run the experiment

We can now go ahead; just follow these steps:

- At the top of EcosimPro's Main Window you will see a drop-down menu on the Toolbar. "Schematic View" is the default option when you start the application, but in this example we are going to select the option "Code View":



- We will need to use the library DEFAULT_LIB and must check that it is open in the Workspace. A list of open libraries is displayed at the top of the Workspace which is on the left-hand side of the screen. If the library is not displayed, go to File > Open > Library... and select the library DEFAULT_LIB
- From EcosimPro's Main Window, select File > New > Source File... This will open a dialogue box where we key in the name of the new code file and select the library with which we wish to associate it. Type in the name "equation_test" and select the library DEFAULT_LIB from the list. A new window will open in the top left-hand part of the screen



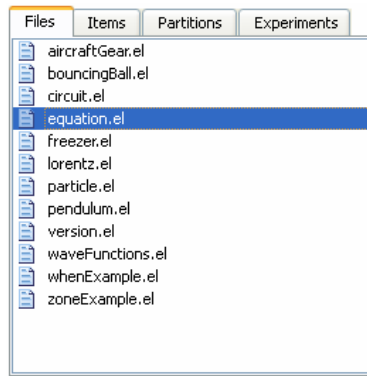
The component to be simulated in EL is like this:

```
COMPONENT equation_test
  DATA
    REAL tau = 0.6      "delay time (seconds)"
  DECLS
    REAL x, y
  CONTINUOUS
    y' = (x - y) / tau
  END COMPONENT
```

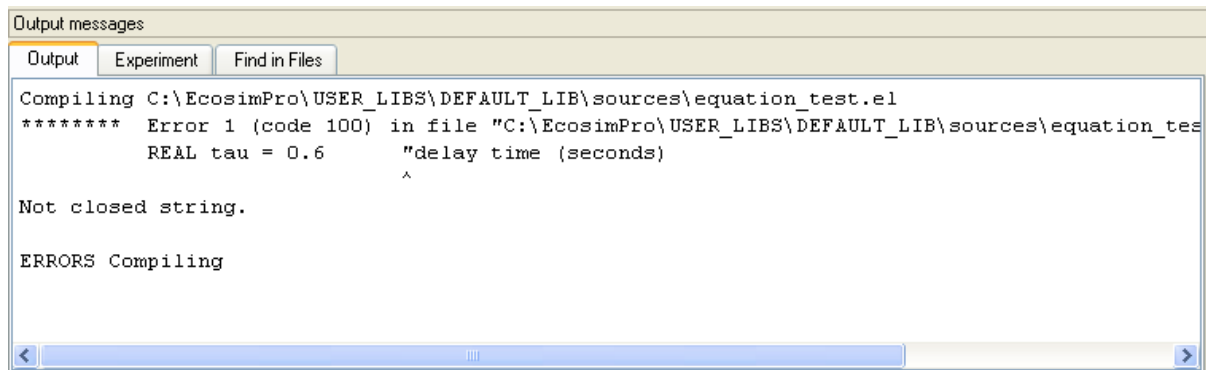
- Insert the code:

```
COMPONENT equation_test
  DATA
    REAL tau = 0.6      "delay time (seconds)"
  DECLS
    REAL x, y
  CONTINUOUS
    y' = (x - y) / tau
  END COMPONENT
```

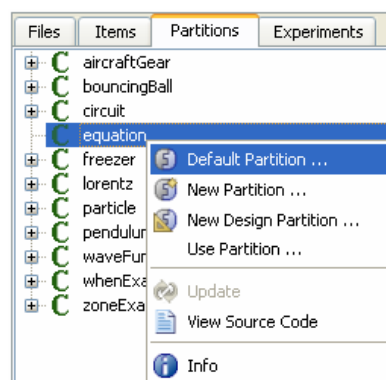
- Compile this file by selecting the library "DEFAULT_LIB" in the top of the Workspace and under the "Files" tab at the bottom of the Workspace right-click on the file name (equation_test.el) and select "Compile"



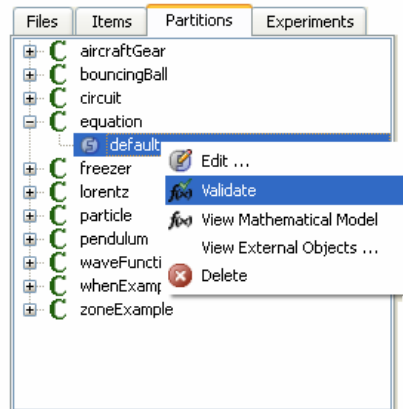
- If the source code is correctly written, the message "Compilation OK" will appear in the "Output" window (at the bottom of the screen). If not an error message is displayed. You can click over the error message (bottom window) and the cursor automatically points to the erroneous line. Correct the error and compile again. The following example shows an error message indicating an unclosed string (the string associated with the variable must be enclosed in double quote)



- Check that the component has been added to the DEFAULT_LIB library. At the bottom of the Workspace, select the "Items tab" (with the library DEFAULT_LIB selected) and check that the component "equation_test" is in the list
- Before creating an experiment for this component, a partition has to be created. In the bottom window of the Workspace, select the tab "Partitions". Right-click on the component name (equation_test) and select the option "Default Partition...". Type in a name for the partition (by default it is "default") and click OK. This generates a default mathematical model for this component. Right-click in the new partition and select the option "Validate". This checks that the partition we have created is correct and we can create experiments for it



- A "+" sign should appear to the left of the component name. Right-click it to display the default partition.

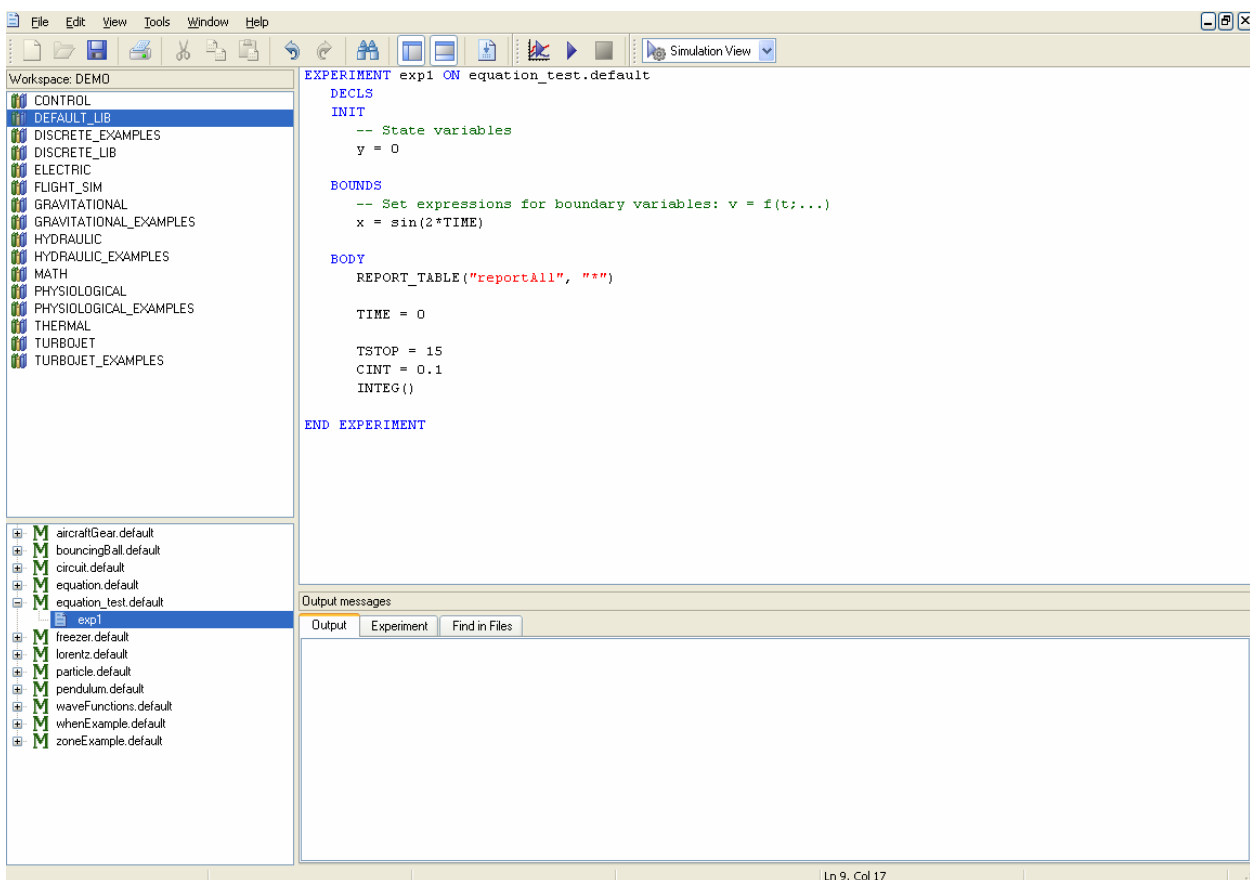


- Select now the “Experiments” tab where the experiments are managed. Right-click “equation_test.default” and select “new experiment”. Type in the name of the experiment and click OK. This will create a new default experiment. A window will open displaying the experiment code file for editing. Replace the boundary condition “x” with the expression:

x = sin(2*TIME)

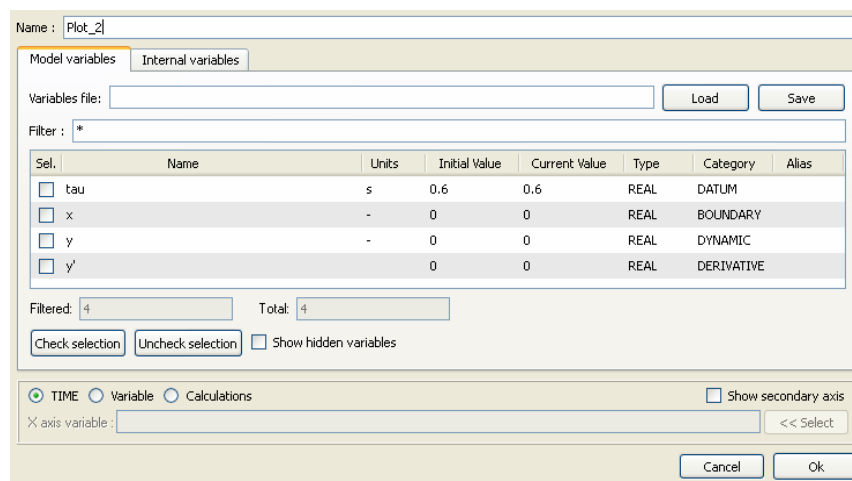
```
BOUNDS  -- set expressions for boundary variables: v = f(t,...)
x = sin(2*TIME)
```

- Save this experiment by clicking File and selecting the Save option. You can also save it by clicking the “Save” icon on the Toolbar. The final layout should be:

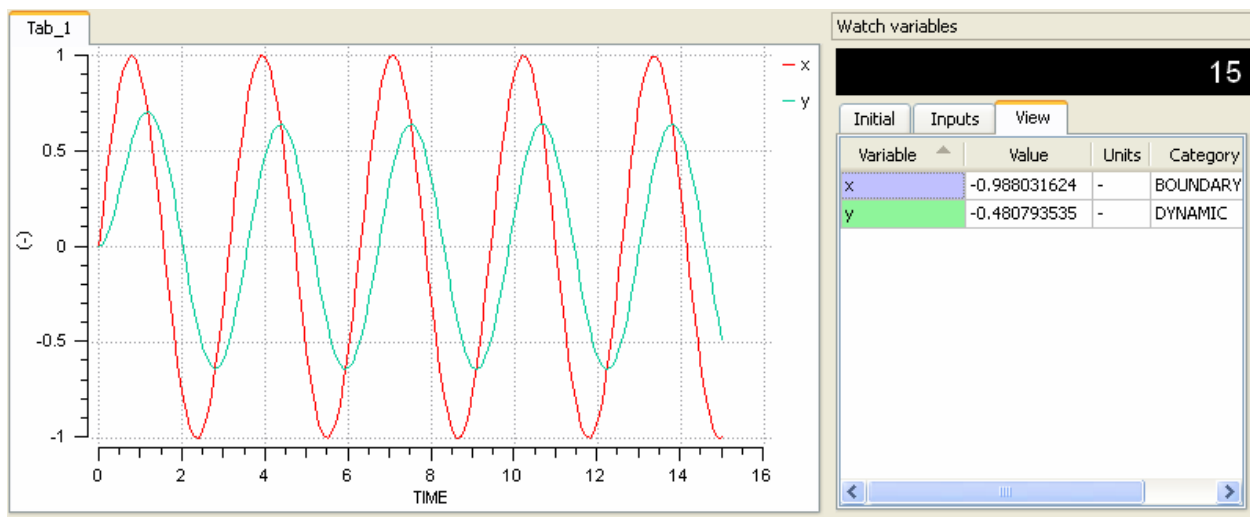




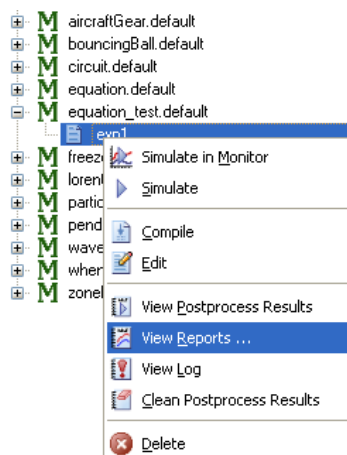
- Basically this experiment integrates the model from 0 to 15 seconds with a communications interval of 0.1 seconds. It also generates a text file report called "reportAll" containing all the variable values during the simulation
- Simulate the experiment: right-click "exp1" within "equation_test.default" (in the Workspace) and select the option "Simulate in Monitor". The experiment will compile automatically and display an error message if the code is not written correctly. It will then simulate the experiment
- EcosimPro automatically generates an executable simulation file
- The Experiments Monitor window opens. Add a new plot with the variables x and y by clicking the icon "New plot" on the Toolbar, then highlight the variables x and y and click OK



- Add a new watch display for the variables x and y. Right click on the watch area, select "Edit watch" from the context menu, select the variables x and y and click OK
- Select "Play" from the Toolbar. We can see in the plot window how the variables change. In the right-hand window we can see how the numeric values of the selected variables change



- The simulation results are saved in a text file called "reportAll". To see the results of the experiment "exp1" return to EcosimPro's Main Window. In the bottom window of the Workspace, right-click "exp1" and select "View Reports...". Select the report you want to open and click OK

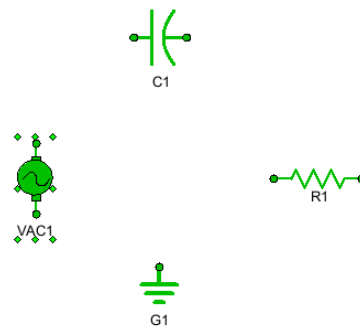



- You can drag and drop your report files to Microsoft Excel
- You can also edit the log file with all the events of simulation
- Congratulations, you have done your first EcosimPro simulation! The steps you have followed are the same even for complex simulations, so you have learned most of the key concepts. Your first component using schematic

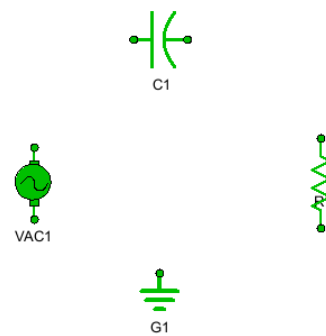
In this section we create a basic electric circuit but this time using the schematic editor instead of EL. The philosophy for creating any other kind of schematic (for chemical, fluid, mechanical, etc) is the same, so having learned these key concepts you can go on to prepare any other kind of simulation schematic.

The simulation schematic will be an electric rectifier using a resistor, a capacitor, a voltage generator and ground. EcosimPro comes with an electric library that contains most of the basic components needed to create electric circuits. The steps to follow in this simulation are:

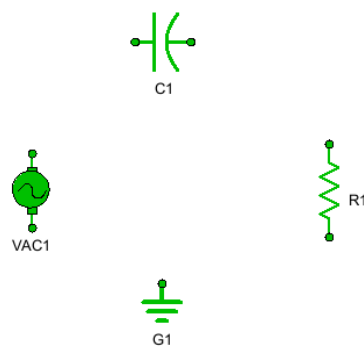
- Open the EcosimPro Main Window (as in the previous section). "Schematic View" is the default option when you start the application and this is where we are going to work in this example. If another view is displayed, change it by selecting "Schematic View" from the drop-down menu on the Toolbar
- Since we are going to work with the "ELECTRICAL" libraries, we need to open a Workspace that contains them. The previous section explains how to do this
- To create a new file select File, New and Schematic. Note that this is different from the first step of the last section, because in this case we will not create a new EL file but a graphical one. You can also create it from "Schematic View" clicking "New" button from the toolbar
- Type in the name of the schematic "my_circuit" and select from the drop-down menu the library with which it will be associated (DEFAULT_LIB)
- Select the ELECTRICAL library in the Workspace. As we have selected "Schematic View", the library symbols (symbols palette) will be displayed at the bottom of the Workspace. If symbols have not been created for the libraries the palette will be empty. The display will look like this:
- Now you can start to design the circuit by dragging the components from the palette onto the canvas
- First drag the components, C (Capacitor), R (Resistor); VAC (Voltage source) and G (Ground), and drop them onto the canvas. Use the layout illustrated below:



- To save the design, click the “save” icon on the Toolbar or click File > Save
- Continue designing the circuit. Rotate R1 90 degrees to the right to align the connection points so they can be connected. To rotate a symbol, select it and click 
- The final layout should look like this:



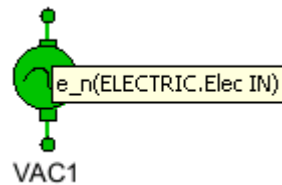
- To relocate the component tags, hold down the SHIFT key, left-click the tag you wish to move and drag it to the desired position



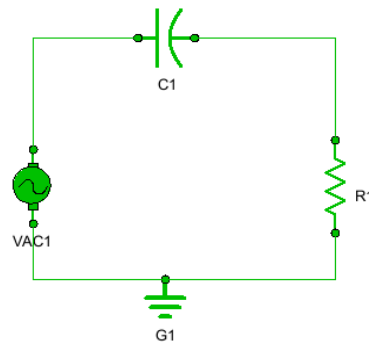
- To connect the components, select the connection button on the right-hand toolbar



- Now you can connect components by first left-clicking the source gate, then clicking the different points through which you want the connection to pass up to the destination gate, where you end with one final click
- To confirm that the cursor is really on top of a gate, when you move it over one with the connector option selected, a box pops up with an anchor and a tag showing the name of the gate




- Make all the connections as shown below (R1 lower port is connected to both G1 and VAC1):



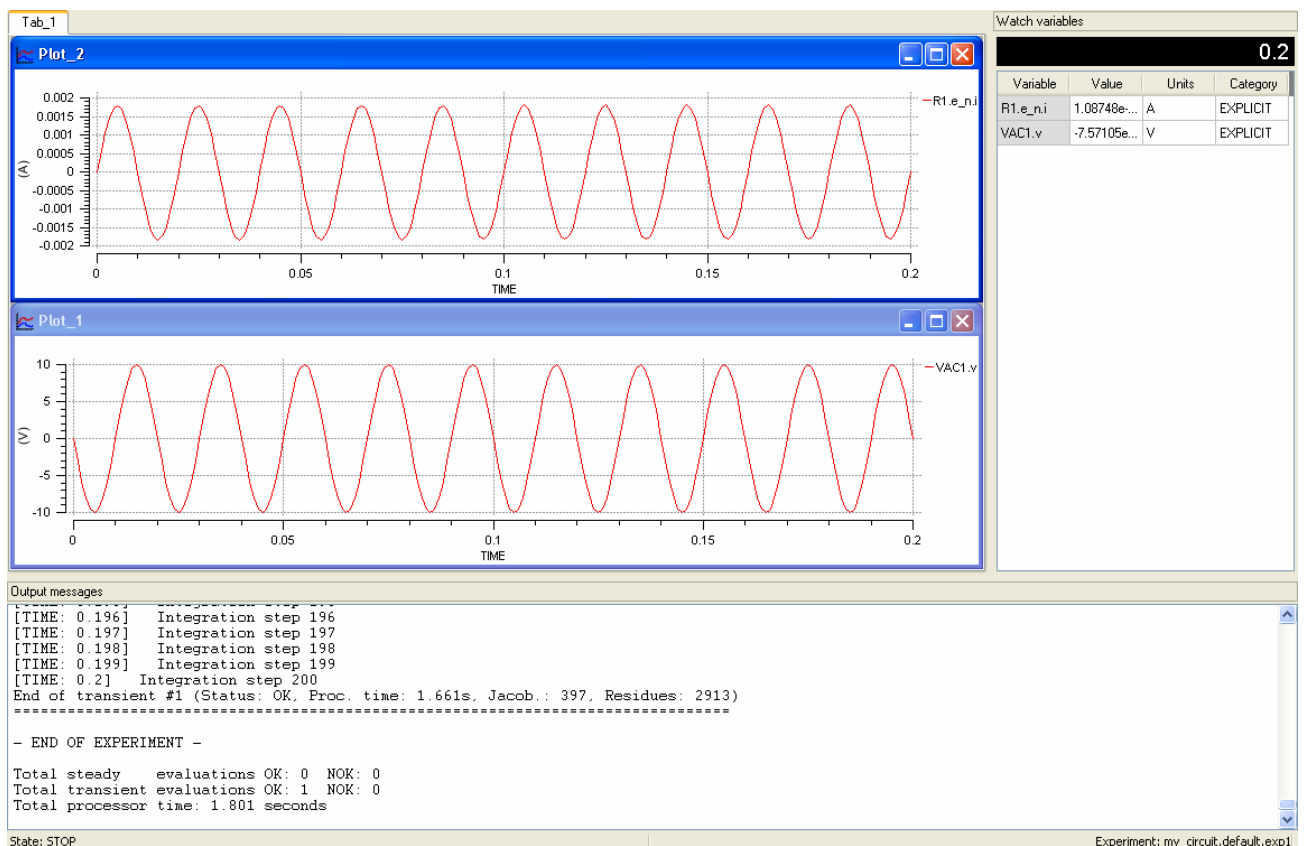
- Don't worry if the lines are not very straight, you will have time later to improve your skill with Schematics. Now, as an example of how to customize the components' attributes, we will change the capacitance of C1 to 0.00005 Farads and the resistance from R1 to 5500 Ohms. To change any of these attributes double-click on the corresponding symbol to open the Attributes Editor. For the Resistance it looks like this:

Library: ELECTRIC				
Type: R				
Name: R1				
<input checked="" type="checkbox"/> Show label				
Name	Type	Value	Units	Description
DATA				
R	REAL	1000.	Ohm	Resistance at reference temperature (Ohm)
TR	REAL	300.	K	Resistor temperature (K)
TR1	REAL	0.	Ohm/K	Linear temperature coefficient (Ohm/K)
TR2	REAL	0.	Ohm/K ²	Quadratic temperature coefficient (Ohm/K ²)

- Change the value of R to the new value and click OK (the row changes colour because the new value differs from the default one for R). Do the same with the Capacitor. This circuit will be a new EcosimPro component. To compile it in EcosimPro, select the Library Toolbar and click . EcosimPro then automatically compiles the EL code generated by the Schematic, displaying a successful compilation message when finished
- You can go to the Code View and check that it has been added to the library files. Look under the Files tab for DEFAULT_LIB library; "my_circuit.el" should be there



- As explained in the previous section, we have to create a partition for the component before we can run the simulation. Under the tab "Partitions", right-click "my_circuit" and select "Default Partition...". Type in the name of the new partition and click OK
- Now it is ready for the simulation. Proceed as in the previous section. Go to the Simulation View. Select "my_circuit.default" (if the name of the partition you have created is "default") from the DEFAULT_LIB library, right-click the option "New experiment" and type in a name
- A default experiment text should appear in the editing window. Change the TSTOP to 0.2 seconds and CINT (Communications Interval) to 0.001 seconds
- Save the experiment
- Move the mouse over the experiment name in the Workspace area and select the option Simulate in Monitor
- The Experiments monitor comes up. Add two plots: one for the voltage of the capacitor and the other to plot the current in the resistor (by using the intermediate port e_p or e_n). Then click Start simulation. The full simulation of the circuit is displayed



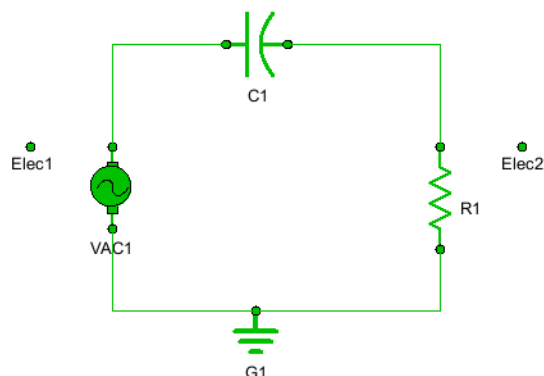
Congratulations! You have just completed your first simulation in EcosimPro, and without needing to know anything about the EcosimPro Simulation Language (EL). And most importantly, you have relearned the key concepts for preparing simulations using Schematics.

8.2. Reusing a component created with Schematic

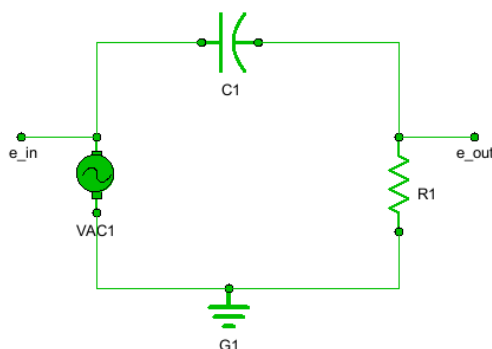
After creating a component with Schematic you may want to reuse it in a new component in other schematics. This section explains how.

We will reuse the previous electric circuit. Obviously, to reuse the circuit we need to add an external connection port, otherwise we cannot connect this complete component to another one.

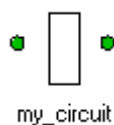
From the Electrical palette, drag and drop two connection points onto the component. They are called Elec ports, represented by a big green dot. Drop the first one close to the VAC output and the other close to the resistor input like this:



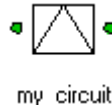
- Customize the connection points. The point close to the source will be type IN (input) and the one close to the resistor will be type OUT (output). Double-click them to edit. Rename the first as “e_in” and the second as “e_out”. Remember to change the Direction for “e_out” to OUT
- Connect the connection points to the component ports as shown below, and save it in the DEFAULT_LIB library



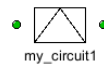
- Compile the schematic again
- You will see that the palette now contains a symbol called “my_circuit” and because it has not been edited, it will appear with the default symbol



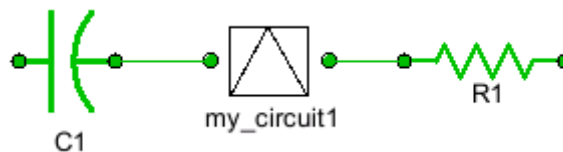
- To edit the symbol, right-click it and select the option “Edit symbol”. You can redesign your component with a drawing like this:



- Close and save the design
- Start a new schematic. Access the DEFAULT_LIB library. The symbol should be on the palette. You can drag and drop it onto the canvas, for example:



- Now you can go to the ELECTRICAL library and create a new electric circuit based on this component. For example:



- When you finish you can compile this new component and start the process of simulation again
- By now you should understand the basic idea: use basic components to create more sophisticated ones, create an icon for them and then reuse the new icon as if it were a simple component. This concept is very powerful and allows you to create complex simulations from simple basic components.

The same principles applied here to electric circuits can be applied to any other discipline: fluid, chemical, propulsion, etc.

8.2.1.1. The bouncing ball example

In this example we will model a rubber ball which, dropped from a certain height, bounces successively on the ground until it stops.

It is assumed that the ball accelerates at the speed of gravity as it falls. Each time it bounces, it loses 20% of its speed.

It must take into account when the simulation has to be stopped. This will be when the ball no longer bounces on the ground. We are crossing into the numeric world and must tread carefully, so we need to allow a certain margin of error.

This can be modelled writing an event into the equation that if the height is less than a certain tolerance, a message is generated and the simulation stops (TSTOP = TIME).

The final model in EL is as follows:

```
COMPONENT bouncingBall
  DATA
    REAL g = 9.806    "gravity (m/s**2)"
    REAL k = 0.8      "restitution coefficient"
  DECLS
    REAL h = 10       "height (m)"
  DISCRETE
    -- event when bouncing on the ground
    WHEN ( h < 0 ) THEN
      h' = - k * h'
    END WHEN
    -- event to detect stopping the simulation
```

```

    WHEN ( h < -0.003 ) THEN
      PRINT("End of simulation")
      TSTOP = TIME
    END WHEN
  CONTINUOUS
    h'' = -g
END COMPONENT

```

We can insert this text into an EL file (as explained in the previous chapter) and generate a default partition and an experiment. We run the following experiment in the Experiments Monitor from 0 to 15 seconds with a communications interval of 0.1 seconds.

```

EXPERIMENT exp1 ON bouncingBall.default
  INIT
    -- Dynamic variables
    h = 10.
    h' = 0.

  BODY
    REPORT_TABLE("reportAll", " * ")

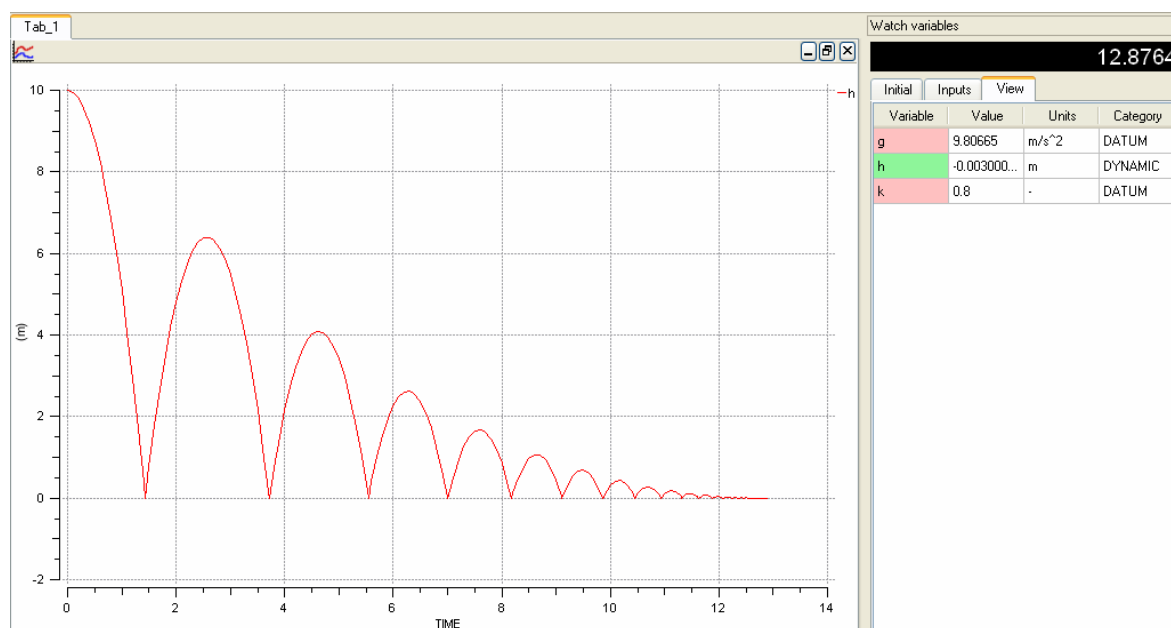
    RDIGITS = 8

    TSTOP = 15.
    CINT = 0.1
    INTEG()

END EXPERIMENT

```

We then plot the variable "h" which should be something similar to the following:



We see that at time 12.8764 the ball doesn't bounce any more and the simulation is automatically stopped.



9. More simulations in the installation

The default installation of the program includes several examples of libraries and experiments. This chapter gives an introduction to some examples.

The intention here is not to provide a detailed understanding of the models but to permit the user to appreciate the power of this program for modelling both simple and complex physical and chemical problems.

9.1. Examples in the DEFAULT_LIB library

The default library contains miscellaneous examples. Some of them are the following:

9.1.1. WHEN example

This is an example concerning the use of discrete events using WHEN statements and delayed assignments. Basically our intention is to maintain a temperature within certain minimum and maximum limits. The component we create is:

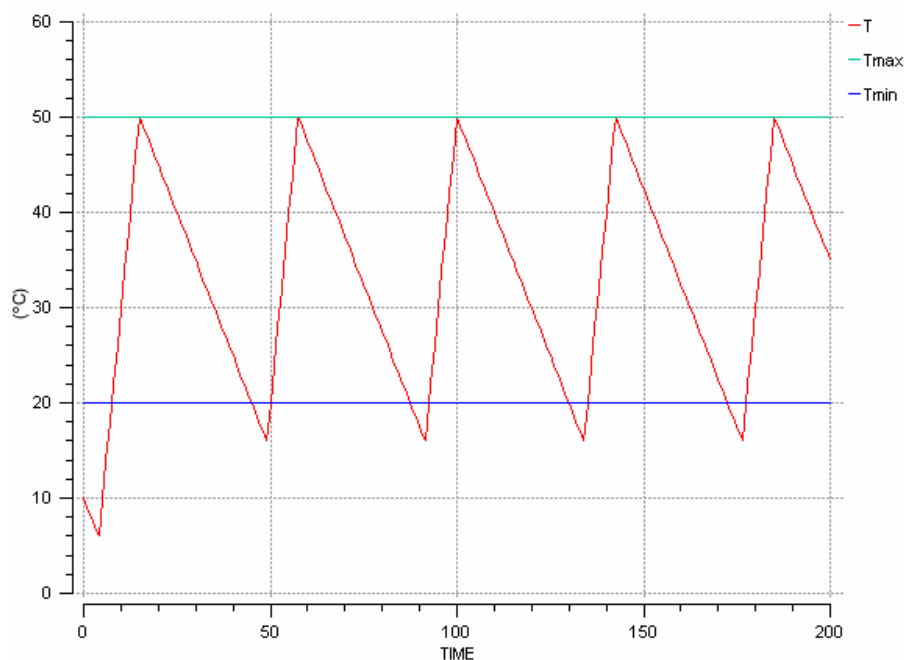
```

COMPONENT whenExample
  DATA
    REAL Tmin = 20
    REAL Tmax = 50.
  DECLS
    BOOLEAN HeaterON
    DISCR REAL HeaterPower
    REAL T = 10.
  DISCRETE
    WHEN (T < Tmin) THEN
      HeaterON = TRUE
      HeaterPower = 50. AFTER 4.
    END WHEN
    WHEN (T > Tmax) THEN
      HeaterON = FALSE
      HeaterPower = 0.
    END WHEN
  CONTINUOUS
    T' = 0.1 * (HeaterPower - 10)
END COMPONENT

```

The continuous part of this model merely introduces a dynamic law for the temperature based on the heater power. Two discrete events are introduced into the discrete part. The first is when the temperature falls below the limit Tmin. At this moment we connect the heater and we assign it a power, but this power is delayed for 4 seconds. This creates a time event at current time plus 4 seconds. The second discrete event models the opposite behaviour. When the temperature rises above the Tmax value, we disconnect the power to the heater.

To execute the default experiment, select "Simulation View" from the drop-down menu on the Toolbar, select the library DEFAULT_LIB, click "+" in "whenExample.default" at the bottom of the Workspace, click "exp1" and select the option "Simulate". This experiment takes the form of a transient case from 0 to 200 seconds with a communications interval of 1 second. It generates a report called "reportAll" containing all the simulation results. If you select the option "Simulate in Monitor" you should get the following result:



9.1.2. Copying graphs to Word

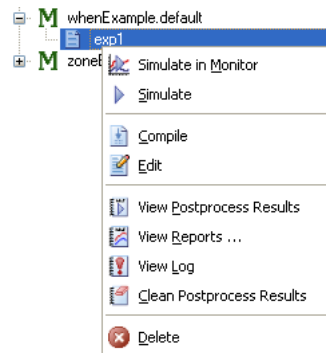
After running a simulation, it is straightforward to copy and paste graphs from the simulation monitor to Office applications such as Microsoft Word®. Simply proceed as follows:

- On the Simulation Monitor select the graph to be copied using the mouse
- Select the option Edit > Copy from the menu
- This can also be done from the contextual menu by right-clicking the graph and selecting the option “Copy”
- In your Office application(e.g. Word) select paste
- Using the mouse, enlarge the small icon to the desired size

9.1.3. Analyzing results in other Office applications

The user can easily visualize the simulation results by using standard applications like Excel®. To do this you should use the report file generated when the simulation was run. In the last example, the file was called "reportAll". Simply proceed as follows:

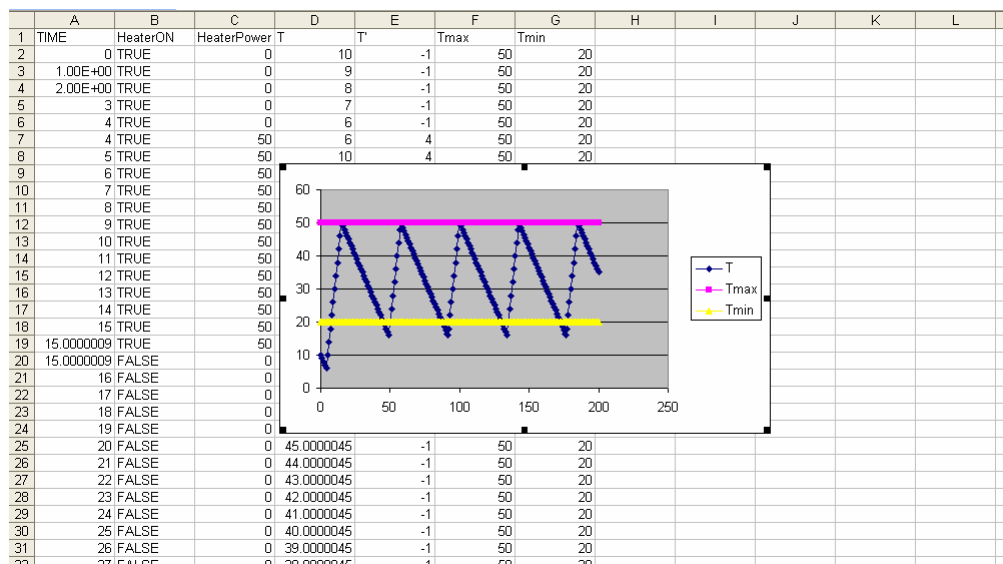
- In Simulation View, right-click the experiment name
- Select the option “View Reports...”



A window appears with the existing report files

You can open Excel® and drag-and-drop the file to Excel®

- Now that the results file is in Excel®, you can create graphics, reports, etc. For example:



9.1.4. The freezer

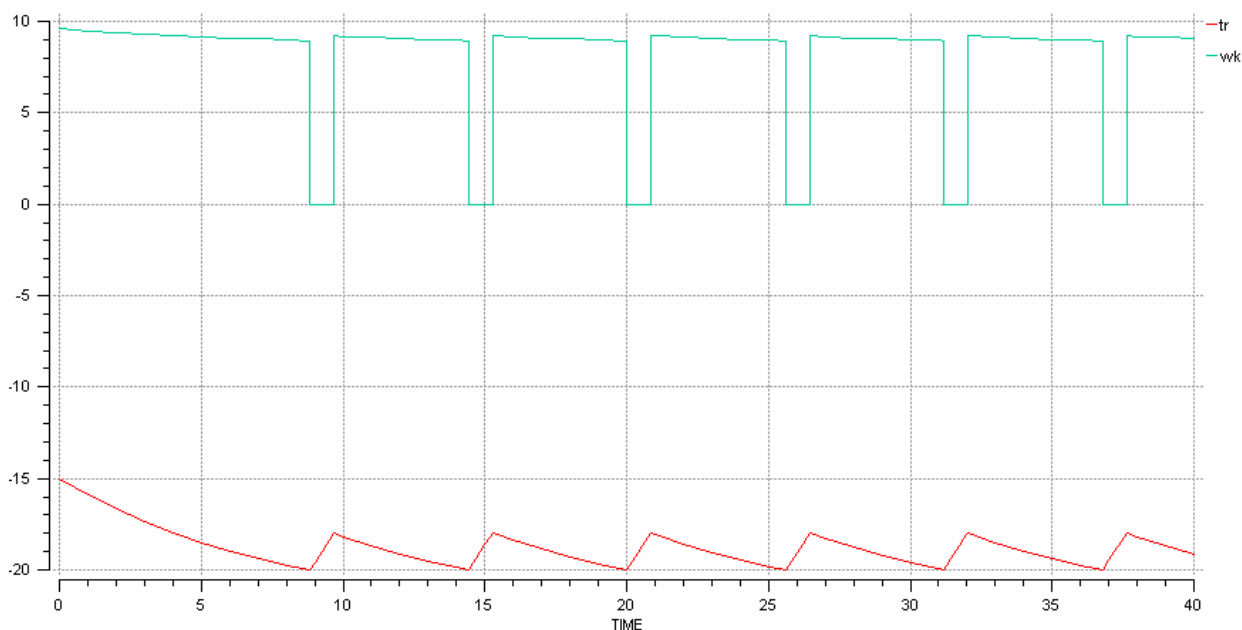
We can take a look at the freezer example from the DEFAULT_LIB library. This model is explained in the Math Algorithms & Simulation Guide. The objective is to model a freezer with all the thermodynamic equations.

The temperature inside the freezer should be maintained between -18 and -20 degrees. This is modelled using a boolean variable which shows whether the freezer is on or off:

```
WHEN ( tr > -18 ) THEN
    compressorON = TRUE
END WHEN
WHEN ( tr < -20 ) THEN
    compressorON = FALSE
END WHEN
```

We can run the experiment from 0 to 15 seconds with a communications interval of 0.1 seconds.

We can plot the temperature inside the freezer (tr) and the power consumption. It is possible to see how the compressor is switched on and off



ZONE Example

This example aims to teach you how to use the ZONE command. We will define a maximum and a minimum value for a function. The component is listed below:

```
COMPONENT zoneExample
  DATA
    REAL ymax = 0.8
    REAL ymin = -0.5
    REAL tau = 0.01
  DECLS
    REAL dy
    REAL x
    REAL y
  CONTINUOUS
    dy = (x - y) / tau
    y' = ZONE (y > ymax AND dy > 0) 0.
          ZONE (y < ymin AND dy < 0) 0.
          OTHERS dy
END COMPONENT
```

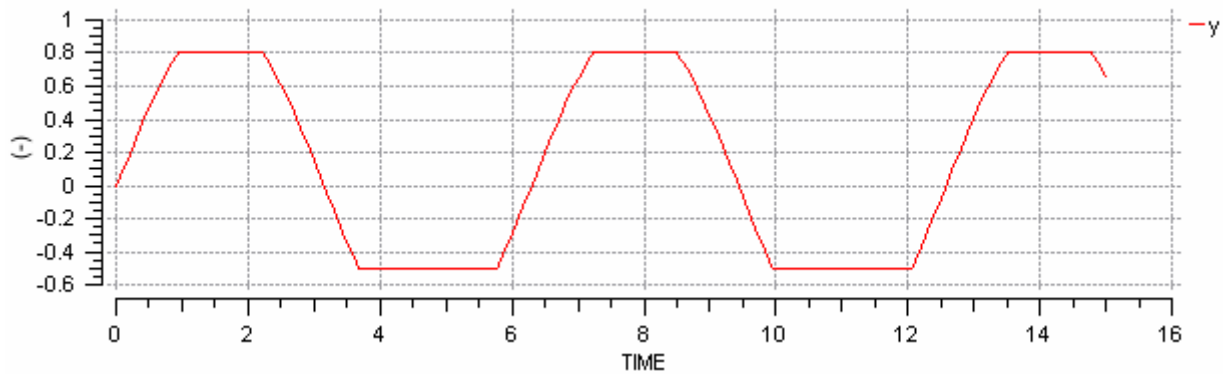
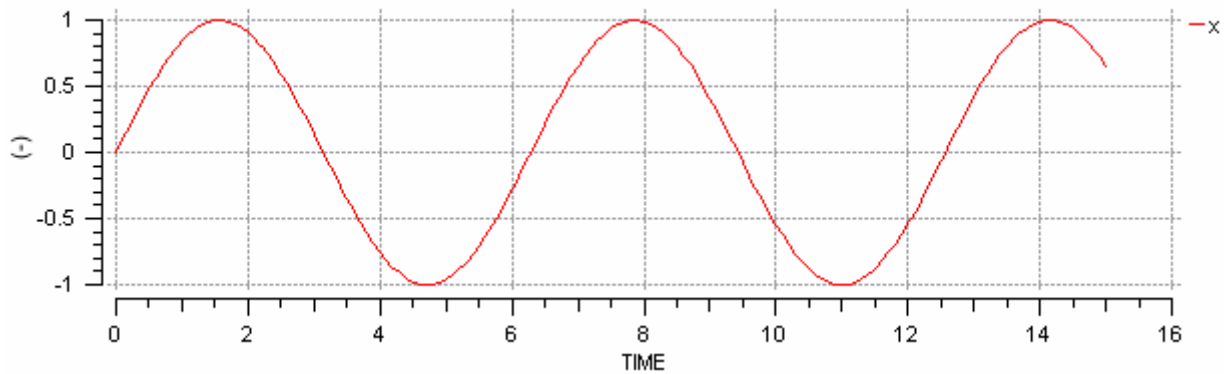
We use ZONE to make variable "y" constant when it is higher or lower than the limits, and it will be the solution of the differential equation in another case. To solve this equation, the evolution of variable "x" must be known. This will be added to the experiment list, in the BOUNDS block.

```
EXPERIMENT exp1 ON zoneExample.default
  DECLS
  INIT
    -- Dynamic variables
    y = 0
  BOUNDS
    x = sin(TIME)
```



```
BODY
  REPORT_TABLE("reportAll", " * ")
  TIME = 0
  TSTOP = 15
  CINT = 0.1
  INTEG()
END EXPERIMENT
```

The result is illustrated in the figure below, and it shows how the evolution of "y" is cut when it reaches the limits.



9.1.5. Lorentz equations

Lorentz' system of equations is a nonlinear differential system of equations. The solution is always rounding two space points, never reaching them. These two points are the steady solutions of the system, and it is easy to obtain them:

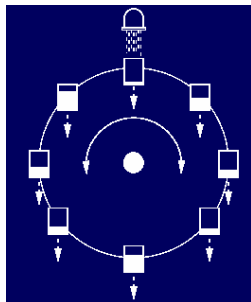
$$A = (\sqrt{b(r-1)}, \sqrt{b(r-1)}, (r-1)) \quad B = (-\sqrt{b(r-1)}, -\sqrt{b(r-1)}, (r-1))$$

The Lorentz equations were discovered by Ed Lorentz in 1963 as a very simplified model of convection rolls in the upper atmosphere. These same equations later appeared in studies of lasers and batteries, and in a simple chaotic waterwheel that can be easily built.

Lorentz found that, for certain settings, the trajectories of this system never settle down to a fixed point, never approach a stable limit cycle, and yet never diverge to infinity. What Lorentz discovered was at the time unheard of in the mathematical community, and was largely ignored for many years. Now this beautiful attractor is the most well known strange attractor that chaos has to offer.

A simple physical model of the Lorentz equations at work is a leaky waterwheel. A waterwheel built from paper cups with equal sized holes in the bottom of each cup is allowed to turn freely under the force of a steady stream of water poured into the top cup. With a slow flow of water, the water leaks out fast enough so that friction keeps the waterwheel from moving. With just a little more flow the waterwheel will pick a direction and spin in that direction forever. If the flow is increased further the waterwheel does not settle into a stable cycle. Instead it spins in one

direction for a bit, then slows down and starts to spin in the other. The waterwheel will constantly change its direction of spin, and never in a repeating predictable manner. Here is a picture of the waterwheel. Many more sophisticated but similar systems have been built, and they all show the same chaotic behaviour.



Lorentz equations are actually three differential equations, a first order equation for each of the y_1 , y_2 and y_3 components of the trajectory position. They are given as:

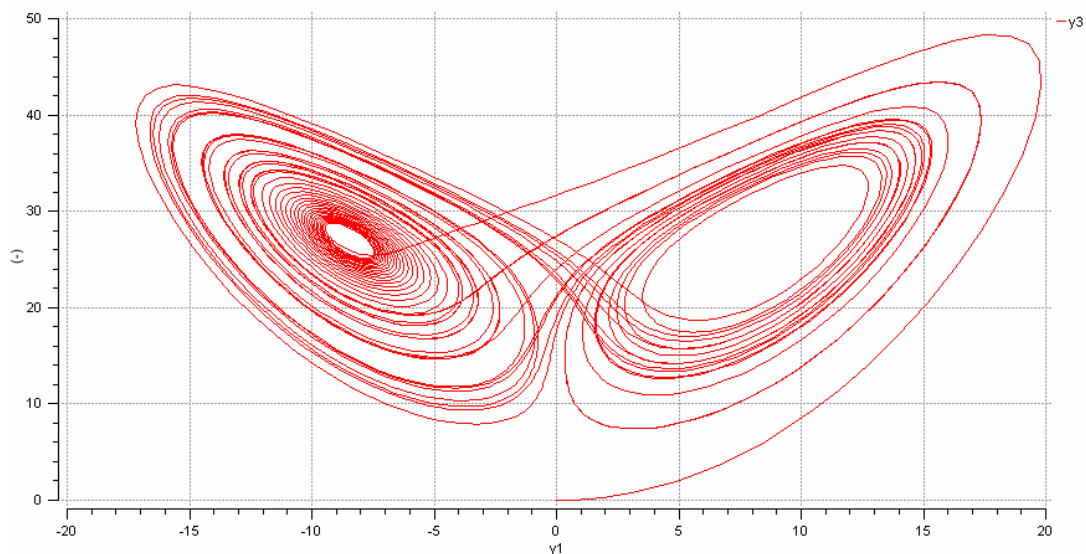
$$\begin{aligned}y_1' &= \text{alfa}*(y_2 - y_1) \\y_2' &= r*y_1 - y_2 - y_1*y_3 \\y_3' &= y_1*y_2 - b*y_3\end{aligned}$$

where r , b and alfa are parameters that change the behaviour of the system. There are a lot of resources available if you wish to study the Lorentz equations in detail. These equations are usually the first chaotic differential equations introduced in any book on chaos. Ed Lorentz' paper (1963) is also a very good source of information.

An component modelling the equations is listed below, and the default experiment can be run in the DEFAULT_LIB library, as in previous examples.

```
COMPONENT Lorentz
  DATA
    REAL alfa = 10
    REAL b = 8/3
    REAL r = 28
  DECLS
    REAL y1 = 0
    REAL y2 = 1
    REAL y3 = 0
  CONTINUOUS
    y1' = alfa*(y2 - y1)
    y2' = r*y1 - y2 - y1*y3
    y3' = y1*y2 - b*y3
END COMPONENT
```

The following graphic shows the evolution of the solution in the XZ plane, but similar graphics could be obtained in other planes.



9.2. Examples in the ELECTRICAL library

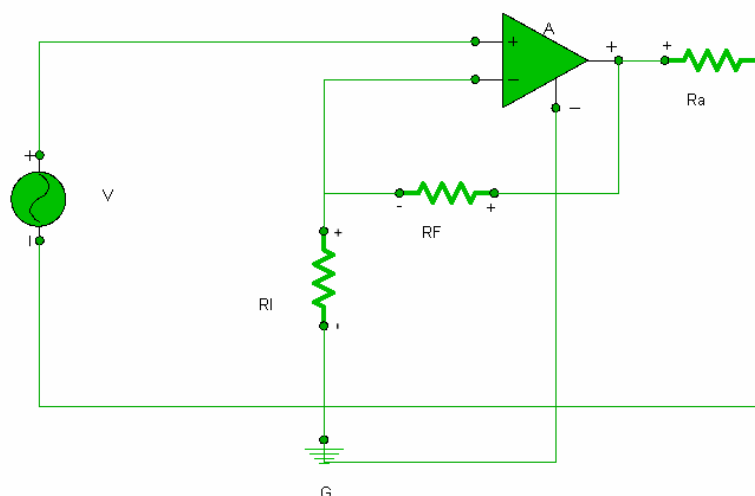
The software includes an Electrical library with components such as capacitors, resistors, diodes, thyristors, etc. The user can examine the source code of that library by selecting the tab Library and then deploying the ELECTRICAL library.

To examine the source code of this library, under the tab "Work Files" select ELECTRICAL library. Some files appear like ELEC_CompBasic.el, ELEC_CompIdeal.el, etc. The user can double-click any of these files to view them. All the components appear with their math formulations using EL language.

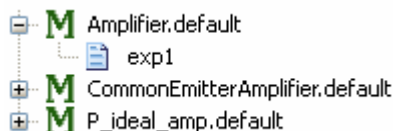
9.2.1. Diode bridge example

The user can run some experiments for this library. They are located in the ELECTRICAL_EXAMPLES library which is used to create experiments for the ELECTRICAL library.

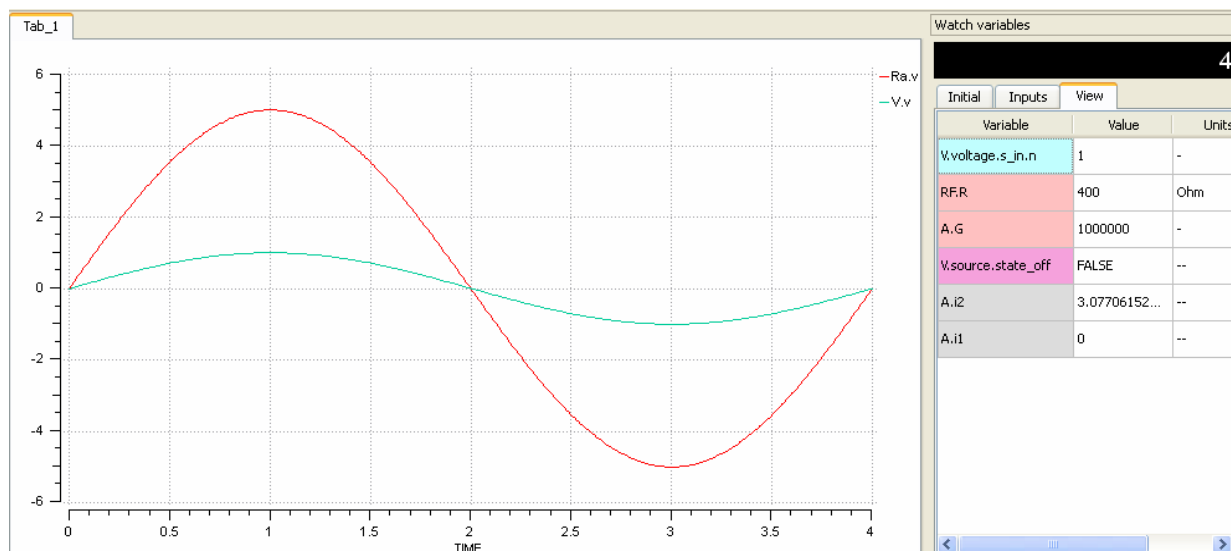
Select "Schematic View" from the drop-down menu on the Toolbar, then File > Open > Schematic... (or click the Open button on the Toolbar), select the library ELECTRICAL_EXAMPLES, the schematic "Amplifier" and click OK.



Since all the intermediate steps have already been carried out for this example, we can directly run a simulation for this circuit. Go to "Simulation View", select the library ELECTRICAL_EXAMPLES in the Workspace and open the tree "Amplifier.default".



Right-click on the experiment name (exp1) and select the option "Simulate in Monitor". Click Start simulation in the Monitor window. The result is as follows:

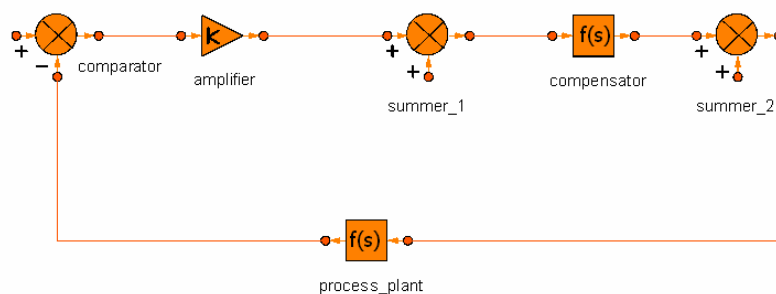


In this example you can see the voltage evolutions and how the amplifier works.

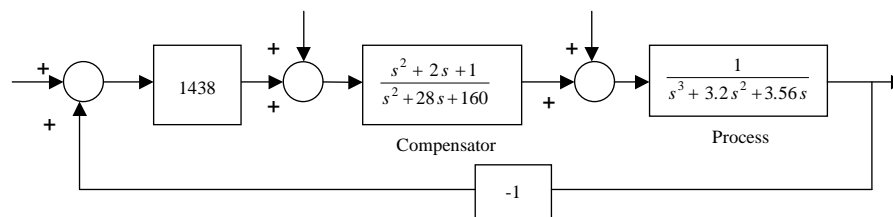
9.3. Examples in the CONTROL library

You can navigate in the CONTROL library in the same way as in the ELECTRICAL library. Typical controls such as P, PI; PID, etc, are modelled in EL.

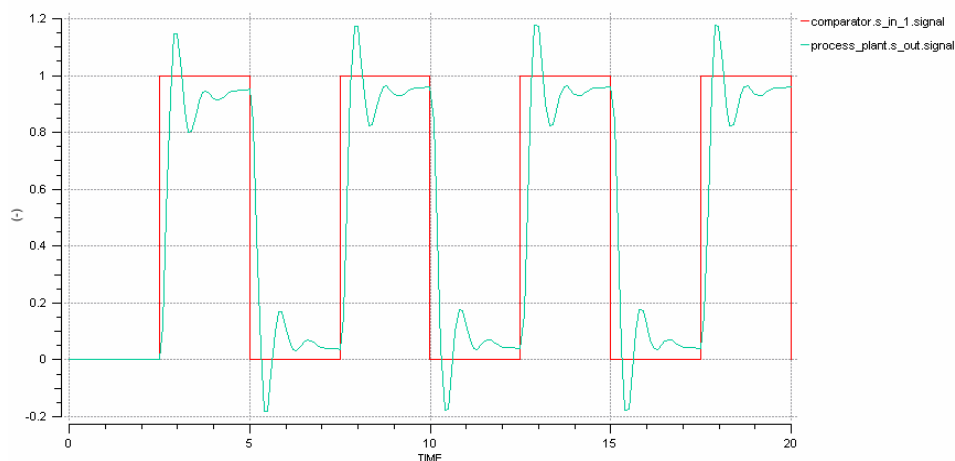
Any CONTROL experiment from the CONTROL_EXAMPLES library can be run. For instance, you can run the experiment prepared for the "linearSystem" component. The schematic of this component is as follows (obtained following the same steps as before):



This model represents a control system which receives the transfer functions as shown in the following figure:



In "Simulation View" select the partition linearSystem.default and run the simulation on the Experiment Monitor. This experiment creates a plot with two variables: "comparator.s_in_1.signal" and "process_plant.s_out.signal". The second variable should follow the first, and the result obtained is as follows:

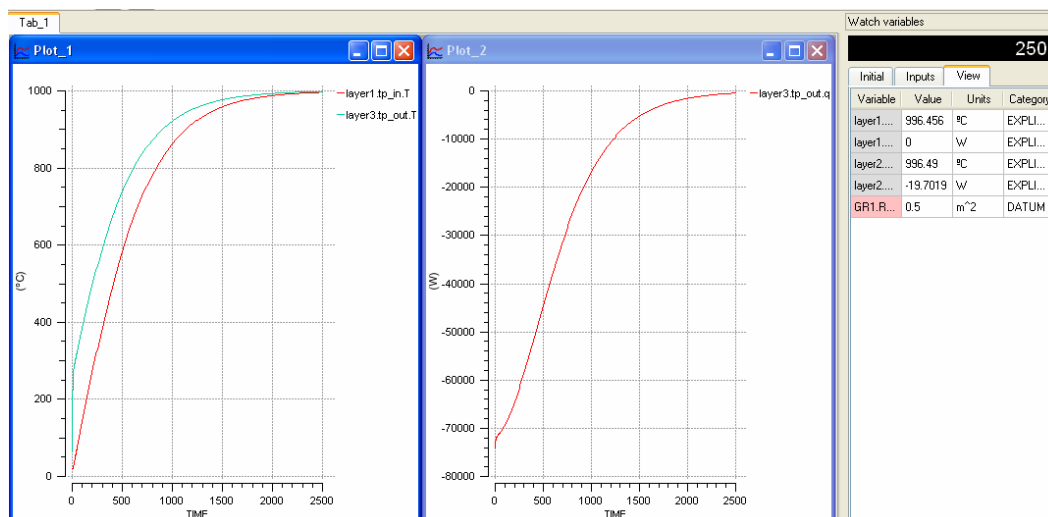


The figure shows the frequency signal to be followed by a series of pulses and the process output.

9.4. Examples in the THERMAL library

This is another library included in the tool. You can navigate through the components and source files as before. Typical thermal components such as conductors, diffusive thermal nodes, etc, are defined.

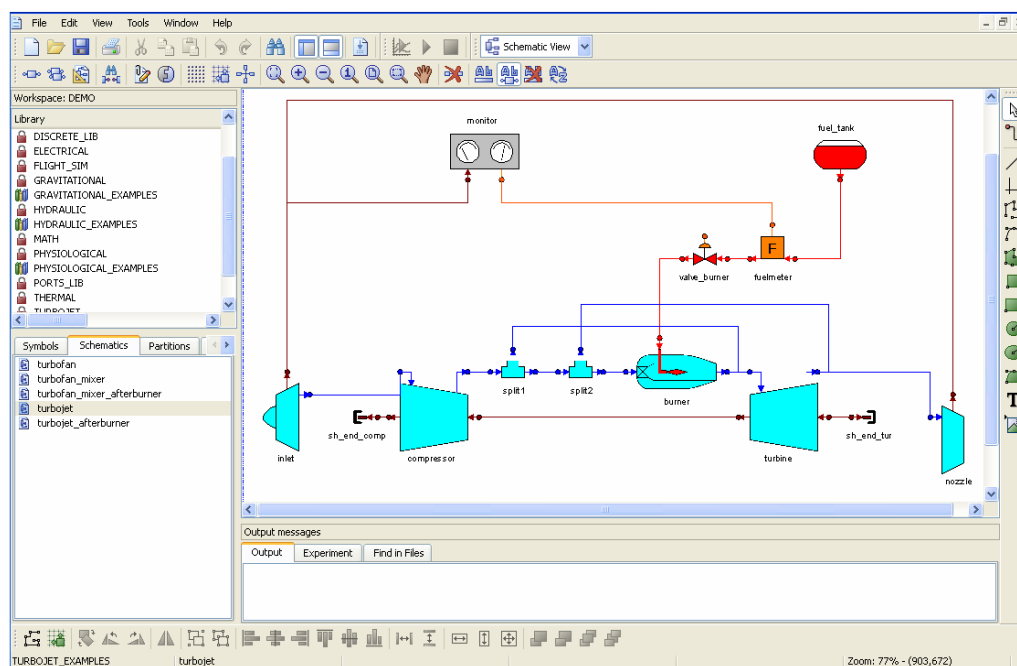
It is possible to run an experiment from the THERMAL_EXAMPLES library following the same steps indicate above; for example, the "Wall_3" model. This model contains three walls. The external layers are made of GFC material and the inner one of aluminium. The associated experiments can be run and the temperature evolution on each layer can be seen:



9.5. Examples in the TURBOJET library

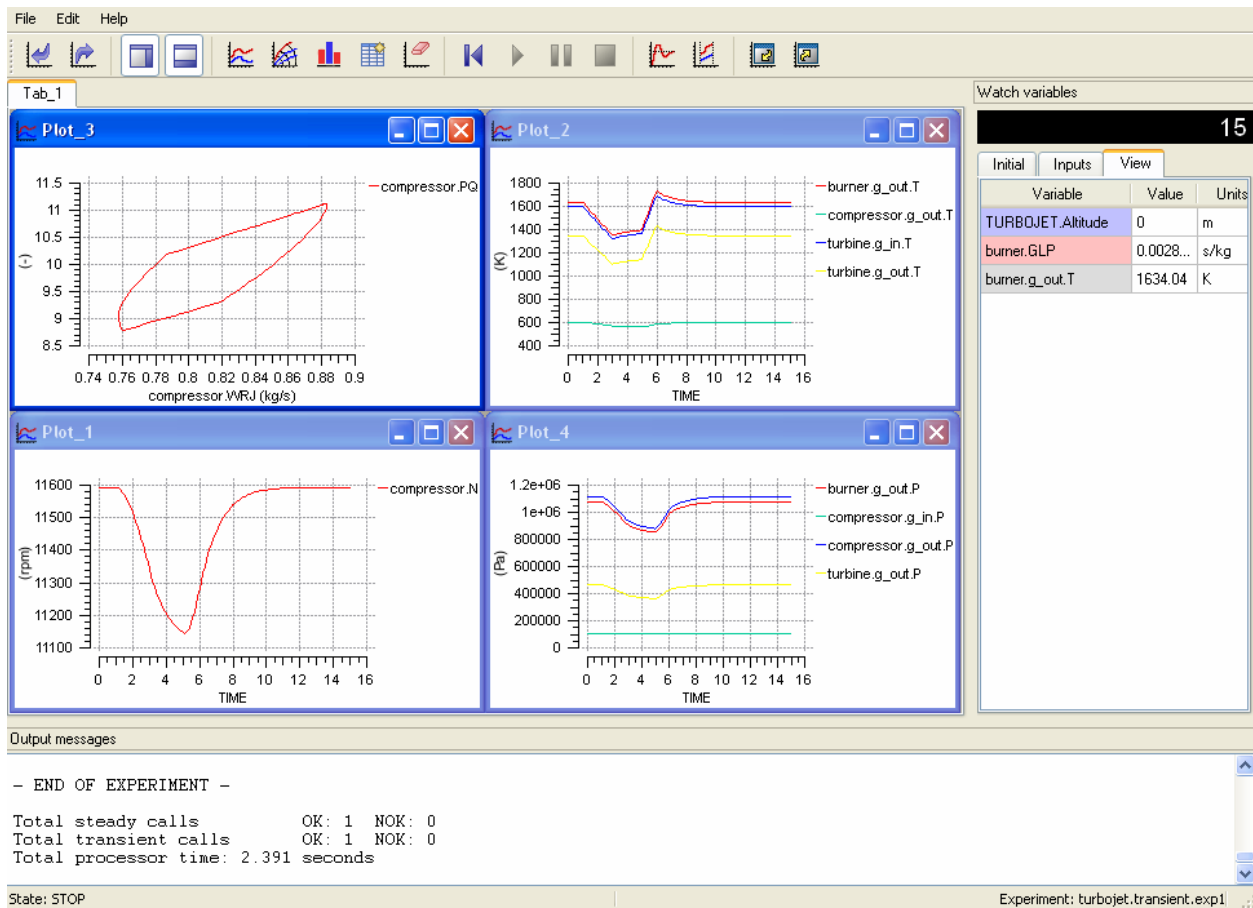
This library contains some typical components for turbojet modelling, such as compressors, turbines, burners, etc.

It is very impressive if you try to simulate a complete turbojet model. To do this, edit the file "turbojet.eds". This model represents a single spool turbojet engine. The components are: diffuser, compressor, two bypasses, combustion chamber, turbine, nozzle and a component to monitor the thrust and the specific fuel consumption.



For these components three different partitions have been defined: one for design called "Design" and used for steady state studies of this model, "Prediction" for multiple steady states, and finally "Transient" to study a transient state. There is not much room for detail in this brief explanation, but many physical problems are modelled here.

To run this model we suggest the user runs the "Transient" partition experiment. The following results are obtained:



The results are displayed when the fuel flow is varied and the pressures in the outlets of the most important elements are as illustrated in "Plot1".

EA Internacional
Magallanes, 3. 28015 Madrid. SPAIN.
E-mail: info@ecosimpro.com
Web: www.ecosimpro.com



EMPRESARIOS AGRUPADOS